A Full Key Recovery Attack on HMAC-AURORA-512

Yu Sasaki

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
sasaki.yu@lab.ntt.co.jp

Abstract. In this note, we present a full key recovery attack on HMAC-AURORA-512 when 512-bit secret keys are used and the MAC length is 512-bit long. Our attack requires 2^{257} queries and the off-line complexity is 2^{259} AURORA-512 operations, which is significantly less than the complexity of the exhaustive search for a 512-bit key. The attack can be carried out with a negligible amount of memory. Our attack can also recover the inner-key of HMAC-AURORA-384 with almost the same complexity as in HMAC-AURORA-512. This attack does not recover the outer-key of HMAC-AURORA-384, but universal forgery is possible by combining the inner-key recovery and 2nd-preimage attacks. Our attack exploits some weaknesses in the mode of operation.

keywords: AURORA, DMMD, HMAC, Key recovery attack

1 Description

1.1 Mode of operation for AURORA-512

We briefly describe the specification of AURORA-512. Please refer to Ref. [2] for details. An input message is padded to be a multiple of 512 bits by the standard MD message padding, then, the padded message is divided into 512-bit message blocks $(M_0, M_1, \ldots, M_{N-1})$.

In AURORA-512, compression functions $F_k : \{0,1\}^{256} \times \{0,1\}^{512} \rightarrow \{0,1\}^{256}$ and $G_k : \{0,1\}^{256} \times \{0,1\}^{512} \rightarrow \{0,1\}^{256}$, two functions $MF : \{0,1\}^{512} \rightarrow \{0,1\}^{512}$ and $MFF : \{0,1\}^{512} \rightarrow \{0,1\}^{512}$, and two initial 256-bit chaining values H_0^U and H_0^D are defined[1].

The algorithm to compute a hash value is as follows.

1. `for` k=0 `to` $N-1$ {
2. $\quad H_{k+1}^U \leftarrow F_k(H_k^U, M_k)$.
3. $\quad H_{k+1}^D \leftarrow G_k(H_k^D, M_k)$.
4. $\quad$ `if`$(0 < k < N-1) \wedge (k \bmod 8 = 7)$ {
5. $\quad\quad$ `temp` $\leftarrow H_{k+1}^U \| H_{k+1}^D$

[1] F_k and $F_{k'}$ are identical if $k \equiv k' \bmod 8$. G_k and $G_{k'}$ also follow the same rule.

6. $H^U_{k+1} \| H^D_{k+1} \leftarrow MF(\texttt{temp})$.
7. }
8. }
9. Output $MFF(H^U_N \| H^D_N)$.

1.2 HMAC

HMAC [7] is an algorithm to compute MACs when a key and a message are input. According to Ref. [3], the minimal recommended length for the secret key is L, where L is the size of the hash function output. Therefore, it is reasonable to use 512-bit keys for HMAC with 512-bit output hash functions, and use 384-bit keys for 384-bit output hash functions. The HMAC algorithm to compute an output with a hash function H and an initial value H_0 when a key K and a message M are input is as follows.

$$K_0 \leftarrow Pad(K), \tag{1}$$

$$\text{temp} = H(H_0, (K_0 \oplus \texttt{ipad}) \| M), \tag{2}$$

$$\text{HMAC}{-}H(M) = H(H_0, (K_0 \oplus \texttt{opad}) \| \text{temp}), \tag{3}$$

where, `ipad` and `opad` are constant values defined in the specification of HMAC, and $Pad(\cdot)$ is a padding process of K. In $Pad(\cdot)$, if the size of K is shorter than the block length, zeros are appended to the end of K to make its length be the block length, and use as K_0. If the size of K and the block length are identical, K is used as K_0.

2 Full key recovery attack on HMAC-AURORA-512

According to the support document of AURORA [2], AURORA-512 can be used for HMAC only to produce 512-bit MACs. Our attack is a full key recovery attack on HMAC-AURORA-512 when the key size is 512 bits and the MAC length is 512 bits. Our attack does not recover the secret key K itself, but recover the values of chaining variables after $K \oplus \texttt{ipad}$ is processed and $K \oplus \texttt{opad}$ is processed. Recovering these values have the same effect as recovering K with respect to forging MAC values. The number of queries in this attack is 2^{257} and the off-line complexity is 2^{259} AURORA-512 operations. The attack can be carried out with a negligible amount of memory.

The attack procedure is as follows. In this attack, we mainly ask 1-block messages (including padding bits) as queries. The structure to process a 1-block message in HMAC-AURORA-512 is illustrated in Fig. 1.

1. Prepare 2^{257} different messages that are the same length but shorter than 448 bits so that the length of padded messages does not exceed 1-block. Let M^i be prepared messages. Ask all M^i to the oracle, and obtain corresponding $\text{HMAC}_K(M^i)$.

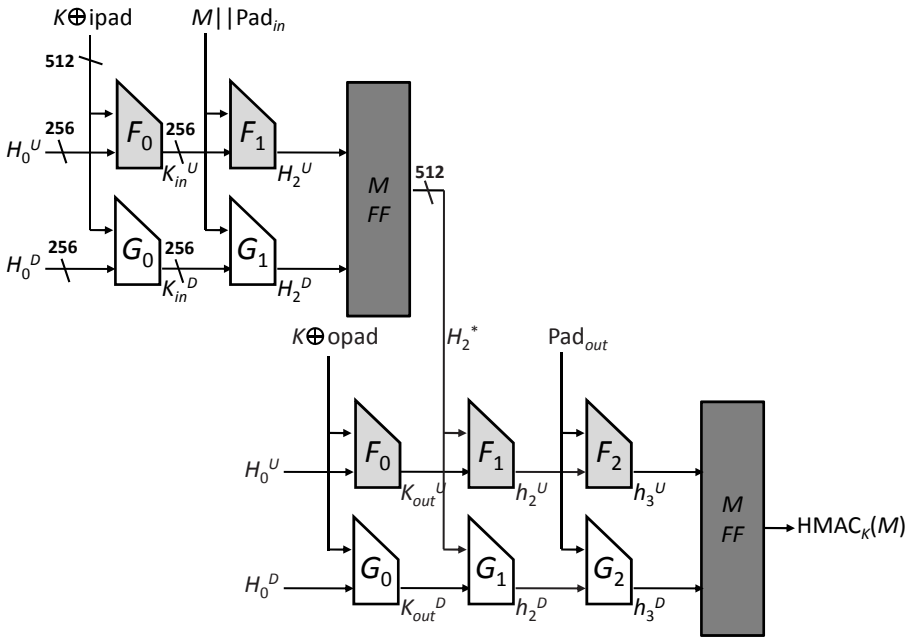

Fig. 1. Structure to process a 1-block message in HMAC-AURORA-512

2. Find message pairs $(M^j, M^{j\prime})$ in which $\mathrm{HMAC}_K(M^j)$ and $\mathrm{HMAC}_K(M^{j\prime})$ are a collision. Due to the computation structure, a pair of messages have the following five possibilities to be a collision.
   Case 1: H_2^Us collide and H_2^Ds collide.
   Case 2: Case 1 does not occur and H_2^*s collide.
   Case 3: Case 1, 2 do not occur and h_2^Us collide and h_2^Ds collide.
   Case 4: Case 1, 2, 3 do not occur and h_3^Us collide and h_3^Ds collide.
   Case 5: Case 1, 2, 3, 4 do not occur and HMAC values collide.
   Therefore, we expect to obtain several collisions in this Step.
3. To detect a collision of Case 1 in Step 2, ask $M^j \| \mathtt{Pad}_{in} \| x$ and $M^{j\prime} \| \mathtt{Pad}_{in} \| x$ for any x to the oracle, and check whether $\mathrm{HMAC}_K(M^j \| \mathtt{Pad}_{in} \| x)$ and $\mathrm{HMAC}_K(M^{j\prime} \| \mathtt{Pad}_{in} \| x)$ are a collision or not. If they are a collision, $(M^j, M^{j\prime})$ is a desired pair with a negligible error probability.
4. Let $(M^{j1}, M^{j1\prime})$ be a colliding pair of Case 1 in Step 2. First, we exhaustively search for K_{in}^U by computing $F_1(K_{in}^U, M^{j1})$ and $F_1(K_{in}^U, M^{j1\prime})$ for all 2^{256} K_{in}^U and check whether the computed values are a collision or not. If they are a collision, the corresponding K_{in}^U is the correct value. Similarly, we detect K_{in}^D by computing $G_1(K_{in}^D, M^{j1})$ and $G_1(K_{in}^D, M^{j1\prime})$ for all 2^{256} K_{in}^D and check whether the computed values are a collision or not.
5. For all HMAC collision pairs $(M^j, M^{j\prime})$ obtained in Step 2, we compute values of H_2^* and $H_2^{*\prime}$ with recovered K_{in}^U and K_{in}^D. If H_2^* and $H_2^{*\prime}$ are a

collision, we discard that pair. Note, each of the remaining collision pairs are one of Cases 3, 4 or 5 in Step 2.

6. Take a collision pair $(M^{j2}, M^{j2\prime})$ from all remaining collision pairs, and assume this pair is a collision of Case 3. We then recover K^U_{out} and K^D_{out} by the same method as Step 4. Namely, we exhaustively search for K^U_{out} such that $F_1(K^U_{out}, H_2^{*j2}) = F_1(K^U_{out}, H_2^{*j2\prime})$ and K^D_{out} such that $G_1(K^D_{out}, H_2^{*j2}) = G_1(K^D_{out}, H_2^{*j2\prime})$.
7. With recovered K_{in} and K_{out}, compute $\mathrm{HMAC}_K(M)$ for any M that are already asked to the oracle, and check whether its HMAC value match with the one obtained from the oracle. If matched, that K_{out} is the correct value. Otherwise, discard the pair $(M^{j2}, M^{j2\prime})$ and go back to Step 6. Repeat the attack by choosing a different collision pair until K_{out} is recovered.

2.1 Complexity and success probability

At Step 1, we ask 2^{257} queries to the oracle. At Step 2, the probability that the collision of each case is obtained can be considered as independent. According to [8, Theorem 3.2], the probability of obtaining a collision for $\log_2 N$-bit output hash function, with trying $\theta \cdot N^{1/2}$ different messages is as follows.

$$1 - e^{-\frac{\theta^2}{2}} \tag{4}$$

Eq. 4 becomes approximately 0.86 when $\theta = 2$. Therefore, we expect to obtain a collision of each case with a probability of 0.86. To successfully recover K_{in} and K_{out}, we need to obtain a collision of Case 1 and a collision of Case 3. By 2^{257} queries, the probability of obtaining these two collisions is $(0.86)^2 \approx 0.75$. This is higher than the probability of obtaining a single collision with 2^{256} queries, which is approximately 0.39. For simplicity, we assume that five collisions in total, a single collision in each case, are obtained. At Step 3, we need two queries for each collision. Hence, if we obtained five collisions, we need eight queries in the worst case, which is negligible compared to Step 1. At Step 4, we compute F_1 $2 \cdot 2^{256}$ times to recover K^U_{in}. For each guess of K^U_{in}, the probability that $F_1(K^U_{in}, M^{j1}) = F_1(K^U_{in}, M^{j1\prime})$ is expected to be 2^{-256}. Hence, we can expect that only one K^U_{in} is chosen as the correct guess. Similarly we compute G_1 $2 \cdot 2^{256}$ times to recover K^D_{in}. As a result, the time complexity for this Step is $2 \cdot 2^{256}$ F_1-operations + $2 \cdot 2^{256}$ G_1-operations $\approx 2^{257}$ AURORA-512 operations. Step 5 costs negligible time. In our assumption, three collisions, one for each of Cases 3, 4, and 5, will remain. Step 6 costs the same complexity as Step 4, which is 2^{257} AURORA-512 operations, and this is repeated three times in the worst case due to Step 7. Therefore, the time complexity for Steps 6 and 7 is $3 \cdot 2^{257}$ AURORA-512 operations. Finally, the total time complexity is 2^{257} AURORA-512 operations for Step 4 and $3 \cdot 2^{257}$ AURORA-512 operations for Step 6, which is 2^{259} AURORA-512 operations.

This attack can be easily carried out if we have a large amount of memory. Moreover, if we apply the memoryless collision search for Step 2, all Steps can be carried out with a negligible amount of memory. To apply the memoryless

collision search, we use the HMAC values obtained from the oracle as the next query. Therefore, Step 1 becomes adaptive. The memoryless collision search of our attack needs the message space of 512 bits[2]. Hence, we use 2-block messages as queries. Due to the increment of the message block, at Step 2, a message pair has 6 possibilities to be a collision. However, since this collision is filtered out at Step 3 with additional two queries, this does not impact on the total attack complexity.

3 Universal forgery on HMAC-AURORA-384

3.1 Inner key recovery attack

AURORA-384 is supporting HMAC for 384-bit MAC length. The computation for AURORA-384 is the same as AURORA-512 but for truncating the last 512-bit value to 384 bits. The structure to process a 1-block message in HMAC-AURORA-384 is illustrated in Fig. 2. The inner key recovery procedure for

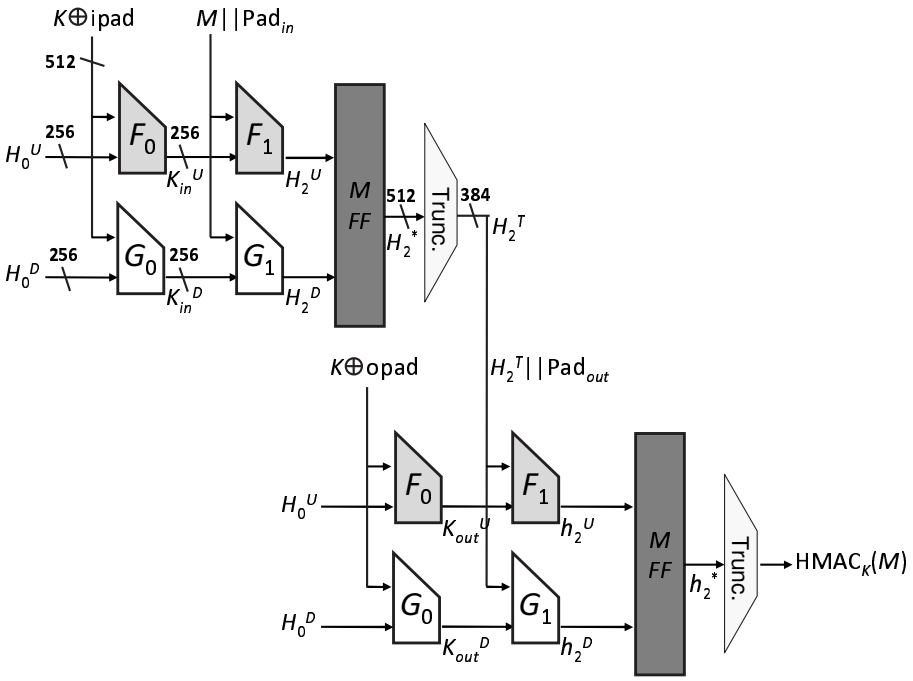

Fig. 2. Structure to process a 1-block message in HMAC-AURORA-384

[2] If message space is much smaller than 512 bits, for example 447 bits, the randomness for the memoryless collision search will collide after $2^{223.5}$ trials and we cannot make 2^{257} different queries.

HMAC-AURORA-384 is almost the same as that of HMAC-AURORA-512. For HMAC-AURORA-384, at Step 2 of the attack procedure, a pair of messages have the following 6 possibilities to be a collision.

Case 1: H_2^Us collide and H_2^Ds collide.
Case 2: Case 1 does not occur and H_2^*s collide.
Case 3: Case 1, 2 do not occur and H_2^Ts collide.
Case 4: Case 1, 2, 3 do not occur and h_2^Us collide and h_2^Ds collide.
Case 5: Case 1, 2, 3, 4 do not occur and h_2^*s collide.
Case 6: Case 1, 2, 3, 4, 5 do not occur and HMAC values collide.

Remember that H_2^T and HMAC values are 384-bit values. By asking 2^{257} queries, we will obtain a single collision pair of Cases 1, 2, 4, 5, and 2^{128} collision pairs of Cases 3 and 6, hence, we will expect to obtain $2^{129}+4$ collisions in total. To recover the inner-key, we need to detect the collision pair of Case 1. At Step 3 of the attack procedure, this can be achieved by asking additional two queries $M^j\|\mathtt{pad}_{in}\|x$ and $M^{j'}\|\mathtt{pad}_{in}\|x$ for each collision pair $(M^j, M^{j'})$. The inner-key recovery procedure at Step 4 is exactly the same, in which we need the time complexity of 2^{257} AURORA-384 operations.

Finally the inner-key is recovered with a query complexity of $2^{257}+2\cdot(2^{129}+4)\approx 2^{257}$ and the time complexity of 2^{257} AURORA-384 operations. This attack can be performed with a negligible amount of memory.

3.2 Universal forgery by combining 2nd-preimage attack

Although our attack cannot recover the outer-key, we can perform the universal forgery on HMAC-AURORA-384 by using the recovered inner-key and applying the 2nd-preimage attack on AURORA-384/-512 [4, 1].

In the universal forgery attack, the attacker has the access to the oracle which returns $\mathrm{HMAC}_k(\cdot)$. For any given message M, our attack can find the value of $\mathrm{HMAC}_k(M)$ without asking M to the oracle. After revealing the inner key, the query complexity of this attack is 1 and the off-line complexity and memory complexity are the same as that of 2nd-preimage attack on AURORA-512, which are 2^{290} AURORA-512 operations and $2^{288}\times 512$ bits of memory in [4] and 2^{291} AURORA-512 operations and $2^{31.5}$ message blocks of memory in [1]. The attack procedure is as follows.

Target:
0. Receive M.
Preparation:
1. Recover the inner-key K_{in} with the attack explained in Section 3.1.
Universal forgery:
2. For the given M, find a 2nd-preimage M' s.t. $\mathrm{AURORA}{-}384(K_{in}, M) = \mathrm{AURORA}{-}384(K_{in}, M')$ by using the 2nd-preimage attack.
3. Ask M' to the oracle, and receive $\mathrm{HMAC}_k(M')$.
4. $\mathrm{HMAC}_k(M')$ is the HMAC value of M.

4 Conclusion

In this note, we presented a full key recovery attack on HMAC-AURORA-512 with 512-bit secret keys and 512-bit output values. Our attack exploits the weaknesses in the mode of operation and recovers K_{in} and K_{out} with 2^{257} queries, 2^{259} off-line computations, and a negligible amount of memory. This attack can also recover the inner-key of HMAC-AURORA-384. Moreover, by combining the inner-key recovery attack and the 2nd-preimage attack, universal forgery can be performed on HMAC-AURORA-384.

Lastly, we emphasize that this attack does not make any impact on security of AURORA-512 as a SHA-3 candidate because, as the security of hash functions used in HMAC, NIST requires only the resistance against distinguishing attack whose query complexity is much less than $2^{n/2}$ and off-line complexity is significantly less than 2^n [6].

Note, collision attacks on AURORA-512 are also presented at Ref. [5, 1].

References

1. Niels Ferguson and Stefan Lucks. Attacks on AURORA-512 and the Double-Mix Merkle-Damgaard transform. Cryptology ePrint Archive, Report 2009/113, 2009. `http://eprint.iacr.org/2009/113`.
2. Tetsu Iwata, Kyoji Shibutani, Taizo Shirai, Shiho Moriai, and Toru Akishita. *AURORA: A Cryptographic Hash Algorithm Family.* AURORA home page `http://www.sony.net/Products/cryptography/aurora/index.html`, (Also available at NIST home page: `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`).
3. H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication.* The Internet Engineering Task Force, 1997. `http://www.ietf.org/rfc/rfc2104.txt`.
4. Yu Sasaki. A 2nd-preimage attack on AURORA-512. Cryptology ePrint Archive, Report 2009/112, 2009. `http://eprint.iacr.org/2009/112`.
5. Yu Sasaki. A collision attack on AURORA-512. Cryptology ePrint Archive, Report 2009/106, 2009. `http://eprint.iacr.org/2009/106`.
6. U.S. Department of Commerce, National Institute of Standards and Technology. *Federal Register /Vol. 72, No. 212/Friday, November 2, 2007/Notices*, 2007. (`http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`).
7. U.S. Department of Commerce, National Institute of Standards and Technology. *The Keyed-Hash Message Authentication Code (HMAC) (Federal Information Processing Standards Publication 198)*, July 2008. (`http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf`).
8. Serge Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security.* Springer, 2006.