Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions

Jee Hea An and Mihir Bellare

Dept. of Computer Science & Engineering, University of California at San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
{jeehea,mihir}@cs.ucsd.edu
URL: http://www-cse.ucsd.edu/users/{jeehea,mihir}

Abstract. Practical MACs are typically designed by iterating applications of some fixed-input-length (FIL) primitive, namely one like a block cipher or compression function that only applies to data of a fixed length. Existing security analyses of these constructions either require a stronger security property from the FIL primitive (eg. pseudorandomness) than the unforgeability required of the final MAC, or, as in the case of HMAC, make assumptions about the iterated function itself. In this paper we consider the design of iterated MACs under the (minimal) assumption that the given FIL primitive is itself a MAC. We look at three popular transforms, namely CBC, Feistel and the Merkle-Damgård method, and ask for each whether it preserves unforgeability. We show that the answer is no in the first two cases and yes in the third. The last yields an alternative cryptographic hash function based MAC which is secure under weaker assumptions than existing ones, although at a slight increase in cost.

1 Introduction

Directly (from scratch) designed cryptographic primitives (for example block ciphers or compression functions) are typically "fixed input-length" (FIL): they operate on inputs of some small, fixed length. However, usage calls for "variable input-length" (VIL) primitives: ones that can process inputs of longer, and varying lengths. Much cryptographic effort goes into the problem of transforming FIL primitives to VIL primitives. (To mention just two popular examples: the various modes of operation of block ciphers address this problem when the given FIL primitive is a block cipher and the desired VIL primitive a data encryption scheme; and the Merkle-Damgård iteration method [16,11] addresses this problem when the given FIL primitive is a collision-resistant compression function and the desired VIL primitive is a collision-resistant hash function.) In this paper, we will address this problem for the design of VIL-MACs in the case where the given FIL primitive is itself a MAC, which corresponds to a weak security assumption on the FIL primitive in this context. Let us begin by recalling some background. We then describe more precisely the problem we consider, its motivation and history, and our results.

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 252–269, 1999.

1.1 Background

MACs. Recall that a message authentication code (MAC) is the most common mechanism for assuring integrity of data communicated between parties who share a secret key k. A MAC is specified by a function g that takes the key k and data x to produce a tag $\tau = g(k, x)$. The sender transmits (x, τ) and the receiver verifies that $g(k, x) = \tau$. The required security property is *unforgeability*, namely that even under a chosen-message attack, it be computationally infeasible for an adversary (not having the secret key k) be able to create a valid pair (x, τ) which is "new" (meaning x has not already been authenticated by the legitimate parties). As the main tool for ensuring data integrity and access control, much effort goes into the design of (secure and efficient) MACs, and many constructions are known. These include block cipher based MACs like the CBC MAC [2] or XOR MACs [9]; hash function based MACs like HMAC [3] or MDx-MAC [20]; and universal hash function based MACs [10,23]. Many of the existing constructions of MACs fall into the category of FIL to VIL transforms. For example the CBC MAC iterates applications of a block cipher (the underlying FIL primitive), while hash function based MACs iterate (implicitly or explicitly) applications of the underlying compression function.

Assumptions underlying the transforms. Analyses of existing block cipher based MACs make stronger assumptions on the underlying FIL primitive than the unforgeability required of the final VIL-MAC. For example, security analyses of the CBC or XOR MACs provided in [6,9] model the underlying block cipher as a pseudorandom function, assumed to be "unpredictable" in the sense of [12], a requirement more stringent than unforgeability.

The security analysis of HMAC[1] provided in [3] makes two assumptions: that the (appropriately keyed) compression function is a MAC and also that the iterated compression function is "weakly collision resistant". Thus, the security of HMAC is not shown to follow from an assumption only about the underlying FIL primitive.

Universal hash function based MACs don't usually fall in the FIL to VIL paradigm, but on the subject of assumptions one should note that they require the use of block ciphers modeled as pseudorandom functions to mask the output of the (unconditionally secure) universal hash function, and thereby use assumptions stronger than unforgeability on the underlying primitives.

1.2 From FIL-MACs to VIL-MACs

The Problem. We are interested in obtaining VIL-MACs whose security can be shown to follow from (only) the assumption that the underlying FIL primitive is itself a MAC. In other words, we wish to stay within the standard paradigm of transforming a FIL primitive to a VIL-MAC, but we wish the analysis to make a minimal requirement on the security of the given FIL primitive: it need not be unpredictable, but need only be a MAC itself, namely unforgeable. This

[1] To be precise, the security analysis we refer to is that of NMAC, of which HMAC is a variant.

is, we feel, a natural and basic question, yet one that surprisingly has not been systematically addressed.

Benefits of reduced assumptions. It is possible that an attack against the pseudorandomness of a block cipher may be found, yet not one against its unforgeability. A proof of security for a block cipher based MAC that relied on the pseudorandomness assumption is then rendered void. (This does not mean there is an attack on the MAC, but it means the MAC is not backed by a security guarantee in terms of the cipher.) If, however, the proof of security had only made an unforgeability assumption, it would still stand and lend a security guarantee to the MAC construction. Similarly, collision-resistance of a compression function might be found to fail, but the unforgeability of some keyed version of this function may still be intact. (This is true for example for the compression function of MD5.) Thus, if the security analysis of a (keyed) compression-function based MAC relied only on an unforgeability assumption, the security guarantee on the MAC would remain.

Another possibility enabled by this approach would be to design FIL-MACs from scratch. Since the security requirement is weaker than for block ciphers, we might be able to get FIL-MACs that are faster than block ciphers, and thereby speed up message authentication.

1.3 Our Results

The benefit (of a VIL-MAC with a security analysis relying only on the assumption that the FIL primitive is a MAC) would be greatest if the construction were an existing, in use one, whose security could now be justified under a weaker assumption. In that case, existing MAC implementations could be left unmodified, but benefit from an improved security guarantee arising from relying only on a weaker assumption. Accordingly, we focus on existing transforms (or slight variants) and ask whether they preserve unforgeability.

CBC MAC. The first and most natural candidate is the CBC MAC. Recall that given a FIL primitive $f\colon \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$ its CBC MAC is the transform CBC[f], taking key $k \in \{0,1\}^\kappa$ and input $x = x_1 \ldots x_n \in \{0,1\}^{ln}$ to return y_n, where $y_i = f(k, y_{i-1} \oplus x_i)$ for $1 \le i \le n$, and $y_0 = 0^l$. We already know that if f is a pseudorandom function then CBC[f] is a secure MAC [6], and the question is whether the assumption that f itself is only a MAC is enough to prove that CBC[f] is a secure MAC. We show that it is not. We do this by exhibiting a f that is a secure MAC, but for which there is an attack showing that CBC[f] is not a secure MAC. (This relies of course on the assumption that some secure FIL-MAC exists, since otherwise the question is void.)

MD method. Next we look at Damgård's method [11] for transforming a keyed compression function $f\colon \{0,1\}^\kappa \times \{0,1\}^{\ell+b} \to \{0,1\}^\ell$ into a full-fledged hash function.[2] Actually our method differs slightly in the way it handles input-length

[2] The construction of Damgård is essentially the same as that of Merkle, except that in the latter, the given compression function is keyless, while in the former, it is keyed. Since MACs are keyed, we must use Damgård's setting here.

variability, which it does by using another key. Our nested, iterated construction, NI[f], takes keys k_1, k_2 and input $x = x_1 \ldots x_n \in \{0,1\}^{nb}$ to return $f_{k_2}(y_n \| \langle |x| \rangle)$, where $y_i = f_{k_1}(y_{i-1} \| x_i)$ for $1 \leq i \leq n$ and $y_0 = 0^\ell$ and $\langle |x| \rangle$ is the length of x written as a binary string of length exactly b bits.

Although the construction is (essentially) the one used in the collision-resistant hash setting, the analysis needs to be different. This is because of two central differences between MACs and hash functions: MACs rely for their security on a secret key, while hash functions (which, in the Damgård setting, do use a key) make this key public; and the security properties in question are different (unforgeability for MACs, and collision-resistance for hash functions).

We show that if f is a secure MAC then so is NI[f]. The analysis has several steps. As an intermediate step in the analysis we use the notion of weak-collision resistance of [3], and one of our lemmas provides a connection between this and unforgeability.

An appropriately keyed version of the compression function of any existing cryptographic hash function can play the role of f above, as illustrated in Section 4.1. This provides another solution to the problem of using keyed compression functions to design MACs. In comparison with HMAC, the nested, iterated construction has lower throughput because each iteration of the compression function must use a key. Implementation also requires direct access to the compression function, as opposed to being implementable only by calls to the hash function itself. On the other hand, the loss in performance is low, it is still easy to implement, and the supporting security analysis makes weaker assumption than that of HMAC.

Feistel. The Feistel transform is another commonly used method of increasing the amount of data one can process with a given FIL primitive. The basic transform doubles the input length of a given function f. The security of this transform as a function of the number of rounds r has been extensively analyzed for the problem of transforming a pseudorandom function into a pseudorandom permutation: Luby and Rackoff [15] showed that two rounds do not suffice for this purpose, but three do. We ask whether r rounds of Feistel on a MAC f result in a MAC. The answer is easily seen to be no for $r = 2$. But we also show that it remains no for $r = 3$, meaning that the 3-round Feistel transform that turns pseudorandom functions into pseudorandom permutations does not preserve unforgeability. Furthermore, even more rounds do not appear to help in this regard.

1.4 Related Work

The FIL to VIL question that we address for MACs is an instance of a classic one, which has been addressed before for many other primitives and has played an important role in the development of the primitives in question. The attraction of the paradigm is clear: It is easier to design and do security analyses for the "smaller", FIL primitives, and then build the VIL primitive on top of them.

The modes of operation of block ciphers were probably the earliest constructions in this area, but an analysis in the light of this paradigm is relatively

recent [5]. Perhaps the best known example is the Merkle-Damgård [16,11] iteration method used in the case of collision-resistant functions. Another early example is (probabilistic) public-key encryption, where Goldwasser and Micali showed that bit-by-bit encryption of a message preserves semantic security [13]. (The FIL primitive here is encryption of a single bit.) Extensive effort has been put into this problem for the case of pseudorandom functions (the problem is to turn a FIL pseudorandom function into a VIL one) with variants of the CBC (MAC) construction [6,19] and the cascade construction [4] being solutions. Bellare and Rogaway considered the problem and provided solutions for TCR (target-collision-resistant) hashing [7], a notion of hashing due to Naor and Yung [18] which the latter had called universal one-way hashing.

Curiously, the problem of transforming FIL-MACs to VIL-MACs has not been systematically addressed prior to our work. However, some constructions are implicit. Specifically, Merkle's hash tree construction [17] can be analyzed in the case of MACs. Bellare, Goldreich and Goldwasser use such a design to build incremental MACs [8], and thus a result saying that the tree design transforms FIL-MACs to VIL-MACs seems implicit here.

2 Definitions

Families of functions. A *family of functions* is a map $F\colon \mathit{Keys}(F) \times \mathit{Dom}(F) \to \mathit{Rng}(F)$, where $\mathit{Keys}(F)$ is the set of keys of F; $\mathit{Dom}(F)$ is some set of input messages associated to F; and $\mathit{Rng}(F)$ is the set of output strings associated to F. For each key $k \in \mathit{Keys}(F)$ we let $F_k(\cdot) = F(k, \cdot)$. This is a map from $\mathit{Dom}(F)$ to $\mathit{Rng}(F)$. If $\mathit{Keys}(F) = \{0,1\}^\kappa$ for some κ then the latter is the key-length. If $\mathit{Dom}(F) = \{0,1\}^b$ for some b then b is called the input length.

MACs. A MAC is a family of functions F. It is a FIL-MAC (fixed-input-length MAC) if $\mathit{Dom}(F)$ is $\{0,1\}^b$ for some small constant b, and it is a VIL-MAC (variable input length MAC) if $\mathit{Dom}(F)$ contains strings of many different lengths. The security of a MAC is measured via its resistance to existential forgery under chosen-message attack, following [6], which in turn is a concrete security adaptation to the MAC case of the notion of security for digital signatures of [14]. We consider the following experiment $\mathrm{Forge}(A, F)$ where A is an adversary (forger) who has access to an oracle for $F_k(\cdot)$:

Experiment $\mathrm{Forge}(A, F)$
  $k \leftarrow \mathit{Keys}(F)$; $(m, \tau) \leftarrow A^{F_k(\cdot)}$
  If $F_k(m) = \tau$ and m was not an oracle query of A
    then return 1 else return 0

We denote by $\mathbf{Succ}^{\mathrm{mac}}_F(A)$ the probability that the outcome of the experiment $\mathrm{Forge}(A, F)$ is 1. We associate to F its insecurity function, defined for any integers t, q, μ by

$$\mathbf{InSec}^{\mathrm{mac}}_F(t, q, \mu) \stackrel{\mathrm{def}}{=} \max_A \{\, \mathbf{Succ}^{\mathrm{mac}}_F(A) \,\} .$$

Here the maximum is taken over all adversaries A with "running time" t, "number of queries" q, and "total message length" μ. We put the resource names in quotes because they need to be properly defined, and in doing so we adopt some important conventions. Specifically, resources pertain to the experiment $\mathrm{Forge}(A, F)$ rather than the adversary itself. The "running time" of A is defined as the time taken by the experiment $\mathrm{Forge}(A, F)$ (we call this the "actual running time") plus the size of the code implementing algorithm A, all this measured in some fixed RAM model of computation. We stress that the actual running time includes the time of all operations in the experiment $\mathrm{Forge}(A, F)$; specifically it includes the time for key generation, computation of answers to oracle queries, and even the time for the final verification. To measure the cost of oracle queries we let Q_A be the set of all oracle queries made by A, and let $Q = Q_A \cup \{m\}$ be union of this with the message in the forgery. Then the number of queries q is defined as $|Q|$, meaning m is counted (because of the verification query involved). Note also that consideration of these sets means a repeated query is not double-counted. Similarly the total message length is the sum of the lengths of all messages in Q. These conventions will simplify the treatment of concrete security.

The insecurity function is the maximum likelihood of the security of the message authentication scheme F being compromised by an adversary using the indicated resources. We will speak informally of a "secure MAC"; this means a MAC for which the value of the insecurity function is "low" even for "reasonably high" parameter values. When exactly to call a MAC secure is not something we can pin down ubiquitously, because it is so context dependent. So the term secure will be used only in discussion, and results will be stated in terms of the concrete insecurity functions.

3 The CBC MAC Does not Preserve Unforgeability

Let $f\colon \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$ be a family of functions. For any fixed integer $n > 0$ we define the associated CBC MAC. It is the family of functions $\mathrm{CBC}[f]\colon \{0,1\}^\kappa \times \{0,1\}^{ln} \to \{0,1\}^l$ defined as follows:

Algorithm $\mathrm{CBC}[f](k, x_1 \ldots x_n)$
$y_0 \leftarrow 0^l$
For $i = 1, \ldots, n$ do $y_i \leftarrow f(k, y_{i-1} \oplus x_i)$
Return y_n

Here $k \in \{0,1\}^\kappa$ is the key, and x_i is the i-th l-bit block of the input message.

We know that if f is a pseudorandom function then $\mathrm{CBC}[f]$ is a secure MAC [6]. Here we show that the weaker requirement that f itself is only a secure MAC does not suffice to guarantee that $\mathrm{CBC}[f]$ is a secure MAC. Thus, the security of the CBC MAC needs relatively strong assumptions on the underlying primitive.

We stress that the number of message blocks n is fixed. If not, splicing attacks are well-known to break the CBC MAC. But length-variability can be dealt with

in a variety of ways (cf. [6,19]), and since the results we show here are negative, they are only strengthened by the restriction to a fixed n.

We prove our claim by presenting an example of a MAC f which is secure, but for which we can present an attack against CBC[f]. We construct f under the assumption that some secure MAC exists, since otherwise there is no issue here at all.

Assume we have a secure MAC g: $\{0,1\}^\kappa \times \{0,1\}^{2m} \to \{0,1\}^m$ whose input length is twice its output length. We set $l = 2m$ and transform g into another MAC f: $\{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$. We show that f is a secure MAC but CBC[f] is not. Below we present f as taking a κ-bit key k and an l-bit input $a = a_1 \| a_2$ which we view as divided into two m-bit halves.

Algorithm $f(k, a_1 a_2)$
$\sigma \leftarrow g(k, a_1 a_2)$
Return σa_1

That is, f_k on any input simply returns g_k on the same input, concatenated with the first half of f_k's input. It should be clear intuitively that f is a secure MAC given that g is a secure MAC, because the output of f contains a secure MAC on the input, and the adversary already knows the data a_1 anyway. The following claim relates the securities more precisely. Its proof can be found in [1].

Claim. Let g, f be as above. Then $\mathbf{InSec}^{\mathrm{mac}}_f(t, q, \mu) \leq \mathbf{InSec}^{\mathrm{mac}}_g(t, q, \mu)$.

We now show that the CBC MAC method is not secure if we use the function f as the underlying base function. The following claim says that there is an attack on CBC[f], which after obtaining the correct tag of only one chosen message, succeeds in forging the tag of a new message. The attack is for the case $n = 2$ of two block messages, so that both the chosen message and the one whose tag is forged have length $2l$.

Claim. There is a forger F making a single $2l$-bit query to CBC[f]$(k, \cdot)$ and achieving $\mathbf{Succ}^{\mathrm{mac}}_{\mathrm{CBC}[f]}(F) = 1$.

Proof. The attacker F is given an oracle for CBC[f]$(k, \cdot)$ and works as follows:

Forger $F^{\mathrm{CBC}[f](k,\cdot)}$
Let a_1, a_2 be distinct m-bit strings and let $x \leftarrow a_1 a_2 0^m 0^m$
$\sigma_2 \sigma_1 \leftarrow \mathrm{CBC}[f](k, x)$
$x'_1 \leftarrow a_1 a_2$; $x'_2 \leftarrow a_1 a_2 \oplus \sigma_1 a_1$; $x' \leftarrow x'_1 x'_2$
Return $(x', \sigma_1 a_1)$

Here F first defined the $2l$ bit message x. It then obtained its l-bit tag from the oracle, and split it into two halves. It then constructed the l-bit blocks x'_1, x'_2 as shown, and concatenated them to get x', which it output along with the claimed tag $\sigma_1 a_1$.

To show that this is a successful attack, we need to check two things. First that the forgery is valid, meaning $\mathrm{CBC}[f](k, x') = \sigma_1 a_1$, and second that the message x' is new, meaning $x' \neq x$.

Let's begin with the second. We note that the last m bits of x' are $a_2 \oplus a_1$. But F chose a_1, a_2 so that $a_2 \neq a_1$ so $a_2 \oplus a_1 \neq 0^m$. But the last m bits of x are zero. So $x' \neq x$.

Now let us verify that $\mathrm{CBC}[f](k, x') = \sigma_1 a_1$. By the definition of f in terms of g, and by the definition of $\mathrm{CBC}[f]$, we have

$$\begin{aligned}\mathrm{CBC}[f](k, x) &= f(k, f(k, a_1 a_2) \oplus 0^m 0^m) \\ &= f(k, g(k, a_1 a_2) a_1) \\ &= g(k, g(k, a_1 a_2) a_1) \| g(k, a_1 a_2)\ .\end{aligned}$$

This implies that $\sigma_1 = g(k, a_1 a_2)$ and $\sigma_2 = g(k, \sigma_1 a_1)$ in the above code. Using this we see that

$$\begin{aligned}\mathrm{CBC}[f](k, x') &= f(k, f(k, a_1 a_2) \oplus (a_1 a_2 \oplus \sigma_1 a_1)) \\ &= f(k, \sigma_1 a_1 \oplus (a_1 a_2 \oplus \sigma_1 a_1)) \\ &= f(k, a_1 a_2) \\ &= \sigma_1 a_1\end{aligned}$$

as desired. □

The construct f above that makes the CBC MAC fail is certainly somewhat contrived; indeed it is set up to make the CBC MAC fail. Accordingly, one reaction to the above is that it does not tell us anything about the security of, say, DES-CBC, because DES does not behave like the function f above. This reaction is not entirely accurate. The question here is whether the assumption that the underlying cipher is a MAC is sufficient to be able to prove that its CBC is also a MAC. The above says that no such proof can exist. So with regard to DES-CBC, we are saying that its security relies on stronger properties of DES than merely being a MAC, for example pseudorandomness.

4 The NI Construction Preserves Unforgeability

Here we define the nested, iterated transform of a FIL-MAC and show that the result is a VIL-MAC.

4.1 The Construction

We are given a family of functions $f\colon \{0,1\}^\kappa \times \{0,1\}^{\ell+b} \to \{0,1\}^\ell$ which takes the form of a (keyed) compression function, and we will associate to this the nested iterated (NI) function $\mathrm{NI}[f]$. The construction is specified in two steps; we first define the iteration of f and then show how to get $\mathrm{NI}[f]$ from that. See Figure 1 for the pictorial description.

Construction. As the notation indicates, the input to any instance function $f(k, \cdot)$ of the given family has length $\ell + b$ bits. We view such an input as divided into two parts: a *chaining variable* of length ℓ bits and a data block of length b bits. We associate to f its *iteration*, a family $\mathrm{IT}[f]\colon \{0,1\}^\kappa \times \{0,1\}^{\leq L} \to$

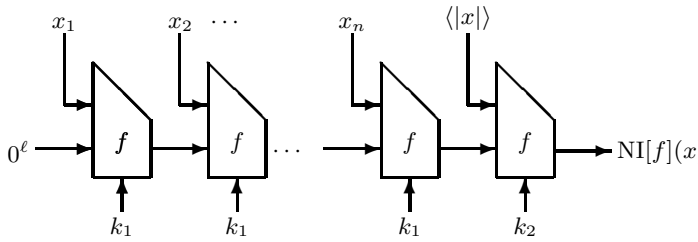

Fig. 1. *The nested, iterated construction of a VIL-MAC given a FIL-MAC f.*

$\{0,1\}^{\ell+b}$, where L is to be defined, and for any key k and string x of length at most L we define:

Algorithm IT[f](k, x)
$y_0 \leftarrow 0^\ell$
Break x into b-bit blocks, $x = x_1 \ldots x_n$
For $i = 1, \ldots, n$ do $y_i \leftarrow f(k, y_{i-1} \| x_i)$
$a \leftarrow y_n \| \langle |x| \rangle$
Return a

Above if $|x|$ is not a multiple of b, some appropriate padding mechanism is used to extend it. By $\langle |x| \rangle$ we denote a binary representation of $|x|$ as a string of exactly b bits. This representation is possible as long as $|x| < 2^b$, and so we set the maximum message length to $L = 2^b - 1$. This is hardly a restriction in practice given that typical values of b are large.

Now we define the family NI[f]: $\{0,1\}^{2\kappa} \times \{0,1\}^{\leq L} \to \{0,1\}^\ell$. A key for this family is a pair $k_1 k_2$ of κ-bit keys, and for a string x of length at most L we set:

Algorithm NI[f]($k_1 k_2, x$)
$a \leftarrow$ IT[f](k_1, x)
Return $f(k_2, a)$

Relation to other constructs. Our f has the syntactic form of a (keyed) compression function. The quantity y_n computed in the code of IT[f] is obtained via the iteration method of Damgård [11]; our iterated function is different only in that it appends to this the length of the input x. The main difference is in the security properties. Whereas Damgård assumes f is collision-resistant and wants to show that IT[f] is too, we assume f is a FIL-MAC and want to show NI[f] is a VIL-MAC. The difference is that for MACs the key is secret while for hash functions it is public, and the notions of security are not the same.

Preneel and Van Oorschot [20] suggest that in designing MACs from iterated hash functions, one might use a keyed compression function with a secret key and keyed output transformation. Modulo the handling of length-variability, this is exactly our construction. Preneel et. al. however did not analyze this construction under the assumption that the compression function is a MAC.

Comparing our construction to HMAC/NMAC, the difference, roughly speaking, is that HMAC is based on a hash function (like MD5 or SHA-1) that uses a compression function that is keyless, and iterated in the Merkle style [16]. Had we instead started with a hash function that iterated a keyed compression function in the Damgård style, and applied the HMAC transform to it, we would end up with essentially our construction. This tells us that the Damgård's setting and construction have a nice extra feature not highlighted before: they adapt to the MAC setting in a direct way.

Another difference between our construction and NMAC lies in how the output of the internal functions of the nested functions are formed. Our internal function $\mathrm{IT}[f]$ appends the length of the message and the appended length is a part of the function's output whereas F (in NMAC) applies the base function once more on the length of the message.

Instantiation. Appropriately keying the compression function of some existing cryptographic hash function will yield a candidate for f above. For example, let $\mathsf{sha\text{-}1}\colon \{0,1\}^{160+512} \to \{0,1\}^{160}$ be the compression function of SHA-1. We can key it via its 160-bit chaining variable. We would then use the 512 bit regular input as the input of the keyed function. This means we must further subdivide it into two parts, one to play the role of a new chaining variable and another to be the actual data input. This means we set $\kappa = \ell = 160$ and $b = 352$, and define the *keyed sha-1* compression function $\mathsf{ksha\text{-}1}\colon \{0,1\}^{160} \times \{0,1\}^{160+352} \to \{0,1\}^{160}$ by

$$\mathsf{ksha\text{-}1}(k, a\|b) \;=\; \mathsf{sha\text{-}1}(k\|a\|b)\;,$$

for any key $k \in \{0,1\}^{160}$, any $a \in \{0,1\}^{160}$ and any $b \in \{0,1\}^{352}$. Now, we can implement $\mathrm{NI}[\mathsf{ksha\text{-}1}]$ and this will be a secure MAC under the assumption that $\mathsf{ksha\text{-}1}$ was a secure MAC on 352 bit messages.

Note that under this instantiation, each application of $\mathsf{sha\text{-}1}$ will process 352 bits of the input, as opposed to 512 in a regular application of $\mathsf{sha\text{-}1}$ as used in SHA-1 or HMAC-SHA-1. So the throughput of $\mathrm{NI}[\mathsf{ksha\text{-}1}]$ is a factor of $352/512 \approx 0.69$ times that of HMAC-SHA-1. Also, implementation of $\mathrm{NI}[\mathsf{ksha\text{-}1}]$ calls for access to $\mathsf{sha\text{-}1}$; unlike HMAC-SHA-1, it cannot be implemented by calls only to SHA-1. On the other hand, the security of $\mathrm{NI}[\mathsf{ksha\text{-}1}]$ relies on weaker assumptions than that of HMAC-SHA-1. The analysis of the latter assumes that $\mathsf{ksha\text{-}1}$ is a secure MAC *and* that the iteration of $\mathsf{sha\text{-}1}$ is weakly collision-resistant; the analysis of $\mathrm{NI}[\mathsf{ksha\text{-}1}]$ makes only the former assumption.

4.2 Security Analysis

Our assumption is that f above is a secure FIL-MAC. The following theorem says that under this condition (alone) the nested iterated construction based on f is a secure VIL-MAC. The theorem also indicates the concrete security of the transform.

Theorem 1. *Let* $f\colon \{0,1\}^{\kappa} \times \{0,1\}^{\ell+b} \to \{0,1\}^{\ell}$ *be a fixed input-length MAC. Then the nested, iterated function family* $\mathrm{NI}[f]\colon \{0,1\}^{2\kappa} \times \{0,1\}^{\leq L} \to \{0,1\}^{\ell}$ *is*

a variable input-length MAC with

$$\mathbf{InSec}^{\mathrm{mac}}_{\mathrm{NI}[f]}(t, q, \mu) \leq \left(1 + \frac{\mu}{b}\right)^2 \cdot \mathbf{InSec}^{\mathrm{mac}}_{f}(t', q', \mu')$$

where $t' = t + O(\mu')$, $q' = \mu/b$, *and* $\mu' = (b + \ell) \cdot \mu/b$.

Tightness of the bound. There is an appreciable loss in security above, with the insecurity of the nested iterated construct being greater than that of the original f by a factor of (roughly) the square of the number μ/b of messages in a chosen-message attack on f. This loss in security is however unavoidable. Iterated constructs of this nature continue to be subject to the birthday attacks illustrated by Preneel and Van Oorschott [20], and these attacks can be used to show that the above bound is essentially tight.

Proof approach. A seemingly natural approach to proving Theorem 1 would be to try to imitate the analyses of Merkle and Damgård [16,11] which showed that transforms very similar to ours preserve collision-resistance. This approach however turned out to be less straightforward to implement here than one might imagine, due to our having to deal with forgeries rather than collisions. Accordingly we take a different approach. We first reduce the question of forgeries to one about a certain kind of collision-resistance, namely "weak-collision resistance", showing that the insecurity of our construct as a MAC can be bounded in terms of its weak collision-resistance and the insecurity of the original f as a MAC. We can bound the weak collision-resistance of the iterated construct in terms of the weak collision-resistance of the original f using the approach of [16,11], and finally bound the weak-collision resistance of f in terms of its insecurity as a MAC. Putting the three together yields the theorem.

Underlying many of these steps are general lemmas, and we state them in their generality since they might be of independent interest. In particular, we highlight the connections between weak collision-resistance and MACs. We need to begin, however, by saying what is weak collision-resistance.

Weak Collision-resistance. In the usual attack model for finding collisions, the adversary is able to compute the hash function for which it seeks collisions; either it is a single, public function, or, if a family F, the key k (defining the map F_k for which the adversary seeks collisions) is given to the adversary. In the weak collision-resistance setting as defined in [3], the adversary seeking to find collisions for F_k is not given k, but rather has oracle access to F_k. Weak collision-resistance is thus a less stringent requirement than standard collision-resistance.

Let $F\colon \{0,1\}^\kappa \times \mathit{Dom}(F) \to \mathit{Rng}(F)$ be a family of functions. To formally define its weak collision-resistance we consider the following experiment. Here A is an adversary that gets an oracle for F_k and returns a pair of points m, m' in $\mathit{Dom}(F)$. It wins if these points are a (non-trivial) collision for F_k.

Experiment FindWeakCol(A, F)
$k \leftarrow \mathit{Keys}(F)$; $(m, m') \leftarrow A^{F_k(\cdot)}$
If $m \neq m'$ and $F_k(m) = F_k(m')$ then return 1 else return 0

We denote by $\mathbf{Succ}^{\mathrm{wcr}}_F(A)$ the probability that the above experiment returns 1. We then define

$$\mathbf{InSec}^{\mathrm{wcr}}_F(t,q,\mu) \stackrel{\mathrm{def}}{=} \max_A \{\, \mathbf{Succ}^{\mathrm{wcr}}_F(A) \,\} \ .$$

As before the maximum is taken over all adversaries A with "running time" t, "number of queries" q, and "total message length" μ, the quantities in quotes being measured with respect to the experiment $\mathrm{FindWeakCol}(A,F)$, analogously to the way they were measured in the definition of $\mathbf{InSec}^{\mathrm{mac}}$ described in Section 2. Specifically the running time is the actual execution time of $\mathrm{FindWeakCol}(A,F)$ plus the size of the code of A. We let $Q = Q_A \cup \{m, m'\}$ where Q_A is the set of all queries made by A. Then $q = |Q|$ and μ is the sum of the lengths of all messages in Q.

Reduction to WCR. We bound the insecurity of the nested construct as a MAC in terms of its weak-collision resistance and MAC insecurity of the original function. The following generalizes and restates a theorem on NMAC from [3]. In our setting h will be $\mathrm{IT}[f]$, and then N becomes $\mathrm{NI}[f]$.

Lemma 1. *Let $f\colon \{0,1\}^\kappa \times \{0,1\}^{\ell+b} \to \{0,1\}^\ell$ be a fixed input-length MAC, and let $h\colon \{0,1\}^\kappa \times D \to \{0,1\}^{\ell+b}$ be a weak collision-resistant function family on some domain D. Define $N\colon \{0,1\}^{2\kappa} \times D \to \{0,1\}^\ell$ via*

$$N(k_1k_2, x) \ = \ f(k_2, h(k_1, x))$$

for any keys $k_1, k_2 \in \{0,1\}^\kappa$ and any $x \in D$. Then N is a MAC with

$$\mathbf{InSec}^{\mathrm{mac}}_N(t,q,\mu) \le \mathbf{InSec}^{\mathrm{mac}}_f(t,q,q(b+\ell)) + \mathbf{InSec}^{\mathrm{wcr}}_h(t,q,\mu)$$

for all t, q, μ.

The proof of the above is an adaptation of the proof in [3] which is omitted here, but for completeness is provided in [1].

To prove Theorem 1 we will apply the above lemma with $h = \mathrm{IT}[f]$. Accordingly our task now reduces to bounding the weak collision-resistance insecurity of $\mathrm{IT}[f]$. But remember that we want this bound to be in terms of the insecurity of f as a MAC. We thus obtain the bound in two steps. We first bound the weak collision-resistance of $\mathrm{IT}[f]$ in terms of the weak collision-resistance of f, and then bound the latter via its insecurity as a MAC.

Weak Collision-Resistance of $\mathrm{IT}[f]$. We now show that if f is a weak collision-resistant function family, then the iterated construction $\mathrm{IT}[f]$ is also a weak collision-resistant function family.

Lemma 2. *Let $f\colon \{0,1\}^\kappa \times \{0,1\}^{\ell+b} \to \{0,1\}^\ell$ be a weak collision-resistant function family. Then,*

$$\mathbf{InSec}^{\mathrm{wcr}}_{\mathrm{IT}[f]}(t,q,\mu) \le \mathbf{InSec}^{\mathrm{wcr}}_f(t, \frac{\mu}{b}, (b+\ell)\frac{\mu}{b})$$

The proof is analogous to those in [16,11] which analyze similar constructs with regard to (standard, not weak) collision-resistance. To extend them one must first observe that their reductions make only black-box use of the underlying

function instances and can thus be implemented in the weak collision-resistance setting via the oracle for the function instance. Second, our way of handling the length variability, although different, can be shown to work. The proof is omitted here, but for completeness is provided in [1].

Given the above two lemmas our task has reduced to bounding the weak collision-resistance insecurity of f in terms of its MAC insecurity. The connection is actually much more general.

Weak Collision-Resistance of any MAC. We show that any secure MAC is weakly collision-resistant, although there is a loss in security in relating the two properties. This is actually the main lemma in our proof, and may be of general interest.

Lemma 3. *Let $g\colon \{0,1\}^\kappa \times Dom(g) \to \{0,1\}^\ell$ be a family of functions. Then,*

$$\mathbf{InSec}^{\mathrm{wcr}}_{g}(t,q,\mu) \leq q^2 \cdot \mathbf{InSec}^{\mathrm{mac}}_{g}(t+O(\mu),q,\mu)$$

The proof of the above is given in Appendix A.

Proof of Theorem 1. We now use the three lemmas above to complete the proof of Theorem 1. Letting $\epsilon = \mathbf{InSec}^{\mathrm{mac}}_{f}(t,q,q(b+\ell))$ for conciseness, we have:

$$\begin{aligned}
&\mathbf{InSec}^{\mathrm{mac}}_{\mathrm{NI}[f]}(t,q,\mu) \\
&\leq \epsilon + \mathbf{InSec}^{\mathrm{wcr}}_{\mathrm{IT}[f]}(t,q,\mu) && (1)\\
&\leq \epsilon + \mathbf{InSec}^{\mathrm{wcr}}_{f}(t,\frac{\mu}{b},(b+\ell)\frac{\mu}{b}) && (2)\\
&\leq \epsilon + \left(\frac{\mu}{b}\right)^2 \mathbf{InSec}^{\mathrm{mac}}_{f}(t+O((b+\ell)\frac{\mu}{b}),\frac{\mu}{b},(b+\ell)\frac{\mu}{b}) && (3)\\
&= \left(1+\frac{\mu}{b}\right)^2 \cdot \mathbf{InSec}^{\mathrm{mac}}_{f}(t',q',\mu')\ , && (4)
\end{aligned}$$

where $t' = t + O(\mu')$, $q' = \mu/b$ and $\mu' = (b+\ell)\cdot\mu/b$. In Equation 1 we used Lemma 1. In Equation 2 we used Lemma 2. In Equation 3, Lemma 3 is used, and the two terms of $\mathbf{InSec}^{\mathrm{mac}}_{f}$ are added in Equation 4 with the larger of the two resource parameters taken as the final resource parameters to obtain the conclusion of the theorem.

5 Feistel Does not Preserve Unforgeability

Let $f\colon \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$ be a family of functions. For any fixed integer $r > 0$, we define the r-round Feistel transform. It is a family of functions $\mathrm{FST}^r[f]\colon \{0,1\}^{r\kappa} \times \{0,1\}^{2l} \to \{0,1\}^{2l}$. Given keys $k_1,\ldots,k_r$ and input LR where $|L| = |R| = l$, we define

Algorithm $\mathrm{FST}^r[f](k_1\ldots k_r, LR)$
$L_0 \leftarrow L$; $R_0 \leftarrow R$
For $i = 1,\ldots,r$ do $Z_{i-1} \leftarrow f_{k_i}(R_{i-1})$; $R_i \leftarrow L_{i-1} \oplus Z_{i-1}$; $L_i \leftarrow R_{i-1}$
Return $L_r R_r$

Here $L_i R_i$ is the $2l$-bit block at the end of the i-th round. The Feistel transform has been used extensively to extend (double) the input size of a given pseudorandom function. Luby and Rackoff have shown that $\mathrm{FST}^3[f]$ is a pseudorandom permutation if f is a pseudorandom function [15]. Here we examine the possibility of $\mathrm{FST}^r[f]$ being a secure MAC under the assumption that f is only a secure MAC.

Luby and Rackoff showed that $\mathrm{FST}^2[f]$ is not pseudorandom even if f is pseudorandom [15]. It is easy to design an attack showing that $\mathrm{FST}^2[f]$ is not a secure MAC even if f is a secure MAC. (This is provided in [1] for completeness.) Here we go on to the more interesting case of three rounds, where the transform is known to be pseudorandomness preserving.

We show however that the three round Feistel transform does not preserve unforgeability. Namely the assumption that f is a secure MAC does not suffice to guarantee that $\mathrm{FST}^3[f]$ is a secure MAC. We prove our claim by presenting an attack against $\mathrm{FST}^3[f]$ when f is the MAC of Section 3 for which we had presented an attack against CBC[f]. Recall that $f\colon \{0,1\}^\kappa \times \{0,1\}^l \to \{0,1\}^l$ was designed in terms of an underlying secure but arbitrary MAC $g\colon \{0,1\}^{2m} \to \{0,1\}^m$, and we set $l = 2m$. Let us now see what happens when we evaluate $\mathrm{FST}^3[f](k_1 k_2 k_3, L_0 R_0)$. We write $L_0 = a_0 \| a_1$ and $R_0 = b_0 \| b_1$ where $|a_0| = |a_1| = |b_0| = |b_1| = m = l/2$ bits, and work through the three Feistel rounds, writing the intermediate results in terms of the notation used in describing the Feistel algorithm above:

$$
\begin{array}{rl|rl}
L_0 & R_0 = & a_0 \| a_1 & b_0 \| b_1 \\
 & Z_0 = & & \mu \| b_0 \\
L_1 & R_1 = & b_0 \| b_1 & a_0 \oplus \mu \| a_1 \oplus b_0 \\
 & Z_1 = & & \mu' \| a_0 \oplus \mu \\
L_2 & R_2 = & a_0 \oplus \mu \| a_1 \oplus b_0 & b_0 \oplus \mu' \| b_1 \oplus a_0 \oplus \mu \\
 & Z_2 = & & \mu'' \| b_0 \oplus \mu' \\
L_3 & R_3 = & b_0 \oplus \mu' \| b_1 \oplus a_0 \oplus \mu & a_0 \oplus \mu \oplus \mu'' \| a_1 \oplus \mu'
\end{array}
$$

Here we have set

$$
\begin{aligned}
\mu &= g(k_1, b_0 b_1) \\
\mu' &= g(k_2, a_0 \oplus \mu \| a_1 \oplus b_0) \\
\mu'' &= g(k_3, b_0 \oplus \mu' \| b_1 \oplus a_0 \oplus \mu)\ .
\end{aligned}
$$

Write $L_3 = a_3 \| a_3'$ and $R_3 = b_3 \| b_3'$. We notice that given the output $L_3 R_3$ and the input $L_0 R_0$, it is possible to extract the values μ, μ', μ'', even without knowledge of any of the keys. Namely $\mu = b_1 \oplus a_0 \oplus a_3'$ and $\mu' = a_3 \oplus b_0$ and $\mu'' = a_0 \oplus \mu \oplus b_3$. Furthermore notice that once an attacker has these values, it can also compute Z_0, Z_1, Z_2, the internal Feistel values. Based on this we will present an attack against against $\mathrm{FST}^3[f]$.

Claim. There is a forger A making four $2l$-bit queries to $\mathrm{FST}^3[f](k,\cdot)$ and achieving $\mathbf{Succ}^{\mathrm{mac}}_{\mathrm{FST}^3[f]}(A) = 1 - O(2^{-l})$.

Proof. The attacker A is given an oracle for $\mathrm{FST}^3[f](k,\cdot)$, where $k = k_1k_2k_3 \in \{0,1\}^{3\kappa}$ is the key. It makes the four queries displayed, respectively, as the first rows of the first four columns in Figure 2. The first two queries generated by A are random. A then generates the next two queries adaptively, using the results of the previous queries. Notice that the third and fourth queries are functions of Z-values occurring in the 3-round Feistel computation on the first two queries. The attacker A can obtain these values using the observation above. Finally, A comes up with the forgery (x,τ), where x and τ are displayed, respectively, as the first and last rows in the fifth column of the same Figure.

query 1	*query 2*	*query 3*	*query 4*	*forgery*
$L_0^1 \mid R_0^1$	$L_0^2 \mid R_0^2$	$L_0^3 \mid R_0^2 \oplus Z_1^1 \oplus Z_1^2$	$R_1^2 \oplus Z_0^3 \mid R_0^2 \oplus Z_1^1 \oplus Z_1^2$	$R_1^1 \oplus Z_0^2 \mid R_0^2$
$\mid Z_0^1$	$\mid Z_0^2$	$\mid Z_0^3$	$\mid Z_0^3$	$\mid Z_0^2$
$R_0^1 \mid R_1^1$	$R_0^2 \mid R_1^2$	$\mid$	$R_0^2 \oplus Z_1^1 \oplus Z_1^2 \mid R_1^2$	$R_0^2 \mid R_1^1$
$\mid Z_1^1$	$\mid Z_1^2$	$\mid$	$\mid Z_1^2$	$\mid Z_1^1$
$R_1^1 \mid R_2^1$	$R_1^2 \mid R_2^2$	$\mid$	$R_1^2 \mid R_0^2 \oplus Z_1^1$	$R_1^1 \mid R_0^2 \oplus Z_1^1$
$\mid Z_2^1$	$\mid Z_2^2$	$\mid$	$\mid Z_2^3$	$\mid Z_2^3$
$R_2^1 \mid R_3^1$	$R_2^2 \mid R_3^2$	$\mid$	$\mid$	$R_0^2 \oplus Z_1^1 \mid R_1^1 \oplus Z_2^3$

Fig. 2. Contents of the queries and the intermediate/final results and the forgery.

Notice that in queries 3 and 4 in Figure 2, the rows after certain values (Z_0^3, Z_2^3) are empty. They are omitted for simplicity because only those two values (Z_0^3, Z_2^3) are needed to form the next queries or the forgery, and the rest of the values are not needed. In the actual attack, those values are computed from the outputs of the oracle.

To show that this is a successful attack, we need to check two things. First that the forgery is valid, meaning $\mathrm{FST}^3[f](k,x) = \tau$, and second that the message x is new, meaning $x \notin \{x_1 \ldots x_4\}$.

We can easily see that the forgery is valid by examining the values in the table. The second requirement that x is new can be achieved with high probability if the adversary chooses the strings L_0^1, L_0^2, and L_0^3 randomly. If the said strings are chosen randomly, then the l-bit left-half of each queried string becomes random and the probability of the forgery string x matching any one of the four queried strings is very small $(O(2^{-l}))$. This means that the probability of the forgery being new is $1 - O(2^{-l})$ as shown in the claim. Hence, the above attack succeeds with high probability $(1 - O(2^{-l}))$ as desired. □

Acknowledgments

Thanks to the Crypto 99 program committee for their comments.

The first author was supported in part by a NSF Graduate Fellowship. The second author was supported in part by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

References

1. J. An and M. Bellare, "Constructing VIL-MACs from FIL-MACs: Message authentication under weakend assumptions," Full version of this paper, available via `http://www-cse.ucsd.edu/users/mihir`.
2. ANSI X9.9, "American National Standard for Financial Institution Message Authentication (Wholesale)," American Bankers Association, 1981. Revised 1986.
3. M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
4. M. Bellare, R. Canetti and H. Krawczyk, "Pseudorandom functions revisited: the cascade construction and its concrete security," Proceedings *of the* 37th *Symposium on Foundations of Computer Science*, IEEE, 1996.
5. M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation," Proceedings *of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.
6. M. Bellare, J. Kilian and P. Rogaway, "The security of cipher block chaining," *Advances in Cryptology – Crypto* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.
7. M. Bellare and P. Rogaway, "Collision-Resistant Hashing: Towards Making UOWHFs Practical," *Advances in Cryptology – Crypto* 97 *Proceedings*, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997.
8. M. Bellare, O. Goldreich and S. Goldwasser, "Incremental cryptography with application to virus protection," Proc. 27th Annual Symposium on the Theory of Computing, ACM, 1995.
9. M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," *Advances in Cryptology – Crypto* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
10. L. Carter and M. Wegman, "Universal Hash Functions," *Journal of Computer and System Science*, Vol. 18, 1979, pp. 143–154.
11. I. Damgård, "A Design Principle for Hash Functions," *Advances in Cryptology – Crypto* 89 *Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
12. O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions,"*Journal of the ACM,* Vol. 33, No. 4, 210–217, (1986).
13. S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Science*, Vol. 28, 1984, pp. 270–299.
14. 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, pp. 281–308, April 1988.

15. M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," *SIAM Journal of Computing*, Vol. 17, No. 2, pp. 373–386, April 1988.
16. R. Merkle, "One way hash functions and DES," *Advances in Cryptology – Crypto* 89 *Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
17. R. Merkle, "A certified digital signature," *Advances in Cryptology – Crypto* 89 *Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
18. M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," Proceedings *of the* 21st *Annual Symposium on Theory of Computing*, ACM, 1989.
19. E. Petrank and C. Rackoff, CBC MAC for real time data sources. *DIMACS Technical Report* 97-26, 1997.
20. B. Preneel and P. van Oorschot, "MD-x MAC and building fast MACs from hash functions," *Advances in Cryptology – Crypto* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
21. R. Rivest, "The MD5 message-digest algorithm," IETF RFC 1321 (April 1992).
22. FIPS 180-1. Secure Hash Standard. Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., April 1995.
23. Wegman and Carter, "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, Vol. 22, 1981, pp. 265–279.

A Proof of Lemma 3

Let C be an arbitrary collision-finder attacking the security of the function g as a weak collision-resistant function and having resources at most t, q and μ as previously defined. We want to upper bound the weak-collision insecurity of g in terms of insecurity of g as a MAC. To do this, we specify an adversary (forger) A that uses the collision-finder C to attack the security of the function family g as a MAC.

Our algorithm for A is composed of two sub-algorithms: A_1 and A_2. They get an oracle for $g_k(\cdot)$ so that they can mount a chosen-message attack and eventually output a pair (m, τ). They run C providing answers to C's queries using the oracle they are given. Each of them is allowed to have resources at most t',q' and μ' as defined before. We denote by q_c the number of queries made by C. The algorithms are specified as follows:

Algorithm $A_1^{g_k(\cdot)}$
  For $i = 1, \ldots, q_c$ do
    $C \rightarrow x_i$
    $C \leftarrow g_k(x_i)$
  $C \rightarrow (y_1, y_2)$
  Choose $m \stackrel{R}{\leftarrow} \{1, 2\}$
  Let $n = \{1, 2\} - \{m\}$
  Return $(y_m, g_k(y_n))$

Algorithm $A_2^{g_k(\cdot)}$
  Choose $i \stackrel{R}{\leftarrow} \{1, \ldots, q_c\}$; $j \stackrel{R}{\leftarrow} \{1, \ldots, i-1\}$
  For $s = 1, \ldots, i$ do
    $C \rightarrow x_s$
    If $s < i$
      $C \leftarrow g_k(x_s)$
  Return $(x_i, g_k(x_j))$

To handle the case where both of the final strings y_1 and y_2 that C outputs were queried by the collision-finder C, we have the forger A_2. For the other case, where at least one of the strings that C outputs was not queried by C, we have the forger A_1. A_2 makes two random guesses $(i, j,\ 1 \leq j < i \leq q_c)$ for the positions in which the queries for the output strings y_1, y_2 are made by C among its query positions $\{1, \ldots, q_c\}$. Without making the query for the guessed second position i, A_2 outputs the forgery using the unqueried string x_i as the new message and the output of the queried string $g_k(x_j)$ as its tag. A_1 randomly chooses one of the output strings of C and outputs that string as the message and uses the other string to obtain its tag.

We now upper bound $\mathbf{Succ}_g^{\mathrm{wcr}}(C)$ in terms of $\mathbf{Succ}_g^{\mathrm{mac}}(A_1)$ and $\mathbf{Succ}_g^{\mathrm{mac}}(A_2)$. For notational convenience, we define the following. Let "C Succeeds" denote the event where the experiment FindWeakCol(C, g) outputs 1. Let $\Pr[\cdot]$ denote the probability of some event occurring in the experiment FindWeakCol(C, g). And let E denote the event where, in the two output strings of C, at least one of them is unqueried.

Forge(A_1, g) will output 1 when FindWeakCol(C, g) outputs 1 (the event "C Succeeds") *and* at least one of the two output strings of C is unqueried (the event E) *and* A_1 chooses the unqueried string correctly. Since A_1 chooses the string randomly (out of the two strings), the probability of choosing right is at least $1/2$. This means that $\mathbf{Succ}_g^{\mathrm{mac}}(A_1) \geq \frac{1}{2}\Pr[C\ \mathit{Succeeds} \wedge E]$. For the algorithm A_2, its success requires that it guesses the correct positions for the two queried strings in addition to the success of C with both of its output strings queried. Since the two positions are randomly chosen among q_c numbers, the probability of choosing the correct position pair among the $\binom{q_c}{2}$ possible position pairs is at least $1/\binom{q_c}{2}$. Notice that the event where both of the output strings of C are queried is $\bar{E}$ (from the definition of the event E). Hence, $\mathbf{Succ}_g^{\mathrm{mac}}(A_2) \geq \Pr[C\ \mathit{Succeeds} \wedge \bar{E}]/\binom{q_c}{2}$. Putting all this together we have

$$\begin{aligned}
\mathbf{Succ}_g^{\mathrm{wcr}}(C) &= \Pr[C\ \mathit{Succeeds}] \\
&= \Pr[C\ \mathit{Succeeds} \wedge E] + \Pr[C\ \mathit{Succeeds} \wedge \bar{E}] \\
&\leq 2\mathbf{Succ}_g^{\mathrm{mac}}(A_1) + \binom{q_c}{2} \cdot \mathbf{Succ}_g^{\mathrm{mac}}(A_2) \\
&\leq \frac{q_c^2 - q_c + 4}{2} \cdot \mathbf{InSec}_g^{\mathrm{mac}}(t', q', \mu')\ . \qquad (5)
\end{aligned}$$

The analysis of the resource parameters is omitted here, but for completeness is provided in [1].

Regarding Equation 5, the multiplicative factor $\frac{q_c^2 - q_c + 2}{2}$ is less than q^2 since we know that $q_c \leq q$. Combining all this, we obtain the conclusion of Lemma 3.