Cryptanalysis of AVALANCHEv1

Andrey Bogdanov, Martin M. Lauridsen, and Elmar Tischhauser
{anbog,mmeh,ewti}@dtu.dk

DTU Compute, Technical University of Denmark, Denmark

Abstract. AVALANCHEv1 [1] is a round-1 submission to the ongoing CAESAR competition for authenticated encryption. In this note, we describe how to recover the entire authentication and encryption key material in $2^{n/2}$ time, where n is the key length of the underlying block cipher, that is, 2^{64}, 2^{128}, and 2^{96} for the three parameters sets of AVALANCHEv1 based on AES-128, -256, and -192, respectively.

1 Introduction

We note that the specification of AVALANCHEv1 leaves some room for interpretation. In the aspects relevant to our attacks, we assume the following:

- The nonce has sizes $|N| \in \{80, 160, 128\}$ for key lengths $n \in \{128, 256, 192\}$, resp.
- The nonce N is randomly generated.
- The counter c is initialized to $c = 0$.
- The tag length is $|T| = 128$ as well as the security parameters k and p are 128-bit.
- The $(n + 256)$-bit key K consists of three independent parts, $K = (K_P, k, p)$.

1.1 AVALANCHEv1

AVALANCHEv1 uses AES to process a message M of m blocks and associated data A of arbitrary length to a ciphertext C of $m + 1$ blocks and an authentication tag T. It does not support a public message number, instead a nonce N is generated by the encryption algorithm itself.

The input to AVALANCHEv1 with a specified secret key $K = (K_P, k, p)$, is a 3-tuple (M, A, K) of message and associated data. The output is a 4-tuple (N, A, C, T) of nonce, associated data, ciphertext, and tag. The scheme uses two main algorithms; PCMAC for message processing and RMAC for processing associated data. The interfaces and outputs of the two algorithms are

$$(N, C, \tau_P) = PCMAC(M) \qquad \text{and} \qquad \tau_A = RMAC(A).$$

The final tag T is computed as $T = \tau_P \oplus \tau_A$.

1.2 PCMAC

The encryption with PCMAC is illustrated in Figure 1. The padded message is denoted $M[1] \cdots M[m]$ and the ciphertext $C[0] \cdots C[m]$. r is generated at random. The counter c is initialized to 0.

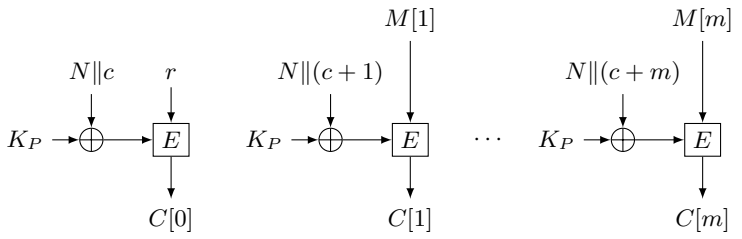

Fig. 1. Message processing with PCMAC

1.3 RMAC

The output of RMAC is an intermediate tag τ_A of 128 bits. RMAC uses the secrets k and p, p being a randomly chosen 128-bit prime and k chosen at random from $\{\lfloor p/2 \rfloor + 1, \ldots, p-1\}$. The intermediate tag τ_A is determined as

$$\tau_A = (1 \| A) \cdot k \mod p.$$

2 Recovering the PCMAC key

The critical part of PCMAC is that the encryption key for E (see Figure 1) depends on the nonce and counter. This facilitates key collision attacks, similar to McOE-X [2]. Our attack works in an *offline phase* (see Algorithm 1) and an *online phase* (see Algorithm 2), where ϵ denotes the empty bit string. Both are called with the same, arbitrary single-message block M. The offline phase outputs a list L which is used in the online phase.

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

In the offline phase we build a table of size ℓ of AVALANCHEv1 encryptions of the same message block, using different keys. In the online phase we request the encryption of the same single-block message M in total $2^n/\ell$ times. By the birthday paradox, we expect to see a collision in the oracle output $C[1]$ in the online phase and the list L from the offline phase. As the nonce N is public, we can then recover the secret key K_P by adding it to the stored nonce z and key y. We verity candidate keys using an additional encryption. Obviously, choosing $\ell = 2^{n/2}$ gives the best overall complexity, using just $2 \cdot 2^{n/2}$ time and memory in the order of $2^{n/2}$ to store L. Memoryless versions of time-memory tradeoff can be used here as well [3].

3 Recovering the RMAC secret parameters

To recover (k, p), we use the attack described above to recover the secret K_P. We furthermore ask for encryption and tag of some arbitrary message block; once with empty associated data, i.e. $A = \epsilon$, and once with $A = 0$, i.e. a single zero bit. Let the corresponding outputs of AVALANCHEv1 be (N, ϵ, C, T) and $(N', 0, C', T')$, where $T = \tau_P \oplus \tau_A$ and $T' = \tau'_P \oplus \tau'_A$.

With K_P in hand, we can ourselves compute τ_P and τ'_P using PCMAC. Consequently, we obtain τ_A and τ'_A. Using the definition of RMAC, we observe that for the case where $A = \epsilon$ we directly obtain $\tau_A \equiv k \mod p$, but since $k \in \{\lfloor p/2 \rfloor + 1, \ldots, p-1\}$ we have $k = \tau_A$. Now, for the case where $A = 0$, we find

$$\begin{aligned} \tau'_A &\equiv (1 \| 0) \cdot k \mod p \\ \Leftrightarrow \tau'_A &\equiv 2k \mod p \\ \Leftrightarrow p &= 2k - \tau'_A. \end{aligned}$$

We therefore obtain the secret parameters (k, p) of RMAC with a complexity of two one-block encryption queries.

Summarising, we have recovered all $n + 256$ bits of secret key material in about $2^{n/2}$ time.

References

1. Basel Alomair. AVALANCHEv1. http://competitions.cr.yp.to/round1/avalanchev1.pdf.
2. Florian Mendel, Bart Mennink, Vincent Rijmen, and Elmar Tischhauser. A Simple Key-Recovery Attack on McOE-X. In *CANS*, volume 7712, pages 23–31. Springer, 2012.
3. Jean-Jacques Quisquater and Jean-Paul Delescaille. How Easy is Collision Search. New Results and Applications to DES. CRYPTO'89, vol. 435 of LNCS, pp. 408-413, Springer-Verlag, 1989.