維基百科

SHA-1

維基百科，自由的百科全書

SHA-1（英語：Secure Hash Algorithm 1，中文名：安全雜湊演算法1）是一種密碼雜湊函式，美國國家安全局設計，並由美國國家標準技術研究所（NIST）發布為聯邦資料處理標準（FIPS）[2]。SHA-1可以生成一個被稱為訊息摘要的160位（20位元組）雜湊值，雜湊值通常的呈現形式為40個十六進位數。

2005年，密碼分析人員發現了對SHA-1的有效攻擊方法，這表明該演算法可能不夠安全，不能繼續使用[3]，自2010年以來，許多組織建議用SHA-2或SHA-3來替換SHA-1[4][5][6]。Microsoft[7]、Google[8]以及Mozilla[9][10][11]都宣布，它們旗下的瀏覽器將在2017年停止接受使用SHA-1演算法簽章的SSL憑證。

2017年2月23日，CWI Amsterdam與Google宣布了一個成功的SHA-1碰撞攻擊[12][13]，發布了兩份內容不同但SHA-1雜湊值相同的PDF檔案作為概念證明。[14]

2020年，針對SHA-1的選擇字首衝突攻擊已經實際可行。建議盡可能用SHA-2或SHA-3取代SHA-1。[15][16]

SHA-1

概述	
設計者	美國國家安全局
首次發布	1993年（SHA-0） 1995年（SHA-1）
系列	安全雜湊演算法家族
認證	FIPS PUB 180-4， CRYPTREC（監測）
密碼細節	
摘要長度	160位元
分組長度	512位元
結構	Merkle–Damgård結構
重複回數	80
最佳公開破解	
A 2011 attack by Marc Stevens can produce hash collisions with a complexity between $2^{60.3}$ and $2^{65.3}$ operations.[1] 截至2015年10月，沒有實際的碰撞公開。	

目錄

- **SHA-0和SHA-1**
  - SHA-0的破解
  - SHA-1的破解
- **SHA-1演算法**
- **參考文獻**

SHA-0和SHA-1

最初載明的演算法於1993年發布，稱做安全雜湊標準（Secure Hash Standard），FIPS PUB 180。這個版本現在常被稱為SHA-0。它在發布之後很快就被NSA撤回，並且由1995年發布的修訂版本FIPS PUB 180-1（通常稱為SHA-1）取代。SHA-1和SHA-0的演算法只在壓縮函式的訊息轉換部份差了一個位元的循環位移。根據NSA的說法，它修正了一個在原始演算法中會降低雜湊安全性的弱點。然而NSA並沒有提供任何進一步的解釋或證明該弱

點已被修正。而後SHA-0和SHA-1的弱點相繼被攻破，SHA-1似乎是顯得比SHA-0有抵抗性，這多少證實了NSA當初修正演算法以增進安全性的聲明。

SHA-0和SHA-1可將一個最大2^{64}位元的訊息，轉換成一串160位元的訊息摘要；其設計原理相似於MIT教授Ronald L. Rivest所設計的密碼學雜湊演算法MD4和MD5。

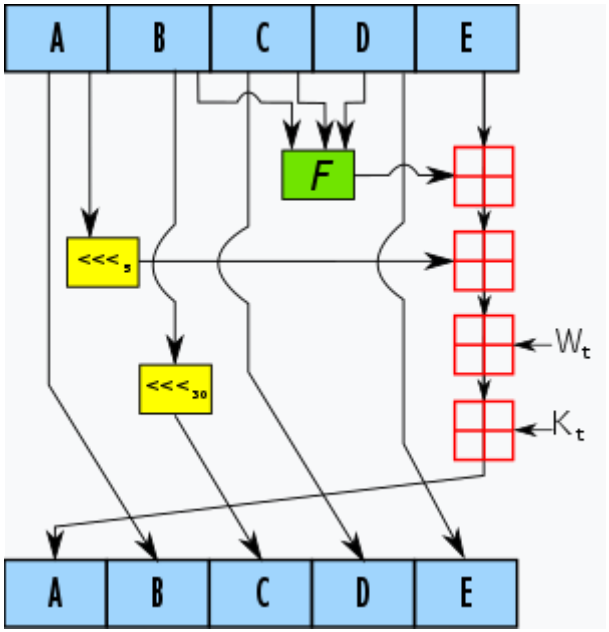
SHA-0的破解

在CRYPTO 98上，兩位法國研究者提出一種對SHA-0的攻擊方式[17]：在2^{61}的計算複雜度之內，就可以發現一次碰撞（即兩個不同的訊息對應到相同的訊息摘要）；這個數字小於生日攻擊法所需的2^{80}，也就是說，存在一種演算法，使其安全性不到一個理想的雜湊函式抵抗攻擊所應具備的計算複雜度。

2004年時，Biham和Chen也發現了SHA-0的近似碰撞，也就是兩個訊息可以雜湊出幾乎相同的數值；其中162位元中有142位元相同。他們也發現了SHA-0的完整碰撞（相對於近似碰撞），將本來需要80次方的複雜度降低到62次方。

2004年8月12日，Joux, Carribault, Lemuet和Jalby宣布找到SHA-0演算法的完整碰撞的方法，這是歸納Chabaud和Joux的攻擊所完成的結果。發現一個完整碰撞只需要2^{51}的計算複雜度。他們使用的是一台有256顆Itanium2處理器的超級電腦，約耗80,000 CPU工時。

2004年8月17日，在CRYPTO 2004的Rump會議上，王小雲，馮登國（Feng）、來學嘉（Lai），和於紅波（Yu）宣布了攻擊MD5、SHA-0和其他雜湊函式的初步結果。他們攻擊SHA-0的計算複雜度是2^{40}，這意味的他們的攻擊成果比Joux還有其他人所做的更好。請參見MD5安全性。



SHA-1壓縮演算法中的一個迴圈。A, B, C, D和E是這個state中的32位元文字；*F*是會變化的非線性函式；$<<<_n$代表bit向左循環移動n個位置。n因操作而異。田代表modulo 2^{32}之下的加法，K_t是一個常數。

2005年二月，王小雲和殷益群、於紅波再度發表了對SHA-0破密的演算法，可在2^{39}的計算複雜度內就找到碰撞。

SHA-1的破解

鑒於SHA-0的破密成果，專家們建議那些計劃利用SHA-1實作密碼系統的人們也應重新考慮。在2004年CRYPTO會議結果公布之後，NIST即宣布他們將逐漸減少使用SHA-1，改以SHA-2取而代之。

2005年，Rijmen和Oswald發表了對SHA-1較弱版本（53次的加密迴圈而非80次）的攻擊：在2^{80}的計算複雜度之內找到碰撞。

2005年二月，王小雲、殷益群及於紅波發表了對完整版SHA-1的攻擊，只需少於2^{69}的計算複雜度，就能找到一組碰撞。（利用生日攻擊法找到碰撞需要2^{80}的計算複雜度。）

這篇論文的作者們寫道；「我們的破密分析是以對付SHA-0的差分攻擊、近似碰撞、多區塊碰撞技術、以及從MD5演算法中尋找碰撞的訊息更改技術為基礎。沒有這些強力的分析工具，SHA-1就無法破解。」此外，作者還展示了一次對58次加密迴圈SHA-1的破密，在2^{33}個單位元運算內就找到一組碰撞。完整攻擊方法的論文發表在2005年八月的CRYPTO會議中。

殷益群在一次面談中如此陳述：「大致上來說，我們找到了兩個弱點：其一是前置處理不夠複雜；其二是前20個迴圈中的某些數學運算會造成不可預期的安全性問題。」

2005年8月17日的CRYPTO會議尾聲中王小雲、姚期智、姚儲楓再度發表更有效率的SHA-1攻擊法，能在2^{63}個計算複雜度內找到碰撞。

2006年的CRYPTO會議上，Christian Rechberger和Christophe De Cannière宣布他們能在容許攻擊者決定部分原訊息的條件之下，找到SHA-1的一個碰撞。

在密碼學的學術理論中，任何攻擊方式，其計算複雜度若少於暴力搜尋法所需要的計算複雜度，就能被視為針對該密碼系統的一種破密法；但這並不表示該破密法已經可以進入實際應用的階段。

就應用層面的考量而言，一種新的破密法出現，暗示著將來可能會出現更有效率、足以實用的改良版本。雖然這些實用的破密法版本根本還沒誕生，但確有必要發展更強的雜湊演算法來取代舊的演算法。在「碰撞」攻擊法之外，另有一種反譯攻擊法（Pre-image attack），就是由雜湊出的字串反推原本的訊息；反譯攻擊的嚴重性更在碰撞攻擊之上，但也更困難。在許多會應用到密碼雜湊的情境（如用戶密碼的存放、檔案的數位簽章等）中，碰撞攻擊的影響並不是很大。舉例來說，一個攻擊者可能不會只想要偽造一份一模一樣的檔案，而會想改造原來的檔案，再附上合法的簽章，來愚弄持有公鑰的驗證者。另一方面，如果可以從密文中反推未加密前的使用者密碼，攻擊者就能利用得到的密碼登入其他使用者的帳戶，而這種事在密碼系統中是不能被允許的。但若存在反譯攻擊，只要能得到指定使用者密碼雜湊過後的字串（通常存在影檔中，而且可能不會透露原密碼資訊），就有可能得到該使用者的密碼。

2017年2月23日，Google公司公告宣稱他們與CWI Amsterdam合作共同建立了兩個有著相同的SHA-1值但內容不同的PDF檔案，這代表SHA-1演算法已被正式攻破。[18]

2020年，針對SHA-1的選擇字首衝突攻擊已經實際可行。建議盡可能用SHA-2或SHA-3取代SHA-1。在數位簽章領域，用更安全的SHA-2或SHA-3替換SHA-1已經變得急迫。[19][20]

SHA-1演算法

以下是SHA-1演算法的虛擬碼：

```
Note: All variables are unsigned 32 bits and wrap modulo 2^32 when calculating
```

```
Initial variables:
h0 := 0x67452301
h1 := 0xEFCDAB89
h2 := 0x98BADCFE
h3 := 0x10325476
h4 := 0xC3D2E1F0
```

```
Pre-processing:
append the bit '1' to the message
append k bits '0', where k is the minimum number >= 0 such that the resulting message
    length (in bits) is congruent to 448(mod 512)
append length of message (before pre-processing), in bits, as 64-bit big-endian integer
```

```
Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15
```

```
    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w[i] := (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1
```

```
    Initialize hash value for this chunk:
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
```

```
    Main loop:
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f := (b and c) or ((not b) and d)
            k := 0x5A827999
        else if 20 ≤ i ≤ 39
            f := b xor c xor d
            k := 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f := (b and c) or (b and d) or(c and d)
            k := 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f := b xor c xor d
            k := 0xCA62C1D6
```

```
        temp := (a leftrotate 5) + f + e + k + w[i]
        e := d
        d := c
        c := b leftrotate 30
        b := a
        a := temp
```

```
    Add this chunk's hash to result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e
```

```
Produce the final hash value (big-endian):
digest = hash = h0 append h1 append h2 append h3 append h4
```

上述關於f運算式列於FIPS PUB 180-1中，以下替代運算式也許也能在主要迴圈裡計算f：

```
(0  ≤ i ≤ 19): f := d xor (b and (c xor d))         (alternative)

(40 ≤ i ≤ 59): f := (b and c) or (d and (b or c))   (alternative 1)
(40 ≤ i ≤ 59): f := (b and c) or (d and (b xor c))  (alternative 2)
(40 ≤ i ≤ 59): f := (b and c) + (d and (b xor c))   (alternative 3)
```

參考文獻

1. Marc Stevens. Attacks on Hash Functions and Applications (PDF). PhD thesis. 19 June 2012 [2016-12-15].（原始內容 (PDF)存檔於2017-03-18）.
2. 存档副本 (PDF). [2016-12-15].（原始內容存檔 (PDF)於2013-02-17）.
3. Schneier, Bruce. Schneier on Security: Cryptanalysis of SHA-1. February 18, 2005 [2016-12-15].（原始內容存檔於2017-04-14）.
4. NIST.gov - Computer Security Division - Computer Security Resource Center. [2016-12-15].（原始內容存檔於2011-06-25）.
5. Stevens1, Marc; Karpman, Pierre; Peyrin, Thomas. The SHAppening: freestart collisions for SHA-1. [2015-10-09].（原始內容存檔於2017-04-19）.
6. Bruce Schneier. SHA-1 Freestart Collision. Schneier on Security. 8 October 2015 [2016-12-15].（原始內容存檔於2017-01-28）.
7. Windows Enforcement of Authenticode Code Signing and Timestamping. Microsoft. 2015-09-24 [2016-08-07].（原始內容存檔於2016-10-05）.
8. Intent to Deprecate: SHA-1 certificates. Google. 2014-09-03 [2014-09-04].
9. Bug 942515 - stop accepting SHA-1-based SSL certificates with notBefore >= 2014-03-01 and notAfter >= 2017-01-01, or any SHA-1-based SSL certificates after 2017-01-01. Mozilla. [2014-09-04].（原始內容存檔於2014-09-07）.
10. CA:Problematic Practices - MozillaWiki. Mozilla. [2014-09-09].（原始內容存檔於2017-05-06）.
11. Phasing Out Certificates with SHA-1 based Signature Algorithms | Mozilla Security Blog. Mozilla. 2014-09-23 [2014-09-24].（原始內容存檔於2017-04-25）.
12. CWI, Google announce first collision for Industry Security Standard SHA-1. [2017-02-23].（原始內容存檔於2017-02-23）.
13. Announcing the first SHA1 collision. Google Online Security Blog. 2017-02-23.（原始內容存檔於2017-04-24）.
14. SHAttered. [2017-02-23].（原始內容存檔於2017-04-12）.
15. 存档副本 (PDF). [2020-09-09].（原始內容存檔 (PDF)於2020-09-10）.
16. Critical flaw demonstrated in common digital security algorithm. media.ntu.edu.sg. [2020-09-08].（原始內容存檔於2020-04-19）（美國英語）.
17. Chabaud and Joux, 1998
18. Announcing the first SHA1 collision. Google Online Security Blog. 2017-02-23 [2017-02-24].（原始內容存檔於2017-04-24）.
19. 存档副本 (PDF). [2020-09-09].（原始內容存檔 (PDF)於2020-09-10）.
20. Critical flaw demonstrated in common digital security algorithm. media.ntu.edu.sg. [2020-09-08].（原始內容存檔於2020-04-19）（美國英語）.