AVALANCHEv1.1[1]

Submitter: Basel Alomair

Designer: Basel Alomair
alomair@uw.edu

2014.04.7

[1]Summary of differences between AVALANCHEv1 and AVALANCHEv1.1 are as follows:
1. Fixing the input/output parameters to bring the cipher into compliance with the call for submission.
2. Explicit mention to the End-of-Message character that will be appended to the end of plaintext before processing.
3. Explicit mention to the key-recovery attack and its effect on the security bounds, along with a possible fix.

Chapter 1

Specification

1.1 Parameters

AVALANCHEv1.1 has three parameters: key length, public message number length, and tag length. Parameter space: Each parameter is an integer number of bytes. The key length is either 48 bytes (384 bits), 56 bytes (448 bits), or 64 bytes (512 bits). The public message number length depends on the key length. The tag length is 16 bytes.

1.2 Recommended parameter sets

Throughout the rest of the document, lengths will be measured in bytes and bits interchangeably.

1. Primary recommended parameter set AVALANCHEv1.1: 64-byte (512-bit) key, 20-byte (160-bit) public message number, 16-byte (128-bit) tag.
2. Secondary recommended parameter set AVALANCHEv1.1: 56-byte (448-bit) key, 16-byte (128-bit) public message number, 16-byte (128-bit) tag.
3. Third recommended parameter set AVALANCHEv1.1: 48-byte (384-bit) key, 10-byte (80-bit) public message number, 16-byte (128-bit) tag.

1.3 Authenticated encryption

The input to authenticated encryption is a tuple (P, A, λ, N, K), where P is a variable-length plaintext, A is a variable-length associated data, λ is the empty string for secret message number (i.e., the cipher does not support secret message number), N is a fixed-length public message number, and K is a fixed-length key. The number of bits in N is the public message length. The number of bits in K is the key length. Since AVALANCHEv1.1 uses two algorithms: an authenticated cipher (PCMAC of Chapter 8) for P and a MAC (RMAC of

Chapter 9) for A, K consists of two independent keys, one for each algorithm. Let $K = (K_P, K_A)$ be the AVALANCHEv1.1 key, where K_P is the key for the authenticated cipher to process P and K_A be the key for the MAC to process A. There is no secret message number. Although there are no requirements on how to choose the public message number, N, the cipher may lose its security if two messages with the same public message number are encrypted with the same key.[1] The maximum number of bits in P is dependent on the key length and the public message number length. In particular, PCMAC utilizes the use of a counter, c, that is of length $|c| = |K_P| - |N|$; the length of P is at most $128 \cdot (2^{|c|} - 1)$ bits. There is no constraint on the length of A.

The output of the authenticated cipher is a variable-length integrity aware ciphertext, C. C consists of two components: the encryption of the plaintext and the authentication tag of the plaintext, associated data, and public message number. That is, $C = C_P||T$, where C_P is the encryption of the plaintext, T is the authentication tag, and $||$ denotes the concatenation operation. C_P is obtained by encrypting P with the PCMAC mode of encryption of Chapter 8, so the number of bytes in C_P equals the number of bytes in P plus one cipher block, i.e., 16-bytes (see Chapter 8 for detailed description of PCMAC). The tag T is an authenticator of P, A, and N obtained by xoring the authentication tag of P and N (the output of PCMAC of Chapter 8) and the authentication tag of A (the output of RMAC of Chapter 9); the number of bytes in T is the tag length.

Details are as follows. The two algorithms constituting AVALANCHEv1.1 are the authenticated cipher (PCMAC) and the standalone MAC (RMAC). On input a tuple (P, A, λ, N, K), AVALANCHEv1.1 will call PCMAC with (P, N, K) to produce a ciphertext-tag pair (C_P, τ_P). Simultaneously, (A, K) will be passed to RMAC to produce an authentication tag τ_A. We assume that the input plaintext message consists of characters, each represented by a unique byte (e.g., an ASCII code). We also assume that a unique character is dedicated to represent the end of the plaintext (for instance, the byte '0000 0011' is a unique representation the End-of-Text (EOT) character in ASCII). When a plaintext message is passed to the algorithm, it will be appended by a unique End-of-Message (EOM) byte before processing (see Chapter 8 for details). RMAC is a universal hashing based MAC that requires the hashed image to be encrypted with a one-time pad. Since RMAC is used alongside PCMAC in AVALANCHEv1.1, τ_P will be used as the one-time pad to encrypt the hashed image of RMAC (as can be seen in the security analysis of Chapter 8, τ_P is practically a one-time pad). The resulting tag T will be the xor of τ_P and τ_A. That is, the output of AVALANCHEv1.1 will be $(C = C_P||\tau_P \oplus \tau_A)$.

[1] Another alternative is to make the public message number empty and generate a nonce internally in the AVALANCHEv1.1 algorithm.

Chapter 2

Security Goals

There is no secret message number. The cipher does not promise any integrity or confidentiality if the legitimate key holder uses the same public message number for two different messages under the same key.

The numbers in the table are actually on different scales. Integrity level is measured by the expected number of online forgery attempts for a successful forgery, while confidentiality is measured by the expected number of key guesses to find the secret key. Any successful forgery or successful key guess should be assumed to completely compromise confidentiality and integrity of all messages.

While there is no limit for the length of the associated data, Table 2.1 assumes legitimate key holder does not encrypt a plaintext exceeding the maximum size specified in Section 1.3 ($\approx 2^{103}$ bits for AVALANCHE256v1.1, $\approx 2^{71}$ bits for AVALANCHE192v1.1, and $\approx 2^{55}$ bits for AVALANCHE128v1.1).

Note that there are two different numbers in the confidentiality of plaintext row. The smaller confidentiality bound is due to the birthday-like key-recovery attack discussed in Appendix B, while the larger bound is included to point out the optional mitigation of such an attack by whitening plaintext blocks (by xoring them with a random number, as discussed in Appendix B).

Table 2.1: Security of different versions of AVALANCHEv1.1

Security goal	AVALANCHE128v1.1 Bits of security	AVALANCHE192v1.1 Bits of security	AVALANCHE256v1.1 Bits of security
Confidentiality of P	65 (128)*	97 (192)*	129 (256)*
Integrity of P	127	127	127
Integrity of A	120	120	120

Chapter 3

Security Analysis

The current version of AVALANCHE uses AES as the underlying blockcipher. Hence, security is based on the security of AES. In particular, as can be seen in the security analysis in Chapter 8, security is based on the property that the output of AES is indistinguishable from a random sequence. The detailed security analyses of PCMAC and RMAC are provided in Chapter 8 and Chapter 9, respectively.

Chapter 4

Features

The main advantage of this cipher is complete parallelization. That is, the entire cipher operation can be performed during a single AES call. Below is the advantages and disadvantages of AVALANCHEv1.1 compared to AES-GCM.

Advantages: As mentioned above, the main advantage of AVALANCHEv1.1 is total parallelizability. On the other hand, AES-GCM uses the well-known two-pass approach of Encrypt-then-MAC (EtM). That is, AES-GCM first encrypts the plaintext using the counter mode of encryption to produce the ciphertext; then, the ciphertext is authenticated using a standard universal hash-function family based MAC. Although MACing using universal hashing is quite fast, its time is non-negligible. AVALANCHEv1.1, unlike AES-GCM, is a single-pass authenticated cipher: the encryption and authentication are performed on the same round. The two algorithms constituting AVALANCHEv1.1 are PCMAC (Chapter 8) to process the plaintext, and RMAC (Chapter 9) to process the associated data. In PCMAC, all AES calls can be performed in parallel. In addition, unlike AES-GCM, there is no need to universally hash ciphertext blocks to compute the authentication tag, nor there is a ned to universally hash plaintext blocks before encryption as in the case of IAPM [12] and OCB [19]. The universal hashing is only used to authenticate the associated data in RMAC, which can also be performed in parallel with the blockcipher calls to encrypt the plaintext blocks in PCMAC. Consequently, the effective time of the entire AVALANCHEv1.1 operation is a single AES call.

Disadvantage: the main disadvantage of AVALANCHEv1.1 is that the key to each blockcipher is different. However, due to the parallelizable nature of AVALANCHEv1.1, key scheduling can be performed in parallel for each block. That is, typically, one will do key scheduling for AES in advanced then use the key for all blocks. In AVALANCHEv1.1, the same thing can be done, but for all blockciphers, which is the same as scheduling for one AES block assuming parallel computation is available.

Chapter 5

Design Rationale

The main purpose of AVALANCHE is high throughput. This is achieved via complete parallelization of authenticated encryption. As discussed in the previous chapter, AVALANCHE has the advantage over AES-GCM in that it is a single-pass authenticated cipher in which all blockcipher calls can be performed in parallel. Thus, the total effective time of AVALANCHE is one AES call.

The designer has not hidden any weaknesses in this cipher. All choices inside AES were amply explained during the AES competition. The security proof for AVALANCHEv1.1 rules out weaknesses outside AES.

Chapter 6

Intellectual Property

At present, there are no patents or patent applications for either PCMAC nor RMAC. If any of this information changes, the submitter will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

Chapter 7

Consent

The submitter hereby consents to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitter understands that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitter understands that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitter acknowledges that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter understands that if he disagrees with published analyses then he is expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitter understands that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Chapter 8

PCMAC

In this chapter, the detailed description of PCMAC (Parity Check MAC) is given. PCMAC is one of the core algorithms constituting AVALANCHEv1.1: it is called to produce an integrity aware ciphertext.

8.1 Notations and Preliminaries

For a binary string s, the length of s in bits is denoted by $|s|$. For any two strings a and b, $(a||b)$ denotes any operation that allows the reconstruction of a and b from $(a||b)$. When the lengths of a and b are known, the concatenation operation is an example of such operations. For a positive integer β, $\{0,1\}^\beta$ denotes a binary string of length β-bits, and $\{0,1\}^*$ denotes a binary string of arbitrary length. For a non-empty set $\mathcal{F}$, we denote by $f \xleftarrow{\$} \mathcal{F}$ the selection of a member of $\mathcal{F}$ uniformly at random and assigning it to f. Throughout the rest of the paper, random variables will be represented by bold font symbols, whereas the corresponding non-bold font symbols represent specific values that can be taken by these random variables.

8.1.1 blockciphers

Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions from $\mathcal{D}$ to $\mathcal{R}$ indexed by keys $\mathcal{K}$. We use $F_K(D)$ as shorthand for $F(K, D)$. F is a family of permutations (i.e. a blockcipher), if $\mathcal{D} = \mathcal{R}$ and $F_K(\cdot)$ is a permutation on $\mathcal{D}$ for each $K \in \mathcal{K}$. If F is a family of permutations, we use $F_K^{-1}(\cdot)$ to denote the inverse of $F_K(\cdot)$ and we use $F^{-1}(\cdot,\cdot)$ to denote the function that takes as input (K, D) and computes $F_K^{-1}(D)$.

We adopt the notion of security for blockciphers formalized in [2]. Let $\mathsf{Perm}(\mathcal{K}, \mathcal{D})$ denote the set of all possible blockciphers (permutations) with key space $\mathcal{K}$ and domain $\mathcal{D}$. Then, the notation $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K}, \mathcal{D})$ corresponds to selecting a random blockciphers. Given a family of functions $E : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ and a key $K \in \mathcal{K}$, define the related-key oracle $E_{RK(\cdot,K)}(\cdot)$ as an oracle that takes two arguments, a function $\phi : \mathcal{K} \to \mathcal{K}$ and an element $M \in \mathcal{D}$, and that returns $E_{\phi(K)}(M)$.

The function ϕ is the related-key-deriving (RKD) function or the key transformation function. Let Φ be a set of functions mapping $\mathcal{K}$ to $\mathcal{K}$. Then Φ is called the set of allowed RKD functions, or allowed key-transformations.

Let $E : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ be a family of functions and let Φ be a set of RKD functions over $\mathcal{K}$. Let $\mathcal{A}$ be an adversary with access to a related-key oracle, and restricted to queries of the form (ϕ, m) in which $\phi \in \Phi$ and $m \in \mathcal{D}$. Then,

$$\begin{aligned}\mathsf{Adv}^{\text{prp-rka}}_{\Phi,E}(\mathcal{A}) = &\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_{RK(\cdot,K)}(\cdot)} = 1\right] \\ &- \Pr\left[K \xleftarrow{\$} \mathcal{K}; \pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K}, \mathcal{D}) : \mathcal{A}^{\pi_{RK(\cdot,K)}(\cdot)} = 1\right] \qquad (8.1)\end{aligned}$$

denotes the prp-rka-advantage of $\mathcal{A}$ in distinguishing a random instance of E from a random permutation using Φ-restricted related keys. We say that E is a secure pseudorandom permutation (prp) against related-key-attacks (rka) if the prp-rka-advantages of all adversaries using reasonable resources is small. For simplicity, we do not differentiate between prp-security and sprp-security (a blockcipher is said to be strong pseudorandom permutation (sprp) if it is

indistinguishable from a random permutation even if the adversary is given an oracle access to the inverse function).

8.1.2 Security Model

To analyze the proposed system, we will use the standard model used to analyze authenticated encryption systems (see, e.g., [19, 13]). Let $\mathcal{E}$ be the underlying encryption algorithm. Depending on the mode of operation, $\mathcal{E}$ may or may not require the use of a nonce for encryption. If the encryption algorithm requires the use of nonces, the input to the algorithm is a nonce-message pair (N, M); otherwise, the input to the encryption algorithm is simply a plaintext message M. The signing oracle internally calls the encryption algorithm and outputs a ciphertext-tag pair. That is, given an encryption algorithm $\mathcal{E}$, on input a key K, and a nonce-message pair (N, M), the signing algorithm $\mathcal{S}_{\mathcal{E}}(K, N, M)$ outputs (c, τ), where c is the ciphertext corresponding to (N, M) and τ is the authentication tag.

Given the decryption algorithm $\mathcal{D}$ corresponding to $\mathcal{E}$, on input a key K, a nonce N, a ciphertext c, and an authentication tag τ, the verifying oracle $\mathcal{V}_{\mathcal{D}}$ outputs a bit, with 1 standing for accept and 0 for reject. We ask for a basic validity condition, namely that authentic tags are accepted with probability one. That is, if $(c, \tau) = \mathcal{S}_{\mathcal{E}}(K, N, M)$, it must be the case that $\mathcal{V}_{\mathcal{D}}(K, N, c, \tau) = 1$ for any encryption/decryption algorithms, key K, nonce N, ciphertext c, and tag τ.

The adversary is a probabilistic polynomial time algorithm, $\mathcal{A}$. The adversary is given oracle access to algorithms $\mathcal{S}_{\mathcal{E}}(K, \cdot, \cdot)$ and $\mathcal{V}_{\mathcal{D}}(K, \cdot, \cdot, \cdot)$ for a random but hidden choice of K. $\mathcal{A}$ can query $\mathcal{S}_{\mathcal{E}}$ to generate a ciphertext-tag pair for a none-message of its choice and ask the verifier $\mathcal{V}_{\mathcal{D}}$ to verify that (N, c, τ) is a valid tuple. Formally, $\mathcal{A}$'s attack on the scheme is described by the following experiment:

1. A random string is selected as the shared secret, K.
2. Suppose $\mathcal{A}$ makes a signing query (N, M). The oracle computes $(c, \tau) \leftarrow \mathcal{S}_{\mathcal{E}}(K, N, M)$, the ciphertext-tag pair, and returns it to $\mathcal{A}$.
3. Suppose $\mathcal{A}$ makes a verify query (N, c, τ). The oracle computes the decision $d = \mathcal{V}_{\mathcal{D}}(K, N, c, \tau)$ and returns it to $\mathcal{A}$.

$\mathcal{A}$ can query the signing oracle q number of times and record the outputs. $\mathcal{A}$ then stops and attempts its forgery.

An adversary is said to be nonce-respecting if she never repeats a nonce. That is, after calling $\mathcal{S}_{\mathcal{E}}$ on (N, M), the adversary never asks its oracle a query (N, M'), regardless of the oracle responses. We emphasize, however, that the nonce used in the forgery attempt may coincide with a nonce used in one of the adversary's queries to the signing oracle. The outcome of running the experiment in the presence of an adversary is used to define security. We say that $\mathcal{A}$ is successful if it is nonce-respecting and makes a verify query (N, c, τ) which

is accepted, for an (N, c, τ) tuple that has not been outputted by the signing oracle $\mathcal{S}_{\mathcal{E}}$.

8.2 PCMAC: Theoretical Construction

In this section, we give the theory behind what we call "Parity Check MAC (or PCMAC)" followed by formal security statements and proofs. Although the theoretical construction might not be of practical interest by itself, the main purpose of this section is to introduce, and better understand, the main ideas that will be used in the construction of the practical system in Section 8.3.

8.2.1 Scheme Description

Assume there exists an invertible function, f, that takes arbitrary-length inputs and produces true random outputs (i.e., an ideal cipher). Let M be the plaintext message to be authenticated and n be a security parameter agreed upon by legitimate users. Divide M into $k := \lceil \frac{|M|}{n} \rceil$ blocks, each of length n-bits, except possibly the k^{th} block. Let σ be the n-bit compressed image of M, evaluated as follows.

$$\sigma = \sum_{i=1}^{k} M[i] \pmod{2^n}, \tag{8.2}$$

where $M[i]$ denotes the i^{th} block of the message M. (For the rest of the paper, we overload $M[i]$ to denote both the binary string in the i^{th} block and its integer representation in a big-endian format; the distinction between the two representations will be omitted whenever it is clear from the context.)

Remark 1. *We note that one can replace the modular addition with the XOR operation without affecting the security of the scheme. Just like modular addition, the set of all possible n-bit strings with the XOR operation form a finite group, which is all that is needed for the security to hold (as can bee seen in the formal proofs to follow).*

Given a plaintext message, M, compute its compressed image according to equation (8.2) and generate a string, r, drawn uniformly at random from $\{0,1\}^n$ (for the rest of the paper, r will be referred to as the coin tosses of the signing algorithm). Using the invertible function, f, encrypt the pair (r, M) to get its corresponding ciphertext

$$c \xleftarrow{\$} f(r, M), \tag{8.3}$$

The authentication tag of M is simply

$$\tau = \sigma + r \pmod{2^n}. \tag{8.4}$$

The pair (c, τ) from equations (8.3) and (8.4) is then transmitted to the intended receiver.

Given c, the intended receiver inverts (decrypt) the ciphertext to obtain the plaintext message, M, and the coin tosses, r. The receiver then breaks M into its n-bit blocks (the $M[i]$'s), computes the modular summation $\sum_i M[i] + r \pmod{2^n}$, and authenticates the message if and only if the summation is congruent to the received tag τ. Formally, the following integrity check must be satisfied to validate the message

$$\tau \stackrel{?}{\equiv} \sum_{i=1}^{k} M[i] + r \pmod{2^n}, \tag{8.5}$$

where τ is the received tag while the $M[i]$'s and r are obtained by decrypting the ciphertext. In what follows, we show that this simple parity check is indeed secure, provided the function f is a true random permutation.

8.2.2 Theorem Statements and Proofs

We start by stating a general lemma; the proof can be found in Appendix A.

Lemma 1. *In the proposed scheme, authentication tags computed according to equation (8.4) are statistically independent of their corresponding plaintext messages. Furthermore, authentication tags corresponding to different messages are mutually independent.*

8.2.3 Authenticity of the Construction

Before we provide an upper bound on the probability of successful forgery, we give an informal discussion on how authenticating the plaintext (i.e., aiming for INT-PTXT) and the true randomness of the encryption algorithm will be used in the new security proofs. Recall that, in standard MAC algorithms, the security is modeled by the adversary's probability of producing a valid authentication tag for a transmitted message.

When computing the tag as a function of the plaintext in authenticated ciphers, however, MACs are fundamentally different. The intended receiver in a authenticated encryption (AE) system will receive a ciphertext-tag pair as opposed to message-tag pair. This implies that the adversary must come up with a ciphertext-tag pair that will be accepted as valid for the forgery attempt to succeed. Consequently, MACs in AE systems possess an advantage over standard MACs as they can benefit from the security of the coupled encryption algorithm (unless the tag is computed as a function of the ciphertext).

The main idea here is that modifying a single plaintext bit will result in changing every ciphertext bit with probability 1/2, due to the true randomness of the ideal cipher. Similarly, modifying a single ciphertext bit will result in changing every plaintext bit with probability 1/2. To give an illustrative example of how the true randomness of the ideal cipher will be utilized to come up with new security proofs, consider an adversary with the knowledge of a valid plaintext-ciphertext-tag tuple (M, c, τ). Assume the adversary is attempting to

forge a valid tag for a plaintext M' that is different than M in only a single bit. In such scenario, even though the adversary can easily predict the correct tag corresponding to M' (since the tag is a simple parity check of plaintext blocks), due to the randomness of the encryption function, the ciphertext corresponding to M' should be uncorrelated to the recorded c. Therefore, it is infeasible for the adversary to predict the correct c', the ciphertext corresponding to M' with a non-negligible probability. On the other hand, assume that the adversary is attempting to authenticate a ciphertext c' that is different than c in a single bit. Since the encryption is a true random function, the plaintext corresponding to c' should be uncorrelated to the recorded M. Therefore, it is infeasible for the adversary to predict the correct τ', the tag corresponding to c' with a non-negligible probability. What the example shows is that, in scenarios where the adversary can predict the correct tag, it is infeasible to predict the correct ciphertext; while in other scenarios in which the adversary can predict the ciphertext, it is infeasible to predict the correct tag.

In what follows we give a formal security treatment of the theoretical construction. Let f be the true random permutation used for encryption and define $\mathsf{Adv}^{\text{auth}}_{\text{AE}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathcal{S}_f(\cdot,\cdot)} \text{ forges}]$ to be $\mathcal{A}$'s advantage in breaking the authenticity of the authenticated cipher when given oracle access to the signing algorithm $\mathcal{S}_f$.

Theorem 1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a true random permutation used as an encryption algorithm and n be the security parameter. Let $\mathcal{A}$ be an adversary making q signing queries before attempting its forgery. Then, $\mathcal{A}$'s advantage of successful forgery against the scheme of Section 8.2.1 is at most $\mathsf{Adv}^{\text{auth}}_{\text{AE}}(\mathcal{A}) \leq 2^{1-n}$.*

Proof: For the rest of the proof, we will denote by M the concatenation of the coin tosses, r, and the plaintext message (i.e., r becomes the first n-bit of the plaintext message). When $q = 0$ it is rather straightforward. It follows directly from the fact that each value of the authentication tag is equally probable (see the proof of Lemma 1).

Now, assume $\mathcal{A}$ has made q signing queries and recorded the sequence

$$\mathsf{Seq} = \Big\{(M_1, c_1, \tau_1), \cdots, (M_q, c_q, \tau_q)\Big\}, \tag{8.6}$$

where M_i, c_i, τ_i are the message, ciphertext, and tag corresponding to the i^{th} signing query, respectively. $\mathcal{A}$ then calls the verify oracle with (c, τ), where $(c, \tau) \neq (c_i, \tau_i)$ for any $i = 1, \cdots, q$ since otherwise $\mathcal{A}$ does not win by definition. We aim to bound the probability that (c, τ) will be validated. Let M be the plaintext message corresponding to the decryption of c, the ciphertext in the forgery attempt. There are two possible strategies for forgery:

1. selective forgery: the adversary attempts to forge a valid ciphertext-tag pair corresponding to a specific plaintext of her choice,
2. existential forgery: the adversary attempts to authenticate a ciphertext-tag pair regardless of the corresponding plaintext (i.e., modify a recorded ciphertext-tag pair in a way undetected by the legitimate receiver).

Call the former $\mathsf{forgery_s}$ and the latter $\mathsf{forgery_e}$. Since the adversary either knows M ($\mathsf{forgery_s}$) or does not ($\mathsf{forgery_e}$), these two possible scenarios span the entire attack vector.

To bound the probability of $\mathsf{forgery_s}$, let $\mathcal{A}$ attempt to falsely authenticate a message, M, of its choice. Since the tag is simply the summation of message blocks, predicting the correct tag is trivial. However, since f is a true random permutation, the corresponding ciphertext is uniformly distributed over the range of f. Consequently, the probability of successful $\mathsf{forgery_s}$ is $2^{-|c|} \leq 2^{-n}$.

To bound the probability of $\mathsf{forgery_e}$, denote by $\mathsf{Collision}$ the event that $\sum_{j=1}^{k_1} M[j] \equiv \sum_{j=1}^{k_2} M_i[j] \pmod{2^n}$ for some $i \in \{1, 2, \cdots, q\}$, where $k_1 = \lceil \frac{|M|}{n} \rceil$, $k_2 = \lceil \frac{|M_i|}{n} \rceil$, and $M_i[j]$ denotes the j^{th} block of the i^{th} message. That is, the message corresponding to the ciphertext in the forgery attempt, M, collides with M_i, one of the recorded messages in the sequence of equation (8.6). Also, we use $\overline{\mathsf{Collision}}$ as the typical notation for the complement of $\mathsf{Collision}$.

Obviously, there are two possible scenarios here: either M will collide with one of the M_i's or it will not. Assume that M collides with M_i for an $i \in \{1, \cdots, q\}$. Then, $(c, \tau_i) \neq (c_i, \tau_i)$ will pass the integrity check. However, since f is a true random function, the probability of collision is

$$\Pr\Big[\mathsf{Collision}\Big] = \Pr\Big[\sum_{j=1}^{k_1} M[j] = \sum_{j=1}^{k_2} M_i[j]\Big] = 2^{-n}. \tag{8.7}$$

Assume now that M does not collide with any of the M_i's. If no collision has occurred, then the adversary's probability of successful forgery is bounded by the probability of predicting the plaintext message corresponding to c, thus predicting the correct tag. That is, similar to the probability of $\mathsf{forgery_s}$, since f is a true random permutation,

$$\Pr\Big[\mathsf{forgery_e}|\overline{\mathsf{Collision}}\Big] = 2^{-|M|} \leq 2^{-n}. \tag{8.8}$$

By equations (8.7) and (8.8), the probability of $\mathsf{forgery_e}$ can be bounded by

$$\begin{aligned}
\Pr\Big[\mathsf{forgery_e}\Big] &= \Pr\Big[\mathsf{forgery_e}|\mathsf{Collision}\Big] \cdot \Pr\Big[\mathsf{Collision}\Big] \\
&\quad + \Pr\Big[\mathsf{forgery_e}|\overline{\mathsf{Collision}}\Big] \cdot \Pr\Big[\overline{\mathsf{Collision}}\Big] \\
&\leq \Pr\Big[\mathsf{Collision}\Big] + \Pr\Big[\mathsf{forgery_e}|\overline{\mathsf{Collision}}\Big] \\
&= 2^{-n} + 2^{-n}.
\end{aligned}$$

Hence, $\max\Big\{\Pr\Big[\mathsf{forgery_s}\Big], \Pr\Big[\mathsf{forgery_e}\Big]\Big\} = 2^{1-n}$, is $\mathcal{A}$'s maximum advantage of successful forgery, and the theorem follows. ■

8.2.4 Privacy of the Construction

There are two pieces of information sent to the intended receiver, the tag and the ciphertext. Since both are functions of the plaintext message, we must

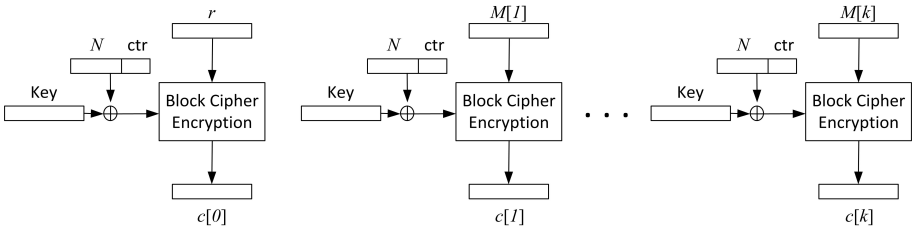

Figure 8.1: Illustration of a counter-based parallelizable mode of encryption. N is a nonce, ctr is a counter that increments each block, r is the coin tosses of the signing algorithm, $M[i]$ is the i^{th} plaintext block of a message consisting of k blocks, and $c[i]$ is the i^{th} ciphertext block. Note that the effective key corresponding to each block is different than all other blocks of the same message and all different messages (due to the use of the nonce and the counter). This is critical for the security of the construction.

show that neither one of them reveals secret information about the confidential message. We start by giving information-theoretic analysis of the privacy of the scheme given the observation of authentication tags.

Theorem 2. *Assume that the coin tosses of $\mathcal{S}_{\mathcal{E}}$, the r's, are shared secrets (e.g., delivered out of band). Then, no information about plaintext messages can be exposed by the observation of authentication tags computed according to equation (8.4).*

Proof: By Lemma 1, each tag is independent of its corresponding message. Therefore, by only observing a single authentication tag, the adversary cannot expose any information about the encrypted message. Assume now the adversary has observed the sequence $\mathsf{Seq} = \{\tau_1, \cdots, \tau_q\}$ of authentication tags. By Lemma 1, different authentication tags are mutually independent. Therefore, the observation of multiple tags gives the adversary no extra information than what a single tag gives individually, and the theorem follows. ■

Recall that the coin tosses, the r's, are delivered to the intended receiver by encrypting them with the underlying encryption algorithm. Theorem 2 implies that, as long as the adversary cannot extract secret information about the r's from observed ciphertexts, authentication tags do not reveal any private information about encrypted messages. In other words, the only way to attack the privacy of the composition is by attacking the security of the underlying encryption algorithm. To complete the analysis of the privacy of the system, it remains to prove the privacy of the underlying encryption algorithm. Different notions of privacy can be used to analyze encryption algorithms. Recent authenticated encryption schemes, however, use the notion of distinguishing the ciphertext from a random string of equal length to analyze the privacy of their schemes (see e.g., [19, 13]). For consistency, we will adopt the same notion of ciphertext randomness to analyze the privacy of our construction. Since the theoretical

construction of this section uses a true random permutation, it is intuitive that the randomness of ciphertexts (thus privacy) is preserved. However, since analysis of practical systems will be dependent on the used block cipher and mode of encryption, we defer the proof of privacy to Section 8.3, where we give an example of a practical construction of encryption algorithms that can be secure using the proposed method.

8.3 PCMAC: From Theory to Practice

In his 1949 foundational work [22], Shannon introduced the notions of confusion and diffusion as required properties for secure encryption algorithms. Confusion refers to the property that correlating the ciphertext and the key is infeasible for unauthorized observers, while diffusion refers to the property that correlating the ciphertext and the plaintext is infeasible for unauthorized observers. Confusion and diffusion have become basic requirements of secure blockciphers. The combination of the confusion and diffusion lead to what is known as the avalanche effect: changing a single plaintext bit or a single key bit results in changing half the ciphertext bits, on average (otherwise stated, each ciphertext bit will change with probability 1/2). Indeed, modern blockciphers have a very good avalanche effect [23].

Obviously, no simple parity check can be a secure MAC in the standard settings. The novelty of this work, however, is to utilize the avalanche effect of modern blockciphers and the special structure of AE systems when only INT-PTXT is sought to come up with such a secure parity check MAC.

Figure 8.1 depicts a parallelizable mode of encryption that can be used alongside the proposed PCMAC to construct a secure authenticated cipher. The mode of encryption of Figure 8.1 utilizes the use of a nonce (the public message number in the case of AVALANCHEv1.1) and a counter. The concatenation of the nonce and the counter is of length equal to that of the blockcipher key. Although there are no restrictions on how the user chooses the nonce, if the same nonce is used twice for two encryption operation under the same key the cipher may lose its security. The maximum length of plaintext messages that can be encrypted using the scheme of Figure 8.1 is exponential in the length of the counter ($2^{|\mathsf{ctr}|} - 1$ times the block size to be exact). As in the standard counter mode, both the nonce, N, and the ciphertext, c, are required for decryption.

8.3.1 PCMAC Description

Let M be the plaintext message to be authenticated and n be a security parameter agreed upon by legitimate users. (For ease of notation, we will assume that n is equal to the size of the blockcipher used to construct the encryption algorithm; we emphasize, however, that n can be different.) Append a unique End-of-Message character to the end of M and divide M into $k := \lceil \frac{|M|}{n} \rceil$ blocks, each of length n-bits, except possibly the k^{th} block. Append the k^{th} block of M with ℓ zeros, where $\ell = n - \Big(|M| \pmod n\Big)$, so that it becomes an n-bit long

string. Let σ be the n-bit compressed image of M, evaluated as follows.

$$\sigma = \sum_{i=1}^{k} M[i] \pmod{2^n}, \tag{8.9}$$

where $M[i]$ denotes the i^{th} block of the message M. Given a plaintext message M, compute its compressed image according to equation (8.9). Generate a string r, drawn uniformly at random from $\{0,1\}^n$, and prepend it to the message M. (As will be demonstrated in the security proof and performance discussion, r will be used in a novel way not only to provide the required randomness, but also to eliminate a blockcipher call that is essential for the security of existing schemes; it can also be used to whiten plaintext blocks before blockcipher encryption to mitigate the key-recovery attack discussed in Appendix B.) Using the underlying encryption algorithm $\mathcal{E}$, encrypt $r||M$ to get the corresponding ciphertext; i.e.,

$$c = \mathcal{E}(N, r||M), \tag{8.10}$$

where N is the public message number. The authentication tag of M is simply

$$\tau = \sigma + r \pmod{2^n}. \tag{8.11}$$

The pair (c, τ) from equations (8.10) and (8.11) is the output of PCMAC. Note that, if $|M| > n \cdot (2^{|\mathsf{ctr}|} - 1)$, the message is treated as multiple messages of lengths less than $n \cdot (2^{|\mathsf{ctr}|} - 1)$.

Given (N, c), the intended receiver decrypts the ciphertext to obtain the plaintext message, M, and the coin tosses, r. The receiver then breaks M into its n-bit blocks (the $M[i]$'s), computes the modular summation $\sum_i M[i] + r \pmod{2^n}$, and authenticates the message if and only if the summation is congruent to the received tag τ. Formally, the following integrity check must be satisfied to validate the message

$$\tau \stackrel{?}{\equiv} \sum_{i=1}^{k} M[i] + r \pmod{2^n}. \tag{8.12}$$

8.3.2 PCMAC Security

To start, we emphasize that we assume that the used blockcipher is secure against related-key attacks. Related-key attacks are attacks that can be launched against blockciphers using the knowledge of some mathematical relation between different blockcipher keys. Such attacks, while theoretically significant in reducing the security of certain blockciphers, require significant resources that make their impact on the security of practical systems very minor [17]. For instance, the most effective related-key attack on AES-256 requires 2^{119} data and time complexity [4]. While such a reduction, from 2^{256} for exhaustive search to 2^{119} for related-key, is theoretically significant, it could take up to 500 *million* years to perform such an attack [17]. Note further that the security against related-key attacks is highly dependent on the blockcipher. For instance, there is no related-key security implications against AES-128 [17].

Let $\mathsf{Adv}^{\text{prp-rka}}_{\Phi,\mathsf{BC}}(\mathcal{A})$ denote adversary's $\mathcal{A}$ advantage of breaking the pseudo-randomness of the blockcipher when launching related key attacks. We say the blockcipher is prp-rka secure if $\mathcal{A}$'s advantage in breaking the prp of the blockcipher using related keys is negligible. Note that prp-security implies that changing one plaintext bit will make every ciphertext bit change with probability 1/2 (the avalanche effect), and vice versa. The added security against related-key attacks implies that changing one bit of the key will make every ciphertext bit change with probability 1/2. The previous two assumptions are Shannon's diffusion and confusion properties which are valid assumptions in modern blockciphers [23]. Define $\mathsf{Adv}^{\text{auth}}_{\text{PCMAC}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathcal{S}_{\mathcal{E}}(\cdot,\cdot)} \text{ forges}]$ to be $\mathcal{A}$'s advantage in breaking the authenticity of PCMAC when given oracle access to the signing algorithm $\mathcal{S}_{\mathcal{E}}$, where $\mathcal{E}$ is the encryption algorithm of Figure 8.1. One gets the following.

Theorem 3. *Fix a blockcipher $\mathsf{BC} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ that is used to construct the mode of encryption of Figure 8.1. Let $\mathcal{A}$ be a nonce-respecting adversary that asks q signing queries totaling at most λ bits of payload before attempting its forgery. Then, there is an adversary $\mathcal{B}$ attacking the prp-rka-security of the blockcipher in which*

$$\mathsf{Adv}^{\text{auth}}_{\text{PCMAC}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{prp-rka}}_{\Phi,\mathsf{BC}}(\mathcal{B}) + 2^{1-n}. \tag{8.13}$$

Furthermore, adversary $\mathcal{B}$ takes the same time adversary $\mathcal{A}$ takes, minus the time of generating the coin tosses and the generation and authentication of tags, and makes at most $2\lceil \lambda/n \rceil + q + 1$ oracle queries.

Proof: Assume $\mathcal{A}$ has made q signing queries and recorded the sequence

$$\mathsf{Seq} = \Big\{(N_1, M_1, c_1, \tau_1), \cdots, (N_q, M_q, c_q, \tau_q)\Big\}, \tag{8.14}$$

where N_i, M_i, c_i, τ_i are the nonce, message, ciphertext, and tag corresponding to the i^{th} signing query, respectively. $\mathcal{A}$ then calls the verify oracle with (N, c, τ), where $(N, c, \tau) \neq (N_i, c_i, \tau_i)$ for any $i = 1, \cdots, q$ since otherwise $\mathcal{A}$ does not win by definition. We aim to bound the probability that (N, c, τ) will be validated. Let M be the plaintext message corresponding to the decryption of c, the ciphertext in the forgery attempt.

The coin tosses, r, plays a pivotal role in this proof. Denote by $c_i[1]$ the first block of ciphertext c_i. First, observe that, for a nonce-respecting adversary, the use of the counter prevents the concatenation of the nonce-counter of the first block of the mode of encryption of Figure 8.1 to be the same as the concatenation of a nonce-counter of any other block in any signing query. Therefore, if $c[1]$, the first ciphertext block of the attempted forgery, is not equal to any of the $c_i[1]$'s, given the prp security of the blockcipher, the resulting coin toss r (the decryption of $c[1]$) will be a random element of $\mathbb{Z}_{2^n}$. Consequently, by Lemma 2, the resulting tag is uniformly distributed over $\mathbb{Z}_{2^n}$ and, hence, the adversary's advantage of successful forgery is 2^{-n}.

Now, assume that $c[1]$ is equal to $c_i[1]$ for an $i \in \{1, \cdots, q\}$. There are two possible scenarios here: either $N = N_i$ or $N \neq N_i$. Let $N \neq N_i$. Then, by

the prp-security of the blockcipher, r is uniformly distributed over $\mathbb{Z}_{2^n}$, where r represents the coin tosses of the attempted forgery (i.e., the decryption of $c[1]$). Therefore, similar to the case in which $c[1] \neq c_i[1]$ for any $i \in \{1, \cdots, q\}$, the adversary's advantage when $c[1] = c_i[1]$ for some $i \in \{1, \cdots, q\}$ but $N \neq N_i$ is 2^{-n}.

Assume now that $c[1]$ is equal to $c_i[1]$ for an $i \in \{1, \cdots, q\}$ and let $N = N_i$. Then, $r = r_i$, where r represents the coin tosses of the attempted forgery and r_i represents the coin tosses of the i^{th} signing query. If $c = c_i$ only $(N, c, \tau) = (N_i, c_i, \tau_i)$ will be a validated and the adversary does not win by definition. It remains now to derive a bound on the probability of successful forgery when $(c[1], N) = (c_i[1], N_i)$ for some $i \in \{1, \cdots, q\}$ but $c \neq c_i$. We consider below the two possible scenarios: $|c| = |c_i|$ and $|c| \neq |c_i|$.

Let $c \neq c_i$ but $|c| = |c_i|$. Then, there must exist at least one ciphertext block $c[d]$ in which $c[d] \neq c_i[d]$. For a prp blockcipher, $M[d]$ cannot be correlated to $c[d]$. Furthermore, for a nonce-respecting adversary, $M[d]$ cannot be correlated to any of the plaintext blocks recorded in the sequence of equation (8.14). Therefore, the same proof of Theorem 1 can be used to show that probability of selective forgery $\mathsf{forgery_s} \leq 2^{-n}$ and probability of existential forgery $\mathsf{forgery_e} \leq 2^{1-n}$. The only difference here is that, since we are not dealing with a true random permutation, one will need access to a BC oracle in order to verify a selective forgery attempt, which translates into needing the pseudorandom permutation assumption (prp), and access to a BC^{-1} oracle in order to verify an existential forgery attempt, which translates into needing the strong pseudorandom permutation assumption (sprp).

Finally, let $c \neq c_i$ and $|c| \neq |c_i|$. Due to the unique end-of-message (EOM) character, if c is a truncation of any observed ciphertext then the decrypted message will not be valid one. Furthermore, for a nonce-respecting adversary, due to the uniqueness of the EOM character, any padding or removal of intermediate bits will make the decryption of the last ciphertext block a random element of $\mathbb{Z}_{2^n}$. Therefore, as in the case of $|c| = |c_i|$, the probability of selective forgery $\mathsf{forgery_s} \leq 2^{-n}$ and probability of existential forgery $\mathsf{forgery_e} \leq 2^{1-n}$, and the theorem follows. ∎

It remains now to prove the privacy of the system. The proof that authentication tags do not reveal any information about their corresponding ciphertexts is the same as in Section 8.2.4. To complete the analysis, we prove in what follows the privacy of the encryption algorithm. As mentioned earlier, there are different notions of privacy. Here, to be consistent with related schemes, such as OCB [19] and CWC [13], we choose to follow the strong notion of distinguishing a ciphertext from a random string of equal length.[1]

Consider an adversary $\mathcal{A}$ who has one of two types of oracles: a real encryption oracle and a fake encryption oracle. The real encryption oracle $\mathcal{E}_K(\cdot, \cdot)$

[1] Another standard privacy notion, used in [3, 12], is indistinguishability under chosen plaintext attacks (IND-CPA) [9], which captures the adversary's inability to distinguish the ciphertext corresponding to a pair of equal-length adversary-selected plaintexts. However, the notion of distinguishing the ciphertext from a random string of equal length implies IND-CPA while the converse is not true in general [19].

takes as input a pair (N, M) and returns a ciphertext $c \leftarrow \mathcal{E}_K(N, M)$. Assume that the length of the ciphertext depends only on the length of the plaintext, that is, $|c| = l(|M|)$. The fake encryption oracle, $\$(\cdot, \cdot)$, takes as input a pair (N, M) and returns a random string $c \xleftarrow{\$} \{0,1\}^{l(|M|)}$. Given adversary $\mathcal{A}$ and the encryption scheme $\mathcal{E}_K$, define

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathcal{E}}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K(\cdot,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} = 1\right]$$

to be $\mathcal{A}$'s advantage of breaking the privacy of the encryption algorithm. One gets the following.

Theorem 4. *Let $\mathcal{E}$ be the encryption algorithm of Figure 8.1 and let BC be the blockcipher used to construct $\mathcal{E}$. Then given a nonce-respecting adversary, $\mathcal{A}$, against $\mathcal{E}$, one can construct an adversary $\mathcal{B}$ against BC such that*

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathcal{E}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{prp-rka}}_{\Phi,\mathsf{BC}}(\mathcal{B}).$$

Furthermore, the experiment for $\mathcal{B}$ takes the same time as the experiment for $\mathcal{A}$ and, if $\mathcal{A}$ makes at most q_e oracle queries totaling at most μ bits of payload data, then $\mathcal{B}$ makes at most $\lceil \mu/n \rceil + q_e$ oracle queries.

Theorem 4 states that, if the blockcipher used to construct the encryption algorithm of Figure 8.1 is a secure pseudorandom permutation against related key attacks, then the resulting encryption algorithm provides data privacy.
Proof: [Theorem 4] Let $\mathcal{B}$ be an adversary against BC that uses adversary $\mathcal{A}$ and that has oracle access to a function $f \xleftarrow{\$} \mathsf{BC}$. Adversary $\mathcal{B}$ runs $\mathcal{A}$ and replies to $\mathcal{A}$'s encryption oracle queries using its own oracle $f(\cdot, \cdot)$ for the blockcipher used in the mode of encryption depicted in Figure 8.1. Adversary $\mathcal{B}$ returns the same bit that $\mathcal{A}$ returns. Then,

$$\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K(\cdot,\cdot)} = 1\right] = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{BC}_{RK(\cdot,K)}(\cdot)} = 1\right],$$

since when $\mathcal{B}$ is given a random instance of BC it runs $\mathcal{A}$ exactly as if $\mathcal{A}$ was given the real encryption oracle. Furthermore,

$$\Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} = 1\right] = \Pr\left[K \xleftarrow{\$} \mathcal{K}; \pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K}, \mathcal{D}) : \mathcal{B}^{\pi_{RK(\cdot,K)}(\cdot)} = 1\right]$$

since $\mathcal{B}$ replies to all of $\mathcal{A}$'s oracle queries with independently selected random strings. Consequently,

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{priv}}_{\mathcal{E}}(\mathcal{A}) &= \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K(\cdot,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} = 1\right] \\
&= \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{BC}_{RK(\cdot,K)}(\cdot)} = 1\right] \\
&\qquad - \Pr\left[K \xleftarrow{\$} \mathcal{K}; \pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K}, \mathcal{D}) : \mathcal{B}^{\pi_{RK(\cdot,K)}(\cdot)} = 1\right] \\
&= \mathsf{Adv}^{\text{prp-rka}}_{\Phi,\mathsf{BC}}(\mathcal{B}),
\end{aligned}$$

and the theorem follows. ∎

Chapter 9

RMAC

In this chapter, the detailed description of RMAC (Residue MAC) is given. RMAC is a universal hash-function based MAC that is used in AVALANCHEv1.1 to produce an authentication tag for the associated data.

9.1 Notations and Preliminaries

In this section we describe the notations and preliminaries that will be used in the chapter.

9.1.1 Notations

- For a positive integer α, $\{0,1\}^\alpha$ denotes the set of all α-bit sequences, whereas $\{0,1\}^*$ denotes the set of all arbitrary length bit sequences.
- We use $\mathbb{F}_p$ as the usual notation for the prime field where operations are performed modulo the prime integer p.
- For a nonempty set $\mathcal{S}$, the notation $s \xleftarrow{\$} \mathcal{S}$ denotes the operation of selecting an element from the set $\mathcal{S}$ uniformly at random and assigning it to s.

9.1.2 Negligible Functions

An important term that will be used in this paper is the definition of negligible functions. A function $\mathsf{negl} : \mathbb{N} \rightarrow \mathbb{R}$ is said to be negligible if for any nonzero polynomial poly, there exists N_0 such that for all $N > N_0$, $|\mathsf{negl}(N)| < \frac{1}{|\mathsf{poly}(N)|}$. That is, the function is said to be negligible if it converges to zero faster than the reciprocal of any polynomial function [8].

9.1.3 Message Authentication Codes

A message authentication code $\mathcal{MAC} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is a symmetric-key primitive consisting of three algorithms: the key generation algorithm ($\mathcal{K}$), the signing algorithm ($\mathcal{S}$), and the verifying algorithm ($\mathcal{V}$). $\mathcal{MAC}$ is defined over some key space KeySp and some message space $\mathsf{MsgSp} = \{0,1\}^*$. The randomized key generation algorithm $\mathcal{K}$ returns a key $k \in \mathsf{KeySp}$. The possibly probabilistic signing algorithm $\mathcal{S}$ takes as input a key $k \in \mathsf{KeySp}$ and a payload message $m \in \mathsf{MsgSp}$, and returns an authentication tag (authentication tag and MAC will be used synonymously in throughout this paper) $\tau \in \{0,1\}^*$. The deterministic verifying algorithm $\mathcal{V}$ takes as input a key $k \in \mathsf{KeySp}$, a message $m \in \mathsf{MsgSp}$, and a tag $\tau \in \{0,1\}^*$, and outputs a decision $d \in \{0,1\}$, where '0' stands for invalid message and '1' stands for valid message. We ask for the basic validity requirement that if $\tau = \mathcal{S}(k,m)$ then it must be the case that $\mathcal{V}(k,m,\tau) = 1$ for any k, m, and τ.

9.1.4 Adversarial Model

We adopt the standard model of existential unforgeability under chosen message attacks (EUF-CMA). The adversary is given oracle access to the signing algorithm $\mathcal{S}(\cdot, m)$. The adversary can call the $\mathcal{S}$ oracle on plaintext messages of her

choice and observe the outputs. After calling the $\mathcal{S}$ oracle for q times, the adversary attempts a forgery by calling the verifying algorithm $\mathcal{V}(\cdot, m, \tau)$ for a tag that has not been attached to the message by the signing oracle. Note that the adversary does not see the secret key k nor the coin tosses of $\mathcal{S}$, if any. If the verified decryption oracle returns 0, the adversary is considered unsuccessful; otherwise, the forgery attempt is said to be successful.

Game 1 (EUF-CMA game)**.**

1. *The challenger draws a key* $k \xleftarrow{\$} \mathcal{K}$ *uniformly at random.*
2. $\mathcal{A}$ *calls the signing oracle a polynomial number of times on messages of its choice and records the corresponding tags.*
3. $\mathcal{A}$ *then calls the verifying oracle on a message-tag pair* (m, τ) *of its choice.*
4. $\mathcal{A}$ *wins the game if* $\mathcal{V}(k, m, \tau) = 1$ *and* τ *has never been attached to* m *by the signing oracle.*

Let $\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{MAC}}(\mathcal{A})$ denotes adversary's $\mathcal{A}$ advantage of successful forgery against the message authentication code $\mathcal{MAC}$. Then, $\mathcal{MAC}$ is said to be existentially unforgeable under chosen message attack if

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{MAC}}(\mathcal{A}) \leq \mathsf{negl}(|k|),$$

where $\mathsf{negl}(|k|)$ is a negligible function in the security parameter.

Since in computationally secure MACs based on universal hash functions the hashed image must be processed with a cryptographic function, another security notion that will be used in this paper is the privacy notion of indistinguishability under chosen plaintext attacks (IND-CPA). Let $\mathcal{A}$ be an adversary who is given oracle access to an encryption algorithm $\mathcal{E}$ and can ask the oracle to encrypt a polynomial number of messages to get their corresponding ciphertexts. The encryption algorithm is said to be IND-CPA secure if the adversary, after calling the signed encryption oracle a polynomial number of times, is given a ciphertext corresponding to one of two plaintext messages of her choice cannot determine the plaintext corresponding to the given ciphertext with an advantage significantly higher than 1/2.

9.2 The Residue Message Authentication Code (RMAC)

9.2.1 The Basic Idea

The main idea of the proposed scheme is to compute authentication tags over a secret prime field. Therefore, unlike standard universal hashing MACs, security is not only obtained from the secrecy of the key, but also from the secrecy of the prime field under which the tag is computed. Since the use of a cryptographic

function to process the hashed image is mandatory in universal hashing MACs to preserve the secrecy of the key, ensuring the secrecy of the prime field comes for free; that is, no extra overhead is needed to guarantee the secrecy of the prime field. The added layer of uncertainty allows the design of a MAC in which only the residue of the message is multiplied by the secret hashing key. To the best of our knowledge, RMAC is the first authentication mechanism in the literature that does not require every single block of the message to be multiplied by a secret key (as in the case of standard universal hashing) or processed with a cryptographic primitive (as in the case of block cipher and cryptographic hash function based MACs). Note that this has nothing to do with "security through obscurity;" all we do is assume the prime modulus is part of the secret key.

Before we describe the details of the proposed scheme, we give a brief background showing that there is a sufficient number of primes to grant security. That is, correctly guessing a prime chosen randomly from the set of equal length (in bits) primes is a negligible function in the bit-length of the prime.

9.2.2 The Prime Number Theorem

Let n be an integer. The prime counting function $\pi(n)$ counts the number of primes less than n. Gauss first proposed that the prime counting function can be approximated by

$$\pi(n) \approx \frac{n}{\ln n} \tag{9.1}$$

and then later refined it to

$$\pi(n) \approx \mathrm{Li}(n), \tag{9.2}$$

where $\mathrm{Li}(n) = \int_2^n \frac{1}{\ln x}\,\mathrm{d}x$ is the logarithmic integral [11]. The relation in equation (9.2) is known as the prime number theorem; the theorem was independently proven by Hadamard [10] and Poussin [16].

The approximations in equations (9.1) and (9.2) were quite accurate. In fact, for a large n, Chebyshev showed that [6]

$$0.89\ \mathrm{Li}(n) < \frac{\pi(n)}{n/\ln n} < 1.11\ \mathrm{Li}(n).$$

Chebyshev also showed that

$$0.922 < \frac{\pi(n)}{n/\ln n} < 1.105$$

and proved that if the limit, $\lim\limits_{n\to\infty} \dfrac{\pi(n)}{n/\ln n}$, does exist, then it must be equal to one [11]. Rosser and Schoenfeld showed that, for all $n \geq 17$, $\pi(n) > \dfrac{n}{\ln n}$ [20].

As a direct consequence of the above results, for a large enough positive integer κ, the number of κ-bit primes can be approximated by

$$\frac{2^\kappa}{\kappa \ln 2} - \frac{2^{\kappa-1}}{(\kappa-1)\ln 2} \approx \frac{2^\kappa}{\kappa \ln 4}. \tag{9.3}$$

Equation (9.3) implies that the number of κ-bit primes is an exponentially increasing function of κ. The following corollary is a direct consequence of the above discussion.

Corollary 1. *If $\mathcal{P}_\kappa$ is the set of all κ-bit prime integers, the probability of guessing a prime number drawn uniformly at random from $\mathcal{P}_\kappa$ is a negligible function in κ.*

9.2.3 Detailed Description

Let κ be the security parameter of message authentication. Formally, RMAC consists of three algorithms: the key generation algorithm $\mathcal{K}$, the signing algorithm $\mathcal{S}$, and the verifying algorithm $\mathcal{V}$. The key generation algorithm takes no input and returns a pair (p, k), where p determines the prime filed under which the MAC is computed and k is the authentication key. The prime p is drawn uniformly at random from $\mathcal{P}_\kappa$, the set of all κ-bit primes, while the authentication key is drawn uniformly at random from the the set $\mathbb{F}_p \backslash \{0, \cdots, \lfloor p/2 \rfloor\}$; that is, k must be greater than $p/2$. Throughout the rest of the paper, we overload notations such as m, k, τ to denote both the binary strings of their respective parameters and their integer representation in a big-endian format; the distinction between the two representations will be omitted as long as it is clear from the context.

On input an arbitrarily long message m, the signing algorithm appends '1' as the most significant bits of the message, $m_{\text{app}} = 1||m$, reduces m_{app} to its residue modulo p,

$$m_{\text{res}} = m_{\text{app}} \pmod{p}, \tag{9.4}$$

and then computes the authentication tag as

$$\tau = \mathcal{E}\Big(k \cdot m_{\text{res}} \pmod{p}\Big), \tag{9.5}$$

where $\mathcal{E}$ is an IND-CPA secure stream cipher. [1]

Note that appending the message with a '1' as the most significant bit serves an important purpose: it guarantees that distinct binary messages correspond to distinct integers (since padding the most significant part of the message with zeros will force its integer value to change). Note also that because $k > p/2$, modular reduction in equation (9.5) will be enforced for all $m_{\text{res}} \neq \{0, 1\}$, which will only happen with negligible probability. The three algorithms constituting RMAC are shown in Figure 9.1.

9.3 Security Statement

In this section, we give a formal security analysis of the proposed RMAC. Compared to existing MACs, in which security proofs typically span multiple pages,

[1] In our implementation of AVALANCHEv1.1, τ_P will be used as the key stream to encrypt $k \cdot m_{\text{res}} \pmod{p}$. That is, the output of RMAC is simply $\tau_A = k \cdot m_{\text{res}} \pmod{p}$ and the final authentication tag of AVALANCHEv1.1 will be $\tau_P \oplus \tau_A$.

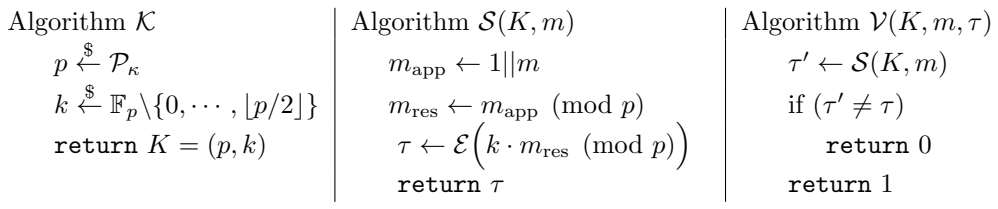

Figure 9.1: The three algorithms constituting the proposed RMAC: the key generating algorithm $\mathcal{K}$, the signing algorithm $\mathcal{S}$, and the verifying algorithm $\mathcal{V}$. The algorithm $\mathcal{E}$ used to encrypt the hashed image can be any IND-CPA secure stream cipher.

the simplicity of the proposed RMAC will be reflected on the simplicity of its security proof.

We prove here the authenticity of the scheme assuming the encryption algorithm in equation (9.5) is a stream cipher (or a block cipher operated in a stream mode, such as the counter mode). Other encryption algorithms might be used but the analysis will be different, thus omitted here.

Let RMAC-$\mathcal{E}$ denote the proposed MAC of Section 9.2 using $\mathcal{E}$ as the underlying encryption algorithm. Let $\mathsf{Adv}^{\text{euf-cma}}_{\text{RMAC-}\mathcal{E}}(\mathcal{A})$ denote adversary's $\mathcal{A}$ advantage of successful forgery against RMAC-$\mathcal{E}$. We give below an information-theoretic bound on the adversary's advantage of successful forgery assuming the plaintext is encrypted by XORing it with a true random binary sequence; i.e., the encryption is a one-time pad (OTP) cipher. The generalization of Theorem 5 to any IND-CPA secure stream cipher is a standard complexity reduction and is informally discussed after the theorem's statement and proof.

Theorem 5. *Let κ be the bit length of the prime modulus used to compute authentication tags according to equations (9.4) and (9.5). Let* RMAC-OTP *denote the proposed MAC of Section 9.2 with an information-theoretically secure OTP cipher as the underlying encryption algorithm. Let $\mathcal{A}$ be an adversary making q signing queries before attempting its forgery. Then,*

$$\mathsf{Adv}^{\text{euf-cma}}_{\text{RMAC-OTP}}(\mathcal{A}) \leq \mathsf{negl}(\kappa).$$

It is standard to pass a complexity-theoretic analog of Theorem 5. One gets the following. Let $\mathcal{E}$ be an IND-CPA secure encryption algorithm that XORs plaintexts with a pseudorandom bit stream (e.g., a stream cipher or a block cipher based encryption using the counter (CTR) mode of operation). Given adversary $\mathcal{A}$ against the RMAC-$\mathcal{E}$, one can construct an adversary $\mathcal{B}$ against $\mathcal{E}$ so that

$$\mathsf{Adv}^{\text{euf-cma}}_{\text{RMAC-}\mathcal{E}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{E}}(\mathcal{B}) + \mathsf{negl}(\kappa). \tag{9.6}$$

Equation (9.6) states that the adversary's advantage of successful forgery is provably secure provided the underlying stream encryption is IND-CPA secure.
Proof: [of Theorem 5] Since the plaintext must be appended with '1' before authentication, we eliminate the trivial case of a known zero tag when the plaintext message consists of all zeros. In addition, attempting forgery by appending zeros as the most significant bits of the plaintext will not trivially succeed since the appended '1' will force the integer value of the plaintext to change. Furthermore, since $k > p/2$, the probability of choosing a message m such that $k \cdot m_{\text{res}} < p$ is negligible for secret p. For the reminder of the proof, for ease of notation, we will drop the subscript from m_{app} and use m to denote the plaintext after appending '1' as its most significant bit.

Assume there is an adversary calling the signing oracle for q times and recording the sequence

$$\mathsf{Seq} = \big\{(m_1, \tau_1), \cdots, (m_q, \tau_q)\big\} \tag{9.7}$$

of message-tag pairs. We aim to bound the probability that a pair (m, τ) of the adversary's choice will be accepted as valid, where $(m, \tau) \neq (m_i, \tau_i)$ for any $i \in \{1, \cdots, q\}$, since otherwise the adversary does not win by definition.

Let $m = m_i + \epsilon_m$ for any $i \in \{1, \cdots, q\}$, where ϵ_m can be any value of the adversary's choice. Let $\mathsf{Collision}$ denote the event that $km \equiv km_i \pmod p$ and let $\overline{\mathsf{Collision}}$ denote its complement. Then,

$$\begin{aligned} \Pr\big[\mathsf{Collision}\big] &= \Pr\big[km \equiv km_i \pmod p\big] &(9.8)\\ &= \Pr\big[k\epsilon_m \equiv 0 \pmod p\big]. &(9.9)\end{aligned}$$

If the prime modulus p is known, the adversary can set ϵ_m to be any integer multiple of p for a collision. However, due to the perfect secrecy of the OTP encryption, no information about p can be obtained by observing the authentication tags. Furthermore, p is chosen randomly from the set of κ-bit primes and, by Corollary 1, guessing the correct value of p is negligible in κ. Consequently,

$$\Pr\big[\mathsf{Collision}\big] \leq \mathsf{negl}(\kappa). \tag{9.10}$$

If no collision occurs, it remains to prove that an adversary cannot construct a pair (m, τ) such that $m = m_i + \epsilon_m$ and τ has been changed appropriately to account for $k\epsilon_m \pmod p$ by flipping the corresponding bits of τ_i. To be able to do that, the adversary must predict the correct value of the secret key. Therefore,

$$\Pr\big[\mathsf{Forgery}|\overline{\mathsf{Collision}}\big] \leq 2^{-(\kappa-2)} = \mathsf{negl}(\kappa). \tag{9.11}$$

Therefore, the probability of successful forgery can be bounded by

$$\begin{aligned} \Pr\big[\mathsf{Forgery}\big] &= \Pr\big[\mathsf{Forgery}|\mathsf{Collision}\big] \cdot \Pr\big[\mathsf{Collision}\big] \\ &\quad + \Pr\big[\mathsf{Forgery}|\overline{\mathsf{Collision}}\big] \cdot \Pr\big[\overline{\mathsf{Collision}}\big] \\ &\leq \Pr\big[\mathsf{Collision}\big] + \Pr\big[\mathsf{Forgery}|\overline{\mathsf{Collision}}\big] \\ &= \mathsf{negl}(\kappa), \end{aligned}$$

and the theorem follows. ∎

Remark 2. *To see how equation (9.6) follows from Theorem 5, observe that in the proof of Theorem 5 the only use of the information-theoretic security of the underlying OTP encryption algorithm is to imply that no information about the authentication key k nor the prime modulus p can be exposed by the authentication tags. Therefore, the difference between using an information-theoretically secure OTP encryption and an IND-CPA secure stream cipher is that the adversary in the latter case has a negligible advantage of exposing the value of k or p by breaking the IND-CPA security of encryption. The rest is just a standard complexity reduction.*

Remark 3. *An alternative approach to prove Theorem 5 is by adopting to the definition of ϵ-almost XOR universal hash families introduced in [14, 18].*[2] *A function $\mathcal{H}$ is said to be ϵ-almost XOR universal, denoted ϵ-AXU, if for all distinct $m, m' \in \mathsf{MsgSp}$, and any $a \in \mathcal{R}$, where $\mathcal{R}$ is the range of the hash function, we have that $\Pr_{h \xleftarrow{\$} \mathcal{H}}[h(m) \oplus h(m') = a] \leq \epsilon$. The proof of Theorem 5 shows that when both the prime modulus p and the secret key k in RMAC are kept secret, then $h(m) = km \pmod p$ is $\mathsf{negl}(\kappa)$-AXU hash family (indeed, the size of the set of all possible messages that are different by multiples of p is negligible compared to the size of the set of all possible messages). This is all that is needed to prove the security of the MAC since, in [14], it has been shown that encrypting the output of an ϵ-AXU hash function with a one-time pad will result in a secure MAC.*

[2] Krawczyk first introduced the definition but with the name ϵ-otp-secure hash families [14]; it was Rogaway, however, who gave it the more common name of ϵ-almost XOR universal [18].

Bibliography

[1] B. Alomair. AVALANCHEv1. In *http://competitions.cr.yp.to/round1/avalanchev1.pdf*. 2014.

[2] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. *Advances in Cryptology–EUROCRYPT'03*, pages 647–647, 2003.

[3] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, 2008.

[4] A. Biryukov and D. Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. In *Advances in Cryptology–ASIACRYPT'09*, pages 1–18. Springer, 2009.

[5] A. Bogdanov, M. Lauridsen, and E. Tischhauser. Cryptanalysis of AVALANCHEv1. In *http://martinlauridsen.info/pub/avalanchev1.pdf*. 2014.

[6] H. Edwards. *Riemann's zeta function*. Dover Pubns, 2001.

[7] W. Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008.

[8] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[9] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[10] J. Hadamard. Sur la distribution des zéros de la fonction ζ (s) et ses conséquences arithmétiques. *Bull. Soc. Math. France*, 24:199–220, 1896.

[11] J. Havil. *Gamma: exploring Euler's constant*. Princeton University Press, 2003.

[12] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology–EUROCRYPT'01*, pages 529–544, 2001.

[13] T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. In *Fast Software Encryption – FSE'04*, volume 3017, pages 408–426. Lecture Notes in Computer Science, Springer, 2004.

[14] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology – CRYPTO'94*, volume 839, pages 129–139. Lecture Notes in Computer Science, Springer, 1994.

[15] F. Mendel, B. Mennink, V. Rijmen, and E. Tischhauser. A simple key-recovery attack on mcoe-x. In *Cryptology and Network Security*, pages 23–31. Springer, 2012.

[16] C. Poussin. *Recherches analytiques sur la théorie des nombres premiers.* Hayez, 1897.

[17] B. Preneel. New Developments in Cryptography. Available at `http://secappdev.org/`, 2012.

[18] P. Rogaway. Bucket hashing and its application to fast message authentication. *Journal of Cryptology*, 12(2):91–115, 1999.

[19] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security – CCS'01*, pages 196–205, 2001.

[20] J. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64–94, 1962.

[21] S. Schwarz. The role of semigroups in the elementary theory of numbers. *Math. Slovaca*, 31(4):369–395, 1981.

[22] C. Shannon. *Communication Theory and Secrecy Systems.* Bell Telephone Laboratories, 1949.

[23] W. Stallings. *Cryptography and Network Security: Principles and Practice.* Prentice-Hall, 2013.

Appendix A

Proof of Lemma 1

The following lemma, a general result known in probability and group theory [21], will be used in the proof of Lemma 1.

Lemma 2. *Let G be a finite group and $\boldsymbol{X}$ a uniformly distributed random variable defined on G, and let $k \in G$. Let $\boldsymbol{Y} = k * \boldsymbol{X}$, where $*$ denotes the group operation. Then $\boldsymbol{Y}$ is uniformly distributed on G.*

Proof: [Lemma 1] Recall that the coin tosses, r, is uniformly distributed over the set of all possible n-bit binary strings, $\{0,1\}^n$. Then, for any possible value of τ computed according to equation (8.4), and any possible plaintext message M, the following holds:

$$\Pr\left[\boldsymbol{\tau} = \tau | \boldsymbol{M} = M\right] = \Pr\left[\boldsymbol{r} = (\tau - \sum_{i=1}^{k} M[i])\right] = 2^{-n}, \tag{A.1}$$

where $M[i]$ denotes the i^{th} block of the plaintext message M, and $\boldsymbol{\tau}$, $\boldsymbol{M}$, and $\boldsymbol{r}$, denote the random variables representing the values of τ, M, and r selected according to their respective distributions.

Furthermore, for an r drawn uniformly at random from $\{0,1\}^n$, by Lemma 2, the resulting tag is uniformly distributed over $\{0,1\}^n$. That is, for any fixed value $\tau \in \mathbb{Z}_{2^n}$, the probability that the tag will take this specific value is given by:

$$\Pr\left[\boldsymbol{\tau} = \tau\right] = 2^{-n}. \tag{A.2}$$

Combining Bayes' theorem [7] with equations (A.1) and (A.2) yields

$$\Pr\left[\boldsymbol{M} = M | \boldsymbol{\tau} = \tau\right] = \Pr\left[\boldsymbol{M} = M\right] \tag{A.3}$$

for any plaintext M and any authentication tag τ. That is, authentication tags are statistically independent of their corresponding plaintext messages. In other

words, the observation of an authentication tag gives no information about its corresponding plaintext message, as required.

To show that different authentication tags are mutually independent, let $\boldsymbol{M_1}$ through $\boldsymbol{M_\ell}$ denote the random variables representing the experiments of drawing messages M_1 through M_ℓ according to any arbitrary probabilistic distribution. Similarly, let $\boldsymbol{\tau_1}$ through $\boldsymbol{\tau_\ell}$ denote the random variables representing the authentication tags corresponding to messages M_1 through M_ℓ, respectively. Further, let $\boldsymbol{r_1}$ through $\boldsymbol{r_\ell}$ be the random variables representing the coin tosses of the signing algorithm $\mathcal{S}_\mathcal{E}$ for the authentication of messages M_1 through M_ℓ, respectively. Recall that the $\boldsymbol{r_i}$'s are mutually independent and identically distributed (iid) uniform random variables drawn from $\{0,1\}^n$. Then, for any possible values of the messages M_1 through M_ℓ with arbitrary joint probability mass function, and all possible values of τ_1 through τ_ℓ, we get:

$$
\begin{aligned}
&\Pr\Big[\boldsymbol{\tau_1} = \tau_1, \cdots, \boldsymbol{\tau_\ell} = \tau_\ell\Big] \\
&\qquad = \sum_{M_1,\cdots,M_\ell} \Pr\Big[\boldsymbol{\tau_1} = \tau_1, \cdots, \boldsymbol{\tau_\ell} = \tau_\ell | \boldsymbol{M_1} = M_1, \cdots, \boldsymbol{M_\ell} = M_\ell\Big] \\
&\qquad\qquad \cdot \Pr\Big[\boldsymbol{M_1} = M_1, \cdots, \boldsymbol{M_\ell} = M_\ell\Big] \qquad \text{(A.4)} \\
&\qquad = \sum_{M_1,\cdots,M_\ell} \Pr\Big[\boldsymbol{r_1} = \Big(\tau_1 - \sum_{i=1}^{k} M_1[i]\Big), \cdots, \boldsymbol{r_\ell} = \Big(\tau_\ell - \sum_{i=1}^{k} M_\ell[i]\Big)\Big] \\
&\qquad\qquad \cdot \Pr\Big[\boldsymbol{M_1} = M_1, \cdots, \boldsymbol{M_\ell} = M_\ell\Big] \qquad \text{(A.5)} \\
&\qquad = \sum_{M_1,\cdots,M_\ell} \Pr\Big[\boldsymbol{r_1} = \Big(\tau_1 - \sum_{i=1}^{k} M_1[i]\Big)\Big] \cdots \Pr\Big[\boldsymbol{r_\ell} = \Big(\tau_\ell - \sum_{i=1}^{k} M_\ell[i]\Big)\Big] \\
&\qquad\qquad \cdot \Pr\Big[\boldsymbol{M_1} = M_1, \cdots, \boldsymbol{M_\ell} = M_\ell\Big] \qquad \text{(A.6)} \\
&\qquad = \sum_{M_1,\cdots,M_\ell} 2^{-n} \cdots 2^{-n} \cdot \Pr\Big[\boldsymbol{M_1} = M_1, \cdots, \boldsymbol{M_\ell} = M_\ell\Big] \qquad \text{(A.7)} \\
&\qquad = \Pr\Big[\boldsymbol{\tau_1} = \tau_1\Big] \cdots \Pr\Big[\boldsymbol{\tau_\ell} = \tau_\ell\Big], \qquad \text{(A.8)}
\end{aligned}
$$

where $M_j[i]$ denotes the i^{th} block of the j^{th} message M_j. Equation (A.6) holds due to the mutual independence of the $\boldsymbol{r_i}$'s; equation (A.7) holds due to the uniform distribution of the $\boldsymbol{r_i}$'s; and equation (A.8) holds due to the uniform distribution of the $\boldsymbol{\tau_i}$'s. Therefore, authentication tags are mutually independent, and the lemma follows. ■

Appendix B

Key-Recovery Attack

Algorithm 1: OFFLINE(M)

Data: Single-block M

$L \leftarrow \emptyset$
for $i = 1, \ldots, \ell$ **do**
  Choose new $K = (K_P, k, p) \in \{0,1\}^{n+256}$
  $(N, \epsilon, C, T) \leftarrow \mathsf{AVALANCHEv1}(M, \epsilon, K)$
  $L \leftarrow L \cup \{(C[1], K_P, N)\}$
end
return L

Algorithm 2: ONLINE(M, L)

Data: Single-block M, List L

for $i = 1, \ldots, 2^n/\ell$ **do**
  Obtain (N, ϵ, C, T) for (M, ϵ) from AE
  if $\exists (x, y, z) \in L : x = C[1]$ **then**
    return $y \oplus ((N \oplus z) \| 0^{n-|N|})$
  end
end
return Failure

Figure B.1: Illustration of the key-recovery attack against AVALANCHEv1 [1] as appeared in [5].

This section addresses a key-recovery attack that was brought to our attention by Bogdanov et al [5], which is based on [15]. The attack consists of two phases: an online phase and an offline phase, as can be seen in Figure B.1. The main idea behind the attack is to build a list L by calling AVALANCHE on different keys with the same single-block message and collecting the outputs. In the online phase, the cipher under attack is called with the same single-block message used to construct the list L. For each call, the output of the cipher is compared to the outputs collected in the list and, if it is equal to one of the ciphertexts in L, the key can be recovered as in Algorithm 2 of Figure B.1. For a better memory-time tradeoff, the size of L is $2^{n/2}$, where n is the length of the blockcipher key [5]. Then, if the online phase consists of calling the cipher $2^{n/2}$ times, it is expected to have a collision between the two sets with high probability, thus recovering the blockcipher key.

An important observation about the attack is that it is based on fixing the plaintext and observing the outputs. Therefore, one way to mitigate this attack is by randomizing the plaintext before encryption (i.e., whitening). This can

be done by xoring plaintext blocks with random strings. Luckily, this can be done in AVALANCHE almost for free. Recall that, in PCMAC, the first block is a random number r. This same r can be used to whiten plaintext block, if needed.[1]

[1] As can be seen in Table 2.1, the complexity of the attack, especially the memory requirement, is impractical for AES192 and AES256.