message (not equal to any of the first three messages) together with its authentication. The adversarial procedure follows:

1. The adversary asks to see the authentication of the message 0. It gets the pair $\left(f^{(1)}_{a'}(0), f^{(2)}_{a''}(1, f^{(1)}_{a'}(0))\right)$. Define $\beta_0 \stackrel{\text{def}}{=} f^{(1)}_{a'}(0) = f_{a'}(0)$.
2. The adversary asks to see the authentication of the message 1. It gets the pair $\left(f^{(1)}_{a'}(1), f^{(2)}_{a''}(1, f^{(1)}_{a'}(1))\right)$. Define $\beta_1 \stackrel{\text{def}}{=} f^{(1)}_{a'}(1) = f_{a'}(1)$.
3. The adversary asks to see the authentication of the message $(0, \beta_0)$. It gets the pair $\left(f^{(2)}_{a'}(0, \beta_0), f^{(2)}_{a''}(2, f^{(2)}_{a'}(0, \beta_0))\right)$.
4. The adversary outputs the message $(1, \beta_1)$ with the authentication got in the previous query, i.e., $\left(f^{(2)}_{a'}(0, \beta_0), f^{(2)}_{a''}(2, f^{(2)}_{a'}(0, \beta_0))\right)$.

In order to see that this forgery is indeed valid, note first that it is enough to show that $f^{(2)}_{a'}(0, \beta_0) = f^{(2)}_{a'}(1, \beta_1)$ (regardless of the value of a''). Now, by definition of $f^{(2)}$:

$$\begin{aligned}
f^{(2)}_{a'}(0, \beta_0) &= f_{a'}(f_{a'}(0) \oplus \beta_0) \\
&= f_{a'}(0) \\
&= f_{a'}(f_{a'}(1) \oplus \beta_1) \\
&= f^{(2)}_{a'}(1, \beta_1)
\end{aligned}$$

and we are done.

[12] B. Preneel and P. van Oorschot MDx-MAC and Building Fast MACs from Hash Functions. In *Advances in Cryptology – Crypto '95 Proceedings*, Lecture Notes in Computer Science Vol. 963, Springer-Verlag, pp. 1-14, 1996.

[13] G. Tsudik, Message Authentication with One-Way Hash Functions. In *Proceedings of Infocom 92*, IEEE Press, pp. 2055-2059, 1992.

[14] R. Rivest, The MD5 message digest algorithm. IETF Networking Working Group, RFC-1321, April 1992.

[15] P. Rogaway. Bucket Hashing and its Application to Fast Message Authentication. *Advances in Cryptology – Crypto '95 Proceedings*, Lecture Notes in Computer Science Vol. 963, Springer-Verlag, pp. 29-42, 1996.

[16] V. Shoup. On fast and Provably Secure Message Authentication based on Universal Hashing. *Advances in Cryptology – Crypto '96 Proceedings*, Lecture Notes in Computer Science Vol. 1109, Springer-Verlag, pp. 313-328, 1996.

[17] D. Stinson. Universal Hashing and Authentication Codes. *Designs, Codes and Cryptography*, vol. 4, No. 4, pp. 369-380, 1994.

[18] M. Wegman and L. Carter. New Hash Functions and their use in Authentication and Set Equality. *Journal of Computer and System Sciences*, Vol. 18, No. 2, pp. 143-154, 1979.

A A Flaw in one of the authentications suggested in [3]

In their conference version, [3] suggest a few ways to deal with the authentication of variable-length messages. One of these suggestions appears to be good also for the case that the length of the message is not known in advance. However, this suggestion has a flaw and it is not secure. In this appendix, we would like to point out this insecurity. We would like to stress that this is not the major result in [3] but only one of a few suggestions meant to deal with variable-length authentication.

The suggested authentication is called *Two steps MAC* and it uses two secret keys a', a'' (or alternatively, uses one secret key a to produce the two secrets $a' \stackrel{\text{def}}{=} f_a(0)$ and $a'' \stackrel{\text{def}}{=} f_a(1)$). The tag is defined as follows:

$$MAC_{a',a''}(x) = \left(f_{a'}^{(m)}(x), f_{a''}^{(2)}(m, f_{a'}^{(m)}(x))\right)$$

where $x = x_1 \cdots x_m$. We are going to present a counter example to the security of this suggestion. Note that our suggestion for a secure protocol is a simplification of this. Namely:

$$EMAC_{a',a''}(x) = f_{a''}(f_{a'}^{(m)}(x)).$$

In order to show that the suggestion in [3] is not secure, we present an adversarial procedure which asks to see authentications of three messages, and then it produces a fourth

References

[1] 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, Springer-Verlag, pp. 1-15, 1996.

[2] M. Bellare, R. Guerin, 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, Springer-Verlag, pp. 15-28, 1995.

[3] 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, Springer-Verlag, pp. 341-358, 1994. An updated version can be found in the personal URL's of the authros. See for example, http://www.cs.ucdavis.edu/ rogaway/papers/.

[4] M. Bellare and P. Rogaway. Collision Resistent Hashing: Towards Making UOWHFs Practical. *Advances in Cryptology – Crypto '97 Proceedings*, Lecture Notes in Computer Science 1294, Springer-Verlag, pp. 471-484, 1997.

[5] A. Berendschot, B. den Boer, J.P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C.J.A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, J. Vandewalle. Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040). *Lecture Notes in Computer Science 1007, RIPE Integrity Primitives*, Springer-Verlag, 1995 (226 pp.).

[6] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions . In *Journal of the ACM*, Vol. 33, No. 4, 210-217, (1986).

[7] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attack. In *SIAM Journal on Computing* , Vol. 17, No. 2, pp. 281-308, April 1988.

[8] H. Krawczyk, LFSR-based Hashing and Authentication. *Advances in Cryptology – Crypto '94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Springer-Verlag, pp. 129-139, 1994.

[9] ISO/IEC 9797. Data Cryptographic Techniques - Data Integrity Mechanism Using a Cryptographic Check Function Employing a Block Cipher Algorithm. 1989.

[10] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing* , Vol. 17, No. 2, pp. 373-386 April 1988.

[11] M. Luby and C. Rackoff. A Study of Password Security. In *Advances in Cryptology – Crypto '87 Proceedings*, Lecture Notes in Computer Science Vol. 293, Springer-Verlag, pp. 392-397, 1987.

4 Prefix-free message space guarantees security

In this section we observe that if the message space is prefix-free, then the security of CBC MAC with an underlying family of functions F is implied by the security of the family F. Recall that prefix-free means that if a message X is authenticated, then a prefix or an extension of X is never authenticated. We can obtain a prefix-free space of messages by encoding each message with a special last block that can never occur inside a message. More formally, suppose the message space is drawn from an alphabet of blocks which excludes a distinguished block $\perp$ and if we encode each authenticated message by appending the $\perp$ block to the end of the message, then we get that the encoded messages form a prefix-free set of messages.

In this setting, the adversary is allowed to forge only messages from the prefix-free message space. The CBC MAC is not secure if we allow the adversary to forge unrestricted messages. It is possible to construct examples in which the adversary sees authentications on messages from a prefix-free space, and then efficiently forges a new message that does not belong to the prefix-free space.

Theorem 2 *Suppose there is an adversary A that (ϵ, t, σ)-breaks CBC MAC with an underlying block cipher F such that the answered queries and the output query of A form a prefix-free message space, and such that $\sigma \leq 2^{(\ell+1)/2}$. Then there exists an adversary A' that distinguishes the family F from the family $\mathcal{R}_{\ell\to\ell}$ with advantage $\epsilon' = \epsilon - 6 \cdot \sigma^2 \cdot 2^{-\ell} - 2^{-\ell}$, running time $t + c \cdot \ell \cdot \sigma$ (for a small constant c) and number of queries at most σ.*

The proof of this theorem is a simple extension of the proof which was given in [3] for a message space of fixed length messages. We choose not to repeat their proof (which is quite different from the one presented in the previous subsections.) The main modification required in their proof is in redefining *border nodes*. Instead of border nodes being exactly the nodes at depth m, border nodes are defined to be the nodes which the adversary asks to see their content. The rest is an exercise.

5 Conclusion

We have shown that the *encrypted message authentication code* (EMAC) is secure: if there is an attack on this scheme, then an attack with comparable parameters can be set on the underlying block cipher. The EMAC scheme provides a secure solution for authenticating variable length messages with almost no additional cost on that of using CBC MAC. Finally, we have also remarked that the standard CBC MAC is secure if all authenticated messages are drawn from a prefix-free message space.

6 Acknowledgment

We thank Mihir Bellare, Joe Kilian and Moti Yung for helpful discussions. We thank the anonymous referees for their thorough review and very helpful comments.

$EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$ distribution. Thus, the advantage that A' has in distinguishing F from $R_{\ell\to\ell}$ in this case is ϵ_{FF}. The number of queries that A' makes to its oracle g is the number of calls to f_1 needed to compute $EMAC_{f_1,f_2}$ on the set of queries $X_1,\ldots,X_n$ that A'' makes. This is at most $\sigma \overset{\text{def}}{=} \sum_{i=1}^{n}|X_i|$. Finally, the running time of A' is at most the runnning time of A'' plus the time it takes to compute the answers for the queries of A''. The time it takes to compute these answers consists of three terms:

- First, the time required to copy queries and answers from the oracle tape of A to the oracle tape of A'.
- Second, the time required for choosing $f_2 \in F$ at random, according to the distribution of F.
- Third, the time it takes to compute the function $f_2 \in F$ for each of the queries,
- and finally, the time it takes to compute $EMAC_{f_1,f_2}$ given all the values of f_1 and f_2 on the relevant points.

Recall that we denote by T_F the worst-case time it takes to compute a function in F on a string in $\{0,1\}^\ell$, and by C_F the time it takes to choose at random a function in F. The first term is at most $c \cdot \sigma \cdot \ell$ for some small constant c which depends on the computational model. The second requires time C_F. The third is at most $\sigma \cdot T_F$, and the last term is at most $c \cdot \sigma \cdot \ell$. Summing it all up, we get that the time needed by A' in this case is at most $t + 3 \cdot c \cdot \sigma \cdot \ell + \sigma \cdot T_F + C_F$.

Analysis of Procedure 2: In this case, if g is taken from F, then A'' gets an oracle from the $EMAC^{(F,\mathcal{R}_{\ell\to\ell})}$ distribution. Whereas if g is drawn from $\mathcal{R}_{\ell\to\ell}$, then A'' gets the $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{R}_{\ell\to\ell})}$ distribution. Thus, the advantage that A' achieves is ϵ_{RR}. The number of calls that A' makes to g equals the number of queries made by A''. (The function f_2 is used once per query.) The running time of A' is computed similarly to the first case except that we now compute a function $f_1 \in \mathcal{R}_{\ell\to\ell}$ rather than a function in F. What is the cost of computing a value of $f_1(\omega)$ for a string $\omega \in \{0,1\}^\ell$? In practice, one would use a hash table to keep record of previous f_1 values and get a good expected behavior. But for the sake of worst case analysis we assume that previous values set to f_1 are kept in a balanced binary tree. In this case, finding an old value requires time at most $c \cdot \ell \cdot \log(\sigma)$, and the time for choosing a random new value is $c \cdot \ell$. Summing up all steps in Procedure 2 (computed similarly to the analysis of Procedure 1), we get that the running time of A' in this case is at most $t + 3 \cdot c \cdot \sigma \cdot \ell + c \cdot \sigma \cdot \ell \cdot \log(\sigma)$

Combining both cases, get that A' breaks F with advantage at least $\epsilon'/2$, makes at most σ queries, and its running time is bounded by

$$t + 3 \cdot c \cdot \sigma \cdot \ell + c \cdot \sigma \cdot \ell \cdot \log(\sigma) + \sigma \cdot T_F + C_F$$

Setting c appropriately, this fits the parameters of Theorem 1, and we are done with the proof of Theorem 1.

Next, we would like to show that if there is a distinguisher A'' between the family of functions $EMAC^F$ and the set of all functions in $R_{\ell^*\to\ell}$, then there exists a distinguisher A' which distinguishes a random function in F from a random function in $R_{\ell\to\ell}$, and that A' has “similar properties” to those of A'' (or of A) as asserted in Theorem 1.

We first examine the behavior of A'' on the hybrid family of functions $EMAC^{R_{\ell\to\ell}}$. By Lemma 3.3, the advantage that A'' achieves in distinguishing a random function in $R_{\ell^*\to\ell}$ from a random function in $EMAC^{R_{\ell\to\ell}}$ is at most $\sigma^2 \cdot 2 \cdot 2^{-\ell}$ (even if A'' is computationally unbounded). It follows that A'' must achieve advantage at least $\epsilon' \overset{\text{def}}{=} \epsilon - \sigma^2 \cdot 2 \cdot 2^{-\ell} - 2^{-\ell}$ in distinguishing the family of functions $EMAC^{R_{\ell\to\ell}}$ and the family of functions $EMAC^F$. We now show that this advantage can be used to break the block cipher F with advantage at least $\epsilon'/2$.

We use a standard hybrid argument. If A'' tells with advantage ϵ' between using $EMAC$ with a uniformly chosen $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$ and using $EMAC$ with a uniformly at random $f_1, f_2 \in F$, then A'' can also be used to distinguish any of these two distributions with a hybrid distribution in which f_1 is uniformly chosen in $\mathcal{R}_{\ell\to\ell}$ and f_2 is uniformly chosen in F. Denote by $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{R}_{\ell\to\ell})}$ the first distribution, in which a function $EMAC_{f_1,f_2}$ is selected by a uniform choice of $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$. Denote by $EMAC^{(F,F)}$ the second distribution, in which a function $EMAC_{f_1,f_2}$ is selected by a random choice of $f_1, f_2 \in F$ according to thr distribution of the family F, and denote by $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$ the hybrid distribution, in which a function $EMAC_{f_1,f_2}$ is selected by random choices of $f_1 \in \mathcal{R}_{\ell\to\ell}$ and $f_2 \in F$. Let ϵ_{FF} be the advantage of A'' in distinguishing $EMAC^{(F,F)}$ from $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$, and let ϵ_{RR} be the advantage of A'' in distinguishing $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$ from $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{R}_{\ell\to\ell})}$. It follows from the definition of advantage that $\epsilon_{RR} + \epsilon_{FF} \geq \epsilon'$.

Our new adversary A' will use A'' to perform one of the above two distinguishing procedures. With probability 1/2 Machine A' will ask A'' to distinguish $EMAC^{(F,F)}$ from $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$, and with probability 1/2 it will ask A'' to distinguish $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$ from $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{R}_{\ell\to\ell})}$. The advantage that A' will get is $\epsilon_{RR}/2 + \epsilon_{FF}/2 \geq \epsilon'/2$. Let us state each of these two procedures and later analyze why they actually behave as required.

Procedure 1: *Using A'' as a distinguisher of $EMAC^{(F,F)}$ from $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$:* Machine A' gets an oracle to a function g. Intuitively, A' has to decide whether g is drawn from $\mathcal{R}_{\ell\to\ell}$ or from F. To do this, A' sets $f_1 = g$ and chooses at random $f_2 \in F$. It then runs Machine A'' using f_1, f_2 to answer all the question that A'' makes to its oracle $EMAC_{f_1,f_2}$. The oracle is used to compute f_1, and f_2 can be computed by A' since it chose this function earlier.

Procedure 2: *Using A'' as a distinguisher of $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{F})}$ from $EMAC^{(\mathcal{R}_{\ell\to\ell},\mathcal{R}_{\ell\to\ell})}$:* Here, A' gets an oracle to a function g, and again, intuitively, A' has to decide whether g is drawn from $\mathcal{R}_{\ell\to\ell}$ or from F. Machine A' sets $f_2 = g$ and chooses uniformly at random $f_1 \in R_{\ell\to\ell}$. Actually, It is not possible to have a succinct description of f_1 for Machine A'. But such a description is not really needed. A' just keeps record of all past queries to f_1 and answers consistently on repeated queries. Whenever a new query is made to f_1, Machine A' chooses uniformly at random a string $\omega \in \{0,1\}^\ell$ and sets this string as the answer to the new query, keeping a record of the new value $f_1(\omega)$. Finally, Machine A' runs Machine A'' using f_1, f_2 to answer all the question that A'' makes to its oracle $EMAC_{f_1,f_2}$.

Analysis of Procedure 1: In this procedure, if g is taken from F, then A'' gets an oracle to the $EMAC^{(F,F)}$ distribution, whereas if g is drawn from $\mathcal{R}_{\ell\to\ell}$, then A'' gets the

$i-1$ distinct strings $f_2(\alpha_1), \ldots, f_2(\alpha_{i-1})$ is execatly $(i-1)\cdot 2^{-\ell}$. Thus, the probability that the i-th sub-query will cause a collision is at most $4 \cdot 2^{-\ell}$ and the probability of any collision amongst the m sub-queries is at most

$$\sum_{i=1}^{m}(i-1)\cdot 4 \cdot 2^{-\ell} = 2 \cdot m(m-1)\cdot 2^{-\ell}.$$

Since $m \leq \sum_{i=1}^{n} |X_n|$, we get that this probability is bounded above by $(\sum_{i=1}^{n}|X_n|)^2 \cdot 2 \cdot 2^{-\ell}$ and we are done with the proof of the corollary. ■

Let us now finish the proof of Lemma 3.3. First, we note that if the queries of the adversary are collision-free then the view of the adversary is exactly a uniform random choice of m distinct blocks. This is clearly true for the family $\mathcal{R}_{\ell*\to\ell}$. As for $EMAC^{\mathcal{R}_{\ell\to\ell}}$, fix f_1 to be any inner collision-free function on the given queries, and let $\alpha_i = f_1^{(*)}(Y_i)$ be the intermediate values on its queries. The uniform choice of $f_2 \in \mathcal{R}_{\ell\to\ell}$ that does not cause collisions on the set of inputs $\alpha_1, \ldots, \alpha_m$ results in a uniformly chosen m distinct random strings in $\{0,1\}^{\ell}$. Thus, in both cases, given that no collision occurs, the adversary sees m uniformly chosen random distinct blocks. Any m distinct blocks have the same probability. Thus the advantage that the adversary gets in this case is 0. In both cases, the adversary will output 1 with the same probability since its input is distributed equally.

We next assume, in a worst case manner, that if a collision occurs, then the adversary can exactly determine whether the oracle is a random function in $\mathcal{R}_{\ell*\to\ell}$ or a random function in $EMAC^{\mathcal{R}_{\ell\to\ell}}$. Namely, if a collision occurs, then the adversary outputs 1 with probability 1 for an oracle from $EMAC^{\mathcal{R}_{\ell\to\ell}}$, and it outputs 0 with probability 1 if it gets an oracle from $\mathcal{R}_{\ell*\to\ell}$. (This is probably not the case and the adversary probably has worst distinguishing advantage, but we are only computing an upper bound.) It remains to compute the probability that the adversary sees a collision. If the adversary gets an oracle of $EMAC^{\mathcal{R}_{\ell\to\ell}}$, then by Corollary 3.12 it gets to see a collision with probability at most $(\sum_{i=1}^{n}|X_n|)^2 \cdot 2 \cdot 2^{-\ell}$. On the other hand, if the adversary gets a random function in $R_{\ell*\to\ell}$ then the probability that it gets to see a collision is even smaller: at most $(\sum_{i=1}^{n}|X_n|)^2 \cdot (1/2) \cdot 2^{-\ell}$. Thus, the advantage the adversary may achieve in distinguishing the two cases is at most $(\sum_{i=1}^{n}|X_n|)^2 \cdot 2 \cdot 2^{-\ell}$ and we are done with Lemma 3.3.

3.2 Computationally bounded adversaries

In this section we would like to finish the proof of Theorem 1. First, we are given a machine A that (ϵ, t, σ) breaks $EMAC^F$ (see Definition 3.1). From this machine we can easily build a machine A'' that distinguishes between the family of functions $EMAC^F$ and the family $R_{\ell*\to\ell}$. The new machine A'' uses its oracle to answer A's queries. Finally, A'' takes the output of A, (X_n, β), and checks if the forgery is successful by asking its oracle whether $EMAC_{f_1,f_2}(X_n) = \beta$. If the forgery is successful, A'' outputs 1, and otherwise 0. The probability that A'' outputs 1 if the oracle is from $EMAC^F$ is at least ϵ and otherwise exactly $2^{-\ell}$. Thus, A'' has advantage at least $\epsilon - 2^{-\ell}$. The cumulative length of the queries of A'' is exactly σ and the running time of Machine A'' is at most $t + c \cdot \sigma \cdot \ell$: the time it takes to run A and copy queries and responses from the oracle tape of A to the oracle tape of A'' forth and back.

probability of the event in Equation (8) is 0, or ω is a fresh string and each value of $f_1(\omega)$ is equally probable.

In the first case, i.e., $\gamma_j = \omega$, we get

$$f_1^{(*)}(Y_j^1, Y_j^2, \ldots, Y_j^{s-1}) \oplus Y_j^s = \omega. \tag{9}$$

If the prefix $(Y_j^1, Y_j^2, \ldots, Y_j^{s-1})$ is not empty (i.e., $s > 1$), then this prefix must also be a sub-query Y_i, and we may now rewrite Equation (9) as

$$f_1^{(*)}(Y_i) = Y_j^s \oplus \omega. \tag{10}$$

Using Part (1) of Lemma 3.10 this has probability at most $2{\cdot}2^{-\ell}$. If the prefix $(Y_j^1, Y_j^2, \ldots, Y_j^{s-1})$ is empty (i.e., $s = 1$) then $Y_j = Y_j^s$ and we ask whether $Y_j = \omega$. But it is assumed in Part (2) of the lemma that this is not the case, so the probability of this event is 0 and we are done with Lemma 3.11. ■

The implication of this lemma is that for any possible new query, there is little chance that a collision will occur. Thus, no matter how powerful the adversary is, based on seeing the EMAC values of its bunch of queries, it will be "hard" for the adversary to get a new query that causes collision. By induction, we can now compute the probability that the (computationally unbounded) adversary sees a collision on a set of queries X_i, $i = 1, \ldots, n$.

Corollary 3.12 *Consider picking uniformly at random $f_1, f_2 \in \mathcal{R}_{\ell \to \ell}$. Suppose that a computationally unbounded machine A is picking queries $X_1, \ldots, X_n \in \left(\{0,1\}^\ell\right)^*$. The choice of query X_i may depend on the $EMAC_{f_1,f_2}$ values of all the sub-queries of the set $X_1, \ldots, X_{i-1}$, but not on any other information on f_1, f_2. Let the set of all sub-queries of $X_1, \ldots, X_n$ be $Y_1, \ldots, Y_m$ and suppose the cardinality of the set of sub-queries is bounded by $m^2/4 + m - 1 \leq 2^\ell/2$. Then the probability that there is a collision of $EMAC_{f_1,f_2}$ on the set of sub-queries $Y_1, \ldots, Y_m$ is at most $\left(\sum_{i=1}^n |X_n|\right)^2 \cdot 2 \cdot 2^{-\ell}$.*

Proof: We prove the corollary by an induction on the sub-queries. We use a simple ordering on the sub-queries $Y_1, \ldots, Y_m$. We first take all sub-queries of X_1 by order of length, and then we add the sub-queries of X_2 not yet encountered (again, by order of length) and so forth. Since all $EMAC_{f_1,f_2}$ values on all these sub-queries are shown to the adversary, we may think of these sub-queries as being the actual queries of the adversary, where at Step i, it chooses a previous sub-query Y_j, $j < i$, and a new block $\omega \in \{0,1\}^\ell$ and it produces the new sub-query $Y_i = Y_j\omega$.

The probability that a collision occurs at the first sub-query is trivially zero. Now suppose there was no collision amongst the first $i-1$ queries and let us compute the probability that a collision occurs at the i-th query. Since there is no collision among the first $i-1$ queries, we may use Lemma 3.11. By this lemma, no matter how the i-th sub-query is computed, the probability that it will cause an inner collision with any specific previous sub-query is at most $3 \cdot 2^{-\ell}$. Thus, the probability that it will cause an inner-collision with any of these $i-1$ queries is at most $(i-1) \cdot 3 \cdot 2^{-\ell}$. However, a collision may also occur when there is no inner collision. Recall that $EMAC_{f_1,f_2}(Y_i) = f_2(\alpha_i)$ for $\alpha_i = f_1^{(*)}(Y_i)$. Given that the α_j's are all distinct for $j = 1, \ldots, i$, the probability that a random $f_2 \in \mathcal{R}_{\ell \to \ell}$ will map α_i to any of the

An equality here can come either from the inputs to f_1 on each side of the equation being equal, or otherwise, the inputs to f_1 are different, but f_1 maps them both to the same value. We bound the probability of the first case by $2 \cdot 2^{-\ell}$ using Lemma 3.10, and we bound the probability of the second case by $2^{-\ell}$ using Lemma 3.7. We start with the second case.

First, we use the fact that $Y_1, \ldots, Y_m$ is a set of all sub-queries. This means that the computation of $EMAC_{f_1,f_2}$ on all these sub-queries involves evaluating f_1 on exactly the m strings $\gamma_1, \ldots, \gamma_m$. Since f_1 is inner collision-free, then all γ_i's are distinct and the set of values $f_1(\gamma_i)$, for $i = 1, \ldots, m$ is also a set of distinct values. Recall that we are considering Equation (4) for the case that $\gamma_j \neq f_1^{(*)}(Y_k) \oplus \omega$. If the value $\gamma \overset{\text{def}}{=} f_1^{(*)}(Y_k) \oplus \omega$ equals one of the other γ_i for $1 \leq i \leq m$, $i \neq j$ then by inner collision-free property, $f_1(\gamma_j)$ must be different from $f_1(\gamma_i)$ and the probability of Equation (4) holding is zero. So suppose $\gamma \neq \gamma_i$ for any $1 \leq i \leq m$. In this case, the event in mind is that when picking at random $f_1 \in \mathcal{A}$ it holds that $f_1(\gamma)$ equals a given string $f_1(\gamma_j)$. Since the value of $f_1(\gamma)$ is independent of all $f_1(\gamma_i)$ for $i = 1, \ldots, m$, and since f_1 is randomly picked in $\mathcal{A}$ then any string in $\{0,1\}^\ell$ has probability $2^{-\ell}$ to be $f_1(\gamma)$. In particular $\text{Prob}[f_1(\gamma) = f_1(\gamma_j)] = 2^{-\ell}$.

The other case to consider is that the inputs of f_1 in both sides of Equation (4) are equal. In this case, $\gamma_j = \gamma$. What is the probability that this happens? Writing this equation explicitly we get:

$$f_1^{(*)}(Y_j^1, Y_j^2, \ldots, Y_j^{s-1}) \oplus Y_j^s = f_1^{(*)}(Y_k) \oplus \omega. \tag{5}$$

If the prefix $(Y_j^1, Y_j^2, \ldots, Y_j^{s-1})$ is not empty (i.e., $s > 1$), then this prefix must also be a sub-query, since the set of Y_i's is a set of all sub-queries. Denote the index of this query by i and we may now rewrite Equation (5) as

$$f_1^{(*)}(Y_i) \oplus f_1^{(*)}(Y_k) = Y_j^s \oplus \omega. \tag{6}$$

Since the indices i, j and k are fixed and since $Y_j^s \oplus \omega$ is a fixed string, we may use Part (2) of Lemma 3.10 and get that the probability of the event in Equation (6) is at most $2 \cdot 2^{-\ell}$.

If the prefix $(Y_j^1, Y_j^2, \ldots, Y_j^{s-1})$ is empty (i.e., $s = 1$) then we need to bound the probability that $f_1^{(*)}(Y_k) = Y_j^s \oplus \omega$. Here, we get the same bound using Part (1) of Lemma 3.10.

Summing up the two cases, we get that the probability that $f_1^{(*)}(Y_j) = f_1^{(*)}(Y_k\omega)$ is at most $3 \cdot 2^{-\ell}$ and we are done with Part (1) of Lemma 3.11.

We now move to proving Part (2) of Lemma 3.11. the argument is quite similar. Again, fix the index j and the string ω. Since ω is one block, $f_1^{(*)}(\omega) = f_1(\omega)$. By Lemma 3.7, we may rewrite the condition of as:

$$\text{Prob}_{f_1 \in \mathcal{A}}[f_1^{(*)}(Y_j) = f_1(\omega)] \leq 3 \cdot 2^{-\ell}. \tag{7}$$

The event that we are interested in is

$$f_1(\gamma_j) = f_1(\omega). \tag{8}$$

And again, we split the analysis to the case $\gamma_j = \omega$ and $\gamma_j \neq \omega$. In the latter case, either ω equals one of the γ_i, $1 \leq i \leq m$, $i \neq j$ and then by the fact that f_1 is inner collision-free, the previous queries (and sub-queries) the probability that *any* new query will cause an inner collision is small. Thus, there can be no clever way to construct a new query that will cause an inner collision.

On the technical level, recall that we are always considering the set of all sub-queries for the queries made so far. Intuitively, this means that the adversary is always given the $EMAC_{f_1,f_2}$ values of all its sub-queries. To preserve this variant, we consider a new query to be a one-block extension of an existing sub-query. Of course, any independent new query can be built by several block-extensions of the already existing queries.

The assumption in the next lemma is that the number of sub-queries made so far is smaller than $\sqrt{2^\ell/2}$. Note that otherwise, there is a good chance that the adversary has already encountered a collision even if it just randomly selected its queries (Recall that ℓ is the block size).

Lemma 3.11 *Fix any set of n queries $X_1, \ldots, X_n \in \left(\{0,1\}^\ell\right)^*$. Let $Y_1, \ldots, Y_m$ be the set of sub-queries of $X_1, \ldots, X_n$. Let $\beta_1, \ldots, \beta_m$ be any distinct strings in $\{0,1\}^\ell$. Consider picking uniformly at random $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$ and applying $EMAC_{f_1,f_2}$ on each sub-query $Y_1, \ldots, Y_m$. Suppose that the cardinality of the set of sub-queries is bounded by $m^2/4 + m - 1 \leq 2^\ell/2$, then, for any string $\omega \in \{0,1\}^\ell$ it holds that*

1. *for any pair of indices j, k, $1 \leq j, k \leq m$, if $Y_j \neq Y_k\omega$ then*

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_j) = f_1^{(*)}(Y_k\omega) | EMAC_{f_1,f_2}(Y_j) = \beta_j \ \ \forall j = 1, 2, \ldots, m] \leq 3 \cdot 2^{-\ell}.$$

2. *for any index j, $1 \leq j \leq m$, if $Y_j \neq \omega$ then*

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_j) = f_1^{(*)}(\omega) | EMAC_{f_1,f_2}(Y_j) = \beta_j \ \ \forall j = 1, 2, \ldots, m] \leq 3 \cdot 2^{-\ell}.$$

Proof: In case $Y_k\omega = Y_i$ for some $1 \leq i \leq m$, $i \neq j$, then we are done. The probability in Part (1) of the lemma is zero, since it is assumed that there are no collisions on the sub-queries $Y_1, \ldots, Y_m$. The same holds for Part (2), in case $\omega = Y_i$ for some $1 \leq i \leq m$, $i \neq j$. In the sequel we assume that $Y_k\omega$ (or ω) is a new query different from all given sub-queries $Y_1, \ldots, Y_m$.

Using the notations from the proof of Lemma 3.10, we denote by $\mathcal{A}$ all functions in $\mathcal{R}_{\ell\to\ell}$ which are inner collision-free with respect to the set of sub-queries $Y_1, \ldots, Y_m$. Also, we denote the blocks of Query Y_i by $(Y_i^1, Y_i^2, \ldots, Y_i^s)$ (where s is the number of blocks in the query i). Finally, for all $i = 1, \ldots, m$, we denote by γ_i the string $\gamma_i \stackrel{\text{def}}{=} f_1^{(*)}(Y_i^1, Y_i^2, \ldots, Y_i^{s-1}) \oplus Y_i^s$, (If $s = 1$, then by our convention $f_1^{(*)}(\epsilon) = 0^\ell$.) Thus, $EMAC_{f_1,f_2}(Y_i) = f_2(f_1(\gamma_i))$.

We start with the first part of the lemma. Fix the indices j, k and the string ω. By Lemma 3.7, we may rewrite the condition of Part (1) of the lemma as:

$$\mathrm{Prob}_{f_1\in\mathcal{A}}[f_1^{(*)}(Y_j) = f_1^{(*)}(Y_k\omega)] \leq 3 \cdot 2^{-\ell}. \quad (3)$$

The event that we are interested in is $f_1^{(*)}(Y_j) = f_1^{(*)}(Y_k\omega)$ which can be written as

$$f_1(\gamma_j) = f_1(f_1^{(*)}(Y_k) \oplus \omega). \quad (4)$$

that f_1 is chosen in $\mathcal{A}$. Our second observation is that any string assigned to $f_1(\gamma_i)$ that does not cause inner-collision has the same probability. In other words, let $\Omega \subseteq \{0,1\}^\ell$ be the set of strings that do not cause an inner collision when set to $f_1(Y_i)$. The probability that $f_1(\gamma_i) = \omega$ for any string $\omega \in \Omega$ is exactly $1/|\Omega|$. To see that this is true we have to count the number of functions $f_1 \in \mathcal{A}$ for which $f_1(\gamma_j) = \alpha_j$ for $j = 1, \ldots, i-1, i+1, \ldots, m$ and $f_1(\gamma_i) = \omega$. All values involved are now determined. This includes all γ_j, $j = 1, \ldots, m$, all α_j, $j = 1, \ldots, i-1, i+1, \ldots, m$, and the string ω. All other entries of f_1 can be set to any value without any restriction (the only restriction of the class $\mathcal{A}$ is that there is no inner collision on the set $Y_1, \ldots, Y_m$). Thus, the number of functions $f_1 \in \mathcal{A}$ that agree with $f_1(Y_i) = \omega$, for $\omega \in \Omega$, and with $f_1(\gamma_j) = \alpha_j$ for all $j = 1, \ldots, i-1, i+1, \ldots, m$ does not depend on the actual value of ω.

If $\alpha \notin \Omega$ then the probability that $f_1(\gamma_i) = \alpha$ is 0 and we are done. Otherwise, in order to bound the probability that $f_1(\gamma_i) = \alpha$ from above, we must bound the cardinality of Ω from below. Which strings are not in Ω? First, all the strings $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots \alpha_m$ are not in Ω since setting $f_1(\gamma_i)$ to any of these strings would cause an inner collision. Second, let us check the values that are determined by $f_1(\gamma_i)$. All the γ_j's such that Y_j is a direct extension of Y_i are determined by the setting of $f_1(\gamma_i)$. We must have all γ_j, $j = 1, \ldots, m$, distinct. Otherwise, an inner-collision is bound to happen. Therefore, any setting of $f_1(\gamma_i)$ that will cause a collision in the values of γ_j's is also not in Ω.

What is the cardinality of Ω? From the 2^ℓ strings of $\{0,1\}^\ell$ we must substract the $m-1$ strings $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_m$. Next we have to subtract the number of strings that may cause collisions in the set of γ_j's. Let us compute an upper bound on the number of such forbidden strings. Suppose there are t direct extensions of Y_i and $m-t$ queries that are not direct extensions of Y_i. A value is forbidden for $f_1(\gamma_i)$ if there exists an extension string w of Y_i such that $Y_i w$ is one of the t extension queries of Y_i and the value $f_1(\gamma_i) + w$ equals one of the g_j's for the $m-t$ values that are already set. We get $t \cdot (m-t)$ forbidden values at the most. Since $t \cdot (m-t) \leq m^2/4$ we get that the cardinality of Ω is at least $2^\ell - m^2/4 - m - 1 \geq 2^\ell/2$. Thus, the probability that $f_1(\gamma_i) = \alpha$ is at most $2 \cdot 2^{-\ell}$ and we are done with Part (1) of Lemma 3.10.

To show Part (2) of the lemma we use Equation (2) again. Let T be the set of values in $\{0,1\}^\ell$ that are not equal to any of the α_j, $1 \leq j \leq m$, $j \neq k$, $j \neq i$. Summing over possible α_k's in T, we get

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_i) \oplus f_1^{(*)}(Y_k) = \alpha | EMAC_{f_1,f_2}(Y_j) = \beta_j \ \ \forall j = 1, 2, \ldots, m] =$$

$$\sum_{\alpha_k \in T} \mathrm{Prob}_{f_1 \in \mathcal{A}}[f_1^{(*)}(Y_k) = \alpha_k] \cdot$$
$$\mathrm{Prob}_{f_1 \in \mathcal{A}}[f_1^{(*)}(Y_i) = \alpha \oplus \alpha_k \ \ | \ f_1^{(*)}(Y_j) = \alpha_j \ \ \forall j = 1, \ldots, i-1, i+1, \ldots, m]$$

The above is an averaging expression over terms that, by Equation (2), are each smaller than $2 \cdot 2^{-\ell}$, and we are done with the proof of Lemma 3.10. ■

We now want to use Lemma 3.10 to show that it is hard for an adversary to produce a new query that causes a collision (given that there were no collisions in previous queries and sub-queries). Our next lemma asserts that even given the $EMAC_{f_1,f_2}$ value on all the

the probability that $f_1^{(*)}(Y_i)$ equals α? It may depend on the values set on the other queries. To deal with the dependencies here, we use a standard trick. We compute the probability of the event $f_1^{(*)}(Y_i) = \alpha$, after fixing values of all the other values α_j, $j \neq i$. We will show that for any distinct values $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1} \ldots \alpha_m$ all different from α, it holds that

$$\mathrm{Prob}_{f_1 \in \mathcal{A}}[f_1^{(*)}(Y_i) = \alpha \ \mid f_1^{(*)}(Y_j) = \alpha_j \ \ \forall i = 1, \ldots, i-1, i+1, \ldots, m] \leq 2 \cdot 2^{-\ell} \qquad (2)$$

Once we've shown Equation (2) then we get that also for randomly chosen α_j's the same holds (no matter what the distribution by which the α_j's are chosen is). Thus, Equation (1) also holds and we are done.

So it remains to show Equation (2). We will now use the fact that $Y_1, \ldots, Y_m$ is a set of all sub-queries. This fact implies that each prefix of any query Y_i is another query in the set $Y_1, \ldots, Y_m$. Let us look into the term $f_1^{(*)}(Y_j)$. Recall that $f_1^{(*)}(Y_j)$ translates to a series of operations of f_1. Denote the blocks of Y_j by $Y_j = (Y_j^1, Y_j^2, \ldots, Y_j^s)$. We have

$$\begin{aligned} f_1^{(*)}(Y_j) &= f_1^{(s)}(Y_j) \\ &= f_1\left(f_1^{(s-1)}(Y_j^1, \ldots, Y_j^{s-1}) \oplus Y_j^s\right) \end{aligned}$$

(Recall our convention that $f_1^{(0)}(\epsilon) = 0^\ell$ for the empty string ϵ.) If $s = 1$, we get

$$f_1^{(*)}(Y_j) = f_1(Y_j).$$

If $s > 1$, then since the set of Y_w, $w = 1, 2, \ldots, m$ is a set of all sub-queries, then there must be a query Y_k in the set that equals the prefix $(Y_j^1, \ldots, Y_j^{s-1})$ of Y_j, where k satisfies $1 \leq k \leq m$, $k \neq j$. Thus, we get:

$$f_1^{(*)}(Y_j) \ = \ f_1\left(f_1^{(*)}(Y_k) \oplus Y_j^s\right)$$

Denote by γ_j the input to f_1: $\gamma_j \overset{\text{def}}{=} f_1^{(*)}(Y_k) \oplus Y_j^s$ if $|Y_j| > 1$ and $\gamma_j = Y_j$ otherwise. Namely, it holds that

$$EMAC(Y_j) = f_2(f_1(\gamma_j)) \qquad i = 1, \ldots, m$$

Let us look at the string γ_j. Is it a predetermined string? Or does it depend on the choice of $f_1 \in \mathcal{A}$?

The string Y_j^s is predetermined. It is part of the set of sub-queries. The string $\alpha_k = f_1^{(*)}(Y_k)$ is predetermined for all $1 \leq k \leq m$, $k \neq i$. Thus, for all queries Y_j that are not direct extensions of the special query Y_i, the term $\gamma_j = \alpha_k + Y_j^s$ is a predetermined string. It is determined by the values of Y_j and α_k. (We call Query $Z \in \left(\{0,1\}^\ell\right)^*$ a direct extension of Query $W \in \left(\{0,1\}^\ell\right)^*$ if Query Z equals Wz for some block $z \in \{0,1\}^\ell$.) Note, that γ_i is completely predetermined, since the i-th sub-query is not a direct extension of the i-th sub-query. But the value of $f_1(\gamma_i)$ is not predetermined. Actually, we are interested in the probability that $f_1(\gamma_i) = \alpha$.

Once the value of $f_1(\gamma_i)$ is determined, then the values of all γ_j, $j = 1, \ldots, m$ are determined, and so are the values of all $f_1(\gamma_j)$, $j = 1, \ldots, m$. We first observe that the string assigned to $f_1(\gamma_i)$ causes an inner collision with probability 0. This follows from the fact

Since all α_i's are distinct, and since f_2 is uniformly chosen in $\mathcal{R}_{\ell\to\ell}$, the first term is exactly $2^{-\ell m}$: we fix m different values of the function f_2. The numerator is exactly $1/|\mathcal{R}_{\ell\to\ell}|$ since f_1 is picked uniformly at random from $\mathcal{R}_{\ell\to\ell}$. The denominator is independent of the function g. Thus, we get that this expression is the same for all functions g which are inner collision-free, and we are done. ■

Recall that the value $EMAC(X)$ consists of an intermediate computation of $f_1^{(*)}(X)$ on the query X, and then a final computation of $f_2(f_1^{(*)}(X))$. In what follow, we show that the intermediate values $f_1^{(*)}(X_i)$'s are almost random even if the final values of the $EMAC(X_i)$'s are fixed. Furthermore, we will show that the exclusive-or of two intermediate values are also almost random. We formalize this in the following lemma.

Lemma 3.10 *Fix any set of n queries $X_1,\ldots,X_n \in \left(\{0,1\}^\ell\right)^*$. Let $Y_1,\ldots,Y_m$ be the set of sub-queries of $X_1,\ldots,X_n$. Let $\beta_1,\ldots,\beta_m$ be any distinct strings in $\{0,1\}^\ell$. Consider picking uniformly at random $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$ and applying $EMAC_{f_1,f_2}$ on each sub-query $Y_1,\ldots,Y_m$. Suppose the cardinality of the set of sub-queries is bounded by $m^2/4+m-1 \leq 2^\ell/2$, then for any string $\alpha \in \{0,1\}^\ell$ it holds that*

1. *for any $1 \leq i \leq m$,*

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_i) = \alpha | EMAC_{f_1,f_2}(Y_j) = \beta_j \quad \forall j = 1,2,\ldots,m] \leq 2\cdot 2^{-\ell}$$

2. *for any pair of indices $i \neq k$, $1 \leq i,k \leq m$,*

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_i) \oplus f_1^{(*)}(Y_k) = \alpha | EMAC_{f_1,f_2}(Y_j) = \beta_j \quad \forall j = 1,2,\ldots,m] \leq 2\cdot 2^{-\ell}$$

Proof: Let us fix an arbitrary index $1 \leq i \leq m$ and fix an arbitrary $\alpha \in \{0,1\}^\ell$. We show Part (1) of the lemma for the fixed i and α. Since we know that all the output strings β_i's $1 \leq i \leq m$ are distinct, then it follows that the strings $\alpha_i \stackrel{\mathrm{def}}{=} f_1^{(*)}(Y_i)$, $1 \leq i \leq m$, must also be distinct. Namely, the function f_1 must be inner collision-free on these queries.

Note that the β_i's, the X_i's, and the Y_i's are arbitrary and predetermined. They are not random variables. The distribution is taken over the choice of $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$. In what follows, we only discuss the conditional distribution in which $EMAC_{f_1,f_2}(Y_i) = \beta_i$, for $i = 1,2,\ldots,m$. By Lemma 3.7, we know that all inner collision-free functions f_1 in $\mathcal{R}_{\ell\to\ell}$ are equally likely in this distribution. Define by $\mathcal{A}$ the set of all inner collision-free functions for the predetermined set of sub-queries. Note that the class $\mathcal{A}$ is determined by the set of sub-queries $Y_1,\ldots,Y_m$. Using Lemma 3.7 and the new notation, we may rewrite Part (1) of the lemma as:

$$\mathrm{Prob}_{f_1\in\mathcal{A}}[f_1^{(*)}(Y_i) = \alpha] \;\leq\; 2\cdot 2^{-\ell}. \tag{1}$$

Here, the probability is over a the random uniform choice of a function $f_1 \in \mathcal{A}$.

Our life would have been easier if we didn't have the restriction that f_1 is inner collision-free. Had f_1 been drawn from $\mathcal{R}_{\ell\to\ell}$ uniformly at random, it would have been the case that the chance to get any fixed value α for $f_1^{(*)}(Y_i)$ would have been $2^{-\ell}$. But this is not the case here. We have a distribution of randomly chosen f_1 in $\mathcal{A}$. For this distribution, what is an inner collision on this set of queries is equally likely to be the one used, given all the $EMAC_{f_1,f_2}$ values on the set of sub-queries. Intuitively, this means that f_2 "hides" whatever happened before invoking it, unless there was a collision. In the second lemma (see Lemma 3.10 below) we use this first lemma to show that given the $EMAC_{f_1,f_2}$ values on the set of sub-queries of the queries made so far, there is almost no information on the intermediate values that f_2 was computed on. Recall that in EMAC we first compute $\alpha_i \stackrel{\text{def}}{=} f_1^{(*)}(Y_i)$ on a sub-query Y_i, and then we invoke f_2 once on the value obtained, i.e., compute $f_2(\alpha_i)$. The claim in this second lemma is that given the $EMAC_{f_1,f_2}$ values of all the sub-queries of the queries made so far (and assuming there was no collision), it is "impossible" to tell what the intermediate α_i's values are. Using this, we show in the third lemma (see Lemma 3.11 below) that the probability that a new query will cause a collision, given the $EMAC_{f_1,f_2}$ values on all sub-queries of the queries made so far, is small. Intuitively, any specific query will only cause a collision to a small fraction of all the possible f_1's. After showing this, we finally prove that Lemma 3.3 holds. Let us start with the first lemma.

Lemma 3.7 *Fix any set of n queries $X_1, \ldots, X_n \in \left(\{0,1\}^\ell\right)^*$. Let $Y_1, \ldots, Y_m$ be the set of sub-queries of $X_1, \ldots, X_n$. Let $\beta_1, \ldots, \beta_m$ be any distinct strings in $\{0,1\}^\ell$. Consider picking uniformly at random $f_1, f_2 \in \mathcal{R}_{\ell \to \ell}$ and applying $EMAC_{f_1,f_2}$ on each sub-query $Y_1, \ldots, Y_m$. Then for any two functions $g, g' \in \mathcal{R}_{\ell \to \ell}$ such that there is no inner-collision on $Y_1, \ldots, Y_m$ for $f_1 = g$ nor for $f_1 = g'$, it holds that*

$$\begin{aligned} &\text{Prob}_{f_1,f_2}\left[f_1 = g \mid EMAC_{f_1,f_2}(Y_i) = \beta_i \quad \forall i = 1,2,\ldots,m\right] \quad = \\ &\qquad \text{Prob}_{f_1,f_2}\left[f_1 = g' \mid EMAC_{f_1,f_2}(Y_i) = \beta_i \quad \forall i = 1,2,\ldots,m\right] \end{aligned}$$

Remark 3.8 *Note that the β_i's are distinct and therefore f_1's that cause an inner collision on $Y_1, \ldots, Y_m$ have probability 0 to be the ones used.*

Remark 3.9 *The intuition of this assertion in this lemma is that when given a set of queries whose authentications are distinct, all functions f_1 that do not cause an inner collision on these queries are equally likely to be used in the authentication.*

Proof: Fix a $g \in \mathcal{R}_{\ell \to \ell}$, which doesn't cause inner collision on $Y_1, \ldots, Y_m$. Let $\alpha_i \stackrel{\text{def}}{=} g^{(*)}(Y_i)$ for $i = 1, \ldots, m$ If $f_1 = g$, then the output of the $EMAC_{f_1,f_2}$ is actually $\beta_1 = f_2(\alpha_1), \beta_2 = f_2(\alpha_2), \ldots, \beta_m = f_2(\alpha_m)$. Note that the α_i's must be distinct since it is given that the β_i's are distinct.

Now let's compute the probability that $f_1 = g$ given the output of the authentication. Using Bayes rule:

$$\begin{aligned} &\text{Prob}_{f_1,f_2}\left[f_1 = g \mid EMAC_{f_1,f_2}(Y_i) = \beta_i \quad \forall i = 1,2,\ldots,m\right] \\ &\quad = \quad \text{Prob}_{f_1,f_2}[EMAC_{f_1,f_2}(Y_i) = \beta_i \quad \forall i = 1,2,\ldots,m | f_1 = g] \cdot \\ &\qquad\qquad \cdot \frac{\text{Prob}_{f_1,f_2}[f_1 = g]}{\text{Prob}_{f_1,f_2}[EMAC_{f_1,f_2}(Y_i) = \beta_i \quad \forall i = 1,2,\ldots,m]} \\ &\quad = \quad \text{Prob}_{f_2}[\wedge_{i=1}^m f_2(\alpha_i) = \beta_i] \cdot \frac{\text{Prob}_{f_1}[f_1 = g]}{\text{Prob}_{f_1,f_2}[EMAC_{f_1,f_2}(Y_i) = \beta_i \quad \forall i = 1,2,\ldots,m]} \end{aligned}$$

Remark 3.4 *The advantage (as defined in the preliminaries) is measured over a random choice of a function in the corresponding family and over the coin tosses of the adversary. More specifically, on top of A's coin tosses, in case Machine A gets a random oracle from the family $EMAC^{R_{\ell\to\ell}}$, then the probability is taken over a random choice of $f_1, f_2 \in R_{\ell\to\ell}$, and when Machine A gets an oracle to a random function in $R_{\ell^*\to\ell}$, then the probability is taken over the choice of a random function in $R_{\ell^*\to\ell}$. The queries $X_1, \ldots, X_n$ are determined by the coin tosses of A and the answers of the oracle to its queries.*

Loosely speaking we prove the lemma using the following steps. First, we show that when there are no collisions, i.e., the adversary gets a different value for each query, then the adversary knows "nothing" except for the fact that there were no collisions. Then, we show that the probability that the adversary can "cause" a collision based on its view (even when using its unbounded computational capabilities) is small. We conclude with deducing that the adversary has little advantage in breaking the system.

We start with formalizing the first argument. Since the adversary A is not limited in computational power we may assume it is deterministic. Namely, its queries and final output are set deterministically according to the answers it gets from the oracle. For example, if the oracle is a random function in the family $EMAC^{R_{\ell\to\ell}}$, then the runs of Machine A, i.e., the queries it makes and the responses it gets are completely determined by the random selection of f_1 and f_2 in $R_{\ell\to\ell}$. We will prove a bit more than stated, in the sense that we let Adversary A see all the authentications of all the prefixes of its queries: when the adversary A makes a query $X = (x_1, x_2, \ldots, x_t)$, then it not only gets the value of $EMAC_{f_1,f_2}(X)$ but also it gets all the EMAC's of all the prefixes of X. Specifically, A gets to see: $f_2(f_1(x_1))$, $f_2(f_1(f_1(x_1) \oplus x_2))$, $\ldots$, $f_2(f_1(\cdots f_1(f_1(x_1) \oplus x_2) \oplus \cdots \oplus x_t))$. We denote the prefixes of a query X by *sub-queries* of X.

Definition 3.5 *Let $X_1, \ldots, X_n$ be any set of n queries in $\left(\{0,1\}^\ell\right)^*$. We define the* set of sub-queries *of $X_1, \ldots, X_n$ to be the set of all sub-queries of all queries $X_1, \ldots, X_n$. The set also includes the given queries $X_1, \ldots, X_n$.*

To make the following discussion clear, we also need to define what collisions and inner collisions are.

Definition 3.6 A collision: *Let $X_1, \ldots, X_n$ be n strings in $\left(\{0,1\}^\ell\right)^*$, and let f_1, f_2 be two functions in $\mathcal{R}_{\ell\to\ell}$. We say that there occurs a* collision *of $EMAC_{f_1,f_2}$ on the queries $X_1, \ldots, X_n$ if there exists a pair of indices $1 \le i < j \le n$ for which $EMAC_{f_1,f_2}(X_i) = EMAC_{f_1,f_2}(X_j)$. We say that there is an* inner collision *if the collision occurs before invoking f_2. Namely, if there exists a pair of indices $1 \le i < j \le n$ for which $f_1^{(*)}(X_i) = f_1^{(*)}(X_j)$. If there is no collision, we will say that f_1, f_2 are* collision-free *for the given set of queries. If there is no inner collision, we will say that f_1 is* inner collision-free *for the given set of inputs.*

Note that inner collision is determined by f_1 (and does not depend on the choice of f_2). Note also, that if there is an inner collision on queries $X_1, \ldots, X_n$ with respect to f_1, then for any f_2 there is a collision on $X_1, \ldots, X_n$ with respect to f_1, f_2.

In the first lemma (see Lemma 3.7 below) we assert that if there is no collision on the queries done so far (including on their set of sub-queries) then any f_1 that does not cause

C_F be the time required to choose a function according to the distribution of the family F. Let us now state our main theorem.

Theorem 1 *Let $F \subseteq \mathcal{R}_{\ell \to \ell}$ be a family of functions. If there exists an adversary A that (ϵ, t, σ) succeeds in breaking $EMAC^F$, then there exists an adversary A' for distinguishing a randomly chosen function according to the distribution of F from a uniformly chosen random function in $\mathcal{R}_{\ell \to \ell}$ with the following properties. Adversary A' achieves an advantage of at least $\epsilon/2 - \sigma^2 \cdot 2^{-\ell} - 2^{-\ell}/2$ after making at most σ queries and working in time at most*

$$t + c \cdot \sigma \cdot \ell \cdot \log(\sigma) + \sigma \cdot T_F + C_F$$

for some small constant c.

Remark 3.2 *The constant c is a small number which depends on the computational model. The advantage of A' (see Section 2) is defined over the random tape of A' and the random choice of $f \in F$.*

Proof: We extend the proof in [3] to deal with EMAC. The proof in [3] consists of two main parts.

1. They start by checking the possibility of distinguishing a random function in $\mathcal{R}_{\ell * \to \ell}$ from a random function in $CBC - MAC^{\mathcal{R}_{\ell \to \ell}}$. Intuitively, this can be thought of as using CBC-MAC with the best block cipher: a random function in $\mathcal{R}_{\ell \to \ell}$. They show that even a computationally unbounded adversary cannot gain too much advantage in this case.

2. Second, they use the first step to show that if an adversary can distinguish the family $\mathcal{R}_{\ell * \to \ell}$ from $CBC - MAC^F$, then this adversary can be used to build another adversary that breaks the underlying family F with comparable resources and with comparable advantage.

We follow these steps for EMAC. The main difficulty (in both cases) is in the proof of Step (1).

3.1 The information theoretic case

We start with Step (1), i.e., we show that it is hard to distinguish $EMAC^{\mathcal{R}_{\ell \to \ell}}$ from the of random functions $\mathcal{R}_{\ell * \to \ell}$, even if the distinguisher is computationally unbounded.

Lemma 3.3 *Let $\ell \geq 1$ be the block length. Let A be a probabilistic oracle Turing machine (an adversary) that makes queries to either a random instance of $EMAC^{R_{\ell \to \ell}}$ or to a random instance of $R_{\ell * \to \ell}$. Suppose Machine A makes queries $X_1, \ldots, X_n$ the cumulative length of which, i.e., $\sigma = \sum_{i=1}^{n} |X_i|$, is less then $2^{(\ell+1)/2}$, then Machine A has advantage at most $\sigma^2 \cdot 2 \cdot 2^{-\ell}$.*

of a block is larger than the length of a key and that $f(b)$ $(b = 0, 1)$ is truncated to get the keys. Otherwise, one may take more values of $f(0)$, $f(1)$, $f(2)\ldots$) It is easy to show that if F is a pseudo random family of functions then this additional step does not foil the security of the system, and we ignore this point in the sequel.

We would like to make a remark about the empty string. Our assumption is that a user does not authenticate the empty string. Thus, during the proof we do not consider the empty string to be a legal query. Like in Subsection 2.1 above, for the sake of the analysis we assume that $f_1^{(0)}(\epsilon)$ is the zero string 0^ℓ, and thus $EMAC_{f_1,f_2}(\epsilon)$ is defined to be $f_2(0^\ell)$. It is not clear what a real system will do with the empty string. Our recommendation is not to allow such authentication. However, if a system behaves according to the definition in this paragraph, then the proof in this paper can be easily modified to cover such a system as well.

3 Encrypted CBC MAC is secure

In this section, we would like to show that the security of the family $EMAC^F$ is implied by the security of the family F. In the main theorem of [3] (where all messages had the same length) and also in case the message space is prefix-free, the adversary is limited in his queries, and this is used in the proof. Here, the adversary is allowed to make any query. (Actually, we will not allow more then a reasonable (i.e., an exponential) number of queries, and we will not allow queries of exponential length.)

To state the theorem we would like to define what it means that the adversary succeeds in breaking the authentication scheme. Our definition is parametrized to allow quantitative analysis later. In the sequel, we denote by ℓ the length of the blocks in the block cipher. Namely, the functions in the family F are from $\{0,1\}^\ell$ into $\{0,1\}^\ell$. Also, we denote by $|X|$ the number of blocks in the string X.

Definition 3.1 *Let $F \subseteq \mathcal{R}_{\ell\to\ell}$ be a family of functions. Let A be a probabilistic oracle Turing machine (the adversary). Consider the following stochastic experiment. First, two functions f_1, f_2 are selected according to the distribution on the family F. Then, A gets to see the $EMAC_{f_1,f_2}$ authentication of messages $X_1, X_2, \ldots, X_{n-1}$ which it chooses adaptively, i.e., X_i is chosen based upon the values $EMAC_{f_1,f_2}(X_j)$, $j = 1, \ldots, i-1$, and upon the random tape of A. We say that an adversary A (ϵ, t, σ) succeeds in breaking the $EMAC^F$ scheme if the following three conditions hold:*

1. **Probability of success:** *With probability at least ϵ, over the choice of $f_1, f_2 \in F$ and over the random coins of Machine A, Machine A outputs $(X_n, EMAC_{f_1,f_2}(X_n))$ for X_n which is different from all the previous queries $X_1, \ldots, X_{n-1}$.*

2. **Time complexity:** *Machine A runs in time at most t.*

3. **Query complexity:** *The number of queries and their length satisfy: $\sum_{i=1}^{n} |X_i| \leq \sigma$. (Note that the length of the forgery also counts.)*

Let us also define T_F to be the time complexity of the block cipher F. Namely, T_F is the maximum over all $g \in F$ and all $\omega \in \{0,1\}^\ell$ of the time it takes to compute $g(\omega)$. Also, let we mean that for a family $F = (C, D)$ we pick a function in C according to the distribution D.

In this paper, we would like to check the ability of A to tell between a function chosen from one specific family C_1 and a function chosen at random from a second family of functions C_2. We denote the *advantage* of A in making this distinction by $advantage_A(C_1, C_2)$, and define it as:

$$advantage_A(C_1, C_2) \stackrel{\text{def}}{=} \left|\text{Prob}[A^{C_1} = 1] - \text{Prob}[A^{C_2} = 1]\right|.$$

The distribution is a random choice of a function in the family C_1 or C_2 according to the distribution defined for the family. (This notation follows [6].)

A message space Ω is a set of strings in $\{0, 1\}^*$. We say that a message space Ω is prefix-free if there are no two distinct strings $x_1, x_2 \in \Omega$ such that x_1 is a prefix of x_2.

Finally, we are going to talk about block ciphers and denote the block length by ℓ. We assume that this parameter is input to all machines discussed in this paper. In particular, each of the adversaries gets ℓ in its input. We also assume that the key length k is efficiently computable given ℓ. (Efficient here is in a liberal sense, i.e., polynomial in ℓ and not in the length of the (binary) representation of ℓ).

2.1 The CBC MAC

Given a block cipher which uses a random key $a \in \{0, 1\}^k$, we define a family of functions F which includes a function for each possible key in $\{0, 1\}^k$. Note that two different keys may indicate the same function. The corresponding distribution is a uniform random choice of a key in $\{0, 1\}^k$ and using the key to determine the function of the block cipher.

Given a family of functions F from $\{0, 1\}^\ell$ to $\{0, 1\}^\ell$ (denoted the underlying family of functions) the CBC MAC authentication scheme MAC^F is defined by choosing at random a function $f \in F$ (unknown to the adversary) according to the distribution of the family, and then the authentication of a message x of m blocks, i.e., $x = (x_1, x_2, \ldots, x_m)$ is defined as:

$$f^{(m)}(x_1, \ldots, x_m) = f(f(\ldots f(f(x_1) \oplus x_2) \oplus \ldots \oplus x_{m-1}) \oplus x_m).$$

It will be convenient to also define $f^{(0)}(\epsilon) = 0^\ell$ (for the empty string ϵ). Sometimes, we prefer not to specify the length of x, and then we will write $f^{(*)}(x)$, which means the same as above with m being the number of blocks in the input message x.

2.2 Encrypted CBC MAC

A variant of CBC MAC which we call $EMAC$ (encrypted CBC MAC) is defined as follows. Let F be a family of functions. Choose two functions $f_1, f_2 \in F$ independently according to the distribution of the family. On a message $X = (x_1, x_2, \ldots, x_m)$ as above, we define

$$EMAC_{f_1,f_2} = f_2(f_1^{(m)}(x)).$$

We denote by $EMAC^F$ the family of functions obtained by using EMAC with the family F. We remark that if only one secret function f is given instead of the pair f_1, f_2, one may use the strings $f(0)$ and $f(1)$ to specify the two functions f_1, f_2. (We assume that the length so that birthday attacks become infeasible. Our proof of security should be consider as complementing these birthday attacks. We remark that these attacks are the best known today for CBC MAC or for EMAC and that there is still a gap between the proven security of these authentication schemes and the known attacks.

1.4 Outline of the paper

In the following section we give the definitions and notations used throughout the paper. In Section 3 we show that EMAC is (almost) as secure as the underlying block cipher used, and in Section 4 we argue that it is secure to use CBC MAC on a prefix-free message space (given that the underlying family is secure).

2 Preliminaries

We (basically) use the notation of [3]. The set of all functions from $\{0,1\}^\ell$ to $\{0,1\}^\ell$ is called $\mathcal{R}_{\ell\to\ell}$. Also, let the (infinite) set of all functions from $(\{0,1\}^\ell)^*$ to $\{0,1\}^\ell$ be called $\mathcal{R}_{\ell^*\to\ell}$.

Let A be a probabilistic, oracle, Turing machine with access to an oracle f (think of f as an authentication function which A can access, or alternatively as a random function) then we denote by $\text{Prob}[A^f = 1]$ the probability that A outputs 1 when accessing the function f as an oracle. For a finite family of functions F, we denote by $\text{Prob}[A^F = 1]$ the probability that A, when accessing an oracle f which is randomly chosen from F outputs 1. We will also consider the infinite set of functions from $(\{0,1\}^\ell)^*$ to $\{0,1\}^\ell$. In this case, instead of thinking of $A^{\mathcal{R}_{\ell^*\to\ell}}$ as a machine which uses a random function in this infinite set, one may think of the oracle answering each question of the machine with a random string in $\{0,1\}^\ell$. Of course, if the same question is asked twice, the same answer will be given.

In many cases, we will consider a set of functions, and a random choice of a function in the set. We shall define a *family of functions* to be a pair $C = (S, D)$ where S is a set of functions and D is a corresponding distribution by which a function in the set is picked. The three families of functions that will be discussed with relation to the $EMAC$ scheme are

1. The family $EMAC^{\mathcal{R}_{\ell\to\ell}}$ is the set of functions $EMAC_{f_1,f_2}$ for all $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$. The corresponding distribution is a uniform and independent random choice of $f_1, f_2 \in \mathcal{R}_{\ell\to\ell}$.

2. The family $EMAC^F$ for a block cipher F is the set of functions $EMAC_{f_1,f_2}$ for all $f_1, f_2 \in F$. The corresponding distribution is a an independent random choice of two encryption functions $f_1, f_2 \in F$ according to the distribution of the cipher block F.

3. The family $\mathcal{R}_{\ell^*\to\ell}$ is the set of all functions in $\mathcal{R}_{\ell^*\to\ell}$ and the corresponding distribution is a uniform random choice of a function in the set. Similarly, the family $\mathcal{R}_{\ell\to\ell}$ is the set of all functions in $\mathcal{R}_{\ell\to\ell}$ and the corresponding distribution is a uniform random choice of a function in the set.

One may define the families $CBC - MAC^F$ and $CBC - MAC^{\mathcal{R}_{\ell\to\ell}}$ in a similar manner. We will sometimes abuse notations by refering to the family as the set of functions. When we say that we pick a function at random from a family F according to the distribution of F,

1.3 Related works

There are various approaches to authentication other than CBC MAC. Wegman and Carter [18] suggested to hash a message using an almost universal$_2$ family of hash functions and then encrypt it (using probabilistic encryption). Following that, efficient applications of this procedure were suggested by Krawczyk [8], Stinson [17], Shoup [16], and Rogaway [15]. See [15] for more details and references.

We also mention the work of Bellare, Guerin, and Rogaway [2], who suggested a new type of (provably secure) authentication based on performing exclusive-or's on encryptions of the input blocks. This allows parallelizability and incrementality.

Bellare, Canetti, and Krawczyk [1] introduce two methods for message authentication code: NMAC and HMAC. Both bear resemblance to our EMAC, but are also different. In the NMAC construction, they use two keys as in the EMAC, applying first some secure compression hash function on the message with one key, and then applying the same compression function with a second key on the of the previous iteration. A similar thing is done in the HMAC except for the use of only one key, together with a predetermined mask which is bit-wise exclusive-or-ed with the key. Their constructions and proofs are not applicable to the EMAC construction since the EMAC deals with an iteration that is not (weakly) collision resistant for non-fixed size inputs. This bad property of the CBC MAC foils the proof in [1] and therefore a proof is needed that the EMAC is indeed secure.

The ANSI standard X9.19 augments CBC-MAC using a DES cipher. The standard suggests using two keys like our EMAC. The first is used to get a CBC-MAC on the message using DES, then the second is used to *decrypt* the result with DES, and last, the first key is used again to encrypt the outcome. This was done primarily to prevent exhaustive key attacks, but has the same effect on variable-length message attacks that EMAC does. Our analysis may be augmented to argue for the security of this method as well. The construction of ANSI X9.19 involves an extra operation which, in view of our result, is not necessary if the underlying block cipher is secure (as a pseudo-random function family). To the best of our knowledge, there is no published rigorous proof of security for the ANSI X9.19 construction, which is different from the EMAC.

The construction of ANSI X9.19 involves an extra operation which, in view of our result, is not necessary if the underlying block cipher is secure (as a pseudo-random function family).

Independently of this work, Bellare and Rogaway [4] consider the case in which private keys are not allowed. They construct a mechanism to do that, which has a similar weakness when variable-length messages are hashed. They suggest the following mechanism to deal with this problem: Hash the message using one key (in their scenario the key is revealed to the adversary at a later stage) to get $h_{k_1}(X)$, and then append the length of X to $h_{k_1}(X)$ and hash them together using a different key k_2. This construction is closely related to our EMAC. It needs a bit more work in the second stage, but the proof of security becomes easier.

Last, let us mention the birthday attacks of Preneel and Oorschot [12]. They show that MAC's of a particular form can be broken if enough (and/or long enough) messages are authenticated so that a *collision* occurs, i.e., the breaker finds two different messages that get the same hash value. These attacks are applicable to the CBC MAC and to the EMAC as well. Of course, the security parameters of the EMAC (or CBC MAC) should be set encoded messages form a prefix-free set of messages. The proof that CBC MAC is secure when invoked on a prefix-free message space is an easy augmentation of the proof for fixed length messages given in [3].

Let us make an important point here. The security of the system guarantees only that the adversary cannot forge a legal message taken from the prefix-free space. The adversary is capable of forging messages that are not from the predetermined prefix-free message space and it must be made sure that the legal users accept only authentications on messages from the predetermined message space.[2]

We remark that making such assumptions on the behavior of the system is on one hand reasonable, but on the other hand, relies on the system being used as it was meant to be used. The security of the system would be compromised if the system is later modified to authenticate the concatenation of a few messages (treating it as if it was a single message to authenticate), thus loosing the prefix-free property. Therefore, we recommend using EMAC even though one more operation of f is needed per message.

1.2 The theoretical framework

We follow the approach suggested by Bellare, Kilian and Rogaway [3]. We describe it here briefly, and we refer the reader to their paper for details and motivation.

We would like to show that if the underlying block cipher is secure then a message authentication code is also secure. We call a block cipher secure if it is a pseudo-random function family with respect to efficient computation. Namely, consider the block cipher as a family of functions, so that each key determines a function in the family. Then we assume that this family is a pseudo-random family of functions. This approach to modeling the security of a block cipher was suggested by Luby and Rackoff [10, 11], and the notion of a family of pseudo random functions was suggested by Goldreich, Goldwasser and Micali [6].

We say that a MAC from $\left(\{0,1\}^\ell\right)^*$ to $\{0,1\}^\ell$ is secure if it resists existential forgery under adaptive message attack. This adopts the viewpoints of Goldwasser, Micali and Rivest [7] with regards to signature schemes. We shall follow [3] and actually show that the message authentication code is secure in a very strong sense, i.e., that the EMAC we suggest, or the application of CBC MAC on prefix-free message space, is actually a family of pseudo random functions.

Bellare, Kilian and Rogaway have also provided an *exact security* analysis. Namely, instead of arguing that if the underlying cipher block is robust against polynomial time attack then the authentication procedure is robust against polynomial time attack, they actually make a more exact statement. They state that if the authentication can be attacked by an adversary running in time t, making n queries to the authentication scheme, and achieving "advantage" ϵ (for definition of advantage see Section 2), then there exists an adversary which attacks the underlying cipher in time t', makes at most n' queries to the cipher, and achieves advantage ϵ', where t', n', and ϵ' are explicitly given as functions of n, t, ϵ. In this way, it is possible to study non-asymptotic ciphers schemes, such as DES (which is only defined on a block of 64 bits). We follow this approach.

[2] For example, upon seeing $t_0 = MAC_a(0)$ the adversary knows that $t_0 = MAC_a(0, t_0)$.

The function $f_a : \{0,1\}^\ell \to \{0,1\}^\ell$ is some underlying block cipher such as the Data Encryption Standard (DES) and a is the secret key. Thus, $f_a^{(m)}$ is a function that takes a message of $m \geq 1$ blocks (or $m\ell$ bits) and assigns it a tag of one block. The CBC MAC is an International Standard [9], widely used, especially with DES as the underlying block cipher.

It is well known that the use of the CBC MAC for variable length messages is not secure, and a few rules of thumb for the correct use of the CBC MAC are known by folklore. For example, it is easy to show that after examining a few authentications, an adversary that doesn't know the secret key can produce a valid authentication of a message that hasn't yet been authenticated.

Until recently, no solid theoretical ground was suggested to deal with the security of this method. The main interest is whether the security of the cipher block f (e.g. DES) implies the security of the CBC MAC $f^{(m)}$. Bellare, Kilian, and Rogaway [3] were the first to study this problem. They show that CBC MAC is secure when applied to messages of fixed length. They also show variants of CBC MAC that are secure for the case of variable length messages when the length of the message is known in advance, i.e., before the message is given to the authentication procedure.

1.1 This work

We study the case of real-time applications, in which the length of the message is not known in advance. These include many important uses such as fax, real-time speech transmission, real-time camera sources of video transmission, and other human-driven multimedia interactions. We study how to use the popular CBC MAC approach in this scenario.

First, we consider the following variant of CBC MAC to deal securely with the problem of variable (unknown) length messages. This variant was first suggested in [5]. We use the secret key a to produce two secret keys $a' = f_a(0)$ and $a'' = f_a(1)$. Using a' and a'' we define:

$$EMAC_{a',a''}(x) \stackrel{\text{def}}{=} f_{a''}(f_{a'}^{(m)}(x))$$

where m is the number of blocks in x. Namely, we first use a' to perform a CBC authentication of the message. This can be done block by block as they are input to the authentication procedure. Given $f_{a'}^{(m)}(x)$ we perform one more encryption and get $EMAC_{a',a''}(x)$. Note that for each block x_i we only use the encryption function once. The additional invocation of $f_{a''}$ is done only once at the end. Therefore, the efficiency of this authentication is virtually the same as the standard CBC MAC. We call this authentication *EMAC* (for encrypted CBC MAC).[1] We then provide a rigorous proof that this method is secure. Our proof is an extension of the proof in [3].

Next we argue that the cipher block chaining (CBC) message authentication code (MAC) is secure when applied to a prefix-free message space. If the part of the message which is authenticated includes the (usually hidden) "end of message" character, then this condition (of prefix-free messages) holds. More formally, suppose the message space is drawn from an alphabet of blocks which excludes a distinguished block $\perp$ and if we encode each authenticated message by appending the $\perp$ block to the end of the message, then we get that the

[1] Actually, a different method to deal with unknown lengths was suggested in [3]. Their method seems to have a flaw. See Appendix A for details.

CBC MAC for Real-Time Data Sources

Erez Petrank* Charles Rackoff†

Abstract

The Cipher Block Chaining (CBC) Message Authentication Code (MAC) is an authentication method which is widely used in practice. It is well known that the use of the CBC MAC for variable length messages is not secure, and a few rules of thumb for the correct use of the CBC MAC are known by folklore. The first rigorous proof of the security of CBC MAC, when used on fixed length messages, was given only recently by Bellare, Kilian and Rogaway [3]. They also suggested variants of CBC MAC that handle variable-length messages but in these variants the length of the message has to be known in advance (i.e., before the message is processed).

We study CBC authentication of real-time applications in which the length of the message is not known until the message ends, and furthermore, since the application is real-time, it is not possible to start processing the authentication only after the message ends.

We first consider a variant of CBC MAC, we call *encrypted CBC MAC* (EMAC) which handles messages of variable unknown lengths. Computing EMAC on a message is virtually as simple and as efficient as computing the standard CBC MAC on the message. We provide a rigorous proof that its security is implied by the security of the underlying block cipher. Next, we argue that the basic CBC MAC is secure when applied to a prefix-free message space. A message space can be made prefix-free by authenticating also the (usually hidden) last character which marks the end of the message.

Keywords: Message authentication, Real time, Cipher block chaining, Block ciphers.

1 Introduction

The Cipher Block Chaining (CBC) Message Authentication Code (MAC) is an authentication method which is widely used in practice. To authenticate a message $X = (x_1, x_2, \ldots, x_m)$ amongst parties who share the secret key a, the following tag is added to the message:

$$f_a^{(m)}(x) \stackrel{\text{def}}{=} f_a(f_a(\cdots f_a(f_a(x_1) \oplus x_2) \oplus \cdots \oplus x_{m-1}) \oplus x_m).$$

*Corresponding author. IBM Haifa Research Laboratory, Matam, Haifa 31905, Israel. . Part of this work was done while the author was visiting the University of Toronto, and the center of Discrete Math and Theoretical Computer Science (DIMACS). Email: erezp@haifa.vnet.ibm.com

†Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4. This work was supported in part by an operating grant from the Natural Sciences and Engineering Council of Canada, and by ITRC, an Ontario Centre of Excellence. Email: rackoff@cs.toronto.edu.