L-collision Attacks against Randomized MACs

Michael Semanko

Department of Computer Science & Engineering,
University of California at San Diego,
9500 Gilman Drive,
La Jolla, California 92093, USA.
msemanko@cs.ucsd.edu

Abstract. In order to avoid birthday attacks on message authentication schemes, it has been suggested that one add randomness to the scheme. One must be careful about how randomness is added, however. This paper shows that prefixing randomness to a message before running the message through an iterated MAC leads to an attack that takes only $O\left(2^{(l+r)/3} + \max\{2^{l/2}, 2^{r/2}\}\right)$ queries to break, where l is the size of the MAC iteration output and r is the size of the prefixed randomness.

Keywords: MACs, message authentication codes, randomness, L-collision, birthday attacks

1 Introduction

1.1 Problem

A message authentication scheme allows people to ensure that messages can travel between them without being altered. These schemes are basic cryptographic primitives, and thus they are widely used. Message authentication schemes consist of three major algorithms: one for computing a message's authentication tag, one for checking that a message's authentication tag is valid, and one for picking a key. By tagging a message before it is transmitted, one can protect the message from being maliciously altered. One popular form of message authentication is the iterated message authentication scheme.

Iterated message authentication schemes are those message authentication schemes which break up the message into smaller blocks and then make repeated use of a small keyed function on these blocks. The keyed function takes as input a block of the message and the previous result of the function to yield the next result. This is repeated for each block of the message, and the final result of the function is then output. CBC-MAC [1], HMAC [2], NMAC [2], and EMAC [7] are all examples of iterated MACs. All of these iterated MACs suffer from the same security flaw: internal collision attacks [8], referred to more generally as birthday attacks. Internal collision attacks can break any deterministic, stateless, iterated MAC in $O\left(2^{l/2}\right)$ queries, where l is the size of the internal chaining value.

M. Bellare (Ed.): CRYPTO 2000, LNCS 1880, pp. 216–228, 2000.

1.2 Possible Solutions

In an attempt to make more efficient MACs, people have come up with methods to try to avoid birthday attacks. There are two obvious possibilities: either make the MAC function stateful or make it probabilistic. We briefly discuss each.

Stateful Iterated MACs. Stateful iterated MACs are iterated MACs which maintain s bits of state between queries, and access this state in the tagging algorithm. The state is then changed so that a different state is always presented as the input to the transformation. The authentication tag then is the l-bit output of the MAC prepended to the s bits of state. These iterated MACs can actually completely avoid the internal collision attack, as shown in [4]. The counter-based XOR-MAC, XMACC, requires approximately $O\left(2^l\right)$ authenticated messages to create a forgery, for instance [4]. XMACC achieves this by prepending a counter to messages before they are run through the iteration.

Stateful MACs have problems of their own which make them less used. It is often not convenient to maintain those s bits of state between authentications; for example, you cannot use a stateful iterated MAC if you want multiple senders to share an authentication key. All of the systems that should be able to authenticate would have to share the same s bits of state in order to maintain the security of the signatures. This is very difficult to do if there are many machines or if those machines may not always be connected.

Randomized Iterated MACs. Using randomized iterated MACs is another approach that could be taken. A randomized iterated MAC is a iterated MAC which has a probabilistic tagging algorithm. The authentication tag in this case is the l-bit output of the MAC prepended to the r bits of randomness.

A randomized scheme seems like it would be the answer to all of these problems. It does not need to maintain state between authentications. It can be used on multiple machines with the same key. What is missing? The problem with randomized schemes is that no one is really sure how secure randomized schemes are. Security proofs of randomized MAC schemes become difficult if one tries to prove security beyond the birthday bound. Designing a randomized authentication scheme that provably avoids birthday attacks was the topic of [3]. This paper succeeded only by tripling to quadrupling the size of the authentication tag. No one yet has designed a randomized MAC scheme with a proof of security that beats the birthday bound without a significant increase in authentication tag size or computation time.

Two approaches which are often suggested are to either prepend or append a random message block to the message before the keyed function iterations are performed. These approaches are promising because they only add a small amount of computation time and do not require a lot of randomness. There have not yet been results showing the security of either of these schemes.

1.3 Background and Related Work

Bellare, Killian, and Rogaway's paper [5], provides a formal analysis of the security of CBC-MAC. This paper is a good starting point from which the effects of

birthday attacks can be seen on message authentication schemes in use. Later, Preneel and van Oorschot described the internal collision attack on MACs in [8]. This attack works upon all deterministic iterated MACs. They propose using keyed hash functions with large outputs to mitigate the effects of this attack. Although this works, it leads to MACs which may be less efficient than possible.

In Bellare, Guerin, and Rogaway's paper [4], the XOR-MAC schemes were introduced. One of these schemes, XMACC, was the first MAC which provably avoided the birthday attack problem. XMACC is a stateful MAC, however, and thus has all of the problems associated with a stateful MAC scheme. For a while, the only way to not be affected by birthday attacks was to use a stateful counter-based scheme.

More recently, Bellare, Goldreich, and Krawczyk provided a randomized, stateless scheme in [3] which is secure beyond the birthday bound in terms of the size of the hashed message; however, is not nearly as secure in terms of the size of the entire authentication tag because they had to transmit multiple random numbers in the tag.

Part of the problem with the term "birthday bound" lays in the fact that there are many parameters to a MAC, and even more to a randomized iterated MAC. An increase in any of these parameters can generally lead to an increase in the security of the MAC. In the best case, we would like to be able to increase the security of MACs, without a significant increase in the key size, computation time, authentication tag size, blocksize, or randomness.

1.4 Results

This paper shows a new form of birthday attack, called the L-collision attack. This attack applies to schemes that prepend random data before MACing. The L-collision attack allows an adversary to create a forgery with a constant probability in only $O\left(2^{(l+r)/3} + \max\{2^{l/2}, 2^{r/2}\}\right)$ queries where l is the size of the internal chaining value of the randomized MAC, and r is the size of the randomness.

Section 2 provides basic definitions that allow us to examine iterated message authentication schemes formally. Section 3 describes how the internal collision attack works, as this is the basis for our new attack. Section 4 presents a variation upon MAC schemes, which is based upon the idea of prepending a random block. This scheme is often secure against birthday attacks, but it falls to our new attack.

2 Definitions

This section presents the basic definitions of message authentication schemes, their security, and iterated message authentication schemes.

2.1 Function Families

Function families are vital components to MACs, and thus we wish to treat them formally. We do so in the format of [5].

Definition 1. *Let* h: $\{0,1\}^k \times \{0,1\}^i \to \{0,1\}^j$ *be a map. We say that* h *is a function family. We define* $\{0,1\}^k$ *as the keyspace of* h, $\{0,1\}^i$ *as the domain of* h, *and* $\{0,1\}^j$ *as the range of* h. *Then for each particular key* K, *we define a map* h_K: $\{0,1\}^i \to \{0,1\}^j$ *such that* $\forall u \in \{0,1\}^i, h_k(u) = h(k,u)$. *We say that* h_K *is a particular instance of* h.

Function families can be seen in such primitives as keyed-hash functions and block ciphers. Often, when proving security, we do not want to consider the effects of a particular block cipher or other keyed function on the security of a scheme. Thus, we replace the function with a function drawn at random from the set of all functions.

2.2 Message Authentication Schemes

We first define the syntax of message authentication schemes so that we may examine their security. This definition is a combination of the schemes used in [5] and [4].

Definition 2. *A message authentication scheme* $\mathcal{MA} = (\mathsf{Tag}, \mathsf{Vf}, \mathsf{Key})$ *consists of three algorithms as follows:*

- *A randomized key generation algorithm,* Key, *which returns a key from the set* $\{0,1\}^k$.
- *A tagging algorithm,* Tag *which can be either randomized or stateful. It takes a key* K *and a message* M, *and returns a tag* σ *from* $\{0,1\}^t$.
- *A deterministic verification algorithm,* Vf, *which takes a key* K, *a message* M, *and a tag* σ *to return a bit* v. *We say that* σ *is a valid tag for a message* M *under a key* K *if* $\mathsf{Vf}(M, \sigma, K) = 1$.

We require that $\mathsf{Vf}(M, \mathsf{Tag}(M,K), K) = 1$ *for all* $M \in \{0,1\}^*$. *The scheme is said to be deterministic if the tagging algorithm is deterministic.*

The security of a message authentication scheme is based upon the probability that an adversary without the secret key can create a forgery, which is a correct message/tag pair such that the verification procedure considers the pair valid.

2.3 Iterated Message Authentication Schemes

There are no interesting known general attacks on message authentication schemes as they are given above. We need to define a more specific message authentication scheme in order to be able to talk about actual attacks. For this reason, we now define iterated message authentication schemes. Figure 1 shows the general layout of an iterated MAC tagging algorithm, and below we present a definition based upon [8].

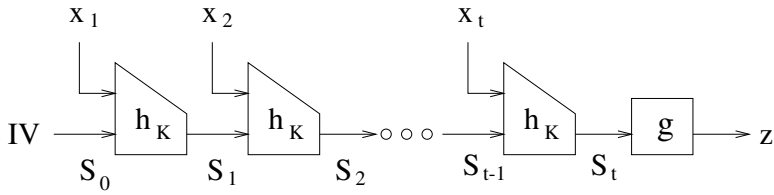

Fig. 1. A general scheme for iterated MACs

Definition 3. *Let* h: $\{0,1\}^k \times \{0,1\}^b \times \{0,1\}^l \to \{0,1\}^l$ *be a function family with domain* $\{0,1\}^b \times \{0,1\}^l$*, and let* g: $\{0,1\}^l \to \{0,1\}^o$ *be an output transformation. We associate to* h *and* g *the following iterative message authentication scheme* $\mathcal{IMA} = (\mathsf{Tag}_h, \mathsf{Vf}_h, \mathsf{Key})$*:*

- Tag_h *first divides a message* M *to be tagged into* b*-bit blocks, labeled* $x_1, \ldots, x_t$*, where* $M = x_1 || \ldots || x_t$ *(where* $||$ *denotes concatenation). We call* k *the keysize of the MAC,* b *the blocksize of the iterated MAC,* l *the chaining variable size, and* o *the output size of the MAC.*
- Tag_h *makes use of the keyed function* h *iteratively to retrieve a value according to the algorithm* $h^*_K(x_1 || \ldots || x_t)$*, which we define below.*

function $h^*_K(x_1 || \ldots || x_t)$
  $S_0 \leftarrow IV$
  for $i = 1, \ldots, t$ ***do***
    $S_i \leftarrow h_K(x_i, S_{i-1})$
  endfor
  return S_t

Note that this definition of iterated MACs includes the well-known CBC-MAC by making $h_k(x_i, S_{i-1}) = f_k(x_i \oplus S_{i-1})$, where f_k is the block cipher to be used in CBC-MAC. For this paper, we shall be set the output transformation g to be the identity function. Thus, the chaining variable size and the output size of the iterated MACs in this paper shall both be l. Changing the attack to deal with different output transformations can be done in the same way as in [8].

3 Birthday Attacks

If we wish to avoid birthday attacks with a message authentication scheme, we must be able to define what a birthday attack is. Usually, the type of birthday attacks that MAC designers attempt to avoid are known as internal collision attacks. The general idea behind an internal collision attack is that if we find two messages with the same n block suffix but different m block prefixes that lead to the same output, then we are likely to be able to change the suffix and still have the two messages have the same output. Such collisions may be found because iterated MACs have an internal state. A collision on the output means

that at some point, the internal state of the two MACs was probably the same. Given the same internal state, and the same suffixes, the MAC will be likely to output the same tags.

For completeness, we define the internal collision attack, as given in [8], more formally in the Appendix.

4 A New Attack

We first present a basic modification to the iterated MAC scheme, having a random message prefix. Many random prefix versions of MACs seem to be secure against internal collision attacks. We then present the L-collision attack, a new attack which breaks these schemes. Finally, we find the probability that the attack produces a valid forgery of an unqueried message for any number of queries.

4.1 Random Prefix Versions of Message Authentication Schemes

The iterated message authentication schemes that we examine in this paper are modified by adding a random prefix to the message to be tagged before the tagging algorithm is run. We denote a random prefix version of a scheme by prepending it with RP-. For example, CBC-MAC with a random prefix added to the message can be refered to as RP-CBC-MAC. We shall use the phrase RP-MAC to denote a general iterated MAC with a random prefix.

One thing to note about RP-MACs is that internal collision attacks do not seem to work. The reason for this is that although one may get two messages to collide on some output, one cannot get any further queries with one of the two random numbers used in the collision. These random numbers are prepended to the messages, and thus become part of the prefix. We are thus unable to generate a query to the third message in the internal collision attack because we cannot get the same prefix again.

4.2 The L-collision Attack

We now describe a new attack that can be used to defeat a random prefix scheme. This is not a length-based attack, and any MAC which would normally fall to an internal collision attack would fall to this new attack if random data were first prepended.

We first define the idea of a collision formally so that the similarities can be seen in the definition of a collision and an L-collision.

Definition 4. *Let A be a set, and let C be a list of elements from the set A. We say that the list C contains a collision if there exist at least two elements of the list C that are the same.*

The idea of a collision can be extended by thinking about what sort of collisions can occur when we take the list C from the Cartesian product of two sets. An

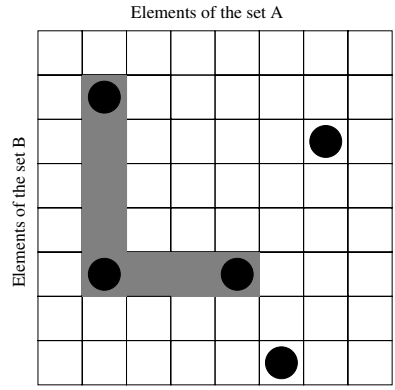

Fig. 2. An L-collision given 5 queries, where there are only 8 possible random values and 8 possible output values.

L-collision, intuitively, is a chain of two collisions in this Cartesian product. One collision is in the first set, and the second collision is in the second set. These two collisions contain a single common point. A picture of this can be seen in Figure 2. In the figure, the x-axis is indexed by the elements of the first set, the y-axis is indexed by the elements of the second set, and the black balls represent the elements of the list C. An L-collision is highlighted in the figure. The L-shape that is formed by the collision chain is the reason we refer to these chains as L-collisions. We now state the formal definition for an L-collision.

Definition 5. *Let A, B be two sets, and let C be a list of elements from the set $A \times B$. We say that the set C contains an L-collision over A and B if there exist distinct elements $(a, b), (a, y), (x, b) \in C$. We call these elements the collision points. We call (a, b) the pivot of the L-collision, (a, y) the A-collision end, and (x, b) the B-collision end.*

For the L-collision attack, we let one of the sets be the possible random values that could be prefixed to the message, and we let the other set be the possible output values. By getting a certain L-collision over these sets, we can forge a message.

The messages queried have two m block message prefixes, and two n block message suffixes. There are four different ways these prefixes and suffixes can be combined into complete messages. By generating enough queries on three of these combinations, we hope to forge the fourth combination.

Definition 6. *Let m, n be any positive integers. Then we define an L-collision attack as an attack performed by an adversary A that, given an oracle $g = \mathsf{Tag}(K, \cdot)$ attacks a MAC scheme as follows:*

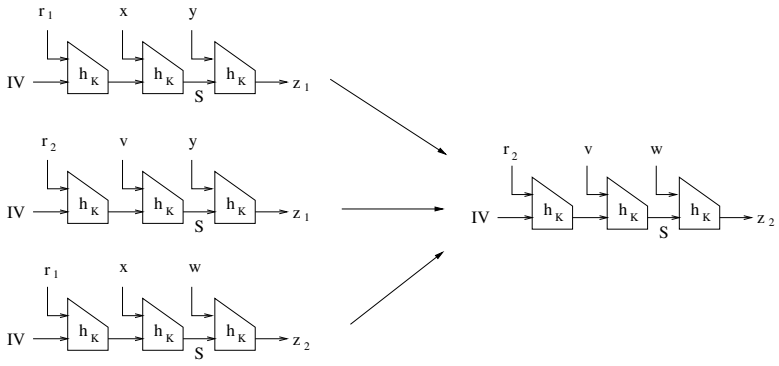

Fig. 3. An L-collision on the three messages M_1, M_2, and M_3 allows us to forge the fourth message M_4. Notice that the internal state S is the same before the third block in all of the messages. For clarity, $m = 1$ and $n = 1$ in this example.

function A^g
  $x \xleftarrow{R} \{0,1\}^{mb}$
  $v \xleftarrow{R} \{0,1\}^{mb} - \{x\}$
  $y \xleftarrow{R} \{0,1\}^{nb}$
  $w \xleftarrow{R} \{0,1\}^{nb} - \{y\}$
  $M_1 \leftarrow x \,||\, y$; $M_2 \leftarrow v \,||\, y$
  $M_3 \leftarrow x \,||\, w$; $M_4 \leftarrow v \,||\, w$
  for $i = 1$ ***to*** 3 ***do***
    for $j = 1$ ***to*** $\lfloor \frac{q}{3} \rfloor$ ***do***
      $(r_{ij}, z_{ij}) \leftarrow g(M_i)$
    endfor
  endfor
  if $\exists\, e, d, f$ *such that* $r_{1d} = r_{3e}$ *and* $z_{1d} = z_{2f}$, ***then***
    return $(M_4, (r_{2f}, z_{3e}))$

Figure 3 shows us why the L-collision attack works. Because the internal state S is the same in all of the messages at some point, we know that the outputs depend only upon the final block(s) which occur after this point. Thus, we know that the outputs of the fourth message and the second message are going to be the same. The theorem below specifies the parameters for the L-collision attack.

Theorem 1. *For any deterministic, non-stateful iterated message authentication scheme MAC, there is a constant $c > 0$ such that an adversary A using the above strategy requires only*

- q_s *tagging queries,*
- q_v *verification queries, and*
- $c(b + l)(q_s + q_v)$ *time*

to be able to forge a message for the random prefix version of MAC, RP-MAC, with probability $\epsilon \approx \left(\frac{q_s^3}{162 \cdot 2^{l+r}}\right)\left(1 - \frac{n}{n+1}\right)$*, where* n *is the number of blocks in the suffixes* y *and* w*, and* q_s *satisfies* $\max\{2^{l/2}, 2^{r/2}\} < q_s < 2^{(l+r)/3}$.

If we assume that the MAC is CBC-MAC or has a similar combine-then-permute structure, the above result is similar except that $\epsilon \approx \left(\frac{q_s^3}{162 \cdot 2^{l+r}}\right)$. We leave the proof of these theorems for later. In order to be able to complete this proof, we must first be able to find the probability that a randomly chosen set contains L-collisions over two sets.

4.3 L-collision Probabilities

We now can turn our attention away from the cryptographic problem for a while, to examine the probabilistic problem of the frequency of L-collisions. We can represent L-collisions in a rectangle, with rows representing the elements of set A and columns representing the elements of the set B.

Definition 7. *Suppose we pick* q *elements randomly and independently from the set* $A \times B$ *where* $|A| = M$ *and* $|B| = N$*. Then* $L(M, N, q)$ *denotes the probability of at least one L-collision over* A *and* B *within these elements.*

We now provide a lemma that gives us the approximate probability of finding such an L-collision.

Lemma 1. *Let* $L(M, N, q)$ *be defined as above. Then,*

$$L(M, N, q) \approx \frac{q^3}{6MN} \tag{1}$$

for all q *such that* $max\{\sqrt{M}, \sqrt{N}\} < q < (MN)^{\frac{1}{3}}$.

The results of this lemma can be seen intuitively through the following argument. The probability that any given triple, $(a, b), (c, d), (e, f)$ taken randomly from $A \times B$, is an L-collision is equal to the probability that $a = c$ times the probability that $b = f$, since these events are independent. Thus, a triple is an L-collision with probability of $\frac{1}{MN}$. If we chose three elements from the q random queries, we have $\frac{q^3}{6}$ ways of choosing the elements without regard to order. Putting this together, we expect to see an L-collision in the queries with probability $\frac{q^3}{6MN}$.

4.4 Analysis of L-collision Attack

Now that we know bounds on the probability that there is an L-collision, we can prove Theorem 1.

Proof (Theorem 1). We first show that the algorithm provides a valid forgery in the case where the **if** statement is true. So, suppose there exists an e, d, f such

that $r_{1d} = r_{3e}$ and $z_{1d} = z_{2f}$. Then we know that $(r_{1d}, z_{1d}) = \mathsf{Tag}(K, x \mid\mid y)$ and $(r_{2f}, z_{2f}) = \mathsf{Tag}(K, v \mid\mid y)$ collide on the iteration outputs since $z_{1d} = z_{2f}$. The message prefixes which are likely to have led to this collision are $r_{1d} \mid\mid x$ and $r_{2f} \mid\mid v$ (note that the randomness is prefixed to the message). Because the suffixes of both messages are the same, we have a birthday style collision here. According to [8], the collision occurs before the n block suffix y with probability $\left(1 - \frac{n}{n+1}\right)$ for a random function and with probability 1 for a CBC-style iterative function with the block cipher replaced by a random permutation. Once we have this collision, all we need is one query of $r_{1d} \mid\mid x$ with a different suffix, and we will be able to forge the message $r_{2f} \mid\mid v$ with that same suffix.

We get this additional query by the fact that we know $(r_{3e}, z_{3e}) = \mathsf{Tag}(K, x \mid\mid w)$. Since $r_{3e} = r_{1d}$, we have the same prefix, $r_{1d}||x$ with a different suffix, w. Because it is likely that the output will be the same, we replace the prefix of the third message with the colliding prefix of the second message. This gives us to $r_{2f}||v||w$, a valid forgery under the output z_{3e}. It is clear that $v \mid\mid w$ has never previously been queried, since the only queries were to $x \mid\mid y$, $x \mid\mid w$, and $v \mid\mid y$.

We now need a lower bound on the probability that the **if** statement is be true. Notice that the if statement corresponds to there being an L-collision over $\{0,1\}^r$ and $\{0,1\}^l$, where we set the pivot to be (r_{1d}, z_{1d}), we set the $\{0,1\}^r$-collision end to be (r_{3e}, z_{3e}), and we set the $\{0,1\}^l$-collision end to be (r_{2f}, z_{2f}).

Lemma 1 shows an approximation of the probability that an L-collision occurs over two sets, given that there are q queries made. We let one of these sets be $\{0,1\}^r$, and the other be $\{0,1\}^l$, and we set the number of collisions made to be q_s. Then, we get a lower bound on the probability of an L-collision in this case to be $L(2^r, 2^l, q_s) \approx \frac{q_s^3}{6 \cdot 2^{l+r}}$. Just the existence of an L-collision is not enough, however. We need the L-collision to be such that the points (r_{1d}, z_{1d}), (r_{3e}, z_{3e}), and (r_{2f}, z_{2f}) are the pivot, the $\{0,1\}^r$-collision end, and the $\{0,1\}^l$-collision end, respectively. The probability that this happens is $\frac{\lfloor \frac{1}{3} q_s \rfloor}{q_s} \cdot \frac{\lfloor \frac{1}{3} q_s \rfloor}{q_s - 1} \cdot \frac{\lfloor \frac{1}{3} q_s \rfloor}{q_s - 2} \approx \frac{1}{27}$. Combining the two above probabilities, we get the approximate probability that the **if** statement is true, $\frac{q_s^3}{27 \cdot 6 \cdot 2^{l+r}} = \frac{q_s^3}{162 \cdot 2^{l+r}}$.

The only thing left to do is show that we can find this collision in time proportional to q_s, and thus finding the collision takes no more time than computing the MAC in the first place. We can do this by keeping a hash table of outputs and randomness. Then we also have hash tables of output collisions and randomness collisions. Whenever we find that we have an output collision by the output hash table, we can add this collision to the output collision hash table. Similarly for randomness collisions. We then search through all of the M_2 query results. If any of them have elements in both the row collision and column collision hash table, we are done. This whole procedure takes linear time in q_s, concluding the proof. ■

Given Theorem 1, we can see that only $O\left(2^{(l+r)/3} + \max\{2^{l/2}, 2^{r/2}\}\right)$ queries are required to get a constant probability of the collision chain occurring. While

this attack requires more queries than normal birthday attacks, it can turn out to cause almost as many problems. We can illustrate this by looking at CBC-MAC with a 64-bit block cipher. Without the random bits prepended, we saw that we could get a forgery in about 2^{32} queries. After prepending 64 random bits, we now require about 2^{43} queries. This is almost as unacceptable, given the computing power of today's machines. Given the doubling in size of our message authentication tag, we would hope that the security would also be doubled. The L-collision attack shows that this is not the case.

5 Conclusion

Now that we know an attack against message authentication schemes with a random prefix, there is still the open question of whether there are any non-stateful schemes which avoid collision-based attacks on tagging algorithm queries. The problem with such a request is that there are many tradeoffs which can be made to achieve better security. Most schemes involve either greatly increased computation time or greatly increased key size. What we would like to see is a simple scheme which requires just one block cipher key, that has near ideal security.

This paper has shown that there is an inherent upper bound to the security that can be achieved by prepending randomness to messages in an iterated MAC scheme. Because this bound is less than ideal, it may be in the best interest of MAC cryptanalysts to examine other ways of combining randomness with the message.

This paper has also shown that the design of randomized MAC schemes is more difficult than first imagined. The birthday bound seems to be one of many bounds between current message authentication schemes and ideal schemes. Usually, when a MAC scheme is designed, one can immediately see how birthday attacks apply. L-collision attacks are much more difficult to see, and when they apply, they can be much trickier to avoid.

Acknowledgments

This paper would not have been possible without the guidance of my advisor, Mihir Bellare. I would like to give additional thanks to David Wagner for sheparding my paper, and for providing his comments on my paper drafts. His willingness to assist has taken a stressful load off of my shoulders.

The work done on this paper was supported in part by Mihir Bellare's 1996 Packard Foundation Fellowship in Science and Engineering and NSF CAREER Award CCR-9624439.

References

1. ANSI X9.9. American National Standard for Financial Institution Message Authentication (Wholesale), American Bankers Association, 1981. Revised 1986.

2. 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.
3. M. Bellare, O. Goldreich, and H. Krawczyk. Stateless Evaluation of Pseudorandom Functions: Security beyond the Birthday Barrier. *Advances in Cryptology – Crypto* 99 *Proceedings*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
4. M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseduorandom Functions. *Advances in Cryptology – Crypto* 95 *Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
5. M. Bellare, J. Killian, 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.
6. A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press. 1996.
7. E. Petrank and C. Rackoff. CBC-MAC for Real-Time Data Sources. Dimacs Technical Report, 97-26, 1997.
8. B. Preneel and P. van Oorschot. MDx-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.

6 Appendix

6.1 Internal Collision Attacks

We formally describe internal collision attacks below.

Definition 8. *Let $\mathcal{MA}$ be an iterated message authentication scheme with block-size b, and let m, n be any positive integers. Then the internal collision attack is performed by an adversary B, given an oracle $g = \mathsf{Tag}(K, \cdot)$, according the following algorithm.*

function B^g
  $y \xleftarrow{R} \{0,1\}^{nb}$; $w \xleftarrow{R} \{0,1\}^{nb} - \{y\}$
  for $i = 1$ ***to*** q ***do***
    $x_i \xleftarrow{R} \{0,1\}^{mb}$
    $M_i \leftarrow x_i \,||\, y$
    $z_i \leftarrow g(M_i)$
  endfor
  if $\exists\ d, e$ *such that* $z_d = z_e$, ***then***
    $M \leftarrow x_d \,||\, w$; $M' \leftarrow x_e \,||\, w$
    $z \leftarrow g(M)$
    return (M', z)
  else return null

When we get two different messages that give us the same output, we say that there is a *collision*. This adversary succeeds with probability $\left(1 - \frac{n}{n+1}\right)$

whenever there is a collision (see [8]). This is because any collision that occurs is likely to occur in the prefixes x_d and x_e, and thus regardless of what we append to them (as long as they maintain the message length) we still have a collision. A well-known result of [8] is that these collisions are expected to occur when there are $\sqrt{2} \cdot 2^{(l/2)}$ queries. This is generalized by the following theorem:

Theorem 2. *For any deterministic, non-stateful iterated message authentication scheme MAC, there is a constant $c > 0$ such that an adversary B using the internal collision attack requires only*

- *q_s tagging queries,*
- *q_v verification queries, and*
- *$c(b + l)(q_s + q_v)$ time*

to be able to forge a message with probability $\epsilon = \left(1 - e^{-\frac{q_s^2(n+1)}{2^{l+1}}}\right)\left(1 - \frac{n}{n+1}\right)$ where n is the number of blocks in the suffixes y and w and where q_s satisfies $q_s < \frac{2^{l/2}}{n}$.

This theorem is implied by the results of [8].

Notice that the internal collision attack queries only messages of the same length. This is important because collisions that occur when messages are of one length, might not occur when messages are another length. Consider for instance the case where the length of the message is prepended before querying. It is also important to notice that this attack only takes $\Theta(q)$ time. This is because for each query, we can immediately detect whether the output collided with a previous output or not using a hash table keyed to the outputs.