PRELIMINARY DRAFT

Key-Collision Attacks Against RMAC

TADAYOSHI KOHNO[*]

October 21, 2002

Abstract

In [Bih02] Biham introduced the notion of a key-collision attack. In this work we show how to extend Biham's key-collision technique to attack RMAC [JJV02] (we do so in the form of a "trade-off attack" against multiple users). We also present related-key attacks against the RMAC. These attacks work by exploiting a "unique" interaction between different instances of the mode's keys (and not some related-key weakness of the underlying block cipher). Because of the resource requirements of these attacks, these attacks are mainly of theoretical interest.

Keywords: RMAC, key-collision attacks, related-key attacks, multi-user setting.

[*]Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-mail: `tkohno@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/tkohno`. Supported by a National Defense Science and Engineering Graduate Fellowship.

Contents

1 Introduction

Jaulmes, Joux, and Valette's RMAC construction [JJV02] is a new randomized message authentication scheme. Similar to Petrank and Rackoff's DMAC construction [PR97] and Black and Rogaway's ECBC construction [BR00], the RMAC construction is a CBC-MAC variant in which an input message is first MACed with standard CBC-MAC and then the resulting intermediate value is enciphered with one additional block cipher application. Rather than using a fixed key for the last block cipher application (as DMAC and ECBC do), RMAC enciphers the last block with a randomly chosen (but related) key. Consequently, RMAC directly exposes the underlying block cipher to related-key attacks [Bih93]. We are interested in attacks that do not exploit some related-key weakness of the underlying block cipher, but some property of the RMAC mode itself.

The observation we make is that if two users each tag a large number of messages, then with relatively high probability they will each tag a message with the same randomly chosen final key. When this occurs we say that the two users' keys "interact." This key interaction enables a Biham-style key-collision attack [Bih02] (which we present as a "trade-off" attack against multiple users). For purposes of this discussion, assume that the key length of RMAC's underlying block cipher is equal to the underlying block cipher's block length l. The RMAC tagging algorithm uses two keys for a total key length of $\alpha = 2l$; the RMAC tag length is also α.

Table 1 describes the resources for our trade-off attack in more detail. In the standard setting (an adversary attacking one user) there is an attack against RMAC requiring the user to tag $O(2^{\alpha/2})$ messages of the attacker's choice. For our trade-off attack, we require $O(2^{\alpha/4})$ users to each tag $O(2^{\alpha/4})$ messages of the attacker's choice. After the users do this, the attacker will be able to forge a message from one of those users. The resources required for our trade-off attack (with respect to the number of users attacked and the number of messages tagged per user) are between the number of users necessary to obtain a total key collision, $O(2^{\alpha/2})$, and the resources necessary to mount a standard attack against one user. This is why we call our attack a "trade-off attack:" the trade-off is between the number of users attacked and the number of messages an attacker must force each user to tag. As Table 1 shows, if an attacker tried to use the standard attack to attack $O(2^{\alpha/4})$ users, then the attacker would need each user to tag $O(2^{3\alpha/8})$ messages, which is more than for our trade-off attack. We argue that there may be situations where it is easier for an attacker to force $O(2^{\alpha/4})$ users to each tag $O(2^{\alpha/4})$ messages than for the attacker to force a single user to tag $O(2^{\alpha/2})$, or even $O(2^{3\alpha/8})$, messages.

The attacks we describe can be converted to a Biham-style key-collision attack [Bih02] against a single user by "simulating" a set of other users. In this case the attack requires $O(2^{\alpha/4})$ chosen message requests for that one user and $O(2^{3\alpha/4})$ offline computations. Details will appear in the full version of this paper.

2 RMAC

Terminology and notation. Formally, a message authentication scheme consists of three algorithms: a key generation algorithm, a tagging algorithm, and a verification algorithm. The key generation algorithm generates a random key. The tagging algorithm, on input a key and a message, outputs a tag (or MAC) for that message. The verification algorithm, on input a key, a message, and a candidate tag, returns accept if the candidate tag is a valid tag for the message and returns reject otherwise.

When presenting pseudocode, we use $\leftarrow$ to denote assignment from right to left, we use $\oplus$ to denote the XOR operation, and we use $\|$ to denote concatenation.

RMAC. We now describe the RMAC construction in more detail. The RMAC construction is parameterized by choice of an underlying block cipher F. Let k denote the block cipher's key length, let l denote the block cipher's block length, and let $F_K(B)$ denote the application of block cipher F on an l-bit block B with a k-bit key K. The RMAC algorithm uses a total of $\alpha = 2k$ bits of key and produces a tag of length $l + k$. Before using the RMAC algorithm, a user first picks two random k-bit keys K_1 and K_2.

Let $\mathsf{RMAC}_{K_1,K_2}(M)$ denote the application of the RMAC tagging algorithm on a message M using keys K_1 and K_2. We assume that the length of M is a multiple of the block size. Pseudocode for the RMAC tagging algorithm is presented below (see also Figure 1):

```
$\mathsf{RMAC}_{K_1,K_2}(M)$
```

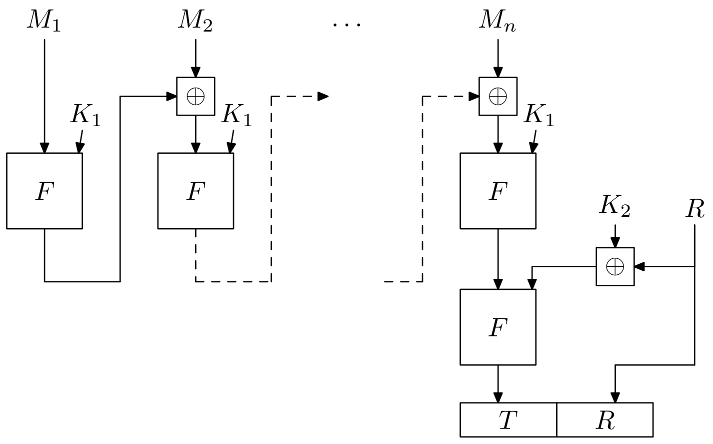

Figure 1: The RMAC tagging algorithm with keys K_1, K_2 on input $M_1 \| M_2 \| \cdots \| M_n$. The underlying block cipher is denoted F. The randomness R is chosen anew on each invocation. The resulting tag is $T \| R$.

```
Parse $M$ into $l$-bit blocks $M_1 \| M_2 \| \cdots \| M_n$
$C_0 \leftarrow 0$
For $i = 1$ to $n$ do
    $C_i \leftarrow F_{K_1}(M_i \oplus C_{i-1})$
$R \leftarrow$ random $k$-bit value
$T \leftarrow F_{K_2 \oplus R}(C_n)$
Return $(T, R)$
```

The RMAC verification algorithm, $\mathsf{RVER}_{K_1,K_2}(M,(T,R))$, returns accept if (T,R) is a valid tag for M and returns reject otherwise; the verification algorithm is defined in the natural way.

3 The Multi-User Attack Model

The standard notion of security for message authentication schemes is that of unforgeability [BKR94]. Formally, consider an adversary $\mathcal{A}$ attacking a message authentication scheme (such as RMAC). We represent the legitimate user of a message authentication scheme with a "tagging oracle" $\mathsf{RMAC}_{K_1,K_2}(\cdot)$ (where K_1, K_2 are randomly chosen keys). The tagging oracle is a "black box" that takes a message as input and uses the keys K_1, K_2 to compute and return the RMAC tag of the message. We give adversary $\mathcal{A}$ access to this tagging oracle so that she can obtain an RMAC tag for any message of her choice. (Note that $\mathcal{A}$ only has input-output access to the tagging oracle; she cannot access the keys K_1, K_2.) A query to the tagging oracle corresponds to $\mathcal{A}$ forcing a user to tag a message of $\mathcal{A}$'s choice. We also give $\mathcal{A}$ access to a verification oracle $\mathsf{RVER}_{K_1,K_2}(\cdot,\cdot)$. The verification oracle represents the original user's intended correspondent. Adversary $\mathcal{A}$ "wins" if she can find a message-tag pair $(M,(T,R))$ such that $\mathsf{RVER}_{K_1,K_2}(M,(T,R))$ returns accept and $\mathcal{A}$ never queried the oracle $\mathsf{RMAC}_{K_1,K_2}(\cdot)$ with input M. Intuitively, a scheme is unforgeable if the probability that any adversary with reasonable resources "wins" is "small."

The RMAC construction motivates a new definition of security for message authentication schemes. The motivation for this new definition will become clearer once we present our attacks. For now, it suffices to mention that because each application of the RMAC tagging algorithm uses a different key for the last block cipher application (i.e., $K_2 \oplus R$ where R is randomly chosen for each message), there is a chance that different users of RMAC (with different keys) will "interact." Consider, for example, two users using the RMAC tagging algorithm. The first user might use keys K_1 and K_2. The second user might use keys K_1' and K_2'. Even if $K_2 \neq K_2'$, there is a chance that the two users will use the same key for the last block cipher application (by picking random R, R' such that $K_2 \oplus R = K_2' \oplus R'$). Moreover, assuming each user tags enough messages, the probability that two users will interact in this way is much higher than the probability that two users will randomly select the same pair of master keys.

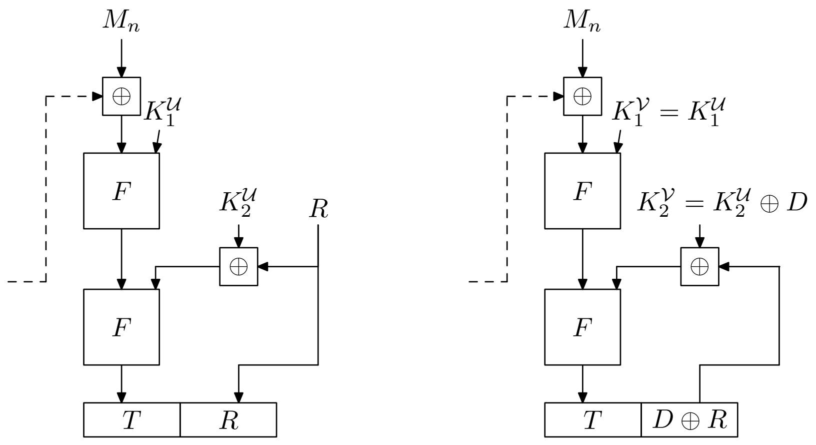

Figure 2: The RMAC known-difference related-key attack (Section 4). The figure on the left shows how user $\mathcal{U}$ constructs a tag (T, R) for a message M in response to an adversary's query. The figure on the right suggests why user $\mathcal{V}$ will accept $(T, D \oplus R)$ as a tag for message M.

Under the standard notion of unforgeability one cannot model attacks that play off the interaction between different users. This is because an adversary in the standard model is restricted to attacking one user at a time. In our new model we give an adversary access to multiple tagging oracles and corresponding verification oracles. The keys for each tagging oracle-verification oracle pair are chosen uniformly at random from the set of all possible keys. The adversary's goal is to make *any* of the verification oracles accept a message that was not tagged by its corresponding tagging oracle. This model corresponds nicely to an observation about real-world security requirements for message authentication schemes. First note that a message authentication scheme (or any cryptographic scheme) will often be used by many different users simultaneously, albeit each user will probably use different keys. Consider, for example, Internet users who use SSL. Clearly an adversary with access to *all* simultaneous SSL connections should not be able to efficiently break *any* user's SSL connection with high probability. Our model captures this requirement.

4 Known-Difference Related-Key Attack

We begin with a known-difference related-key attack against two users using the RMAC message authentication scheme. Let $\mathcal{U}$ and $\mathcal{V}$ denote the two users and let $K_1^{\mathcal{U}}, K_2^{\mathcal{U}}$ be user $\mathcal{U}$'s keys and let $K_1^{\mathcal{V}}, K_2^{\mathcal{V}}$ be user $\mathcal{V}$'s keys. Assume for this attack that $K_1^{\mathcal{U}} = K_1^{\mathcal{V}}$. Assume also that the adversary knows the difference D between $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$; i.e., the adversary knows $D = K_2^{\mathcal{U}} \oplus K_2^{\mathcal{V}}$. Although this attack may seem somewhat unrealistic (requiring that the two users share the same first key and that the attacker knows a priori the difference between the second keys), it motivates the attacks in the following subsections.

The adversary begins by having $\mathcal{U}$ tag a message M. Let (T, R) be the resulting tag; i.e., $(T, R) \leftarrow \mathsf{RMAC}_{K_1^{\mathcal{U}}, K_2^{\mathcal{U}}}(M)$. Since $K_2^{\mathcal{V}} = K_2^{\mathcal{U}} \oplus D$, user $\mathcal{V}$ will accept $(T, D \oplus R)$ as a tag for message M; i.e., $\mathsf{RVER}_{K_1^{\mathcal{V}}, K_2^{\mathcal{V}}}(M, (T, D \oplus R))$ will accept. See Figure 2. The query $(M, (T, D \oplus R))$ to the $\mathsf{RVER}_{K_1^{\mathcal{V}}, K_2^{\mathcal{V}}}(\cdot)$ oracle is considered a valid forgery because the adversary forces the user $\mathcal{V}$ to accept a message that was not tagged by one of the party's involved in $\mathcal{V}$'s session.

5 Unknown-Difference Related-Key Attack

Consider again the scenario in which an adversary is attacking the RMAC usage of two users $\mathcal{U}$ and $\mathcal{V}$. As before, let $K_1^{\mathcal{U}}, K_2^{\mathcal{U}}$ and $K_1^{\mathcal{V}}, K_2^{\mathcal{V}}$ respectively denote the two users' pairs of keys. Again assume that $K_1^{\mathcal{U}} = K_1^{\mathcal{V}}$. Unlike in Section 4, however, assume that the adversary does not know the difference D between the keys $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$.

To mount a variant of the attack in Section 4, an adversary must first learn the difference between the keys $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$. One way to learn this difference is shown in the following pseudocode. As discussed in Section 3, the following adversary is given access to the tagging oracles $\mathsf{RMAC}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(\cdot)$ and $\mathsf{RMAC}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(\cdot)$ and the corresponding verification oracles $\mathsf{RVER}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(\cdot,\cdot)$ and $\mathsf{RVER}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(\cdot,\cdot)$.

```
Adversary
    $M, M' \leftarrow$ any two distinct messages
    For $i = 1$ to $2^{k/2}$ do        // have $\mathcal{U}, \mathcal{V}$ tag $M$ $2^{k/2}$ times
        $(T_i^{\mathcal{U}}, R_i^{\mathcal{U}}) \leftarrow \mathsf{RMAC}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(M)$
        $(T_i^{\mathcal{V}}, R_i^{\mathcal{V}}) \leftarrow \mathsf{RMAC}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(M)$
    For each pair of indices $i, j \leq 2^{k/2}$ such that $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$
        $(T, R) \leftarrow \mathsf{RMAC}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(M')$
        $\mathsf{RVER}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(M', (T, R_i^{\mathcal{U}} \oplus R_j^{\mathcal{V}} \oplus R))$        // forgery attempt
```

The above adversary begins by having both $\mathcal{U}$ and $\mathcal{V}$ repeatedly tag some message M. After each user generates approximately $2^{k/2}$ tags, we expect at least one collision between the last block cipher key used by $\mathcal{U}$ and the last block cipher key used by $\mathcal{V}$. That is, we expect to find two indices $i, j \leq 2^{k/2}$ such that $K_2^{\mathcal{U}} \oplus R_i^{\mathcal{U}} = K_2^{\mathcal{V}} \oplus R_j^{\mathcal{V}}$. The adversary detects this collision by looking for indices i, j such that $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$. When a collision $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$ occurs due to the above internal key collision, the adversary learns that the difference between the two keys $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$ is $R_i^{\mathcal{U}} \oplus R_j^{\mathcal{V}}$. The adversary can use its knowledge of this difference to mount the attack in Section 4.

Assuming that the underlying block cipher is a family of independent random functions, after $2^{k/2}$ tagging requests per user we expect approximately 2^{k-l} additional collisions $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$ at random (not due to the above internal key collision). Thus we expect that an adversary may have to perform 2^{k-l} forgery attempts before it successfully forges a message. Provided that the underlying block cipher's key size k is not much larger than its block size l, an adversary will succeed after only a few forgery attempts.

6 Trade-Off Attack

We shall now describe a trade-off attack against RMAC in which the attacker does not a priori know any information about the relationship between different users' keys. This attack is a "trade-off attack" because, under our multi-user setting, the attack trades-off the number of users the attacker must attack with the number of messages the attacker must force each user to MAC. This attack extends the attacks of Section 4 and Section 5.

Consider the adversary shown in the following pseudocode. For this attack, we assume the adversary has tagging and verification oracle access to $2^{k/2}$ different users (where each user's keys are independently chosen at random). Let $\mathsf{RMAC}_{K_1^u,K_2^u}(\cdot)$ and $\mathsf{RVER}_{K_1^u,K_2^u}(\cdot,\cdot)$ represent the uth user's tagging and verification oracles. As stated in Section 3, the adversary wins if it can force any of the users to accept a message which that user did not previously tag. To simplify the exposition, we present the attack in two phases; it should be clear that, if desired, the two phases can be interwoven.

```
Adversary
    Phase One:
        $M \leftarrow$ any message
        For $u = 1$ to $2^{k/2}$ do        // for each of $2^{k/2}$ users
            For $i = 1$ to $2^{k/2}$ do        // have user $u$ tag $M$ $2^{k/2}$ times
                $(T_i^u, R_i^u) \leftarrow \mathsf{RMAC}_{K_1^u,K_2^u}(M)$
    Phase Two:
        For each pair of distinct users $u, v$ and for each pair of indices $i, j$
        such that $T_i^u = T_j^v$
            $M' \leftarrow$ a message not previously tagged by user $v$
            $(T, R) \leftarrow \mathsf{RMAC}_{K_1^u,K_2^u}(M')$
            $\mathsf{RVER}_{K_1^v,K_2^v}(M', (T, R_i^u \oplus R_j^v \oplus R))$        // forgery attempt
```

	Number of Users	Messages per User
Standard Attack	1	$2^{\alpha/2}$
Standard Attack	$2^{\alpha/4}$	$2^{3\alpha/8}$
Trade-Off Attack	$2^{\alpha/4}$	$2^{\alpha/4}$
Total Key Collision	$2^{\alpha/2}$	—

Table 1: Comparison of attacks against RMAC; α is both RMAC's total key length and RMAC's total tag length.

The intuition behind the above attack is that if two users u and v collide on their first keys K_1^u and K_1^v, then the adversary will be able to mount the attack in Section 5. In more detail: given $2^{k/2}$ users, we expect two users u and v to share the same first key $K_1^u = K_1^v$. After tagging $2^{k/2}$ messages each, we expect to find two indices i and j such that $K_2^u \oplus R_i^u = K_2^v \oplus R_j^v$. We detect this collision by looking for users u, v and indices i, j such that $T_i^u = T_j^v$. For the attack as presented in the above pseudocode, we expect to find approximately one such collision.

Unfortunately, the signal-to-noise ratio of this attack (as compared to the attack in Section 5) is greatly reduced; we expect up to approximately $2^{2k-l} + 2^k$ collisions $T_i^u = T_j^j$ at random. Since a low signal-to-noise ratio is handled by brute forcing through the noise, a small signal-to-noise ratio only mean an inversely proportional large cost for the second phase of the attack. If we tolerate a cost of approximately 2^k oracle queries for the second phase of the attack (recalling that RMAC uses two keys for a total key length of $2k$), then the above attack works for k up to l.

Why is this a "trade-off attack?" The most interesting aspect of this attack is the unexpected interaction between users' keys in RMAC and the corresponding trade-off between the number of users attacked and the number of messages each user must tag. For simplicity of exposition, we let $\alpha = 2k$ and make the reasonable assumption that $k = l$. The value α is the total key length (and the total output length) of the RMAC message authentication scheme scheme.

Before discussing the trade-off, we first describe a more "standard" attack against RMAC. Consider an adversary attacking one user's use of RMAC. The adversary forces the user to repeatedly tag messages of the form $\$\|X$ where X is some fixed sequence of blocks and $\$$ represents a sequence of random blocks chosen anew for each of the adversary's chosen-plaintext requests. After the user tags on the order of $2^{\alpha/2}$ such messages, we expect a total collision between two α-bit tags generated by the user. This total collision implies an internal collision in the internal state of RMAC's CBC iteration. An adversary can then exploit the techniques from [PvO95] to forge a message: assume that $A\|X$ and $B\|X$ were the two messages that generated colliding tags. Then a tag generated for any message $A\|Y$, $Y \neq X$, will be a valid tag for $B\|Y$.

The requirements for the "standard" attack in the above paragraph are as follows: an adversary must be able to obtain oracle access to *one* user and the adversary must be able to force that user to tag approximately $2^{\alpha/2}$ messages of the attacker's choice. The attacker itself must exert on the order of $2^{\alpha/2}$ work. Now consider our "trade-off attack" against RMAC. The requirements for this attack are as follows: an adversary must be able to obtain access to $2^{\alpha/4}$ users and the adversary must be able to force all $2^{\alpha/4}$ users to tag $2^{\alpha/4}$ messages of the attacker's choice. As with the attack in the previous paragraph, the attacker itself must exert on the order of $2^{\alpha/2}$ work. The trade-off occurs because the attacker, rather than having to force one user to tag $2^{\alpha/2}$ messages, can force $2^{\alpha/4}$ users to tag $2^{\alpha/4}$ messages each. The resources required for our trade-off attack lie between the resources required to mount the standard attack in the above paragraph and the number of users we need before we expect a total key collision between two users.

To paint a more complete spectrum, we can also extend the "standard" attack to the multi-user setting as follows. Assume an adversary attacks $2^{\alpha/4}$ users and that the adversary has each user tag $2^{3\alpha/8}$ messages using the "standard" strategy. Then we expect to find a total tag collision in the tags generated by one of those users; i.e., the standard attack will succeed against one of the $2^{\alpha/4}$ users attacked. As Table 1 shows, when both the trade-off attack and the standard attack are applied in the multi-user setting against $2^{\alpha/4}$ users, the trade-off attack requires less chosen-plaintexts per user than the standard attack.

Why is this important? The existence of this trade-off attack raises several concerns about the RMAC design (and about any scheme that selects a new related key upon every invocation). In particular, we show that because RMAC uses a large number of keys, different RMAC users can “interact” in a very unusual way. While it is true that for any cryptographic scheme there is a slight probability that two users will randomly select the same set of keys, the problem with users accidentally using the same key is compounded when a cryptographic scheme (like RMAC) uses a different key upon each invocation. We also note that RMAC’s design makes it vulnerable to scheme-level related-key attacks (Sections 4–5) and potential related-key attacks against the underlying block cipher [Bih93].

7 Key-Collision Attacks

The trade-off attack in the previous section can be converted to a Biham-style key-collision attack [Bih02] against a single user in which the adversary forces that user to tag approximately $2^{\alpha/4}$ messages and the adversary performs approximately $2^{3\alpha/4}$ offline computations. Details will appear in the final version of this paper.

Acknowledgments

The author thanks Mihir Bellare, Chanathip Namprempre, Gregory Neven, Adriana Palacio, and David Wagner for helpful comments.

References

[Bih93] E. Biham. New types of cryptanalytic attacks using related keys. In T. Helleseth, editor, *Advances in Cryptology – Eurocrypt ’93*. Springer-Verlag, 1993.

[Bih02] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, 84, 2002.

[BKR94] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Y. Desmedt, editor, *Advances in Cryptology – Crypto ’94*. Springer-Verlag, 1994.

[BR00] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In M. Bellare, editor, *Advances in Cryptology – Crypto 2000*. Springer-Verlag, 2000.

[JJV02] É. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In *Fast Software Encryption 2002*. Springer-Verlag, 2002.

[PR97] E. Petrank and C. Rackoff. CBC MAC for real-time data sources, 1997. DIMACS Technical Report 97-26. Available at `http://dimacs.rutgers.edu/TechnicalReports/abstracts/1997/`.

[PvO95] B. Preneel and P. C. van Oorschot. MDx-MAC and building fast MACs from hash functions. In D. Coppersmith, editor, *Advances in Cryptology – Crypto ’95*. Springer-Verlag, 1995.