又一个安全屏障坍塌，SHA-1 已被实际攻陷

潘文伦

2017 年 2 月 23 日，CWI Amsterdam 与 Google 的研究人员宣布找到了实现 SHA-1 碰撞的实用性技术方案，并展示了世界上第一例 SHA-1 碰撞实例 [1]，该研究标志着 SHA-1 算法在长期理论研究预警之后终于走向生命末期。

1995 年，美国国家标准技术研究所（NIST）确定 SHA-1 为哈希函数标准，并被广泛使用。大量理论分析表明该算法安全性存在问题，且 NIST 在 2011 年正式弃用该算法，但鉴于找到实际碰撞的计算复杂度及代价极大，似乎不切实际，因此，目前 SHA-1 仍广泛应用于文件和 TLS 证书签名，以及软件完整性校验等。不过，现在一切已成为过去，研究人员们找到了构造碰撞切实可行的方法：只需 9,223,372,036,854,775,808 约 2^{63} 次 SHA-1 调用，即大概 6500 个 CPU 年与 110 个 GPU 年的计算量就可构造出碰撞。受该碰撞攻击的影响，目前很多基于 SHA-1 的应用都不再安全，如源代码管理系统 Git 目前依然使用 SHA-1 作为文件的消息指纹、安卓设备中 APP、升级包等模块的签名机制、电子邮件 PGP/GPG 签名、文件完整性校验等。下面简要介绍 SHA-1 攻击事件的影响及攻击历史、攻击方法等。

1. 事件的影响及反应

直接受影响的应用

1. 基于 SHA-1 提供文件完整性和一致性的应用（File integrity and identification，SHA1(file)），例如

a. 使用 SHA-1 来保证完整性和提供备份功能的 Git、Mercurial、SVN 等版本控制系统；

b. 为保证正确的软件下载和升级更新（如操

作系统 ISO 的检验码、软件安装程序的校验码等），软件开发者基于 SHA-1 提供的软件指纹；

2. 提供和使用基于 SHA-1 数字签名的应用（Sign(SHA1(salt, file))）：

a. 时间戳（Timestamping）；

b. 文件认证（文件的数字签名）；

c. 证书数字签名（基于 IETF 的 TLS/SSL 1.2 及以下版本的协议并基于 SHA-1 算法颁发的证书），如图 1 所示；

●尚未更新的 Windows7 加密文件系统证书、微软根证书颁发机构（Microsoft Root Certificate Authority）；

●高达 10% 的信用卡支付系统（招商银行一网通支付平台 API 开发商户校验码）；

●微信（财付通支付商户平台）、支付宝（蚂蚁金服商户平台）、百度（百度钱包企业版商户系统）；

●基于 PGP/GPG 协议的邮件的数字签名；

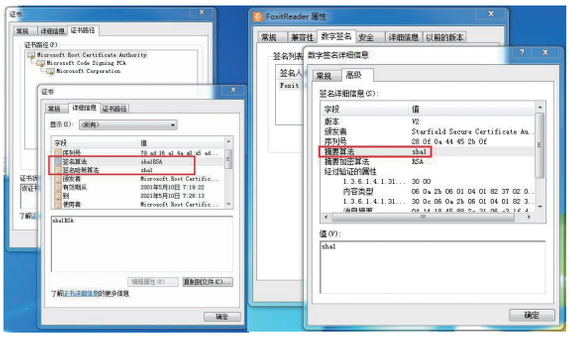
因此由 SHA-1 提供证书保护的文件、邮件、软件、网站将被视为不安全的。

3. 基于 SHA-1 的承诺（Commitments, SHA1(commit, r)）；

4. 基于 SHA-1 的取证、鉴证（Forensics, SHA1(key, evidence)）；

5. 基于 SHA-1 的工作量证明（Proof-of-work）系统，如基于 SHA-1 的电子货币；

6. 协议中基于 SHA-1 的随机预言机（random Oracle），例如挑战应答协议中的随机 Oracle（SHA1(key, random_challenge)）。



(a) 微软根证书颁发机构　(b) FoxitReader 软件的数字签名

图 1：受影响的应用

不直接受影响的应用（不要求强抗碰撞安全的应用）

1. 带密钥的 SHA-1 散列应用（Keyed Hash Application based on SHA-1, SHA1(key, message)）；

2. 基于 SHA-1 的密钥派生应用（Key derivation applications of hash functions, SHA1(salt, password)）；

3. 基于 SHA-1 的密码存储（Password Storage, SHA1(salt, password)）；

4. 基于 SHA-1 的伪随机数生成器（Pseudorandom generation, SHA1(key, nonce, 1), SHA1(key, nonce, 2), …）；

5. 基于 SHA-1 的安全密钥生成（Generation of secure keys, SHA1(physical entropy)）。

2. 散列函数简介及 SHA-1 分析历史

密码散列函数

密码散列函数是一种提供“完整性、认证性和随机性”的散列函数（也被称为哈希函数）。散列函数将任意长度（一些密码散列函数的应用考虑安全性的需求，会对消息长度

有所限制）的消息映射为固定长度的摘要（比如 160 比特、256 比特或 512 比特等）。密码散列函数 H 需要根据应用需求满足一定抗攻击的安全性。抗攻击的安全性分为以下三类：对于输出的消息摘要长度为 n 比特的密码散列函数 H：

1. 抗碰撞攻击：找到任意两个不同的消息 x 及 x' 使得它们具有相同的散列值，即 H(x) = H(x')，是计算上困难的。安全的密码散列函数应当使碰撞攻击的复杂度达到 $2^{n/2}$。

2. 抗原像攻击：对于给定的消息摘要 V，找到消息 x 满足 H(x)=V 是计算上困难的。安全的密码散列函数应当使原像攻击的复杂度达到 2^n。

3. 抗第二原像攻击：对于给定的消息 x，找到另一个不同的消息 x'，使得 H(x)=H(x')，是计算上困难的。安全的密码散列函数应当使第二原像攻击的复杂度达到 2^n。

SHA-1 安全散列算法简介

SHA-1（英语：Secure Hash Algorithm 1，中文名：安全散列算法 1）是一种密码散列函数，由美国国家安全局设计， 并由 NIST 发布为联邦数据处理标准 (FIPS)[2]。SHA-1 生成的消息摘要长度为 160 比特。

SHA-1 基于 NIST 早两年发布的 SHA-0 安全散列算法设计。SHA-0 在发布之后很快就被 NSA 撤回，并由 1995 年发布的修订版本 FIPS PUB 180-1（通常称为 SHA-1）替换。SHA-1、SHA-0 的设计原理相似于早期的密码学散列算法 MD4 和 MD5，它们都采用了一种称为 Merkel-Damgård 结构 (MD-structure)，即使用相同的一个压缩函数进行迭代，来处理经过填充和分段后的消息块序列，它们对消息的处理过程如下：

1. 填充和分块：将输入的消息 M 进行长度填充（填充过程可逆），使其长度为消息块长度 b 的整数倍，从而将其分成长度固定为 b 的消息块，即 $M=m_1 \| m_2 \| \ldots\ldots \| m_e$ ；

2. 初始化：将链接状态初始化为一个算法规范指定的固定初始化值 IV ；

3. 迭代压缩：链接状态由一个压缩函数 h 进行迭代更新，每次更新处理一个消息块，直到处理完最后一个消息块 m_l；

4. 输出：链接状态经由一个输出变换 g 处理，生成散列值输出。

SHA-1 的压缩函数 h 包含 80 个循环的状态更新变换，如图 2 所示。其中 A,B,C,D 和 E 是这个 state 中的 32 位文字；F 是会变化的非线性函数；<<n 代表 bit 向左循环移动 n 个位置， n 因操作而异。田代表模 2^{32} 加法，K_t 是一个常数。

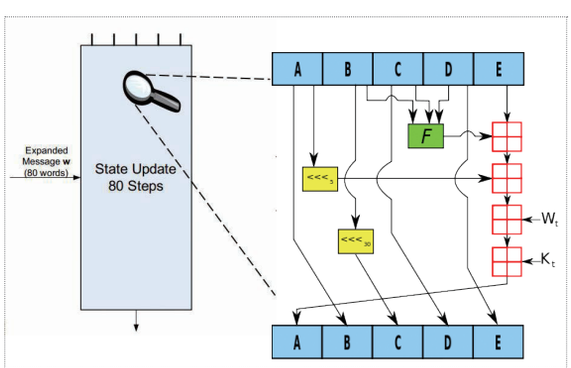


图 2：SHA-1 的压缩函数

SHA-1 的攻击历史

1. 1995 年，NIST 公布 SHA-1[3]；

2. 2004 年，王小云团队在 CRYPTO 的 Rump 会议上宣布了攻击 MD5 和 SHA-0 等散列函数攻击结果，攻击 SHA-0 的复杂度为 2^{40}。鉴于此，专家们认为 SHA-1 的安全强度可能也存在问题，随后，NIST 宣布他们将逐渐减少使用 SHA-1，改以 SHA-2 替代；

3. 2005 年，王小云等在 CRYPTO 会议上发表论文，给出对完整版本 SHA-0 的碰撞攻击，计算复杂度为 2^{39} [4]，并同时发表了对完整版 SHA-1 的碰撞攻击，计算复杂度为 2^{69}[5]。在 CRYPTO 2005 会议尾声及 CT-RSA 2006 会议上，王小云等声称已将对 SHA-1 碰撞攻击的复杂度降低到 2^{63}[6]，该结果在 2007 年被 Martin Cochran 等验证 [7]；

4. 2006 年，Christian Rechberger 等在 CRYPTO 的 Rump 会议上声称找到一个碰撞，该碰撞允许攻击者选定消息的一部分；

5. 2007 年，Mendel 等在 CRYPTO 2007 的 Rump 会议上声称找到了完整版 SHA-1 的碰撞，攻击复杂度约为 $2^{60.x}$ [8]；

6. 对非完整轮数的 SHA-1，人们也进行了大量分析 [9,10,11,12] 等；

7. 还有一些关于 SHA-1 的理论分析文献声称碰撞的计算复杂度可降低到更低的程度，但发布者缺少进一步的研究和分析的细节，人们很难验证攻击的正确性及是否能达到所声称的复杂度，故无法确定哪个结果是关于 SHA-1 碰撞攻击的最佳结果。2011 年，RFC6194 [13] 认为首次碰撞 [5] 是最优结果，复杂度为 2^{69}；

8. 2013 年，Stevens 等公布了一个关于完整版 SHA-1 的碰撞攻击，复杂度为 2^{61}[14]。在 EUROCRYPT 2013 会议上，Stevens 等进一步改进了对完整版 SHA-1 的攻击 [14]；但是，公开已知的碰撞尚未被实际构造出来。

9. 2016 年，Stevens 等在 EUROCRYPT 2016 会议上发表了对 SHA-1 底层压缩函数的全轮碰撞攻击，使用了一种从中间开始的技术 (using a start-from-the-middle approach) 并利用了一个高效的 GPU 计算框架。这一攻击只需要使用 64 个 GPU 计算 10 天，相当于在 GPU 上进行 $2^{57.5}$ 次 SHA-1 调用。随后，CABForum Ballot 152 立即撤回了其基于 SHA-1 签发的 HTTPS 证书，IETF 的 TLS 协议 1.3 版本完全弃用 SHA-1。

10. 2017 年 2 月 23 日，包括 Stevens 在内的 CWI Amsterdam 与 Google 的研究人员公布了第一个公开的碰撞，攻击复杂度为 $2^{63.1}$[1]。

3. 此次实际攻击的方法简介

此次实际攻击基于 Stevens 在 2013 年 EUROCRYPT 会议上发表的理论碰撞攻击 [14]。该理论攻击基于最优的联合局部碰撞 (optimal joint local-collsion) 分析技术。当时作者对攻击复杂度的理论估计是 2^{61} 次 SHA-1 函数的调用。这一攻击是至今已知的最优理论攻击。本次的实际攻击与理论攻击在计算复杂度的估计上有所差异，实际攻击的复杂度被估计为 $2^{63.1}$ 次 SHA-1 函数的调用。这一差异源于两个方面，一方面，在理论攻击的复杂度估计中并没有包含使用 GPU 进行计算时所固有的成比例的效率损失，也没有考虑跨

越多个数据中心进行分布式大规模计算时额外的效率损失。另一方面，构造第二个近似碰撞攻击（如下所述）在实际上比文献中的理论估计要复杂得多。

本次的攻击是一种相同前缀碰撞攻击(identical-prefix collision attack)。攻击过程大致如下：

1. 用一个相同的消息前缀 P 产生相同的链接状态值。

2. 从相同的链接状态值出发：

a. 构造第一个近似碰撞消息块对(near-collision block pair)：第一个消息块对 $(M_1^{(1)}, M_1^{(2)})$ 具有一定的差分，使得在输出的链接状态值上产生一个小的差分。

b. 构造第二个近似碰撞消息块对(near-collision block pair)：第二个消息块对 $(M_2^{(1)}, M_2^{(2)})$ 也具有一定的差分，此差分会使得输出的链接状态的差分与第一个近似碰撞产生的差分相抵消。

3. 在碰撞了的消息前缀下链接任意的消息后缀 S，从而产生整体的碰撞。即：

$SHA1(P \| M_1^{(1)} \| M_2^{(1)} \| S)=SHA1(P \| M_1^{(2)} \| M_2^{(2)} \| S)$.

其中的关键步骤是构造两个近似碰撞消息块对，构造的过程大致如下：

1. 选择合适的干扰向量 (selection of the disturbance vector)；

2. 构造非线性差分路径 (construction of the non-linear differential path)；

3. 确定差分路径对所有 SHA-1 压缩函数操作步上带来的攻击限制条件 (determine attack conditions over all steps)；

4. 找到除了差分路径带来的限制条件之外其它的额外限制条件，以便提前终止 (find additional conditions beyond fixed diff. path for early-stop)；

5. 若条件充分，确定对 SHA-1 压缩函数前面几个操作步上的攻击限制条件的可解性 (if necessary fix solvability of attack conditions over first few steps)；

6. 找到消息修改规则，来加快碰撞的搜索 (find message modification rules to speed-up collision search)；

7. 根据前面几个步骤确定近似碰撞攻击算法，写出攻击算法的实现代码 (write the attack algorithm)；

8. 运行攻击算法的实现代码，找到一对近似碰撞消息块 (finally, run the attack to find a near-collision block pair)。

具体地，构造第一个近似碰撞消息块对的算法直接采用了 [14] 一文中提供的公开源码，只是将使用的消息前缀 P 加以修改，并将其修改为适合在多个数据中心之间进行大规模并行计算的代码。

构造第二个近似碰撞消息块来抵消第一个带来的差分，所采用的步骤与构造第一个碰撞消息块的步骤相同（如上所示），但要比构造第一个消息块对要复杂得多。作者在这一过程中结合了很多已知的密码分析技术，并使用了 GPU 进行更高效的碰撞搜索，另外还引入两个额外的新技术：

1. 第一个技术为寻找额外的状态条件 (Find additional state conditions)，它使得攻击者可以利用在 SHA-1 的压缩函数第 23 步

之后的多个额外的差分路径来增加成功概率，并且在不阻碍使用提前终止技术的同时增加更多的自由度；

2. 第二个技术为使用 SAT 求解器来确定对 SHA-1 压缩函数前面几步之上引进的限制条件的可解性，这一技术是克服一个严重问题的必要条件。这个严重问题是由在 SHA-1 的压缩函数前面几步之上定义的一个高度过定义化的方程系统所导致的。这一问题若无法解决将威胁这次实际攻击项目的顺利完成。

总体而言，攻击综合使用了以下技术才得以成为现实：

1. 相同前缀碰撞攻击 (identical-prefix collision attack)；

2. 利用差分路径 (differential paths) 搜索构造碰撞；

3. 利用局部碰撞 (local collision)、干扰向量 (Disturbance Vector，DV) 解决消息块线性扩展带来的差分一致性问题；

4. 利用类似于中间相遇的方法来构造一个定制的差分路径 (tailored differential path，称为非线性路径，Non-Linear path) 来解决对任意干扰向量都能使得输入链接状态差分经过前面的 16 步之后能够链接到后面的局部碰撞差分上的问题（后面步骤上的局部碰撞差分被称为线性路径，(Linear path)，源于线性消息扩展）。

5. 去除冗余的最优差分路径 (differential paths)，使得到一个小而有效的一定量的差分路径条件上的缩减，从而使得所得到的方程系统更容易判断可解性；

6. 使用优化的联合局部碰撞分析 (Joint-local collision) 技术来确定攻击的整个方程系统；

7. 增加除差分路径带来的限制条件之外的额外的限制条件 (additional state conditions)，来增加成功概率，并且在不阻碍使用提前终止技术的同时增加更多的自由度；

8. 使用 SAT 求解器来确定在 SHA-1 压缩函数前几步之上得到的方程系统的可解性；

9. 利用中立比特 (neutral bits) 和 boomerages 作为消息修改规则 (message modification rules)，加快碰撞的搜索；

10. 基于 CPU-GPU 混杂架构的超大规模并行计算系统。

另外，此次实际攻击给出了两个具有相同散列值的 PDF 文档。要使这成为现实还有赖于作者提出的优化的 JPEG 技术。它们构造的两个 PDF 文档依赖于对 JPEG 图像的不同的解析，类似于 Gebhardt 等人的 TIFF 技术和 Albertini 等人的 JPEG 技术。在这些基本的技术基础之上，作者还使用了“very low-level 'wizard' JPEG features” 以便对于所有常见的 PDF 阅读器都有效，并允许非常大的 JPEG 文件以便构造多页 PDF 文件的碰撞。

更多工作的细节尚未公之于众，作者声明将给实施补救的安全措施留够足够多的时间，随后再将更多的技术细节公开，这包括他们优化的 JPEG 技术、攻击的源代码和使用的密码分析工具。

关于此次实际攻击方法的更多介绍请参见 SHAttered 网站上给出的技术文档 [1]。

4. 研究团队介绍

此次对 SHA-1 的实际攻击结果是在荷兰阿姆斯特丹 Centrum Wiskunde & Informatica (CWI) 的密码研究组与 Google 安全隐私和反滥用研究组中的研究人员长期合作之下得到的，研究人员包括：Marc Stevens (CWI Amsterdam)，Elie Bursztein (Google)，Pierre Karpman (CWI Amsterdam)，Ange Albertini (Google)，Yarik Markov (Google)，Alex Petit Bianco (Google)，Luca Invernizzi (Google)，Clement Baisse (Google)。本次攻击的关键人员是 CWI 年轻的 Marc Stevens，他的硕士、博士和博士后的工作全部围绕着攻击 MD5 和 SHA-1 展开，曾经在各大顶级密码会议上发表过相关的多个突破性的研究结果。基于 Marc Stevens 长期的攻击技术和攻击经验积累、Google 提供的强有力的计算资源支持、CWI 研究团队和 Google 研究团队的通力合作，SHA-1 算法的实际碰撞得以在 2017 年 2 月 23 日成为现实。

附：参考文献

[1] https://shattered.io/.

[2] https://zh.wikipedia.org/wiki/SHA-1.

[3] F. P. NIST, "180-1: Secure hash standard," 1995.

[4] X. Wang, H. Yu, and Y. L. Yin, Efficient Collision Search Attacks on SHA-0, pp. 1 - 16. Berlin,Heidelberg: Springer Berlin Heidelberg, 2005.

[5] X. Wang, Y. L. Yin, and H. Yu, Finding Collisions in the Full SHA-1, pp. 17 - 36. Berlin, Heidelberg:Springer Berlin Heidelberg, 2005.

[6] X. Wang, "Cryptanalysis of hash functions and potential dangers," Invited Talk at CT-RSA, vol.2006, 2006.

[7] M. Cochran, "Notes on the wang et al. 2^{63} SHA-1 differential path." Cryptology ePrint Archive, Report 2007/474, 2007. http://eprint.iacr.org/2007/474.

[8] F. Mendel, C. Rechberger, and V. Rijmen, "Update on SHA-1," rump session of CRYPTO, vol. 2007,2007.

[9] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby, Collisions of SHA-0 and Reduced SHA-1, pp. 36 - 57. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[10] C. De Canni è re and C. Rechberger, Finding SHA-1 Characteristics: General Results and Applications, pp. 1 - 20. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[11] C. De Canni è re, F. Mendel, and C. Rechberger, Collisions for 70-Step SHA-1: On the Full Cost of Collision Search, pp. 56 - 73. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[12] E.A.Grechnikov, "Collisions for 72-step and 73-step SHA-1: Improvements in the method of characteristics." Cryptology ePrint Archive, Report 2010/413, 2010. http://eprint.iacr.org/2010/ 413.

[13] T. Polk, L. Chen, S. Turner, and P. Hoffman, "Security considerations for the SHA-0 and SHA-1 message-digest algorithms," tech. rep., 2011.

[14] M. Stevens, New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis, pp. 245 - 261. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.