Breaking H^2-MAC Using Birthday Paradox

Fanbao Liu[1,2], Tao Xie[1] and Changxiang Shen[2]

[1] School of Computer, National University of Defense Technology, Changsha, 410073, Hunan, P. R. China
[2] School of Computer, Beijing University of Technology, 100124, Beijing, P. R. China
liufanbao@gmail.com

Abstract. H^2-MAC was proposed to increase efficiency over HMAC by omitting its outer key, and keep the advantage and security of HMAC at the same time. However, as pointed out by the designer, the security of H^2-MAC also depends on the secrecy of the intermediate value (the equivalent key) of the inner hashing. In this paper, we propose an efficient method to break H^2-MAC, by using a generalized birthday attack to recover the equivalent key, under the assumption that the underlying hash function is secure (weak collision resistance). We can successfully recover the equivalent key of H^2-MAC in about $2^{n/2}$ on-line MAC queries and $2^{n/2}$ off-line MAC computations with great probability. Moreover, we can improve the attack efficiency by reducing the on-line MAC queries, which can't be done concurrently. This attack shows that the security of H^2-MAC is totally dependent on the (weak) collision resistance of the underlying hash function, instead of the PRF-AX of the underlying compression function in the origin security proof of H^2-MAC.

Keywords: H^2-MAC, Equivalent Key Recovery, Weak Collision Resistance, Birthday Paradox.

1 Introduction

HMAC [2, 1], a derivative of NMAC, is a practically and commonly used, widely standardized MAC construction nowadays. HMAC has two advantages. First, HMAC can directly make use of current hash functions, the most widely used ones are based on Merkle-Damgård construction [4, 7], as black boxes. Second, it is provable secure under the assumption that the compression function of the underlying hash function is a pseudo random function (PRF) [1].

For an iterated hash function H with Merkle-Damgård construction, HMAC is defined with secret prefix approach [9], by

$$\mathrm{HMAC}_{(k_{\mathrm{in}},\, k_{\mathrm{out}})}(M) = H(k_{\mathrm{out}}||H(k_{\mathrm{in}}||M))$$

where M is an input message with arbitrary length, k_{in} and k_{out} are secret b-bit keys derived from a base key K.

However, HMAC has a drawback of managing its secret keys. It has to call the secret keys twice to complete the MAC computation. In ISC 2009, Yasuda proposed H^2-MAC [15], a variant of HMAC, which aims to remedy the drawback of HMAC and keep its advantages and security at the same time. H^2-MAC is defined by removing the outer key of HMAC.

H^2-MAC is proven to be a secure PRF (pseudorandom function) under the assumption that the underlying compression function is a PRF-AX [15].

This year, Wang [11] proposed an equivalent key recovery attack to H^2-MAC instantiated with the broken MD5 [8, 12, 14], combining the technologies used in [3] and [13], with complexity about 2^{97} on-line MAC queries.

Our contributions. We propose the first equivalent key recovery attack that breaks the security of H^2-MAC instantiated with secure hash functions, without related key setting. The attack is based on the assumption that the underlying hash function is (weak) collision resistance (WCR), which is stronger than the PRF assumption of the underlying compression function in origin security proof in [15]. Moreover, our attack is suitable to all of the H^2-MACs instantiated with secure[1] iterated hash functions, since our attack is based on the birthday paradox. This attack is also applicable to NMAC [6], in a related key setting.

We implement the equivalent key recovery attack to H^2-MAC through a generalized birthday attack, which contains two groups. First, we get the corresponding MAC values of H^2-MAC by on-line queries in group G_1, using different messages. Second, we directly compute the values of $H(H(C||m))^2$, called H^2, in group G_2 through off-line, where Cs and ms can be both randomly generated. If the number of queries in G_1 is $2^{n/2}$ and the number of computation in G_2 is also $2^{n/2}$, then, there is a pair (m, m') of that the inner hashing part of H^2-MAC and H^2 that equates with great probability [5]. Hence, the equivalent key of H^2-MAC is recovered by computing the corresponding value of H^2.

Since the on-line MAC queries to the H^2-MAC oracle can't be done concurrently, and the off-line computations of the H^2 can be completed in parallel, we can improve the attack efficiency by reducing the on-line queries and increasing the number of off-line computations at the same time, without reducing the success ratios of the attack. Moreover, once the off-line computation is done, it can be reused while recovering the equivalent key of other H^2-MACs with different keys.

Organization of this paper. We introduce some preliminaries and background, such as birthday paradox, in section two. In section three, We break the security of H^2-MAC by using a generalized birthday attack with two groups, based on the assumption that the underlying hash function is (weak) collision resistance. Further, in section four, we show some optimizations, such as enlarging the success probability of the equivalent key recovery attack to H^2-MAC. We conclude the paper in the last section.

[1] In this paper, a secure hash function means it is WCR.

[2] The secret key of H^2-MAC is replaced with a constant, for example, the IV of the underlying hash function.

2 Preliminaries

In this section, we first present some notations, and then we recall the birthday paradox in brief, at last, we present a brief description of H^2-MAC.

2.1 Notations

Let h be a compression function mapping $\{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$, and let H be a concrete hash function mapping $\{0,1\}^* \rightarrow \{0,1\}^n$. Let IV be the initial chaining variable of H. Let k denote a secret key with b bits. K denote a secret key with n bits. $x||y$ denotes the concatenation of two bit strings x and y. $|G|$ denotes the number of elements of the set G. $\oplus$ means the bit wise exclusive OR. $pad(M)$ denotes the padding bits of M in Merkle-Damgård style. H^2 means that the secret key to H^2-MAC is replaced with a constant C or known to everybody, hence, H^2 can be also viewed as the application of the underlying hash function H twice.

2.2 Birthday Paradox

The famous birthday paradox is stated as follows [5]: "Let r be the number of the students in a classroom and let $q(r)$ be the probability that at least two students in this classroom have the same birthday. The minimal value of r is 23 for $q(r) \geq 1/2$."

A generalized variant. Given two groups G_1 with r elements, G_2 with s elements drawn uniformly and independently at random from $\{0,1\}^n$, find $x_1 \in G_1$ and $x_2 \in G_2$, such that $x_1 = x_2$.

The probability $\Pr(|G_1 \cap G_2|) = i$ that there are i distinct elements in the intersection of the two groups is denoted by $P(2^n, r, s, i)$. $P(2^n, r, s, i)$ converges towards a Poisson distribution $\wp_\lambda(i)$ with parameter λ, where $r \cdot s / 2^n \rightarrow \lambda$, $r, s, 2^n \rightarrow +\infty$ [5].

A solution x_1, x_2 exists with great probability once $r \times s \gg 2^n$ holds, and if the list sizes are favourably chosen, the complexity of the optimal algorithm is $O(2^{n/2})$ [5, 10].

The birthday problem has numerous applications throughout cryptography and cryptanalysis, and the direct application is collision searching.

2.3 Brief Description of H^2-MAC

H^2-MAC [15] was proposed by Yasuda in ISC 2009, it is defined as

$$H^2\text{-MAC}_{(k)}(M) = H() \tag{1}$$

where k is a b-bit key[3]. It is a reduced version of HMAC by removing the outer key.

[3] In the origin paper, $k = K||pad$, where K is a n-bit key, pad is some $(b-n)$-bit fixed padding. Here, we use k just for simplifying the notation.

Compared with HMAC, H^2-MAC can also utilize the underlying hash functions as black box, it is also provable secure under the assumption that the underlying compression function is a PRF-AF [15]. Moreover, H^2-MAC can achieve higher performance and simpler key management over HMAC, especially for short messages, since it only access the secret key once.

However, the key reduction of H^2-MAC introduces another security problem, as pointed out by the designer [15], once the intermediate chaining variable of the inner hashing (the equivalent key of H^2-MAC) is leaked, it can be used to perform a selective forgery attack. Hence, the security of H^2-MAC is also dependent on the secrecy of intermediate chaining variable.

3 Breaking H^2-MAC Using Birthday Paradox

We call $I_k = H(k||M)$ the inner hashing of H^2-MAC, $Oh = H(I_k)$ the outer hashing of H^2-MAC. On-line birthday attack is launched by query the MAC or hash oracle on-line.

3.1 On-Line Birthday Attack for Verifiable Forgery Attack

If we apply the on-line birthday attack to the H^2-MAC oracle, after about $2^{n/2}$ queries, we may get a collision pair (M, M'), which satisfies $H^2\text{-MAC}(M) = H^2\text{-MAC}(M')$. It means $H^2\text{-MAC}(M||pad(M)||x) = H^2\text{-MAC}(M'||pad(M')||x)$ always holds, for arbitrary message x. Hence, we can generate verifiable forgeries to H^2-MAC after a collision pair of H^2-MAC is found. We first query the MAC value of $M||pad(M)||x$, and we get the very MAC value for $M'||pad(M')||x$, eventually.

After $2^{n/2}$ on-line queries, any verifiable forgery to H^2-MAC, based on the collision pair (M, M'), can be made with one additional on-line query.

3.2 Equivalent Key Recovery Attack to H^2-MAC

If we know the inner intermediate chaining variable of H^2-MAC, $I_k = H(k||M)$, we can construct any selective forgery attack to H^2-MAC, based on that. However, it seems that we can't get the value of inner hashing $H(k||M)$ for the application of outer hashing Oh.

Attack principle. To find a way out, we notice that if we consider the inner intermediate chaining variable (the equivalent key) of H^2-MAC as a n-bit input x, then we can view H^2-MAC as a simple hash of $H(x)$. Intuitively, find a collision pair that satisfies $H(x) = H(x')$ is easier than recover the secret key of a MAC, even if H is collision resistance. However, we can't use the birthday attack with one group to recover the equivalent key, because we can't know the value of x and x', even a collision pair (x, x') is found.

Fortunately, we can use the generalized birthday attack with two groups. If we can get a collision pair (M, M') that satisfies $H^2\text{-MAC}_{(k)}(M) = H^2_{(c)}(M')$, and further we have $H(k||M) = H(c||M')$[4], where c is a constant or known parameter set by us. Then, we get the very equivalent key K_e of H^2-MAC, since $K_e = H(k||M) = H(c||M')$. Finally, we know the value of c and M', hence, K_e can be easily computed.

So the equivalent key recovery attack to H^2-MAC is transformed to the problem of finding a collision pair in two groups, where the elements of one group must be computed by on-line query to H^2-MAC, and the elements of the other group can be computed directly through H^2 off-line. Thus problem is the generalized birthday attack with two groups (sometimes, it is also named as meet-in-the-middle attack).

Generalized Birthday Attack to Recover Equivalent Key of H^2-MAC Here, we apply the generalized birthday attack with two groups [5] to H^2-MAC and then recover its equivalent key $K_e = H(k||M_0)$.

We use 1-block message M_0s to generate the corresponding H^2-MAC values, and use 1-block message M_0's to generate the corresponding H^2 values. The overall strategy of equivalent key recovery attack to H^2-MAC is shown as follows.

1. Generate a group one G_1 with $r = 2^{n/2}$ elements, by computing the corresponding values of $H(H(c||M_0'))$ for r different cs and M_0's, which can be randomly generated. Specifically, c can be a pre-chosen constant.
2. Generate a group two G_2 with $s = 2^{n/2}$ elements, by querying the corresponding values to H^2-MAC oracle with the secret key k for s different M_0s, where M_0s are randomly generated.
3. There is a collision pair (M_0, M_0') that not only satisfies $H^2\text{-MAC}_k(M_0) = H_c^2(M_0')$, but also satisfies $H(k||M_0) = H(c||M_0')$ (an inner collision between H^2 and H^2-MAC happens), with great probability [5].
4. Since $H(k||M_0) = H(c||M_0')$, and we know the value of c and M_0', we can compute the value of $K_e = H(k||M_0) = H(c||M_0')$.
5. Let pad_0 and pad_1 be the padding bits of $k||M_0$ and $k||M_0||pad_0||x$, respectively, for arbitrary message x. Hence, we can generate the result of $H(k||M_0||pad_0||x)$ by computing $y = h(K_e, x||pad_1)$, then we compute $H(y)$ further, and finally we get the very value of $H^2\text{-MAC}(k||M_0||pad_0||x)$.

Why inner collision. In the above attack, an inner collision must be found first. The problem is why an inner collision must happens. If we remove the outer hashing of H^2-MAC, we can directly observer that a collision pair (M_0, M_0') can be found with great probability, after querying the oracle of $H(k||M_0)$ and $H(c||M_0')$ with enough times. Moreover, the application of outer hashing of H^2-MAC can't hide the existence of such inner collision.

How to judge the inner collision. After a collision pair (M_0, M_0') satisfying $H^2\text{-MAC}_k(M_0) = H_c^2(M_0')$ is found, we first generate the padding bits pad_0 for M_0 and M_0', where $pad_0 = pad(c||M_0')$. Further, we randomly generate a message x, and append x to $M_0||pad_0$ and $M_0'||pad_0$, respectively. We query

[4] Here, an inner collision happens between H^2-MAC and H^2.

the corresponding MAC value on-line to the H^2-MAC oracle for $M_0||pad_0||x$, and we compute the corresponding value for $M_0||pad_0||x$ off-line using H^2. After that, we further check whether $H^2\text{-MAC}_k(M_0||pad_0||x) = H_c^2(M_0'||pad_0||x)$ still holds. If so, (M_0, M_0') is also an inner collision pair between H^2-MAC and H^2, the attack succeeds. Otherwise, (M_0, M_0') is an outer collision, which will be simply discarded.

Success Probability. We calculate the success probability of the above attack. We notice that $r = s = 2^{n/2}$ (hence $\lambda = r{\cdot}s/2^n = 1$), the probability sp of that at least one inner collision happens is computed as

$$sp = 1 - P(2^n, r, s, 0) = 1 - \wp_\lambda(0) + \varepsilon = 1 - e^{-1} + \varepsilon \geq 0.632$$

where $\varepsilon \leq 10^{-5}$ [5].

Complexity analysis. The elements of group G_1 computed by H^2 need $2^{n/2}$ off-line H^2 computations. The elements of group G_2 computed by H^2-MAC need $2^{n/2}$ on-line H^2-MAC queries. We can store the values of both group using hash table. Then the above algorithm will require $O(2^{n/2})$ time and space.

After an inner collision pair (M_0, M_0') is found, we can apply $H_c^2(M_0')$ to compute the equivalent key of the H^2-MAC. Finally, we can use the recovered equivalent key k_e to launch any selective forgery attack to H^2-MAC without on-line query, based on M_0, which claims that the security of H^2-MAC is broken. Hence, we point out that the security of H^2-MAC is solely dependent on the (weak) collision resistance of the underlying hash function, not the strength of the used key.

However, it is interesting to notice that H^2-MAC is provable secure under the assumption of that the underlying compression function h is a PRF-AX [15], which means that (weak) collision resistance of the underlying hash function can be dropped. Thus proof and assumption obvious violate our result.

4 Some Optimizations over the attack

Here, we discuss some improvements over the equivalent key recovery attack to H^2-MAC, such as enlarging the success probability and achieving more parallelism in the attack.

4.1 Enlarging the success probability

In the above attack, the success probability of that at least one inner collision happens is at least 0.632, which is acceptable sometimes. However, we can enlarge the success probability, through doing more queries, since sp is determined by $P(2^n, r, s, 0)$, which converges towards a Poisson distribution with parameter λ.

For example, if we now want the success probability sp to be $sp \geq 1 - 10^{-4}$, by changing only r and s (but preserving $r = s$ both powers of 2), we can choose $r = s = 2^{n/2+2}$, and then $\lambda = r \cdot s/2^n = 16$, finally, we have

$$sp = 1 - P(2^n, 2^{n/2+2}, 2^{n/2+2}, 0) = 1 - \wp_\lambda(0) + \varepsilon = 1 - e^{-16} + \varepsilon \geq 1 - 10^{-4}$$

where $\varepsilon \leq 10^{-5}$.

4.2 Implementing more parallelism

We notice that the on-line queries to H^2-MAC can't be done concurrently, however, the off-line computations of H^2 can be executed in parallel. For a hash function with $n = 128$ bits result, the on-line queries of $2^{n/2} = 2^{64}$ times can't be completed in practical time. Even if the H^2-MAC oracle can reply at the speed of 10Gbit/second, this could require continuous replying during 25,000 years. However, the 2^{64} off-line computation of H^2 is achievable by utilizing the parallelism computation. For example, if use 1 million PCs in Internet, this work may be done within 9 days.

In a extreme way, we hold the success probability of the equivalent key recovery attack to H^2-MAC with 0.632, but we reduce heavily the on-line queries to the H^2-MAC oracle, at the cost of increasing the off-line computation of H^2. We set $s = 2^{n/4}$ and $r = 2^{3 \cdot n/4}$ (hence $\lambda = r \cdot s/2^n = 1$). The success probability sp of that at least one inner collision happens is computed as

$$sp = 1 - P(2^n, 2^{3\cdot n/4}, 2^{n/4}, 0) = 1 - \wp_\lambda(0) + \varepsilon = 1 - e^{-1} + \varepsilon \geq 0.632$$

where $\varepsilon \leq 10^{-5}$, and $n = 128$.

However, this work still can't be done in practical, since $2^{3\cdot n/4} = 2^{96}$ computation s of H^2 for $n = 128$ is still out of reach.

To do this attack more practically, we reduce the on-line queries to H^2-MAC to $s = 2^{n/2-10} = 2^{54}$ times, which can be done successfully. At the same time, we increase the off-line computations of H^2 to $r = 2^{n/2+10} = 2^{74}$ times, which is also reachable by utilizing computing parallelism. Since $\lambda = r \cdot s/2^n = 1$, the success probability sp of that at least one inner collision happens can be still computed as

$$sp = 1 - P(2^n, 2^{n/2+10}, 2^{n/2-10}, 0) = 1 - \wp_\lambda(0) + \varepsilon = 1 - e^{-1} + \varepsilon \geq 0.632$$

where $\varepsilon \leq 10^{-5}$, and $n = 128$.

5 Conclusion

We recover the equivalent key of H^2-MAC through applying a generalized birthday attack with two groups, based on the assumption of that the underlying hash function is WCR. We can recover thus key in about $2^{n/2}$ on-line queries to H^2-MAC and $2^{n/2}$ off-line H^2 computations. Moreover, this attack can be further optimized, the success probability can be improved by doing more queries and computations, and the on-line queries can also be reduced to the extent of practicality by increasing the off-line computations, which can be done in parallel. Our attack shows that the security of H^2-MAC is totally dependent on the WCR of the underlying hash function, which claims that the security of H^2-MAC is totally broken. WCR is a stronger basis than the assumption of that the underlying compression function is a PRF in the origin paper to prove the security of H^2-MAC [15].

Acknowledgement

This work was partially supported by the program “Core Electronic Devices, High-end General Purpose Chips and Basic Software Products” in China (No. 2010ZX01037-001-001), and supported by the 973 program of China under contract 2007CB311202, and by National Science Foundation of China through the 61070228 project.

References

1. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006, Lecture Notes in Computer Science, vol. 4117, pp. 602–619. Springer Berlin / Heidelberg (2006)
2. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) Advances in Cryptology CRYPTO' 96, Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer Berlin / Heidelberg (1996)
3. Contini, S., Yin, Y.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.) Advances in Cryptology ASIACRYPT 2006, Lecture Notes in Computer Science, vol. 4284, pp. 37–53. Springer Berlin / Heidelberg (2006)
4. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) Advances in Cryptology CRYPTO' 89 Proceedings, Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer Berlin / Heidelberg (1990)
5. Girault, M., Cohen, R., Campana, M.: A Generalized Birthday Attack. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmller, G., Stoer, J., Wirth, N., Gnther, C. (eds.) Advances in Cryptology EUROCRYPT 88, Lecture Notes in Computer Science, vol. 330, pp. 129–156. Springer Berlin / Heidelberg (1988)
6. Liu, F., Shen, C., Xie, T.: On the Security of NMAC and Its Variants. submission to Cryptology ePrint Archive (2011)
7. Merkle, R.: One Way Hash Functions and DES. In: Brassard, G. (ed.) Advances in Cryptology CRYPTO 89 Proceedings, Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer Berlin / Heidelberg (1990)
8. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321 (Informational) (Apr 1992), http://www.ietf.org/rfc/rfc1321.txt, updated by RFC 6151
9. Tsudik, G.: Message authentication with one-way hash functions. SIGCOMM Comput. Commun. Rev. 22, 29–38 (October 1992)
10. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) Advances in Cryptology CRYPTO 2002, Lecture Notes in Computer Science, vol. 2442, pp. 288–304. Springer Berlin / Heidelberg (2002)
11. Wang, W.: Equivalent Key Recovery Attack on H^2-MAC Instantiated with MD5. In: Kim, T.h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) Information Security and Assurance, Communications in Computer and Information Science, vol. 200, pp. 11–20. Springer Berlin Heidelberg (2011)
12. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) Advances in Cryptology EUROCRYPT 2005, Lecture Notes in Computer Science, vol. 3494, pp. 561–561. Springer Berlin / Heidelberg (2005)

13. Wang, X., Yu, H., Wang, W., Zhang, H., Zhan, T.: Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, Lecture Notes in Computer Science, vol. 5479, pp. 121–133. Springer Berlin / Heidelberg (2009)
14. Xie, T., Liu, F., Feng, D.: Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5?. Eurocrypt 2009, Poster Session, Cryptology ePrint Archive, Report 2008/391 (2008), http://eprint.iacr.org/
15. Yasuda, K.: HMAC without the "Second" Key. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C. (eds.) Information Security, Lecture Notes in Computer Science, vol. 5735, pp. 443–458. Springer Berlin / Heidelberg (2009)