Forgery and Key Recovery Attacks for Calico

Christoph Dobraunig, Maria Eichlseder,
Florian Mendel, Martin Schläffer

Institute for Applied Information Processing and Communications
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria

April 1, 2014

1 Calico v8

Calico [3] is an authenticated encryption design submitted to the CAESAR competition by Christopher Taylor. In Calico v8 in reference mode, ChaCha-14 and SipHash-2-4 work together in an Encrypt-then-MAC scheme. For this purpose, the key is split into a Cipher Key K_C and a MAC Key K_M. The plaintext is encrypted with ChaCha under the Cipher Key to a ciphertext with the same length as the plaintext. Then, the tag is calculated as the SipHash MAC of the concatenated ciphertext and associated data. The key used for SipHash is generated by xoring the nonce to the (lower, least significant part of the) MAC Key:

$$(C, T) = \mathrm{Enc}_{\mathrm{Calico}}(K_C \parallel K_M, N, A, P),$$

where $\parallel$ is concatenation, and with $\oplus$ denoting xor, the ciphertext and tag are calculated vi

$$C = \mathrm{Enc}_{\text{ChaCha-14}}(K_C, N, P)$$
$$T = \mathrm{MAC}_{\text{SipHash-2-4}}(K_M \oplus N, C \parallel A).$$

Here, A, P, C denote associated data, plaintext and ciphertext, respectively, all of arbitrary length. T is the 64-bit tag, N the 64-bit nonce, and the 384-bit key K is split into a 256-bit encryption and 128-bit authentication part, $K = K_C \parallel K_M$.

2 Missing Domain Separation

As shown above, the tag is calculated over the concatenation $C \parallel A$ of ciphertext and associated data. Due to the missing domain separation between ciphertext and associated data in the generation of the tag, the following attack is feasible. An attacker who knows a valid tuple (A, C, T) is able to arbitrarily declare the boundary between ciphertext and associated data without changing the value of the tag. More specifically, any prefix of the associated data can be declared as postfix to the ciphertext, or any ciphertext postfix prepended to the associated data instead. In either case, the plaintext will change uncontrollably.

3 MAC Key Recovery

In Calico, SipHash is modified by xoring a nonce to the lower-significance bits of the key. This modification of SipHash makes an attack similar to the one described by Mendel et al. [2]

(and also applied to AVALANCHEv1 [1]) feasible. The attack targets the tag generation to recover the MAC Key, which in turn allows to forge tags for arbitrary associated data and ciphertexts.

We can split the attack into an online phase and an offline phase, where the online phase requires access to an encryption oracle. Algorithm 1 requires 2^{64} online queries for an overall complexity of 2^{64}. Tradeoffs are possible to reduce the number of online queries at the cost of the overall complexity.

Algorithm 1 MAC Key Recovery

Offline Phase:
for $i = 0$ **to** $2^{64} - 1$ **do**
1. Compute tag T for empty plaintext under the MAC key $i \parallel 0$ and nonce $N = 0$.
2. Save pair (T, K_M) in a list.

end for

Online Phase:
for $i = 0$ **to** $2^{64} - 1$ **do**
1. Request tag T for empty plaintext under nonce $N = j$ from encryption oracle.
2. If T is found in the list with a key $i \parallel 0$, then $K_M = i \parallel j$ is a MAC Key candidate.

end for

This produces at least one MAC Key candidate; if necessary, remaining candidates can be filtered with additional offline computations, though their expected number is very small.

Since Calico preserves the plaintext length for the ciphertext, an empty plaintext and associated data will produce an empty input for the MAC, independent of the Cipher Key or nonce. Thus, all offline computations and online queries give tags calculated from the same MAC input, only with varying keys fed to SipHash. The SipHash keys used in the offline phase all have the lower 64 bits set to 0 and the upper 64 bits iterating through all possible values. In the online phase, the SipHash keys have the upper 64 bits set to the original bits of the secret K_M, while the lower bits iterate through all possibilities. Thus, there is exactly one match between the two key lists, which will also produce a colliding tag (though other tag pairs may collide as well). The matching key stored in the offline list gives the upper 64 bits of the correct key, the colliding nonce from the online phase the lower 64 bits.

For tradeoffs with online complexity $2^N < 2^{64}$, replace 2^{64} by 2^N in the online phase and by $2^{(128-N)}$ in the offline phase; the success probability remains 1.

References

[1] Andrey Bogdanov, Martin M. Lauridsen, and Elmar Tischhauser. Cryptanalysis of AVALANCHEv1. `https://groups.google.com/forum/#!topic/crypto-competitions/JLvtsRCBWEY`, `http://martinlauridsen.info/pub/avalanchev1.pdf`, 2014.

[2] Florian Mendel, Bart Mennink, Vincent Rijmen, and Elmar Tischhauser. A Simple Key-Recovery Attack on McOE-X. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *CANS*, volume 7712, pages 23–31. Springer, 2012.

[3] Christopher Taylor. The Calico Family of Authenticated Ciphers Version 8. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/calicov8.pdf`, 2014.