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 naive use of CBC MAC for variable length messages is not secure, and a few rules of thumb for the correct use of 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 present a variant of CBC MAC, called *double MAC* (DMAC) which handles messages of variable unknown lengths. Computing DMAC 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 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 Channing (CBC) Massage 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. DIMACS Center, P. O. Box 1179, Piscataway, NJ 08855-1179. Part of this work was done while the author was visiting the University of Toronto. Email: erez@dimacs.rutgers.edu

†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.

The function $f_a : \{0,1\}^\ell \to \{0,1\}^\ell$ is some underlying block cipher such as 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 [8], widely used, especially with the DES as the underlying cipher block function.

It is well known that the naive use of CBC MAC for variable length messages is not secure, and a few rules of thumb for the correct use of 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 block cipher function 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 suggest the following variant of CBC MAC to deal securely with the problem of variable (unknown) length messages. 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:

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

where m is the number of blocks in x. Namely, we first use a' to perform a regular 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 operation and get $DMAC_{a',a''}(x)$. Note that for each block x_i we only use the encryption function once. The additional invocation of $f_{a''}$ is done only once in the end. Therefore, the efficiency of this authentication is virtually the same as the standard CBC MAC. We call this authentication *DMAC* (for double 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 security of cipher block chaining (CBC) message authentication code (MAC) applied on prefix free message space is secure. 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. The proof of this is an easy augmentation of the proof for fixed length messages given in [3]. We remark that making such assumptions on the behavior

[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.

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 DMAC even though one more operation of f is needed per message.

1.2 A remark on a single message authentication

Sometimes a key is used only once for authentication. For example, when a more complicated cryptographic procedure is used to produce keys, one for each message, and then a simple and efficient cryptographic procedure is used with this key to authenticate one message. The key is then discarded.

We would like to remark that the use of CBC-MAC with a key that is used only once is a safe use of CBC-MAC. In [3] it is asserted that if the adversary only sees authentications of messages of a fixed length, then he "cannot" succeed in producing an authentication of a message of the same length that he hasn't already seen the authentication of. However, their proof can be easily extended to show that the adversary "cannot" produce authentications of messages on any length even different from the length of the messages of all his queries. Thus, seeing an authentication on one message cannot help the adversary authenticate any other message.

1.3 The theoretical framework

We follow the approach suggested by Bellare, Kilian and Rogaway [3]. We describe it here shortly, 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 security of a cipher scheme was suggested by Luby and Rackoff [9, 10], and the notion of a family of pseudo random functions was suggested by Goldreich, Goldwasser and Micali [5].

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 [6] 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 DMAC 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 q 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 q' queries to the cipher, and achieves advantage ϵ', where t', q', and ϵ' are explicitly given as functions of q, 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.

1.4 Related works

There are various approaches to authentication other than CBC MAC. Wegman and Carter [17] suggested to hash a message using 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 [7], Stinson [16], Shoup [15], and Rogaway [14]. See [14] 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 resemblence to our DMAC, but are also different. In the NMAC construction, they use two keys as in the DMAC, 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 DMAC construction since the DMAC 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 DMAC is indeed secure.

The ANSI standard X9.19 augmentats CBC-MAC using a DES cipher. The standard suggests using two keys like our DMAC. 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 DMAC does. The construction of ANSI X9.19 involves an extra operation which, in view of our result, is not really necessary. 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 DMAC.

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 DMAC. 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 [11]. They show that any MAC 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 not avoided in CBC MAC or in our DMAC. Of-course, the security parameters of the DMAC (or CBC MAC) should be set so that birthday attacks would be infeasible. In light of these attacks, our theorem should be interpreted as saying that nothing more than these attacks is possible if the underlying block cipher is secure. In other words, our theorem should be thought of: "If you can do better than birthday attacks then you can break the underlying block cipher scheme". This is formalized in our main Theorem (see Theorem 1 in Section 3).

1.5 Outline of the paper

In the following section we give the definitions and notations used throughout the paper. In Section 3 we show that DMAC 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 random access machine (RAM) 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 $\mathrm{Prob}[A^f = 1]$ the probability that A outputs 1 when accessing the function f as an oracle. For a finite family of functions F, we denote by $\mathrm{Prob}[A^F = 1]$ the probability that A, when accessing an oracle f which is uniformly 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 this paper, we would like to check the ability of A to tell between a function chosen from a specific family F and a random function on the same domain. We denote the *advantage* of A in making this distinction by $advantage_A(F)$. Let F be a family of functions from $\{0,1\}^\ell$ to $\{0,1\}^\ell$, then

$$advantage_A(F) \stackrel{\text{def}}{=} \left|\mathrm{Prob}[A^F = 1] - \mathrm{Prob}[A^{\mathcal{R}_{\ell\to\ell}} = 1]\right|.$$

(This notation follows [5].) We also use a similar definition for a family of functions F from $(\{0,1\}^\ell)^*$ to $\{0,1\}^\ell$. In this case we define

$$advantage_A(F) \stackrel{\text{def}}{=} \left|\mathrm{Prob}[A^F = 1] - \mathrm{Prob}[A^{\mathcal{R}_{\ell^*\to\ell}} = 1]\right|.$$

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 strings $x_1, x_2 \in \Omega$ such that x_1 is a prefix of x_2.

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 imply the same function. When we say that a function $f \in F$ is chosen uniformly at random, we mean that a key is chosen at random in $\{0,1\}^k$ and the implied function is used.

Given a set 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 uniformly at random a function $f \in F$ (unknown to the adversary) 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 Double MAC

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

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

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. 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. We denote the use of DMAC with a family F by $DMAC^F$.

3 Double MAC is secure

In this section, we would like to show that the security of $DMAC^F$ 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.)

Let us state the theorem. 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.

Theorem 1 *Let $F \subseteq \mathcal{R}^{\ell \to \ell}$ be a family of functions. Let f_1, f_2 be two functions selected uniformly and independently in F. Suppose there is an adversary that gets to see the $DMAC_{f_1,f_2}$ authentication of messages $X_1, X_2, \ldots, X_{q-1}$ which he chooses adaptively, and then with probability ϵ is able to forge X_q, i.e., output $X_q, DMAC(X_q)$ for X_q which is different from all the previous queries $X_1, \ldots, X_{q-1}$. Suppose that the adversary runs in time t and that the total length of the queries satisfy $\sum_{i=1}^{q} |X_i| \leq 2^{(\ell+1)/2}$. Then, there exists an adversary A for distinguishing a uniformly chosen function in F from a random function* *of $\{0,1\}^\ell$ to $\{0,1\}^\ell$ with the following properties. Adversary A achieves an advantage of at least $\epsilon/2 - 2^{-\ell} \cdot (\sum_{i=1}^{q} |X_i|)^2 - 2^{-\ell}/2$, after making at most $\sum_{i=1}^{q} |X_i|$ queries and working in time at most $t + c\ell \sum_{i=1}^{q} |X_i|$, for some small constant c.*

The constant c is a small number which depends on the computational model.

Proof: We extend the proof in [3] to deal with this case. The proof in [3] is composed of three stages.

1. They start by considering the information theoretic case in which the function f is really chosen at random between all functions from $\{0,1\}^\ell$ to $\{0,1\}^\ell$. In this case, the adversary is allowed unlimited computational power and we only check how many queries he makes and how long the messages in each query are. In this case, they show that the CBC-MAC (operated on fixed length messages) behaves almost as a random function. Namely, the adversary cannot gain high advantage in distinguishing CBC-MAC form a random function.

2. Second, they argue that if the family F, from which the function f is chosen, is only a family of pseudo-random functions (and not really a random function as in step (1)), then CBC-MAC applied on this family results in a family of pseudo random functions as well.

3. Last, they show that if CBC MAC is a family of pseudo-random function then it is also a secure authentication scheme.

The arguments given in [3] for Steps (2) and (3) are standard, except for the fact that they computed the exact security of the reductions involved. The exact calculations, given in [3], apply to our case too (with easy modifications in the presentations of the parameters). We will spend most of this section on extending the first step so that it matches the DMAC scenario.

3.1 The information theoretic case

Starting with Step (1) in the proof, i.e., the information theoretic case, we are going to show the following lemma.

Lemma 3.1 *Let $\ell \geq 1$ be the block length. Suppose the adversary makes queries $X_1, \ldots, X_n$ the accumulative length of which, i.e., $\sum_{i=1}^{n} |X_i|$, is less then $2^{(\ell+1)/2}$, then the adversary cannot gain advantage greater than $\left(\sum_{i=1}^{n} |X_i|\right)^2 \cdot 2 \cdot 2^{-\ell}$.*

In order to prove the lemma, we use the following steps. First, we show that when there are no collisions, i.e., the underlying function f_1 is evaluated on a different string each time, 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 his view (even when using his 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 is not limited in computational power we may assume he is deterministic. Therefore the distribution space is determined only by the random selection of f_1 and f_2 amongst all functions from $\{0,1\}^\ell$ to $\{0,1\}^\ell$. We will actually prove a bit more then stated, in the sense that we are going to let the adversary see all the authentications of all the prefixes of his queries. Namely, when the adversary makes a query $X = (x_1, x_2, \ldots, x_m)$, then he not only gets the value of DMAC on X but also he gets all the DMAC's of all the prefixes of X. Therefore, he gets: $f_2(f_1(x_1)),\ f_2(f_1(f_1(x_1) \oplus x_2)),\ \ldots, f_2(f_1(\cdots f_1(f_1(x_1) \oplus x_2) \oplus \cdots \oplus x_m))$.

In the first lemma (see Lemma 3.2 below) we assert that if there is no collision on the queries done so far (including all sub-queries) then any f_1 that does not cause collision on this set of queries is equally likely to be the one used, given the view of the adversary. Here, we fix an f_1 and thus the distribution is only on the choice of f_2. In the second lemma (see Lemma 3.3 below) we show that this implies an almost uniform behavior of the values on which f_2 is applied. (Recall that in DMAC we first perform a CBC MAC authentication with f_1 and then we invoke f_2 once on the value got.) Using this, we show in the third lemma (see Lemma 3.4 below) that the probability that a new query will cause a collision, given the view of the adversary so far, is small. Intuitively, any specific query made will only cause a collision to a small fraction of all the possible f_1's. Having this, we prove that the lemma holds. Let us start with the first lemma.

Lemma 3.2 *Suppose the adversary makes queries $X_1, X_2, \ldots, X_n$, and suppose that so far, there was no collision in the values output by DMAC on the all queries and sub-queries. Then all $f_1 \subseteq \mathcal{R}^{\ell\to\ell}$ which do not cause collision on this set of queries and sub-queries, are equally likely to be used in the DMAC, when the function f_2 is uniformly chosen in the set $\mathcal{R}^{\ell\to\ell}$.*

Proof: Fix a $g \in \mathcal{R}^{\ell\to\ell}$, which doesn't cause collision on the given set of queries and all the implied sub-queries. Denote the values of CBC MAC, using the function g on all the queries and sub-queries by $\alpha_1, \alpha_2, \ldots, \alpha_m$. If $f_1 = g$ is used for the DMAC, then the output given to the adversary 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, otherwise the output of DMAC would have collisions.

Now let's ask what is the probability that $f_1 = g$ given the adversary's view: $(X_1, X_2, \ldots, X_n, \beta_1, \ldots, \beta_m)$. The values β_i, $i = 1, \ldots, m$ are the outputs given to the adversary on his set of queries. This includes the DMAC values on all the queries and sub-queries. Using Bayes rule:

$$\begin{aligned}
&\mathrm{Prob}_{f_1,f_2}[f_1 = g|\mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)] \\
&\quad = \mathrm{Prob}_{f_1,f_2}[\mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)|f_1 = g] \cdot \\
&\qquad\qquad \cdot \frac{\mathrm{Prob}_{f_1,f_2}[f_1 = g]}{\mathrm{Prob}_{f_1,f_2}[\mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)]} \\
&\quad = \mathrm{Prob}_{f_2}[\wedge_{i=1}^m f_2(\alpha_i) = \beta_i] \cdot \frac{\mathrm{Prob}_{f_1,f_2}[f_1 = g]}{\mathrm{Prob}_{f_1,f_2}[\mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)]}
\end{aligned}$$

The first term, assuming that all α_i's are distinct is exactly $2^{-\ell m}$ since we fix m different values of the function f_2. The dominator is exactly $1/|\mathcal{R}^{\ell\to\ell}|$ since f_1 is picked 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 collision free, and we are done. ∎

Recall that DMAC involves an intermediate computation of $f_1^{(*)}$ on the query X, and then a final computation of $f_2(f_1^{(*)}(X))$. In the following lemma, we assert that the intermediate values are "hidden" even if the final values of the DMAC are given. Furthermore, we will show that the exclusive or of two intermediate values are also "hidden". We formalize this in the following lemma.

Lemma 3.3 *Let $X_1, \ldots, X_n$ be the queries made so far, let $Y_1, \ldots, Y_m$ be all the sub-queries implied by them (including the X_i's). Also, let $m^2/4 + m - 1 \leq 2^\ell/2$ and let $\beta_1, \ldots, \beta_m$ be the DMAC values given to the adversary in response to his queries, i.e., $\beta_i = DMAC_{f_1,f_2}(Y_i)$. Suppose the β_i's are all distinct (i.e., there is no collision), then for any string $\alpha \in \{0,1\}^\ell$ it holds that*

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

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_i) = \alpha | \mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)] \leq 2 \cdot 2^{-\ell}$$

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

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_i) \oplus f_1^{(*)}(Y_k) = \alpha | \mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)] \leq 2 \cdot 2^{-\ell}$$

Proof: 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 all distinct. Namely, the function f_1 must also be collision free on these queries. Let us also go one step inside the calculation of the cipher block chaining $f_1^{(*)}(Y_i)$ and recall that the last step in this calculation is always an application of f_1 on some string gotten from previous step. So let γ_i be this string. (If $Y_i = (y_1, y_2, \ldots, y_t)$ then $\gamma_i \stackrel{\mathrm{def}}{=} f_1^{(t-1)}(y_1, \ldots, y_{t-1}) \oplus y_t$). Clearly, $\alpha_i = f_1(\gamma_i)$, $1 \leq i \leq m$. Since there are no collisions, we get that all γ_i's must be distinct too.

Note that the β_i's are random variables dependent on f_1 and f_2 and they are given to the adversary, while the α_i's and the γ_i's are random variables that depend only on f_1 and they are not known to the adversary. (An exception is when Y_i is of length one block, and then γ_i is actually Y_i which is determined by the adversary.) We would like to determine the probability that $\alpha_i = \alpha$ for a string α and an index $1 \leq i \leq m$, or the probability that $\alpha_i \oplus \alpha_k = \alpha$, given the view of the adversary so far. To do that, we are going to show that even if we fix the values of all the α_j's except α_i, then the value of $f_1^{(*)}(\alpha_i)$ is almost uniform in $\{0,1\}^\ell$ given the view of the adversary. This implies also that the value of the exclusive or $\alpha_i \oplus \alpha_k$ is almost uniform. From this we conclude that also when we do not fix the values of the α_j's, it still holds that $f_1^{(*)}(\alpha_i)$ is still almost uniform, and also that the value of $f_1^{(*)}(\alpha_i) \oplus f_1^{(*)}(\alpha_k)$ is almost uniform.

Fix all values α_j for all $j \neq i$ such that all these values are distinct and different from the given string α. We shall show that given any such fixing of the α_j's, the probability that $\alpha_i = \alpha$ is at most $2 \cdot 2^{-\ell}$. Thus, this bound on the probability also hold when the α_j's are not fixed, but are rather random variables depending on f_1.

By Lemma 3.2, all collision free f_1's are equally likely given the output $\beta_1, \ldots, \beta_m$. Only a subset of these f_1's agree with the new constraint put through the values α_j $1 \leq j \leq m$, $j \neq i$, and we actually would like to know what the probability of $f_1(Y_i) = \alpha$ is, given this constraint. Thus, the distribution we have is a uniformly chosen function which is collision free, and which satisfies $f_1^{(*)}(Y_j) = \alpha_j$ for all $1 \le j \le m$, $j \ne i$. Note that each function is chosen with equal probability originally, and thus also each function that satisfies the new constraint is equally likely to be chosen. Also, note that the constraint may be written as $f_1(\gamma_j) = \alpha_j$ for all $j = 1, \ldots, i-1, i+1, \ldots, m$. Since γ_i is different from all the γ_j above, and since f_1 is chosen between all collision free functions, we get that the value of $f_1(\gamma_i)$ is uniformly chosen between all values that do not cause a collision for f_1. Let's determine how many strings like that exist.

There are 2^ℓ possible strings to be output, but out of them we have already determined $m-1$ to be output on different inputs, and thus are not allowed. Also, setting $f_1(\gamma_i)$ to be something specific, determines the values of γ_k for all Y_k which are direct extensions of Y_i. Namely, if $Y_i = (y_1, \ldots, y_t)$ then any Y_k of the form $Y_k = (y_1, \ldots, y_t, y_{t+1})$ has its γ_k determined as $f_1(\gamma_i) \oplus y_{t+1}$. So if there are s direct extension of Y_j, then for each of them $m-s$ strings are forbidden, because their inputs should not collide with any input of the other $m-s$ sub-queries which are not direct extensions of Y_j. So the number of forbidden string that follows from this argument is $s(m-s) \le m^2/4$. To summarize, the number of allowed strings is at least $2^\ell - m^2/4 - m - 1 \ge 2^\ell/2$. Since all of them are equally likely to appear when f_1 is chosen as above, we get that the probability that $f_1(Y_i) = \alpha$, given that f_1 is collision free and consistent with $f^{(*)}(Y_j) = \alpha_j$ for $j = 1, 2, \ldots, m$, $j \ne i$, is at most $2 \cdot 2^{-\ell}$ and we are done. ■

Building on this lemma, we can go on and prove that the probability that the prover will manage to find collision given that there was no collision so far is small. Here we shall again assume that the prover sees the DMAC values on all the sub-queries in addition to seeing the DMAC values on the queries themselves. So we are going to ask what is the probability that the prover can get a collision in any extension of the sub-queries done so far. By an extension, we mean adding one block to a query made before. We claim that the probability of a collision, given that there was no collision in the past is small.

We are going to assume that the number of outputs shown to the adversary is smaller than $\sqrt{2^\ell/2}$. Note that otherwise, there is a good chance that the adversary encounters a collision even if he just randomly selects his queries (Recall that ℓ is the block size).

Lemma 3.4 *Let $Y_1, Y_2, \ldots, Y_m$ be all the sub-queries (and queries) made so far and let $\beta_i = DMAC(Y_i)$ for $i = 1, 2, \ldots, m$. Suppose the β_i's are distinct and that $m^2/4 + m - 1 < 2^l/2$. Then, for any string $\omega \in \{0,1\}^\ell$ it holds that*

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

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_j) = f_1^{(*)}(Y_k\omega) | \mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)] \le 3 \cdot 2^{-\ell}.$$

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

$$\mathrm{Prob}_{f_1,f_2}[f_1^{(*)}(Y_j) = f_1^{(*)}(\omega) | \mathrm{View} = (X_1, \ldots, X_n, \beta_1, \ldots, \beta_m)] \le 3 \cdot 2^{-\ell}.$$

Proof: We assume in proving the first part of the lemma that $Y_k\omega$ is not a query made before. We assume the same for ω in the proof of the second part of the lemma. If this was not the case, then the probability of collision would have been zero, since we know that all previous queries are collision free.

We start with the first part of the lemma. One possible case is that Y_j is composed of more than one block, i.e., $|Y_j| > 1$. Denote the blocks of Y_j by $Y_j = (Y_j^1, Y_j^2, \ldots, Y_j^t)$. The event that we are interested in is $f_1^{(*)}(Y_j) = f_1^{(*)}(Y_k\omega)$ (given the view of the adversary) which can be written as

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

One possibility of collision is that the inputs to f_1 on each side of the equation are different, but f_1 maps them both to the same value. If f_1 was computed on the value $f_1^{(*)}(Y_k) \oplus \omega$ for some other query Y_i $(i \neq k)$, then since there are no collisions, then the probability that it would collide with Y_k is 0. So we assume that f_1 was not yet computed on the value $f_1^{(*)}(Y_k) \oplus \omega$. Since all non-colliding f_1's are equally likely given the view so far, then any value of f_1 on the string $f_1^{(*)}(Y_k) \oplus \omega$ appears with the same probability given the current view. Therefore, the probability of forming a collision is at most $2^{-\ell}$.

The other possibility that the event in Equation (1) holds is that the inputs to f_1 collide. Namely,

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

Note that $(Y_j^1, Y_j^2, \ldots, Y_j^{t-1})$ is also a query, since it is a sub-query of another query. Thus, we may apply the second Part of Lemma 3.3 to get that the probability of this event is at most $2 \cdot 2^{-\ell}$. To summarize this case (i.e., $|Y_j| > 1$), we get that the probability of forming a collision is at most $3 \cdot 2^{-\ell}$.

If, on the other hand, Y_j has one block, then we ask what is the probability that

$$f_1(Y_j) = f_1(f_1^{(*)}(Y_k) \oplus \omega). \tag{2}$$

Again, either f_1 is applied on a new value, i.e., $f_1^{(*)}(Y_k) \oplus \omega$ such that this collision occurs, and this happens with probability $2^{-\ell}$, or it holds that

$$f_1^{(*)}(Y_k) = Y_j \oplus \omega.$$

Again, by Lemma 3.3, this happens with probability at most $2 \cdot 2^{-\ell}$ and thus in this case (i.e., Y_j has one block) the probability of a new collision is also at most $3 \cdot 2^{-\ell}$.

Now let us proceed with Part (2) of the lemma. Here we consider the event $f_1^{(*)}(Y_j) = f_1^{(*)}(\omega)$ (given the view of the adversary) which can be written as

$$f_1^{(*)}(Y_j) = f_1(\omega). \tag{3}$$

Again, we partition the analysis according to the number of blocks in Y_j. If Y_j contains one block, then the event in Equation (3) can be written as $f_1(Y_j) = f_1(\omega)$. Recall that ω is not a copy of a query made so far, thus $Y_j \neq \omega$. Also, since f_1 is collision free on all the previous queries, then f_1 was never evaluated on ω. Therefore, any value is equally likely to be $f_1(\omega)$ and we get $f_1(Y_j) = f_1(\omega)$ with probability at most $2^{-\ell}$.

If $Y_j = (Y_j^1, \ldots, Y_j^t)$ for $t > 1$, then the event in Equation (3) can be written as $f_1(f_1^{(*)}(Y_j^1, \ldots, Y_j^{t-1}) \oplus Y_j^t) = f_1(\omega)$. This happens either if the inputs to f_1 are equal, i.e.,

$f_1^{(*)}(Y_j^1, \ldots, Y_j^{t-1}) = Y_j^t \oplus \omega$, or if the inputs are different but f_1 makes them collide. The first happens with probability at most $2 \cdot 2^{-\ell}$ by Lemma 3.3, and the second happens with probability at most $2^{-\ell}$ since f_1 has not been evaluated on ω before. Thus, we are done with Lemma 3.4. ■

Based on this lemma, we may conclude that the probability that a collision occurs after the adversary makes the queries $X_1, X_2, \ldots, X_n$ is at most $\left(\sum_{i=1}^{n} |X_n|\right)^2 \cdot 4 \cdot 2^{-\ell}$.

Corollary 3.5 *Suppose the adversary makes the queries $X_1, \ldots, X_n$, and let $Y_1, \ldots, Y_m$ be all the sub-queries implied by these queries (including the queries themselves), suppose also that $m^2/4 + m - 1 < 2^l/2$. Then the probability that a collision occurs on these queries 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. The order of the sub-queries for the purpose of the induction, is a sequence of sub-queries such that if a sub-query $(y_1, \ldots, y_t)$ appears at location i in the sequence, then the sub-query $(y_1, \ldots, y_{t-1})$ appears at location $j < i$ in the sequence. Clearly there exists such an order. Now, the probability that a collision occurs after the first query is made is zero. Then, the probability that a collision occurs when the $i+1$'st query is made given that no collision occurred so far is at most $i \cdot 4 \cdot 2^{-\ell}$. The reason is as follows. By Lemma 3.4, the new query would cause a collision on the $f_1^{(*)}$ values with any of the queries $Y_1, Y_2, \ldots, Y_i$ with probability at most $3 \cdot 2^{-\ell}$. The probability that applying f_2 on two distinct values would cause a collision is at most $2^{-\ell}$. So the probability that any collision will occur while making the $i+1$'st query is at most $i \cdot 4 \cdot 2^{-\ell}$.

Summing up over all $i = 1, 2, \ldots, m$, we get that the overall probability is at most

$$\sum_{i=1}^{m} i \cdot 4 \cdot 2^{-\ell} \leq \left(\sum_{i=1}^{n} |X_n|\right)^2 \cdot 2 \cdot 2^{-\ell}$$

and we are done with the proof of the corollary. ■

Let us now finish the proof of Lemma 3.1. First, we note that if the queries of the adversary are collision free then the view of the adversary is completely random. Let f_1 be any collision free function on the adversary queries, then what the adversary actually sees is the output of f_2 on m different inputs. As f_2 is chosen at random, this view of the adversary behaves exactly like a random function on his queries. So here the adversary can gain no advantage (no matter what his strategy is). On the other hand, if the adversary encounters a collision, then he might be able to use it to distinguish between the DMAC and a random function. In this case, we make a worst case assumption that the adversary always succeeds. So it remains to use Corollary 3.5 and see that the advantage of the adversary does not exceed $\left(\sum_{i=1}^{n} |X_n|\right)^2 \cdot 2 \cdot 2^{-\ell}$. as needed for Lemma 3.1, and we are done.

3.2 The computational case

Reducing DMAC to the underlying family of functions involves a standard reduction whose exact analysis was done in [3]. The only difference in our case is that we get a distinguisher which distinguishes a randomly chosen pair of functions $f_1, f_2 \in F$ from a randomly chosen pair of random functions $g_1, g_2 \in \mathcal{R}_{\ell \to \ell}$. Our goal is to show the existence of a distinguisher for a single function. Standard hybrid arguments imply that if there exists a distinguisher for a pair which achieves advantage ϵ then there exists another distinguisher for a single function which achieves advantage at least $\epsilon/2$. This smaller advantage is embedded in Lemma 3.6 below.

Note that if we actually want to build this distinguisher (rather than just to state its existence) then we can choose at random one of the two implied testers and use it on the given (as an oracle) function h. This will give a distinguisher with advantage $\epsilon/2$ as needed.

Let us state our final lemma.

Lemma 3.6 *Suppose there exists an algorithm A which makes $q-1$ queries $X_1, \ldots, X_{q-1}$ to $DMAC^{\mathcal{F}}$ and then with probability ϵ produces a query X_q together with $DMAC(X_q)$. Suppose also that A runs in time t and that the sum of lengths of the queries satisfy $\sum_{i=1}^{q} |X_i| \leq 2^{(\ell+1)/2}$. Then there exists an algorithm U that distinguishes the family F from a randomly chosen function of $\{0,1\}^\ell$ to $\{0,1\}^\ell$. The algorithm U runs in time $t + c\ell \sum_{i=1}^{q} |X_i|$ for some small constant c, makes at most $\sum_{i=1}^{q} |X_i|$ queries and achieves an advantage of at least $\epsilon/2 - (\sum_{i=1}^{q} |X_i|)^2 \cdot 2^{-\ell} - 2^{-\ell}/2$.*

This lemma is actually Theorem 1, and we are done.

4 Prefix free message space guarantees security

In this section we argue 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 is authenticated, then we never authenticate a prefix or an extension of this message. This situation is always the case if the part of the message which is authenticated actually includes the last character which marks the end of the message.

After querying about the authentications of prefix free messages, the adversary has to output an authentication of a new message which has not been queried about. The message that the adversary outputs is not required to satisfy the prefix free demand. It may be a prefix or an extension of one of the queries. Still, the adversary has low probability in succeeding to construct such an authentication.

Some notations (following [3]): Let the CBC MAC be used with a family F of functions from $\{0,1\}^\ell$ to $\{0,1\}^\ell$. The success of an algorithm to break the family F is measured by its advantage of distinguishing a random function (from $\{0,1\}^\ell$ to $\{0,1\}^\ell$) from a function chosen uniformly at random in F. The success of an algorithm that tries to break the authentication procedure is measured by the probability that it can forge an authentication of a message whose authentication has not been given to him. The resources that the breaking algorithm use are its running time, and the number of queries that he makes to the function or to the authentication procedure.

Theorem 2 *Suppose there is an adversary A which with probability ϵ succeeds in breaking CBC MAC with an underlying family F. To do that, A examines $q-1$ authentications of messages X_i, $i = 1, \ldots, q-1$, which are prefix free. After working time t, he manages with* *probability at least ϵ to produces a new message X_q which was not questioned before, together with the correct authentication code of X_q. Suppose that $\sum_{i=1}^{q} |X_i| \leq 2^{(\ell+1)/2}$. Then there exists an algorithm U which breaks the family F with advantage $\epsilon' = \epsilon - 6(\sum_{i=1}^{q} |X_i|)^2 2^{-\ell} - 2^{-\ell}$ runs in time $t + c\ell \sum_{i=1}^{q} |X_i|$ (for a small constant c) and makes at most $\sum_{i=1}^{q} |X_i|$ queries.*

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 briefly mark the main differences. The difference is in the main part of the proof - the information theoretic case. The reductions from the information theoretic case to the computational case and the reduction from breaking the MAC as a family of pseudo random functions to breaking the MAC as an authentication scheme remain the same (with the appropriate change of parameters).

In what follows we assume that the reader is familiar with the proof and notations of the original proof of [3]. To augment the proof of the information theoretic case, we redefine *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 game tree is not extended after a border node. Now, the main change in the argument is in the proof of Lemma 3.4 of [3]. In this lemma, it is asserted that if there is no collision in the first n queries then there is little chance of a collision in the next query. Here the fact that the adversary doesn't get to see non-border nodes is used. If a collision occurs, then a "new" node collides with some "old" node X_t whose father's value is not given, and therefore is distributed close to uniformly. In our case this fact remains valid, since if a node X_t exists in the tree, then by the prefix free property, the value of his father is not given to the adversary.

5 Acknowledgment

We thank Mihir Bellare, Joe Kilian and Moti Yung for helpful discussions.

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, N. Koblitz ed, Springer-Verlag, 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, D. Coppersmith, Ed., pp. 340-358, 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, Y. Desmedt, Ed., pp. 340-358, 1994.

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

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

[6] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attack. In *SIAM Journal on Computing* , 17(2):281-308, April 1988.

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

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

[9] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. In *SIAM Journal on Computing* , 17(2), April 1988.

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

[11] 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, D. Coppersmith ed., Springer-Verlag*, 1996.

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

[13] R. Rivest, The MD5 message digest algorithm. IETF RFC-1321, 1992.

[14] P. Rogaway. Bucket Hashing and its Application to Fast Message Authentication. In *Advances in Cryptology – Crypto 95 Proceedings*, pages 29-42, 1995.

[15] V. Shoup. On fast and Provably Secure Message Authentication based on Universal Hashing. In *Advances in Cryptology – Crypto 96 Proceedings*, 1996.

[16] D. Stinson. Universal Hashing and Authentication Codes. In *Designs, Codes and Cryptography*, vol. 4, 369-380, 1994.

[17] M. Wegman and L. Carter. New Hash Functions and their use in Authentication and Set Equality. In *J. of Computer and System Sciences* 22, 265-279, 1981.

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:

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

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 forth 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. He gets the pair $\left(f_{a'}^{(1)}(0), f_{a''}^{(2)}(1, f_{a'}^{(1)}(0))\right)$. Define $\beta_0 \stackrel{\text{def}}{=} f_{a'}^{(1)}(0) = f_{a'}(0)$.

2. The adversary asks to see the authentication of the message 1. He gets the pair $\left(f_{a'}^{(1)}(1), f_{a''}^{(2)}(1, f_{a'}^{(1)}(1))\right)$. Define $\beta_1 \stackrel{\text{def}}{=} f_{a'}^{(1)}(1) = f_{a'}(1)$.

3. The adversary asks to see the authentication of the message $(0, \beta_0)$. He gets the pair $\left(f_{a'}^{(2)}(0, \beta_0), f_{a''}^{(2)}(2, f_{a'}^{(2)}(0, \beta_0))\right)$.

4. The adversary outputs the message $(1, \beta_1)$ with the authentication got in the previous query, i.e., $\left(f_{a'}^{(2)}(0, \beta_0), f_{a''}^{(2)}(2, f_{a'}^{(2)}(0, \beta_0))\right)$.

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

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

and we are done.