The Power of Verification Queries in Message Authentication and Authenticated Encryption

Mihir Bellare* Oded Goldreich† Anton Mityagin‡

Abstract

This paper points out that, contrary to popular belief, allowing a message authentication adversary multiple verification attempts towards forgery is *not* equivalent to allowing it a single one, so that the notion of security that most message authentication schemes are proven to meet does not guarantee their security in practice. We then show, however, that the equivalence does hold for *strong* unforgeability. Based on this we recover security of popular classes of message authentication schemes such as MACs (including HMAC and PRF-based MACs) and CW-schemes. Furthermore, in many cases we do so with a *tight* security reduction, so that in the end the news we bring is surprisingly positive given the initial negative result. Finally, we show analogous results for authenticated encryption.

Keywords: Message authentication, PRFs, Carter-Wegman, symmetric encryption, authenticated encryption.

1 Introduction

Message authentication. A message authentication (ma) scheme allows parties sharing a key K to authenticate data they send to each other. The sender applies a *tag generation* algorithm TG to K and the message M to get a tag Tag, and then sends M, Tag to the receiver. The latter applies a verification algorithm VF to K, a received message, and its accompanying tag, to get an output of 1 (accept) or 0 (reject), indicating whether or not the message should be considered authentic.

Message authentication schemes are pervasive in practice. In a typical usage, public-key cryptography is first used to exchange a private key K, and the latter is then used to authenticate (and also possibly encrypt) data. This happens in Internet security protocols like SSL (used to secure credit card numbers in electronic commerce) SSH (secure remote login) and IPSEC.

Much work goes into obtaining high-performance, secure ma-schemes, and there are a variety of schemes in existence and use, including HMAC [1], block-cipher based MACs [5, 22, 10, 4, 11, 20, 17, 18] and Carter-Wegman (CW) ma-schemes [25, 21, 23, 16, 8, 9].

This paper points out that the notion of security that most of these schemes are proven to meet does not guarantee their security in practice, and then looks into remedies.

*Dept. of Computer Science & Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. Email: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. Supported in part by NSF grants CCR-0098123, ANR-0129617 and CCR-0208842, and by an IBM Faculty Partnership Development Award.

†Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. Email: oded.goldreich@weizmann.ac.il. URL: http://www.wisdom.weizmann.ac.il/~oded/homepage.html.

‡Dept. of Computer Science & Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. Email: amityagin@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/amityagin. Supported in part by above-mentioned grants of first author.

1.1 UF-1$\not\Rightarrow$UF-M

Two definitions. The natural definition of security [5] comes by extending the one for digital signatures [15]. Namely, an adversary, allowed a chosen-message attack (via access to a *tagging oracle* $\mathsf{TG}(K,\cdot)$), should be unable to produce a message-tag pair M, Tag that is *valid* (meaning $\mathsf{VF}(K, M, \mathit{Tag}) = 1$) and for which the message is *new* (meaning, was not queried to the tagging oracle).

This definition however gives the adversary only one verification attempt. An alternative definition [4] gives the adversary, in addition to the tagging oracle, also a *verification oracle* $\mathsf{VF}(K,\cdot,\cdot)$. The adversary should be unable to make a query M, Tag to its verification oracle such that the latter returns 1 but M was not previously queried to the tag oracle.

Let us refer to this second definition as UF-M (<u>U</u>n<u>F</u>orgeability under <u>M</u>ultiple verification queries). Note that the first (standard) definition we discussed above is just the special case of the second one in which only a single verification query is allowed, so henceforth we think of it this way and refer to it as UF-1 (<u>U</u>n<u>F</u>orgeability under a single verification query).

In practice, it is certainly possible for an adversary to make multiple verification attempts. (For example a server authenticating a client under their common key functions as the verification oracle, and an adversary can forward many transmissions to it.) A scheme that is UF-1 but not UF-M is thus clearly not providing adequate security for practical usage.

Equivalent? It turns out that this fact (namely, that UF-M is the "right" notion, not UF-1) is actually quite well understood in the community and literature. Nonetheless, it is the UF-1 notion that most schemes are proven to meet [5, 22, 1, 10, 11, 8, 9, 17, 18, 20]. (Exceptions are [4, 3].) Why?

It appears to be due to the belief that UF-1 $\Rightarrow$ UF-M. More precisely, the belief is that if an adversary A makes v verification oracle queries, its advantage (probability of forgery) is not more than v times that of an adversary B of comparable time making just one verification oracle query. So the belief is that although a difference in concrete security does manifest itself, UF-1 and UF-M are polynomially equivalent. The folklore argument for this is that B can guess one of the verification queries of A to use as its own, single verification query, answering previous ones by "0."

Not equivalent. In Section 3 we show why the above-mentioned folklore argument is incorrect. This leaves open the question of whether or not it can be patched to show the equivalence, but we then go on to show that it cannot. Namely we show in Theorem 4.1 that UF-1 $\not\Rightarrow$ UF-M. We do this by presenting a ma-scheme that is UF-1 but is not UF-M. (Naturally this requires the assumption that some UF-1 ma-scheme exists, otherwise the question is moot.) Thus, contrary to expectation, the UF-1 and UF-M notions are *not* equivalent.

Intuitively, the problem is malleability [12]. In particular multiple verification queries can add power when an adversary can modify a valid tag for a message into another, different valid tag for the same message.

1.2 SUF-1$\Rightarrow$SUF-M

Next we establish a useful positive result, namely that the equivalence (between security under one or many verification queries) does hold under a stronger notion of unforgeability than the one considered above.

Strong unforgeability. Strong unforgeability [6] means the adversary is not only unable to forge a tag for a new message, but also unable to forge a new tag for an old (meaning, already tagged) message. The formalization is with respect to adversaries getting both a tagging and a

$$
\begin{tikzcd}
\text{UF-1} \arrow[r, shift left, "/" marking] \arrow[d, shift right, "/" marking] & \text{SUF-1} \arrow[l, shift left] \arrow[d, shift right] \\
\text{UF-M} \arrow[r, shift left, "/" marking] \arrow[u, shift right] & \text{SUF-M} \arrow[l, shift left] \arrow[u, shift right]
\end{tikzcd}
$$

Figure 1: An arrow represents an implication and a barred arrow represents a separation.

---

verification oracle as above, so again there is both a single verification query version (that we denote SUF-1) and a multiple verification query version (that we denote SUF-M).

We show (cf. Theorem 5.1 and Corollary 5.2) that SUF-1 $\Rightarrow$ SUF-M. Thus, in contrast to (mere) unforgeability, the strong versions of the notions are equivalent.

Relations. Figure 1.2 summarizes the relations between the four notions we have discussed, in the style of [2]. For each pair $\mathbf{A}, \mathbf{B} \in \{\text{UF-1}, \text{UF-M}, \text{SUF-1}, \text{SUF-M}\}$ of notions we show one of the following: (1) An implication $\mathbf{A} \Rightarrow \mathbf{B}$ (this means we prove that any ma-scheme achieving $\mathbf{A}$ also achieves $\mathbf{B}$); or (2) A separation $\mathbf{A} \not\Rightarrow \mathbf{B}$ (this means we give a construction of a ma-scheme that meets notion $\mathbf{A}$ but does not meet notion $\mathbf{B}$). A separation $\mathbf{A} \not\Rightarrow \mathbf{B}$ is always under the (minimal) assumption that there exists a scheme meeting notion $\mathbf{A}$.

1.3 Repair: Establishing UF-M and SUF-M security of known schemes

The fact that UF-1 $\not\Rightarrow$ UF-M means that what has been proved about specific existing ma-schemes does not guarantee their security under multiple verification attempts and hence in practice, but does not mean that these particular schemes are not UF-M. We now ask whether it may be possible, via other routes, to prove UF-M security of existing schemes.

In answering this question, it would be much preferable to avoid re-entering the (sometimes complex) proofs of security of these ma-schemes. Our approach, instead, is to consider some special classes of ma-schemes such that (1) known schemes have been proved to fall into one of these classes, and (2) we can show that any scheme in the class is SUF-1. Then, via our SUF-1 $\Rightarrow$ SUF-M implication, and existing results about the schemes, we can conclude that they are UF-M (and in fact SUF-M) secure. We implement this program as follows.

The first class of ma-schemes we consider are UF-1 MACs. (A ma-scheme is a *message authentication code*, or *MAC*, if its tagging function is stateless and deterministic, and verification is done by applying the tagging function to compute the correct tag of the given message and comparing this with the candidate tag.) It is easy to see (cf. Proposition 6.1) that any UF-1 MAC is in fact SUF-1, and hence (by Corollary 5.2) UF-M and SUF-M. As an application, since HMAC, proven UF-1 in [1], is a MAC, the above tells us that it is UF-M and SUF-M secure.

An important subclass of MACs are PRF based ones, where the tagging function is pseudorandom. These are UF-1 [14, 5] and thus by the above automatically SUF-1, UF-M and SUF-M. Since the tagging algorithms of block-cipher based constructs like various CBC-MACs [5, 10, 22], PMAC [11], TMAC [17] and OMAC [18] are shown in the cited papers to be PRFs, we can conclude that the corresponding MACs are UF-M and SUF-M.

The second class of ma-schemes we consider are Carter-Wegman (CW) ma-schemes [25]. These schemes are randomized or counter-based, and hence not MACs, so Proposition 6.1 does not apply. However, via an extension of the standard proof establishing the UF-1 security of these ma-schemes [21, 23, 16, 8, 9], we can establish their SUF-1 security as well. Hence by Corollary 5.2, they too are UF-M and SUF-M. In particular, UMAC [8, 9] and MMH-MAC [16] are CW ma-schemes shown in the cited papers to be UF-1, and hence are UF-M and SUF-M by the above.

1.4 Reduction tightness and concrete security improvements

The results above are with respect to asymptotic definitions of security, where meeting a notion means that the success probability of any polynomial-time adversary is a negligible function of the security parameter. This is indeed the conceptually appropriate setting in which to study security relations at the first cut. However, for practical purposes, the concrete security of reductions can be important, and we now address this issue.

Quality of the SUF-1 $\Rightarrow$ SUF-M reduction, and its impact. The reduction of Theorem 5.1, which establishes that SUF-1 $\Rightarrow$ SUF-M, is not tight: given a polynomial-time adversary A making v verification attempts and succeeding with probability a, it delivers a polynomial-time adversary B making one verification attempt and succeeding with probability at least $b = a/v$. The above-mentioned results showing UF-M and SUF-M security of various classes of schemes inherit this loss in security through their reliance on Theorem 5.1. Unfortunately this loss can be significant in practice. Consider for example PRF-based MACs using a block cipher with block-length n. Typically [5, 10, 22, 11, 17, 18] it is shown that the probability of forgery under t tagging queries, each of at most m blocks, is roughly $m^2t^2/2^n$. With AES ($n = 128$) and assuming 1Kbyte messages ($m = 2^{13}$) this provides UF-1 security until around $t = 2^{51}$ messages are tagged, which is adequate if we only cared about UF-1, but of course we don't. Obtaining UF-M/SUF-M via Theorem 5.1, we get that the probability of forgery under v verification queries is roughly $vm^2t^2/2^n$, with t, m as before. Now, for example, a guarantee of security against $v = 2^{34}$ verification attempts requires that at most $t = 2^{34}$ messages are tagged. These numbers are not large enough for comfort, and things get substantially worse if we consider legacy ciphers with block length $n = 64$.

No better general reduction. The first question to ask given the above is whether there is a better reduction showing SUF-1 $\Rightarrow$ SUF-M. Proposition 7.1 shows that the answer is no. It does this by presenting a MAC for which there is a polynomial-time attack having constant probability of strong forgeability with v queries, but for which no polynomial-time attack making only one verification query succeeds in strong forgery with probability significantly better than $1/v$. This shows (cf. Corollary 7.2) that no reduction could deliver a reduction factor that is better than that of Theorem 5.1 by more than a constant factor, even for the special case of ma schemes that are MACs.

Improvements in special cases. Although a general concrete-security improvement is ruled out by the above, we are able to find such improvements for important subclasses of ma-schemes. Proposition 7.3 presents an essentially tight reduction of SUF-M to the pseudorandomness of the MAC in the case the latter is PRF-based. Continuing the above example, this will imply that the probability of forgery under v verification queries is roughly $m^2(v^2 + t^2)/2^n$, with v, t, m as before. Now (for 1Kbyte messages) we have a guarantee of security against $v = 2^{50}$ verification attempts even if up to $t = 2^{50}$ messages are tagged.

This is a substantial improvement. It is particularly valuable since so many ma-schemes are PRF-based [5, 10, 22, 11, 17, 18], and we note it can also be applied to HMAC, whose tagging algorithms is often assumed to be a PRF, due to its usage for tasks such as key-derivation. (We note the proof of [1] extends to establish its security as a PRF if we are willing to assume the underlying compression function is a PRF.

1.5 The case of authenticated encryption

There are two notions of integrity for authenticated encryption: integrity of plaintexts (an analog of standard unforgeability for ma-schemes) and integrity of ciphertexts (an analog of strong unforgeability for ma-schemes). In either case we may consider versions with one or more verification attempts (here the verification oracle takes a ciphertext and says whether or not it is valid) leading to four notions, INT-PTXT-1, INT-PTXT-M, INT-CTXT-1, INT-CTXT-M. In the literature, INT-CTXT-1 was defined in [19, 7], and INT-CTXT-M, INT-PTXT-M in [6]. INT-PTXT-1 was not defined prior to our paper. Our main results about ma-schemes extend, and in particular we show that INT-PTXT-1 $\not\Rightarrow$ INT-PTXT-M but INT-CTXT-1 $\Rightarrow$ INT-CTXT-M. This material is covered in Appendix B and the relations are depicted in Figure B.

1.6 Remarks

One might consider a multiple verification version of the definition of digital signatures as well, but it is clearly equivalent to the standard definition of [15] because verification takes place under a key that is public (and in particular available to the adversary).

In practice, "throttling" is often used to limit the number of verification attempts an adversary can make. (The verification server refuses further requests under a key for which some number of verification attempts have been previously rejected). It would be a mistake to think that the use of throttling means that UF-1 security suffices. For one thing, one surely cannot limit a user to just one verification attempt before revoking their key, for a few rejections can occur for natural reasons such as corrupted transmissions. Also, revocation of keys brings key-management costs. Thus we feel that for practice the most desirable situation is to have schemes providing security against a large number of verification attempts.

1.7 Related work

In his recent textbook [13], the second author of this paper noted that UF-1 $\not\Rightarrow$ UF-M, and also that the implication is true for the class of *unique* UF-1 ma-schemes. (In such a scheme there exists for every key and message at most one tag that the verification algorithm accepts.) The present paper represents the submission of this material to a research venue while adding significant improvements, extensions and applications. In particular we establish UF-M for SUF-1 schemes, which is a strictly larger class than the class of unique UF-1 ma-schemes. (Every unique UF-1 ma-scheme is SUF-1, but there are SUF-1 ma-schemes, such as CW-ma-schemes, that are not unique.) Other contributions of the paper not present in the textbook are: consideration of the UF-M security of specific existing ma-schemes in the literature as discussed in Section 1.3 above; tightness of the main reduction and concrete security improvements as discussed in Section 1.4 above; and authenticated encryption as discussed in Section 1.5 above.

2 Definitions

Notation. By $a\|b\|c\|\cdots$ we denote an encoding of objects $a, b, c, \ldots$ into a binary string in such a way that the constituents are uniquely recoverable in linear time. If n is an integer then $\langle n\rangle$ denotes its binary representation and $[n] = \{1, \ldots, n\}$. If x is a string then $|x|$ denotes its length and $x[i]$ denotes its i-th bit, for $i \in [|x|]$. If A is a randomized algorithm then $a \stackrel{\$}{\leftarrow} A(x, y, \cdots)$ means that A is executed on inputs $x, y, \ldots$ with fresh coins and a denotes the outcome. If S is a set then $s \stackrel{\$}{\leftarrow} S$ means that s is chosen uniformly at random from S. If x, y are objects then $x \leftarrow y$ means x is assigned the value y.

Message authentication schemes. A *message-authentication scheme* (ma-scheme) $\Pi = (\mathsf{TG}, \mathsf{VF})$ is a pair of polynomial-time algorithms, where

Experiment $\mathbf{Exp}^{\mathsf{type}}_{\Pi,A}(k)$

$K \xleftarrow{\$} \{0,1\}^{l(k)}$; $\mathsf{count} \leftarrow 0$; $\mathsf{f} \leftarrow 0$; $\mathsf{sf} \leftarrow 0$

Run A on input 1^k, replying to its oracle queries as follows:

– When A makes a query M to $\mathsf{TG}(K,\cdot)$ do

$\quad\quad$ $\mathit{Tag} \xleftarrow{\$} \mathsf{TG}(K,M)$; $\textsc{TagSet}[M] \leftarrow \textsc{TagSet}[M] \cup \{\mathit{Tag}\}$; Return Tag to A

– When A makes a query M, Tag to $\mathsf{VF}(K,\cdot,\cdot)$ do

$\quad\quad$ $d \leftarrow \mathsf{VF}(K,M,\mathit{Tag})$; $\mathsf{count} \leftarrow \mathsf{count}+1$

$\quad\quad$ If $d=1$ then

$\quad\quad\quad$ If $\textsc{TagSet}[M] = \emptyset$ then $\mathsf{f} \leftarrow 1$; If $\mathit{Tag} \notin \textsc{TagSet}[M]$ then $\mathsf{sf} \leftarrow 1$

$\quad\quad$ Return d to A

If (type =UF-1) then return $(\mathsf{count}=1) \wedge \mathsf{f}$

If (type =UF-M) then return f

If (type =SUF-1) then return $(\mathsf{count}=1) \wedge \mathsf{sf}$

If (type =SUF-M) then return sf

Figure 2: Experiment defining our four notions of security.

- The *tag-generation* algorithm TG, which may be randomized or stateful, takes a key $K \in \{0,1\}^{l(k)}$, where $k \in \mathbb{N}$ is the security parameter, and a *message* $M \in \{0,1\}^*$ to return a *tag* $\mathit{Tag} \in \{0,1\}^*$. The function $l(\cdot)$ is the *key-length*.
- The deterministic, stateless *verification* algorithm VF takes a key $K \in \{0,1\}^{l(k)}$, a message $M \in \{0,1\}^*$ and a candidate tag $\mathit{Tag} \in \{0,1\}^*$ to return either 1 (ACCEPT) or 0 (REJECT).

We require the following *completeness condition*: $\mathsf{VF}(K,M,\mathsf{TG}(K,M)) = 1$ with probability 1 for any k, any $K \in \{0,1\}^{l(k)}$, and any $M \in \{0,1\}^*$. We say that Π is a *message authentication code (MAC)* if TG is (stateless and) deterministic, and also $\mathsf{VF}(K,M,\mathit{Tag})$ is defined via: If $\mathit{Tag} = \mathsf{TG}(K,M)$ then return 1 else return 0.

For simplicity we will focus for now on stateless schemes. (Meaning ones where TG is stateless. Verification is always stateless). We also assume that TG applies to any message $M \in \{0,1\}^*$. In some later cases we will want to consider that the $\mathsf{TG}(K,\cdot)$ algorithm only applies to strings in some message space associated to k. (This happens because in practice the messages are often restricted in some way, for example to a maximum length or to length a multiple of some block length.) In that case the completeness condition only applies to messages in this space.

Four security notions. We provide the formal definitions and then some explanations. Let $\Pi = (\mathsf{TG},\mathsf{VF})$ be a message authentication scheme. Let $\mathsf{type} \in \{\text{UF-1}, \text{UF-M}, \text{SUF-1}, \text{SUF-M}\}$. Let A be an adversary with access to a tagging oracle and a verification oracle, and let $k \in \mathbb{N}$. We define the experiment depicted in Figure 2. We let

$$\mathbf{Adv}^{\mathsf{type}}_{\Pi,A}(k) \;=\; \Pr\left[\mathbf{Exp}^{\mathsf{type}}_{\Pi,A}(k) = 1\right]$$

denote the winning probability for A in the experiment in question. We say that Π is type-secure if the function $\mathbf{Adv}^{\mathsf{type}}_{\Pi,A}(\cdot)$ is negligible for any polynomial time adversary A.

In the experiment, naturally, A's tagging oracle queries are answered via $\mathsf{TG}(K,\cdot)$ and its verification oracle queries via $\mathsf{VF}(K,\cdot,\cdot)$. The rest is book-keeping. The flag f is set to 1 if the conditions of standard forgery are met, and the flag sf is set to 1 if the conditions of a strong forgery are met. It is assumed that the set $\textsc{TagSet}[M]$ is initially empty for all M. The counter count keeps track of the number of verification queries made.

PRFs and PRF-based MACs. Let F be a family of functions with domain $\{D_k\}_{k\in\mathbb{N}}$, output length $\tau(\cdot)$ and key-length $l(\cdot)$. This means that $F(K,M)$ is a $\tau(k)$-bit string for each k, each $l(k)$-bit key K and each input $M \in D_k$, and can be computed in polynomial time. Recall [14, 5] that if B is an adversary with an oracle for a function $f\colon\ D_k \to \{0,1\}^{\tau(k)}$ then its prf-advantage is

$$\mathbf{Adv}^{\mathrm{PRF}}_{F,B}(k) \ = \ \Pr\left[\,B^{F(K,\cdot)}(1^k)=1\,\right] - \Pr\left[\,B^{f(\cdot)}(1^k)=1\,\right] ,$$

the first probability being over $K \stackrel{\$}{\leftarrow} \{0,1\}^{l(k)}$ and the second over $f \stackrel{\$}{\leftarrow} \mathsf{Maps}(D_k,\{0,1\}^{\tau(k)})$, where $\mathsf{Maps}(D,R)$ is the set of all functions mapping from domain D to range R. We say that F is a PRF if $\mathbf{Adv}^{\mathrm{PRF}}_{F,B}(\cdot)$ is negligible for all polynomial-time adversaries B. We associate to F the ma-scheme $\Pi[F]=(F,\mathsf{VF})$, where $\mathsf{VF}(K,M,\mathit{Tag})$ returns 1 if $F(K,M)=\mathit{Tag}$ and 0 otherwise. The message space associated to k is D_k. A ma-scheme Π is said to be *PRF-based* if there is a family F such $\Pi=\Pi[F]$. Note that a PRF-based ma-scheme is a MAC.

3 Template of a reduction

Let $\Pi=(\mathsf{TG},\mathsf{VF})$ be a UF-1 secure ma-scheme. As we noted earlier, there is a belief that one can prove it also UF-M secure. Here we will present the template of the "proof" that people appear to have in mind and see why it does not work. This is useful for two reasons: it lends some insight into the later counter-example showing UF-1 $\not\Rightarrow$ UF-M, and the same template will be later appropriately used to show that SUF-1 $\Rightarrow$ SUF-M.

Let A be a polynomial-time UF-M adversary attacking Π, and let $v(\cdot)$ bound the number of verification-oracle queries made by A. (This means $v(k)\le \mathrm{poly}(k)$ is a polynomial-time computable function such that the number of verification queries made by A on input 1^k is at most $v(k)$ with probability one, regardless of how answers to any oracle queries of A are computed.) We want to define a UF-1 adversary B such that for every k the following inequality, which we refer to as $\mathbf{Eq}[A,B,k]$ holds:

$$\mathbf{Eq}[A,B,k]: \qquad \mathbf{Adv}^{\text{UF-1}}_{\Pi,B}(k) \ \ge \ \frac{1}{v(k)}\cdot \mathbf{Adv}^{\text{UF-M}}_{\Pi,A}(k)\ .$$

The idea is for B to guess and output the first successful verification query made by A, answering all others negatively without consulting its verification oracle. The corresponding adversary, which we call B_1, is defined by the part of the code of Figure 3 that omits the boxed statements. (Ignore the boxed statements for now.) Let I be the random variable, in $\mathbf{Exp}^{\text{UF-M}}_{\Pi,A}(k)$, that takes value the first verification query made by A to which the answer of the verification oracle is 1 and the message was not previously queried to the tagging oracle. Now we would like to claim that the guess guess made by B_1 equals I with probability at least $1/v(k)$, and thus $\mathbf{Eq}[A,B_1,k]$ is true.

However, this is not true. For example, A might make a query M to its tagging oracle, get back a tag Tag, and immediately make query M,Tag to its verification oracle. Let us imagine that, after this, it makes another verification oracle query M',Tag' that is valid, with $M'\ne M$, so that it wins with probability one. However, B_1's simulation is inaccurate, because (in the case $\mathsf{guess}\ge 2$) the verification oracle would have returned 1 in answer to A's first query to it, but B_1 returns 0 to A as the answer. So B_1's advantage is 0. (In the case $\mathsf{guess}=1$, B_1 does not win because M was queried to the tagging oracle and is not new).

This seems easily fixed by comparing tags in verification oracle queries to ones returned previously by the tagging oracle. Namely, we consider adversary B_2 of Figure 3 that now includes the boxed statements. For our example A above, B_2's simulation is now correct, and thus $\mathbf{Eq}[A,B,k]$ is true. However, in general, this strategy is still wrong, meaning there are adversaries A for which $\mathbf{Eq}[A,B_2,k]$ does not hold. The reason is malleability. Suppose A begins by making a query M to

Adversary $B_b^{\mathsf{TG}(K,\cdot),\mathsf{VF}(K,\cdot,\cdot)}(1^k)$ $(b \in \{1,2\})$

$\mathsf{count} \leftarrow 0$; $\mathsf{guess} \xleftarrow{\$} [v(k)]$

Run A on input 1^k, replying to its oracle queries as follows:

– When A makes a query M to $\mathsf{TG}(K,\cdot)$ do

$\quad$ $Tag \xleftarrow{\$} \mathsf{TG}(K,M)$; $\boxed{\textsc{TagSet}[M] \leftarrow \textsc{TagSet}[M] \cup \{Tag\}\ ;}$ Return Tag to A

– When A makes a query M, Tag to $\mathsf{VF}(K,\cdot,\cdot)$ do

$\quad$ $\mathsf{count} \leftarrow \mathsf{count} + 1$

$\quad$ If $(\mathsf{count} > \mathsf{guess})$ then Halt

$\quad$ $\boxed{\text{If } (Tag \in \textsc{TagSet}[M]) \text{ then return 1 to } A}$

$\quad$ If $(\mathsf{count} < \mathsf{guess})$ then return 0 to A

$\quad$ If $(\mathsf{count} = \mathsf{guess})$ then $d \leftarrow \mathsf{VF}(K,M,Tag)$; Halt

Figure 3: UF-1 adversaries B_1, B_2 derived from UF-M adversary A. Adversary B_1 omits the boxed statements while B_2 includes them.

---

its tagging oracle, getting back a tag Tag, and is then capable of modifying Tag to some different value $\overline{Tag}$ such that $\mathsf{VF}(K,M,\overline{Tag}) = 1$. Let it then make query $M, \overline{Tag}$ to its verification oracle. As before, imagine that, after this, it makes another verification oracle query M', Tag' that is valid, with $M' \neq M$, so that it wins with probability one. Now, again, B_2's simulation is inaccurate, because (in the case $\mathsf{guess} \geq 2$) the verification oracle would have returned 1 in answer to A's first query to it, but B_2 returns 0 to A as the answer. So B_2's advantage is 0.

This time, it is not clear how to fix B_2, because in general it is not clear how to detect whether $\overline{Tag}$ is a valid tag for M without querying the verification oracle. Theorem 4.1 implies that this difficulty is not surmountable. On the other hand Theorem 5.1 shows that if Π was SUF-1 rather than merely UF-1 then this problem can be overcome to show that Π is also SUF-M.

4 UF-1$\not\Rightarrow$UF-M

The above discussion leaves open the question of whether UF-1 $\Rightarrow$ UF-M or not. Perhaps the proof could be patched? The following theorem implies that it cannot, because the underlying claim is simply not true.

Theorem 4.1 [UF-1 $\not\Rightarrow$ UF-M] Assume there exists a UF-1-secure ma-scheme. Then there exists a UF-1-secure ma-scheme that is *not* UF-M-secure. ▌

The rest of this section is devoted to a proof of Theorem 4.1. By assumption there exists a UF-1-secure ma-scheme $\Pi = (\mathsf{TG}, \mathsf{VF})$. Let l be its key-length. We build from it the ma-scheme $\overline{\Pi} = (\overline{\mathsf{MAC}}, \overline{\mathsf{VF}})$ whose constituent algorithms are specified below:

Algorithm $\overline{\mathsf{MAC}}(K,M)$	Algorithm $\overline{\mathsf{VF}}(K,M,\overline{Tag})$
$\quad Tag \leftarrow \mathsf{TG}(K,M)$	$\quad$ Parse $\overline{Tag}$ as $Tag \| \langle i \rangle$ where $i \in \{0,1,\ldots,\lvert K \rvert\}$
$\quad \overline{Tag} \leftarrow Tag \| \langle 0 \rangle$	$\quad d \leftarrow \mathsf{VF}(K,M,Tag)$
$\quad$ Return $\overline{Tag}$	$\quad$ If ($d = 0$ or $i = 0$) then return d
	$\quad$ If ($d = 1$ and $i \geq 1$) then return $K[i]$

Recall that $K[i]$ denotes the i-th bit of the key K. The key-length of the new scheme is also l. We first note that the scheme satisfies the completeness condition. Indeed, if $\overline{Tag}$ is an output of

$\overline{\mathsf{MAC}}(K, M)$ then it has the form $Tag\|\langle i\rangle$ with $i = 0$. Thus the first "If" statement of $\overline{\mathsf{VF}}(K, M, \overline{Tag})$ applies and returns d, which is 1 by completeness of Π. Now we present an attack to show $\overline{\Pi}$ is not secure against an adversary making more than one query to its verification oracle.

Claim 4.2 $\overline{\Pi}$ is not UF-M-secure.

Proof: Consider the following UF-M-adversary $\overline{A}$ against $\overline{\Pi}$:

Adversary $\overline{A}^{\overline{\mathsf{MAC}}(K,\cdot),\overline{\mathsf{VF}}(K,\cdot,\cdot)}(1^k)$

  Let M_1 be any message and let $\overline{Tag}_1 \xleftarrow{\$} \overline{\mathsf{MAC}}(K, M_1)$
  Parse $\overline{Tag}_1$ as $Tag_1\|\langle 0\rangle$
  For $i = 1, \ldots, l(k)$ do $L[i] \leftarrow \overline{\mathsf{VF}}(K, M_1, Tag_1\|\langle i\rangle)$
  Let M_2 be any message different from M_1
  $\overline{Tag}_2 \xleftarrow{\$} \overline{\mathsf{MAC}}(L, M_2)$; $d \leftarrow \overline{\mathsf{VF}}(K, M_2, \overline{Tag}_2)$

Adversary $\overline{A}$ obtains the tag of a message M_1 using its tagging oracle. The definition of the algorithm $\overline{\mathsf{VF}}(K, \cdot, \cdot)$ then tells us that, above, $L[i] = K[i]$ is the i-th bit of the key K for all $i \in [|K|]$. In other words, $\overline{A}$ has succeeded in recovering the key. Then it can, of course, easily win, by forging the tag of some new message M_2. (Note that its computation of $\overline{Tag}_2$ is not an oracle query. $\overline{A}$ simply runs algorithm $\overline{\mathsf{MAC}}$ with key L and message M_2.) Thus $\mathbf{Adv}^{\text{UF-M}}_{\overline{\Pi},\overline{A}}(k) = 1$. Since $\overline{A}$ is polynomial-time, the claim is justified. ▮

Next we show that $\overline{\Pi}$ retains the UF-1 security of Π. The intuition is simple. In order to make $\overline{\mathsf{VF}}(K, \cdot, \cdot)$ accept, an adversary must have a message-tag pair that is valid for $\mathsf{VF}(K, \cdot, \cdot)$. On the other hand, being limited to one verification oracle query, it cannot make any use of any information that the verification oracle returns in answer to this query, since its game is effectively over once the query is made. Here now is the formal claim and proof.

Claim 4.3 $\overline{\Pi}$ is UF-1-secure.

Proof: Let $\overline{A}$ be any polynomial-time UF-1-adversary attacking $\overline{\Pi}$. We can assume it makes exactly one query to its verification oracle. We construct a polynomial-time UF-1 adversary A attacking Π such that

$$\mathbf{Adv}^{\text{UF-1}}_{\overline{\Pi},\overline{A}}(k) \;\le\; \mathbf{Adv}^{\text{UF-1}}_{\Pi,A}(k)\ . \tag{1}$$

Here is how A works:

Adversary $A^{\mathsf{TG}(K,\cdot),\mathsf{VF}(K,\cdot,\cdot)}(1^k)$

  Run $\overline{A}$ on input 1^k, replying to its oracle queries as follows:
  – When $\overline{A}$ makes a query M to $\overline{\mathsf{MAC}}(K, \cdot)$ do
    $Tag \xleftarrow{\$} \mathsf{TG}(K, M)$; $\overline{Tag} \leftarrow Tag\|\langle 0\rangle$; Return $\overline{Tag}$ to $\overline{A}$
  – When $\overline{A}$ makes a query $M, \overline{Tag}$ to $\overline{\mathsf{VF}}(K, \cdot, \cdot)$ do
    Parse $\overline{Tag}$ as $Tag\|\langle i\rangle$ where $i \in \{0, 1, \ldots, k\}$; $d \leftarrow \mathsf{VF}(K, M, Tag)$; Halt

A does not return to $\overline{A}$ any answer to $\overline{A}$'s (unique) verification-oracle query. (Not knowing K, it would not know how). It simply uses this query to make its own verification-oracle query, and halts. Equation (1) is true because A's simulation of replies to queries of the tagging oracle is perfect, and also because $\overline{\mathsf{VF}}(K, M, Tag\|\langle i\rangle)$ returns 1 only if $\mathsf{VF}(K, M, Tag)$ returns 1. ▮

This concludes the proof of Theorem 4.1. We now make some remarks that help relate this result to upcoming ones.

Remark 4.4 The ma-scheme $\overline{\Pi}$ of our counter-example is not a MAC. Although $\overline{\mathsf{MAC}}(K,\cdot)$ is deterministic, the second requirement of MACs is violated: verification $\overline{\mathsf{VF}}(K,\cdot,\cdot)$ is not done by recomputing $\overline{\mathsf{MAC}}(K,\cdot)$ and comparing it with an input tag. This is relevant because otherwise we could contradict Proposition 6.1. ▮

Remark 4.5 The ma-scheme $\overline{\Pi}$ of our counter-example is not SUF-1 due to the following adversary:

Adversary $\overline{A}^{\overline{\mathsf{MAC}}(K,\cdot),\overline{\mathsf{VF}}(K,\cdot,\cdot)}(1^k)$
  Let M be any message and let $\overline{Tag} \xleftarrow{\$} \overline{\mathsf{MAC}}(K,M_1)$
  Parse $\overline{Tag}$ as $Tag\|\langle 0\rangle$; $d \leftarrow \overline{\mathsf{VF}}(K,M,Tag\|\langle 1\rangle)$

The verification oracle accepts if $K[1]=1$. Since K is chosen at random we have $\mathbf{Adv}^{\text{SUF-1}}_{\overline{\Pi},\overline{A}}(k) = 1/2$. This is relevant because otherwise we would contradict Corollary 5.2. ▮

5 Equivalence in the strong unforgeability case

In contrast to the case of standard unforgeability, under the strong unforgeability definition, the single and multiple verification query versions are equivalent. We prove the equivalence and also state the quantitative relation between the advantages, since this is important for practical applications.

Theorem 5.1 [SUF-1 ⇒ SUF-M] Let $\Pi = (\mathsf{TG},\mathsf{VF})$ be a ma-scheme. Let A be an adversary making at most $v(\cdot)$ verification queries. Then there exists an adversary B, making only one verification query, such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{SUF-M}}_{\Pi,A}(k) \;\le\; v(k)\cdot \mathbf{Adv}^{\text{SUF-1}}_{\Pi,B}(k)\ . \tag{2}$$

Furthermore, the number of tagging queries made by B is the same as the number made by A, and the running time of B is that of A plus $O((v+t)\tau)$ where $t(\cdot)$ is a bound on the number of tagging queries made by A and $\tau(\cdot)$ is a bound on the lengths of messages and tags in A's oracle queries. ▮

The following is immediate:

Corollary 5.2 [SUF-1 ⇒ SUF-M] If a ma-scheme is SUF-1 secure then it is also SUF-M secure. ▮

Since the relations SUF-M ⇒ UF-M and UF-M ⇒ UF-1 are trivial, it follows that a ma-scheme that is SUF-1 secure is also SUF-M, UF-M and UF-1 secure, meaning meets all the four notions we have considered.

Proof of Theorem 5.1: The required adversary B is exactly the one called B_2 in Figure 3. For the analysis, let $(M_1, Tag_1), (M_2, Tag_2), \ldots$ denote the verification-oracle queries made by A in $\mathbf{Exp}^{\text{SUF-M}}_{\Pi,A}(k)$. Let the random variable I take value the smallest index i such that $\mathsf{VF}(K,M_i,Tag_i) = 1$ and $Tag_i \notin \textsc{TagSet}[M_i]$ if such an index exists, and 0 otherwise.

In the simulation of A provided by B, we notice that if $\mathsf{guess} \le I$, then for all $1 \le i < \mathsf{guess}$, the reply provided by B to the i-th verification-oracle query of A is correct, meaning equals $\mathsf{VF}(K,M_i,Tag_i)$. On the other hand the random choice of guess is independent of the view of A as long as B has not halted. So

$$\mathbf{Adv}^{\text{SUF-1}}_{\Pi,B}(k) \;=\; \Pr[\,\mathsf{guess} = I \wedge I \ge 1\,] \;\ge\; \Pr[\,\mathsf{guess} = I\,]\cdot\Pr[\,I \ge 1\,] \;=\; \frac{1}{v(k)}\cdot \mathbf{Adv}^{\text{SUF-M}}_{\Pi,A}(k)\ .$$

This establishes Equation (2). ▮

6 Classes of SUF-1 ma-schemes

Towards applying Corollary 5.2 to existing ma-schemes, we establish SUF-1 for certain important classes.

MACs. Suppose ma-scheme $\Pi = (\mathsf{TG}, \mathsf{VF})$ is a MAC. This implies that for any message M and key K there is only one tag that VF will accept, namely $\mathsf{TG}(K, M)$. Thus:

Proposition 6.1 Let Π be a UF-1 MAC. Then it is also SUF-1. ▮

So by Corollary 5.2, any UF-1 MAC is UF-M and SUF-M. As an application, since HMAC is proven UF-1 in [1], the above shows it is UF-M and SUF-M.

PRF-based MACs. Any PRF-based MAC is UF-M secure [5], and, being a MAC, we can then apply the above to get UF-M, SUF-M security. It is worth stating the concrete security of the reduction underlying this easy result, however, since it provides the backdrop for our later improvement:

Proposition 6.2 [PRF-based MACs are SUF-M, basic reduction] Let F be a PRF with output-length $\tau(\cdot)$. Let A be an adversary making at most $v(\cdot)$ verification queries. Then there exists a PRF adversary B such that for any $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{SUF-M}}_{\Pi[F],A}(k) \;\leq\; \frac{v(k)}{2^{\tau(k)}} + v(k) \cdot \mathbf{Adv}^{\text{PRF}}_{F,B}(k) \;. \tag{3}$$

Furthermore, the number of oracle queries made by B is at most $t(\cdot) + 1$, and the running time of B is that of A plus $O((v + t)\tau)$, where $t(\cdot)$ is a bound on the number of tagging queries made by A. ▮

As an application, various CBC-MACs [5, 10, 22], PMAC [11], TMAC [17] and OMAC [18] are UF-M and SUF-M. (But Proposition 7.3 yields better results.)

CW-ma-schemes. The Carter-Wegman paradigm [25] is now yielding the most efficient ma-schemes. UMAC [8, 9] is the canonical example. These schemes are nonce-based (the nonce is either a counter or a random number) and in particular are not MACs in the sense defined above, so the above does not apply. Via a closer look at the proof of the UF-1 security [21, 23, 16, 8, 9], however, one can establish SUF-1 as well. Details follow. For simplicity we consider only the case where the nonce is a counter.

Let H be a family of functions with domain $\{D_k\}_{k\in\mathbb{N}}$, output length $\tau(\cdot)$ and key-length $l_H(\cdot)$. Let F be a family of functions with domain $\{\{0,1\}^k\}_{k\in\mathbb{N}}$, output length $\tau(\cdot)$ and key-length $l_F(\cdot)$. We associate to them the counter-based CW ma-scheme $\Pi[H, F]$ which has key-length $l_H(\cdot) + l_F(\cdot)$ and the following tagging and verification algorithms. Below, N is a k-bit counter which is initially 0 and then incremented with each invocation of TG.

Algorithm $\mathsf{TG}(HK \| FK, M)$	Algorithm $\mathsf{VF}(HK \| FK, M, Tag)$
$N \leftarrow N + 1$	Parse Tag as (N, σ)
$Tag \leftarrow (N, F(FK, N) \oplus H(HK, M))$	$\sigma' \leftarrow F(FK, N) \oplus H(HK, M)$
Return Tag	If $(\sigma = \sigma')$ then return 1 else return 0

The message space is the domain of H. The proof of the following is in Appendix A.1.

Proposition 6.3 [Security of CW-ma-schemes] Let families H, F be as above, both with output-length $\tau(\cdot)$. Assume H is an $\epsilon(\cdot)$-AXU family and F is a PRF. Let $\Pi[H, F]$ be the associated CW-ma-scheme as above. Let A be any adversary who makes at most $t(k) < 2^k$ queries to its tagging oracle and one query to its verification oracle. Then there exists a PRF adversary B such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{SUF-1}}_{\Pi[H,F],A}(k) \;\leq\; 2^{-\tau(k)} + \epsilon(k) + \mathbf{Adv}^{\text{PRF}}_{F,B}(k)\,. \tag{4}$$

Furthermore, the number of oracle queries made by B is $t(\cdot) + 1$ and the running time of B is that of A plus $O(t\tau)$. ▌

Since UMAC [8, 9] and MMH-MAC [16] were shown to be UF-1 in the cited papers, the above together with Corollary 5.2 shows they are UF-M and SUF-M.

7 Reduction tightness and concrete security improvements

Using Theorem 5.1 to establish UF-M and SUF-M security of ma-schemes results, as per Equation (2), in a factor $v(\cdot)$ loss in the advantage. As we have discussed in Section 1.4, this can be appreciable in practice. This section looks into getting better reductions.

No better SUF-1 $\Rightarrow$ SUF-M reduction. The first, natural question to ask is whether Theorem 5.1 represents the best possible reduction, or whether there is a better one. We claim there is no better one, even for the special case of ma-schemes that are MACs. To establish this we begin with the following, whose proof is in Appendix A.2:

Proposition 7.1 Assume PRFs exist. Then for any polynomial $v(\cdot)$ there exists a MAC Π such that

(1) There exists a polynomial-time adversary A that makes at most $v(\cdot)$ verification queries and achieves $\mathbf{Adv}^{\text{SUF-M}}_{\Pi,A}(k) = 1$, and

(2) For any polynomial-time adversary B making only one verification query, there is a negligible function $s(\cdot)$ such that $\mathbf{Adv}^{\text{SUF-1}}_{\Pi,B}(k) \leq 2/v(k) + s(k)$ for all $k \in \mathbb{N}$. ▌

Now let us see why this Proposition shows that the reduction of Theorem 5.1 cannot be improved. View a reduction as a transform $\mathcal{B}$ that given a ma-scheme Π, a polynomial v and a polynomial-time adversary A making v verification queries, outputs a polynomial-time SUF-1 adversary $\mathcal{B}(\Pi, v, A)$. We say that $\mathcal{B}$ has *reduction factor* $\phi(\cdot)$ if

$$\mathbf{Adv}^{\text{SUF-1}}_{\Pi,\mathcal{B}(\Pi,v,A)}(k) \;\geq\; \phi(v(k)) \cdot \mathbf{Adv}^{\text{SUF-M}}_{\Pi,A}(k)$$

for all Π, A and k. Theorem 5.1 provides a reduction with reduction factor $\phi(n) = 1/n$. Now we claim:

Corollary 7.2 If PRFs exist then the reduction factor of any reduction is $O(1/n)$. ▌

So no reduction has a reduction factor that is better than the one of Theorem 5.1 by more than a constant factor. The proof of Corollary 7.2 is in Appendix A.2.

One might observe that the MAC of Proposition 7.1 is neither SUF-1 nor SUF-M secure. This did not preclude it being useful with regard to establishing the un-improvability of the reduction of Theorem 5.1, for the ma-scheme in that theorem is also not assumed to be secure in any sense. Indeed, a reduction transforms an adversary of one type into an adversary of another type with some relation between the advantages, regardless of whether the underlying scheme is secure or not, so the tightness of its reduction factor may be assessed via its performance on insecure schemes.

Improvements for PRF based MACs. Although the above precludes improved concrete security results for MACs in general, we note that one can improve the reduction for PRF-based MACs given in Proposition 6.2 to show the following, whose proof is in Appendix A.3:

Proposition 7.3 [PRF-based MACs are SUF-M, improved version] Let F be a PRF with output-length $\tau(\cdot)$. Let A be an adversary making at most $v(\cdot)$ verification queries. Then there exists a PRF adversary B such that for any $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{SUF-M}}_{\Pi[F],A}(k) \;\leq\; \frac{v(k)}{2^{\tau(k)}} + \mathbf{Adv}^{\text{PRF}}_{F,B}(k)\;. \tag{5}$$

Furthermore, the number of oracle queries made by B is at most $v(\cdot) + t(\cdot)$, and the running time of B is that of A plus $O((v+t)\tau)$, where $t(\cdot)$ is a bound on the number of tagging queries made by A. ▮

The improvement relative to Proposition 6.2 is to eliminate the factor of $v(k)$ multiplying $\mathbf{Adv}^{\text{PRF}}_{F,B}(k)$ at the cost of allowing B an extra $v(\cdot)$ verification queries. However, typically $t(\cdot) \geq v(\cdot)$, so the number of oracle queries of B in fact only increases by a constant factor and the improvement is essentially for free. Now, under typical choices of $\tau(\cdot)$, the $v(k)2^{-\tau(k)}$ term is negligible compared to the other term, so roughly the bound is better than that of Proposition 6.2 by a factor $v(k)$. This forms the basis for the examples, discussed in Section 1.4, which showed that the improvement has appreciable practical impact. We note that as applications we obtain UF-M, SUF-M security for the CBC-MACs [5, 10, 22], PMAC [11], TMAC [17] and OMAC [18] with concrete security that is better than that envisaged at the time people believed UF-1 $\Rightarrow$ UF-M, even though we now know this implication is not even true.

Acknowledgments

We thank Bodo Moeller for comments on a previous draft.

References

[1] M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[2] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[3] M. Bellare, O. Goldreich and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[4] M. Bellare, R. Guérin and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. *Advances in Cryptology – CRYPTO '95*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.

[5] M. Bellare, J. Kilian and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, Vol. 61, No. 3, Dec 2000, pp. 362–399.

[6] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – ASIACRYPT '00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.

[7] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. *Advances in Cryptology – ASIACRYPT '00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.

[8] J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway. UMAC: Fast and Secure Message Authentication. *Advances in Cryptology – CRYPTO '99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999. [Preliminary version of [9]].

[9] J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway. UMAC: Fast and Secure Message Authentication. `http://www.cs.ucdavis.edu/~rogaway/papers/umac-full.ps`. [Full version of [8]].

[10] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. *Advances in Cryptology – CRYPTO '00*, Lecture Notes in Computer Science Vol. 1880, M. Bellare ed., Springer-Verlag, 2000.

[11] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology – EUROCRYPT '02*, Lecture Notes in Computer Science Vol. 2332, L. Knudsen ed., Springer-Verlag, 2002.

[12] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal of Computing*, Vol. 30, No. 2, 2000, pp. 391–437.

[13] O. Goldreich. Foundations of Cryptography: Volume II, Basic applications. *Cambridge University Press*, 2004.

[14] O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.

[15] S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, Vol. 17, No. 2, 1988, pp. 281–308.

[16] S. Halevi and H. Krawczyk. MMH: Message authentication in software in the Gbit/sec rate. *Fast Software Encryption '97*, Lecture Notes in Computer Science Vol. 1267, E. Biham ed., Springer-Verlag, 1997.

[17] T. Iwata and K. Kurosawa. TMAC: Two-Key CBC MAC. *Topics in Cryptology – CT-RSA '02*, Lecture Notes in Computer Science Vol. 2271, B. Preneel ed., Springer-Verlag, 2002.

[18] T. Iwata and K. Kurosawa. OMAC: One-key CBC MAC. *Fast Software Encryption '03*, Lecture Notes in Computer Science Vol. 2887, T. Johansson ed., Springer-Verlag, 2003.

[19] J. Katz and M. Yung. Unforgeable encryption and chosen-ciphertext secure modes of operation. *Fast Software Encryption '00*, Lecture Notes in Computer Science Vol. 1978, B. Schneier ed., Springer-Verlag, 2000.

[20] E. Jaulmes, A. Joux and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. *Fast Software Encryption '02*, Lecture Notes in Computer Science Vol. 2365, J. Daemen and V. Rijmen ed., Springer-Verlag, 2002.

[21] H. Krawczyk. LFSR-based hashing and authentication. *Advances in Cryptology – CRYPTO '94*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

[22] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *Journal of Cryptology*, Vol. 13, No. 3, 2000, pp. 315–338.

[23] P. Rogaway. Bucket hashing and its application to fast message authentication. *Jounral of Cryptology*, Vol. 12, No. 2, 1999, pp. 91–115.

[24] D. Stinson. Cryptography theory and practice, Second edition. Chapman and Hall/CRC, 2002.

[25] M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279.

A Proofs

This section provides proofs omitted from the body of the paper.

A.1 Proof of Proposition 6.3

Proof of Proposition 6.3: The main part of the security proof considers the thought experiment in which $F(FK, \cdot)$ is replaced by a random function $f(\cdot)$ with output-length $\tau(\cdot)$. Formally, this means we consider the security of the ma-scheme $\Pi[H, R]$ where R is the family of all functions having the same domain and output-length as F. We now bound $\mathbf{Adv}^{\text{SUF-1}}_{\Pi[H,R],A}(k)$.

By assumption A makes at most 2^k tagging queries, thus the tagging oracle never replies to two queries with the same nonce. Let $M, (N, S)$ denote the single verification query made by A. If the nonce N is new, then clearly the probability that the tag is correct is at most $2^{-\tau(k)}$ since f has never before been invoked on input N.

However, there could have been a previous query M' to the tagging oracle returning a tag of the form (N, S'). If $M' = M$ then the only way that (N, S) is a correct tag for M is that $S' = S$, so in this case there is no strong forgery. So assume $M' \neq M$. In that case the argument reduces to that of the UF-1 case. Namely, we have

$$S \oplus S' \;=\; (f(N) \oplus h(M)) \oplus (f(N) \oplus h(M')) = h(M) \oplus h(M')$$

and the probability that this can happen is at most $\epsilon(k)$ by assumption that H is $\epsilon(\cdot)$-AXU. In summary

$$\mathbf{Adv}^{\text{SUF-1}}_{\Pi[H,R],A}(k) \;\leq\; 2^{-\tau(k)} + \epsilon(k) \;.$$

Now one can lift this to the statement of the Proposition, using the assumption that F is a PRF, in a standard way. ▌

A.2 Proofs of Proposition 7.1 and Corollary 7.2

Proof of Proposition 7.1: Let $w(\cdot) = \lfloor \log_2 v(\cdot) \rfloor$. Let F be a PRF with output-length $w(\cdot)$ and let $\Pi = \Pi[F] = (\mathsf{TG}, \mathsf{VF})$ be the associated PRF-based MAC, with message space the domain of F and key-length that of F. (Note by definition $\mathsf{TG} = F$.) Adversary A works as follows:

Adversary $A^{\mathsf{TG}(K,\cdot),\mathsf{VF}(K,\cdot,\cdot)}(1^k)$

  Let M be an arbitrary message in the message space

  For each $\mathit{Tag} \in \{0,1\}^{w(k)}$ do $d(\mathit{Tag}) \leftarrow \mathsf{VF}(K, M, \mathit{Tag})$

Since $\mathsf{TG}(K, M) = F(K, M) \in \{0,1\}^{w(k)}$ we have $\mathbf{Adv}^{\text{SUF-M}}_{\Pi,A}(k) = 1$. The number of verification queries made by A is $2^{w(k)} \leq v(k)$. So we have item (1) of the Proposition. Now suppose B is a polynomial-time adversary making only one verification query. We associate to it the polynomial-time PRF adversary D that, given an oracle $f(\cdot)$, works as follows. It runs B on input 1^k. When B makes a query M to its tagging oracle, D returns $f(M)$ to B. When B makes a query M, Tag to its verification oracle, D halts, outputting 1 if $f(M) = \mathit{Tag}$ and M was not a query to the tagging oracle, and outputting 0 otherwise. Then

$$\Pr\left[D^{F(K,\cdot)}(1^k) = 1 \right] \;=\; \mathbf{Adv}^{\text{SUF-1}}_{\Pi,B}(k) \quad \text{and} \quad \Pr\left[B^{f(\cdot)}(1^k) = 1 \right] \leq \frac{1}{2^{w(k)}} \leq \frac{2}{v(k)} \, ,$$

where in the first case the probability is over a random choice of K and in the second over a random choice of function f. So we have

$$\mathbf{Adv}^{\text{SUF-1}}_{\Pi,B}(k) \leq \frac{2}{v(k)} + \mathbf{Adv}^{\text{PRF}}_{F,B}(k) \, .$$

Since the last function is negligible, we have item (2) of the Proposition. ▮

Proof of Corollary 7.2: Let $v(\cdot)$ be a polynomial and let Π be the MAC associated to it by Proposition 7.1. Let $\mathcal{B}$ be a reduction with reduction factor ϕ. Proposition 7.1 says there is a polynomial-time SUF-M adversary A, making $v(\cdot)$ verification queries, such that

$$\mathbf{Adv}^{\text{SUF-1}}_{\Pi,\,\mathcal{B}(\Pi,v,A)}(k) \;\leq\; \frac{\Theta(1)}{v(k)} \;=\; \frac{\Theta(1)}{v(k)} \cdot \mathbf{Adv}^{\text{SUF-M}}_{\Pi,A}(k)$$

for all k. So $\phi(n) = O(1/n)$. ▮

A.3 Proof of Proposition 7.3

Proof of Proposition 7.3: Adversary B is given input 1^k and an oracle $f\colon D_k \to \{0,1\}^{\tau(k)}$. It runs A on input 1^k. When A makes a query M to its tagging oracle, B responds with $f(M)$. When A makes a query M, Tag to its verification oracle, B returns 1 to A if $f(M) = \mathit{Tag}$ and 0 otherwise. Finally, B outputs 1 if A succeeded in strong forgery, and 0 otherwise. (In the first case it is betting f is an instance of F, and in the second it is betting f was chosen at random from $\mathsf{Maps}(D_k, \{0,1\}^{\tau(k)})$. The analysis is standard. ▮

B Results for authenticated encryption

An authenticated encryption scheme is a symmetric encryption scheme that simultaneously provides privacy and authenticity. We are not concerned here with the privacy. Let us state the definitions of authenticity we consider.

Experiment $\mathbf{Exp}^{\mathsf{type}}_{\Pi,A}(k)$

$K \xleftarrow{\$} \{0,1\}^{l(k)}$; $\mathsf{count} \leftarrow 0$; $\mathsf{fp} \leftarrow 0$; $\mathsf{fc} \leftarrow 0$

Run A on input 1^k, replying to its oracle queries as follows:

– When A makes a query M to $\mathsf{Enc}_K(\cdot)$ do

$C \xleftarrow{\$} \mathsf{Enc}_K(M)$; $\textsc{CtxtSet}M \leftarrow \textsc{CtxtSet}M \cup \{C\}$; Return C to A

– When A makes a query C to $\mathsf{De}^*(\cdot)$ do

$d \leftarrow \mathsf{De}^*_K(C)$; $\mathsf{count} \leftarrow \mathsf{count} + 1$

If $d = 1$ then

If $\textsc{CtxtSet}M = \emptyset$ then $\mathsf{fp} \leftarrow 1$; If $C \notin \textsc{CtxtSet}M$ then $\mathsf{fc} \leftarrow 1$

Return d to A

If (type =INT-PTXT-1) then return $(\mathsf{count} = 1) \wedge \mathsf{fp}$

If (type =INT-PTXT-M) then return fp

If (type =INT-CTXT-1) then return $(\mathsf{count} = 1) \wedge \mathsf{fc}$

If (type =INT-CTXT-M) then return fc

Figure 4: Experiment defining four notions of security of authenticated encryption scheme.

Four security notions. A symmetric encryption scheme $\Pi = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{De})$ is specified as usual via its key-generation, encryption and decryption algorithms. The last either returns a string or the symbol $\perp$ to indicate rejection. The scheme has an associated key-length $l(\cdot)$. We associate to the scheme a *verification algorithm* De^* that on input a key K and ciphertext C runs $\mathsf{De}(K, C)$, returning 0 if the result is $\perp$ and 1 otherwise. We now formally define four notions of integrity. (We use the term integrity synonymously with authenticity.) Let $\mathsf{type} \in \{$INT-PTXT-1, INT-PTXT-M, INT-CTXT-1, INT-CTXT-M$\}$. Let A be an adversary with access to an encryption oracle $\mathsf{Enc}(K, \cdot)$ and a verification oracle $\mathsf{De}^*(K, \cdot)$, and let $k \in \mathbb{N}$. We define the experiment depicted in Figure 4. We let

$$\mathbf{Adv}^{\mathsf{type}}_{\Pi,A}(k) \;=\; \Pr\left[\mathbf{Exp}^{\mathsf{type}}_{\Pi,A}(k) = 1\right]$$

denote the winning probability for A in the experiment in question. We say that Π is type-secure if the function $\mathbf{Adv}^{\mathsf{type}}_{\Pi,A}(\cdot)$ is negligible for any polynomial time adversary A.

In the experiment, access to the encryption oracle allows the adversary to mount a chosen-message attack. The adversary succeeds if it makes the verification oracle accept an "illegitimate ciphertext." Integrity of plaintexts calls a ciphertext illegitimate if the corresponding plaintext was never queried to the encryption oracle, while integrity of ciphertexts calls a ciphertext illegitimate if it was never returned by the encryption oracle. In each case one may allow either one or many verification queries, giving rise to the four notions we have defined. Historically, INT-PTXT-M and INT-CTXT-M are from [6] while INT-CTXT-1 is from [19, 7]. INT-PTXT-1 is new.

Results. We depict in Figure B the relations between the four notions of integrity for symmetric encryption schemes. As usual, arrows represent implications and barred arrows represent separations. INT-CTXT-M $\Rightarrow$ INT-PTXT-M and INT-PTXT-M $\not\Rightarrow$ INT-CTXT-M were known [6], and easily extend to show INT-CTXT-1 $\Rightarrow$ INT-PTXT-1 and INT-PTXT-1 $\not\Rightarrow$ INT-CTXT-1. Our main results are INT-PTXT-1 $\not\Rightarrow$ INT-PTXT-M and INT-CTXT-1 $\Rightarrow$ INT-CTXT-M. We also show that the latter reduction is tight. Formal statements follow.

Theorem B.1 [INT-PTXT-1 $\not\Rightarrow$ INT-PTXT-M] Assume there exists an INT-PTXT-1-secure sym-

$$
\begin{tikzcd}
\text{INT-PTXT-1} \arrow[r, shift left, "/" marking] \arrow[d, shift right, "/" marking] & \text{INT-CTXT-1} \arrow[l, shift left] \arrow[d, shift right] \\
\text{INT-PTXT-M} \arrow[u, shift right] \arrow[r, shift left, "/" marking] & \text{INT-CTXT-M} \arrow[l, shift left] \arrow[u, shift right]
\end{tikzcd}
$$

Figure 5: An arrow represents an implication and a barred arrow represents a separation.

---

metric encryption scheme. Then there exists an INT-PTXT-1-secure symmetric encryption scheme which is *not* INT-PTXT-M-secure. ▮

Proof: By assumption there exists a INT-PTXT-1-secure symmetric key encryption scheme $\Pi = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{De})$. The counterexample is the symmetric key encryption scheme $\overline{\Pi} = (\overline{\mathsf{Kg}}, \overline{\mathsf{Enc}}, \overline{\mathsf{De}})$ whose key-generation algorithm $\overline{\mathsf{Kg}}$ is the same as Kg and whose encryption and decryption algorithms are specified below:

Algorithm $\overline{\mathsf{Enc}}(K, M)$	Algorithm $\overline{\mathsf{De}}(K, \overline{\mathrm{c}})$
$\mathrm{c} \leftarrow \mathsf{Enc}(K, M)$	Parse c as $\mathrm{c} \| \langle i \rangle$ where $i \in \{0, 1, \ldots, \lvert K \rvert\}$
$\overline{\mathrm{c}} \leftarrow \mathrm{c} \| \langle 0 \rangle$	$M \leftarrow \mathsf{De}(K, \mathrm{c})$
Return $\overline{\mathrm{c}}$	If ($M = \perp$ or $i = 0$) then return M
	If ($M \neq \perp$ and $i \geq 1$ and $K[i] = 1$) then return M
	If ($M \neq \perp$ and $i \geq 1$ and $K[i] = 0$) then return $\perp$

This encryption scheme has the following 2 properties:

(1) There exists polynomial-time adversary $\overline{A}$ such that

$$\mathbf{Adv}^{\text{INT-PTXT-M}}_{\overline{\Pi}, \overline{A}}(k) = 1.$$

(2) Let $\overline{A}$ be any polynomial-time INT-PTXT-1-adversary attacking $\overline{\Pi}$. We can assume it makes exactly one query to its verification oracle. Then there exists a polynomial-time INT-PTXT-1 adversary A attacking Π such that

$$\mathbf{Adv}^{\text{INT-PTXT-1}}_{\overline{\Pi}, \overline{A}}(k) \;\leq\; \mathbf{Adv}^{\text{INT-PTXT-1}}_{\Pi, A}(k)\ .$$

The proofs of these claims are analogous to the ones for ma-schemes and thus we omit them. ▮

Theorem B.2 [INT-CTXT-1 $\Rightarrow$ INT-CTXT-M] Let $\Pi = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{De})$ be a symmetric encryption scheme. Let A be an adversary making at most $v(\cdot)$ verification queries. Then there exists an adversary B, making only one verification query, such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{INT-CTXT-M}}_{\Pi, A}(k) \;\leq\; v(k) \cdot \mathbf{Adv}^{\text{INT-CTXT-1}}_{\Pi, B}(k)\ . \tag{6}$$

Furthermore, the number of encryption queries made by B is the same as the number made by A, and the running time of B is that of A plus $O((v + t)\tau)$ where $t(\cdot)$ is a bound on the number of encryption queries made by A and $\tau(\cdot)$ is a bound on the lengths of messages and ciphertexts in A's oracle queries.

Proof: Adversary B is specified below:

Adversary $B^{\mathsf{Enc}(K,\cdot), \mathsf{De}^*(K,\cdot)}(1^k)$

$\quad \mathsf{count} \leftarrow 0\ ;\ \mathsf{guess} \xleftarrow{\$} [v(k)]\ ;\ \textsc{CtxtSet} \leftarrow \emptyset$

Run A on input 1^k, replying to its oracle queries as follows:

– When A makes a query M to $\mathsf{Enc}(K,\cdot)$ do

  $\mathrm{c} \xleftarrow{\$} \mathsf{Enc}(K,M)$; $\textsc{CtxtSet} \leftarrow \textsc{CtxtSet} \cup \{\mathrm{c}\}$; Return c to A

– When A makes a query c to $\mathsf{De}^*(K,\cdot)$ do

  $\mathsf{count} \leftarrow \mathsf{count} + 1$
  If ($\mathsf{count} > \mathsf{guess}$) then Halt
  If ($\mathrm{c} \in \textsc{CtxtSet}$) then return 1 to A
  If ($\mathsf{count} < \mathsf{guess}$) then return 0 to A
  If ($\mathsf{count} = \mathsf{guess}$) then $d \leftarrow \mathsf{De}^*(K,\mathrm{c})$; Halt

For the analysis, let $\mathrm{c}_1, \mathrm{c}_2, \ldots$ denote the verification-oracle queries made by A in $\mathbf{Exp}^{\text{INT-CTXT-M}}_{\Pi,A}(k)$. Let the random variable I take value the smallest index i such that $\mathsf{De}^*(K,\mathrm{c}_i) = 1$ and $\mathrm{c}_i \notin \textsc{CtxtSet}$, and 0 otherwise.

In the simulation of A provided by B, we notice that if $\mathsf{guess} \leq I$, then for all $1 \leq i < \mathsf{guess}$, the reply provided by B to the i-th verification-oracle query of A is correct, meaning equals $\mathsf{De}^*(K,\mathrm{c}_i)$. On the other hand the random choice of guess is independent of the view of A as long as B has not halted. So

$$\begin{aligned}
\mathbf{Adv}^{\text{INT-CTXT-1}}_{\Pi,B}(k) &= \Pr\left[\,\mathsf{guess} = I \wedge I \geq 1\,\right] \\
&\geq \Pr\left[\,\mathsf{guess} = I\,\right] \cdot \Pr\left[\,I \geq 1\,\right] \\
&= \frac{1}{v(k)} \cdot \mathbf{Adv}^{\text{INT-CTXT-M}}_{\Pi,A}(k) \ .
\end{aligned}$$

This establishes Equation (6). ▮

Corollary B.3 [SUF-1 ⇒ SUF-M] If a symmetric encryption scheme is INT-CTXT-1-secure then it is also INT-CTXT-M-secure. ▮

No Better INT-CTXT-1 ⇒ INT-CTXT-M Reduction. Theorem B.2 establishes INT-CTXT-M-security with a factor of $v(k)$ security loss. We claim that this loss is unavoidable and there is no better reduction. To prove it we will use the results for me-schemes from Section 7.

Proposition B.4 Assume PRFs exist. Then for any polynomial $v(\cdot)$ there exists a symmetric encryption scheme Π such that

(1) There exists a polynomial-time adversary A that makes at most $v(\cdot)$ verification queries and achieves $\mathbf{Adv}^{\text{INT-CTXT-M}}_{\Pi,A}(k) = 1$, and

(2) For any polynomial-time adversary B making only one verification query, there is a negligible function $s(\cdot)$ such that $\mathbf{Adv}^{\text{INT-CTXT-1}}_{\Pi,B}(k) \leq 2/v(k) + s(k)$ for all $k \in \mathbb{N}$.

Proof: Consider the MAC scheme $\Pi = (\mathsf{TG},\mathsf{VF})$ from the proof of Proposition 7.1. Using it we construct a symmetric encryption scheme $\Pi^* = (\mathsf{Kg},\mathsf{Enc},\mathsf{De})$ whose constituent algorithms are given below:

Algorithm $\mathsf{Kg}(1^k)$	Algorithm $\mathsf{Enc}(K,M)$	Algorithm $\mathsf{De}(K,\mathrm{c})$
$K \xleftarrow{\$} \{0,1\}^{l(k)}$	$\mathit{Tag} \leftarrow \mathsf{TG}(K,M)$	Parse c as $M \| \mathit{Tag}$, $\mathit{Tag} \in \{0,1\}^{\tau(k)}$
Return K	$\mathrm{c} \leftarrow M \| \mathit{Tag}$	$d \leftarrow \mathsf{VF}(M,\mathit{Tag})$
	Return c	If ($d = 1$) return M; otherwise return $\perp$

Following Proposition 7.1, there exists adversary A' against Π such that $\mathbf{Adv}^{\text{SUF-M}}_{\Pi,A'}(k) = 1$. Such adversary could be naturally modified to the adversary A against encryption scheme Π^*:

Adversary $A^{\mathsf{Enc}(K,\cdot),\mathsf{De}^*(K,\cdot)}(1^k)$

Run $A'(1^k)$ and answer its oracle queries as follows:

$\mathsf{TG}(M)$: c $\leftarrow \mathsf{Enc}(K, M)$; parse c as $M \| Tag$ where $Tag \in \{0,1\}^{\tau(k)}$; return Tag to A'

$\mathsf{VF}(M, Tag)$: c $\leftarrow M \| Tag$; $d \leftarrow \mathsf{De}^*(K, \mathrm{c})$; return d to A'

Note that A' wins SUF-M-experiment against Π exactly when A wins INT-CTXT-M-experiment against Π^* and thus $\mathbf{Adv}^{\text{INT-CTXT-M}}_{\Pi^*,A}(k) = \mathbf{Adv}^{\text{SUF-M}}_{\Pi,A'}(k) = 1$. This proves item (1) of the proposition.

Now consider an arbitrary INT-CTXT-1-adversary B against Π^*. Similarly to the previous case we modify B into the following SUF-1-adversary B' against Π:

Adversary $B'^{\mathsf{TG}(K,\cdot),\mathsf{VF}(K,\cdot,\cdot)}(1^k)$

Run $B(1^k)$ and answer its oracle queries as follows:

$\mathsf{Enc}(M)$: $Tag \leftarrow \mathsf{TG}(K, M)$; c $\leftarrow M \| Tag$; Return c to B

$\mathsf{De}^*(\mathrm{c})$: Parse c as $M \| Tag$, $Tag \in \{0,1\}^{\tau(k)}$; $d \leftarrow \mathsf{VF}(M, Tag)$; return d to B

Note that B wins INT-CTXT-1-experiment against Π^* exactly when B' wins SUF-1-experiment Π. By Proposition 7.1, $\mathbf{Adv}^{\text{SUF-1}}_{\Pi,B'}(k) < 2/v(k) + s(k)$ for some negligible function $s(\cdot)$. Therefore

$$\mathbf{Adv}^{\text{INT-CTXT-1}}_{\Pi^*,B}(k) \;=\; \mathbf{Adv}^{\text{SUF-1}}_{\Pi,B'}(k) \;<\; 2/v(k) + s(k)$$

as claimed. ▮

Note that the symmetric encryption scheme in our counter-example does not provide any privacy, since the plaintext is present in the clear in the corresponding ciphertext. This does not affect the fact that the example we construct shows what it is supposed to. It is possible to extend this example to one where the symmetric encryption scheme does provide privacy.

Recall that Theorem B.2 provides a reduction with a reduction factor $1/n$. Proposition B.4 implies than no better reduction factor is possible:

Corollary B.5 If PRFs exist then the reduction factor of any INT-CTXT-1 $\Rightarrow$ INT-CTXT-M reduction is $O(1/n)$. ▮