From Collisions to Chosen-Prefix Collisions Application to Full SHA-1

Gaëtan Leurent[1] and Thomas Peyrin[2,3]

[1] Inria, France
[2] Nanyang Technological University, Singapore
[3] Temasek Laboratories, Singapore

gaetan.leurent@inria.fr, thomas.peyrin@ntu.edu.sg

Abstract. A chosen-prefix collision attack is a stronger variant of a collision attack, where an arbitrary pair of challenge prefixes are turned into a collision. Chosen-prefix collisions are usually significantly harder to produce than (identical-prefix) collisions, but the practical impact of such an attack is much larger. While many cryptographic constructions rely on collision-resistance for their security proofs, collision attacks are hard to turn into break of concrete protocols, because the adversary has a limited control over the colliding messages. On the other hand, chosen-prefix collisions have been shown to break certificates (by creating a rogue CA) and many internet protocols (TLS, SSH, IPsec).
In this article, we propose new techniques to turn collision attacks into chosen-prefix collision attacks. Our strategy is composed of two phases: first a birthday search that aims at taking the random chaining variable difference (due to the chosen-prefix model) to a set of pre-defined target differences. Then, using a multi-block approach, carefully analysing the clustering effect, we map this new chaining variable difference to a colliding pair of states using techniques developed for collision attacks.
We apply those techniques to `MD5` and `SHA-1`, and obtain improved attacks. In particular, we have a chosen-prefix collision attack against `SHA-1` with complexity between $2^{66.9}$ and $2^{69.4}$ (depending on assumptions about the cost of finding near-collision blocks), while the best-known attack has complexity $2^{77.1}$. This is within a small factor of the complexity of the classical collision attack on `SHA-1` (estimated as $2^{64.7}$). This represents yet another warning that industries and users have to move away from using `SHA-1` as soon as possible.

Keywords: hash function; cryptanalysis; chosen-prefix collision; `SHA-1`; `MD5`

1 Introduction

Cryptographic hash functions are crucial components in many information security systems, used for various purposes such as building digital signature schemes, message authentication codes or password hashing functions. Informally, a cryptographic hash function H is a function that maps an arbitrarily long message

M to a fixed-length hash value of size n bits. Hash functions are classically defined as an iterative process, such as the Merkle-Damgård design strategy [17, 7]. The message M is first divided into blocks m_i of fixed size (after appropriate padding) that will successively update an internal state (also named chaining variable), initialised with a public initial value (IV), using a so-called compression function h. The security of the hash function is closely related to the security of the compression function.

The main security property expected from such functions is *collision resistance*: it should be hard for an adversary to compute two distinct messages M and M' that map to the same hash value $H(M) = H(M')$, where "hard" means not faster than the generic birthday attack that can find a collision for any hash function with about $2^{n/2}$ computations. A stronger variant of the collision attack, the so-called *chosen-prefix* collision attack is particularly important. The attacker is first challenged with two message prefixes P and P', and its goal is to compute two messages M and M' such that $H(P \parallel M) = H(P' \parallel M')$, where $\parallel$ denotes concatenation. Such collisions are much more dangerous than simple collisions in practice, as they indicate the ability of an attacker to obtain a collision even though random differences (thus potentially some meaningful information) were inserted as message prefix. In particular, this is an important threat in the key application of digital signatures: chosen-prefix collisions for `MD5` were demonstrated in [29], eventually leading to the creation of colliding X.509 certificates, and later of a rogue certificate authority [30]. Chosen-prefix collisions have also been shown to break important internet protocols, including TLS, IKE, and SSH [1], because they allow forgeries of the handshake messages.

`SHA-1` is one the most famous cryptographic hash functions in the world, having been the NIST and de-facto worldwide hash function standard for nearly two decades until very recently. Largely inspired by `MD4` [23] and then `MD5` [24], the American National Security Agency (NSA) first designed a 160-bit hash function `SHA-0` [18] in 1993, but very slightly tweaked one part of the design two years later to create a corrected version `SHA-1` [19]. It remained a standard until its deprecation by the NIST in 2011 (and disallowed to be used for digital signatures at the end of 2013). Meanwhile, hash functions with larger output sizes were standardized as `SHA-2` [20] and due to impressive advances in hash function cryptanalysis in 2004, in particular against hash functions of the `MD-SHA` family [32, 34, 35, 33], the NIST decided to launch a hash design competition that eventually led to the standardization in 2015 of `SHA-3` [21].

There has been a lot of cryptanalysis done on `SHA-1`. After several first advances on `SHA-0` and `SHA-1` [5, 3], researchers managed to find for the first time in 2004 a theoretical collision attack on the whole hash function, with an estimated cost of 2^{69} hash function calls [33]. This attack was extremely complex and involved many details, so the community worked on better evaluating and further improving the actual cost of finding a collision on `SHA-1` with these new techniques. Collisions on reduced-step versions of `SHA-1` were computed [8, 11], or even collisions on the whole `SHA-1` compression function [28], which eventually led to the announcement in 2017 of the first `SHA-1` collision [27].

Even though SHA-1 has been broken in 2004, it is still deployed in many security systems, because collision attacks do not seem to directly threaten most protocols, and migration is expensive. Web browsers have recently started to reject certificates with SHA-1 signatures, but there are still many users with older browsers, and many protocols and softwares that allow SHA-1 signatures. Concretely, it is still possible to buy a SHA-1 certificate from a trusted CA, and many email clients accept a SHA-1 certificate when opening a TLS connection. SHA-1 is also widely supported to authenticate TLS and IKE handshake messages.

Main SHA-1 cryptanalysis techniques. Attacks against SHA-1 are based on differential cryptanalysis, where an attacker manages to somewhat control the output of the compression function. Several important ideas were used to turn differential cryptanalysis into an effective tool against hash functions:

Linearization [5]. In order to build differential trails with high probability, a linearized version of the step function is used. Differential trails with a low-weight output difference δ_O can be used to find near-collisions in the compression function (i.e. two outputs that are close to a collision, the distance metric being for example the Hamming distance).

Message modification [2, 33] In a differential attack against a hash function, the attacker can choose messages that directly satisfy some of the constraints of the path, because there is no secret key. While the conditions in the first steps are easy to satisfy, more advanced techniques have been introduced to extend the usage of these degree of freedom to later rounds in order to speed-up collision search: neutral bits (firstly introduced for cryptanalysis SHA-0 [2, 3]), message modifications [33] and boomerangs/tunnels [12, 10].

Non-linear trails [33]. In order to get more flexibility on the differential trails, the first few steps can use non-linearity instead of following the constraints of the linearized step function. This does not affect the complexity of the search for conforming messages (thanks to messages modification techniques), but it allows to build trails from an arbitrary input difference to a good fixed output difference δ_O (or its opposite).

Multi-block technique [5, 33]. The multi-block technique takes advantage of the Davies-Meyer construction used inside the compression function. Indeed, it can be written as $h(x, m) = x + E_m(x)$ where E is a block cipher, and $+$ denotes some group operation. Because of this feed-forward, an attacker can use several differential trails in E, and several near-collisions blocks, to iteratively affect the internal state. In particular, using non-linearity in the first steps, he can derive two related trails $0 \overset{\delta_M}{\rightsquigarrow} \delta_O$ and $\delta_O \overset{-\delta_M}{\rightsquigarrow} -\delta_O$ in E from a single linear trail, by swapping the message pair. When conforming messages are found for each block, this leads to a collision because the internal state differences cancel out (see Figure 1).

Birthday phase for chosen-prefix collisions [29]. Differential techniques have also been used for chosen-prefix collision attacks. An attacker can relax the last steps of the differential trail to allow a set $\mathcal{D}$ of output differences rather than a single δ_O. He can also use several differential trails, and use the

union of the corresponding sets. Starting from two different prefixes P, P', the chosen-prefix collision attack has two stages (see Figure 2):

- In the *birthday stage*, the attacker uses a generic collision search with message blocks m_0, m'_0 to reach a difference $\delta = H(P' \,\|\, m'_0) - H(P \,\|\, m_0)$ in $\mathcal{D}$ with complexity roughly $\sqrt{\pi \cdot 2^n / |\mathcal{D}|}$.
- In the *near-collision stage*, he builds a differential trail $\delta \rightsquigarrow -\delta$ using non-linearity in the first steps, and searches a conforming message to build the chosen-prefix collision.

Multi-block for chosen-prefix collisions [29]. If a collection of differential trails affecting separate parts of the internal state is available, chosen-prefix collision attacks can be greatly improved. In particular, if an arbitrary input difference δ_R can be decomposed as $\delta_R = -\left(\delta_O^{(1)} + \delta_O^{(2)} + \cdots + \delta_O^{(r)}\right)$, where each $\delta_O^{(i)}$ can be reached as the output of a differential trail, the attacker just has to find near-collision blocks with output differences $\delta_O^{(1)}, \ldots, \delta_O^{(r)}$ (see Figure 3).
Alternatively, if this only covers a subset of input differences, the multi-block technique is combined with a birthday stage.

Our contributions. In this work, we describe new chosen-prefix collision attacks against the `MD-SHA` family, using several improvements to reduce the complexity.

1. The main improvement comes from the use of several near-collision blocks, while Stevens uses a single near-collision block [26]. For instance, using two blocks we can start from any difference in the set $\mathcal{S} := \{\delta_1 + \delta_2 \mid \delta_1, \delta_2 \in \mathcal{D}\}$, and cancel it iteratively with a first block following a trail $\delta_1 + \delta_2 \rightsquigarrow -\delta_1$ and a second block following a trail $\delta_2 \rightsquigarrow -\delta_2$ (see Figure 4). The set $\mathcal{S}$ grows with the number of blocks: this reduces the cost of the birthday search in exchange for a more expensive near-collision stage.
   While there are previous chosen-prefix collision attacks using several near-collision blocks [13, 22, 29, 30, 15], these attacks use a collection of differential trails to impact different parts of the state (each block uses a different trail). On the opposite, our technique can be used with a single differential trail, or a collection of trails without any special property. In particular, previous chosen-prefix collision attacks based on a single trail (against `SHA-1` [26] and `MD5` [30, Section 6]) used only one near-collision block.
2. In addition, we use a clustering technique to optimize the near-collision stage, taking advantage of multiple ways to cancel a given difference. For instance, let's assume that we have to cancel a difference δ in the internal state that can be written in two different ways: $\delta = \delta_1 + \delta_2 = \delta'_1 + \delta'_2$, with $\delta_1, \delta'_1, \delta_2, \delta'_2 \in \mathcal{D}$, knowing trails $\delta \rightsquigarrow -\delta_1$ and $\delta \rightsquigarrow -\delta'_1$ with the same message constraints. Then, an attacker can target simultaneously $-\delta_1$ and $-\delta'_1$ for the first near-collision block (and use either a trail $\delta_2 \rightsquigarrow -\delta_2$ or $\delta'_2 \rightsquigarrow -\delta'_2$ for the second block, depending on the first block found). This can reduce the cost of finding the first block by a factor two.

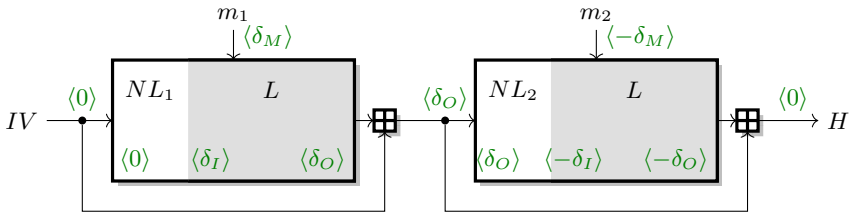

Fig. 1. 2-block collision attack using a linear trail $\delta_I \overset{\delta_M}{\rightsquigarrow} \delta_O$ and two non-linear trails $0 \rightsquigarrow \delta_I$ and $\delta_O \rightsquigarrow -\delta_I$. Green values between bracket represent differences in the state.

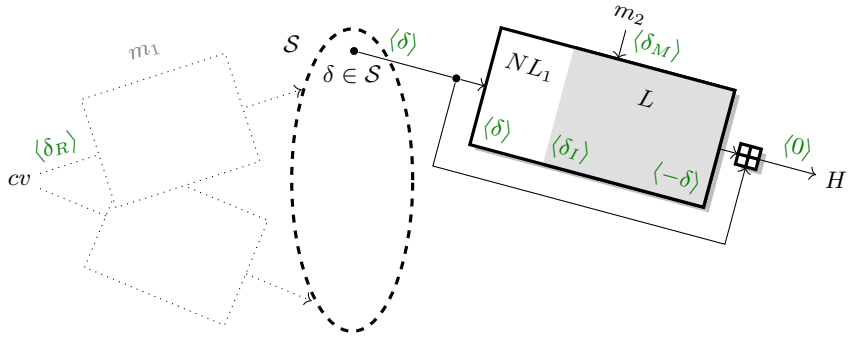

Fig. 2. Single-block chosen-prefix collision attack with a birthday stage. The linear trail $\delta_I \rightsquigarrow \delta_O$ is relaxed to reach a set $\mathcal{S}$ of feasible differences.

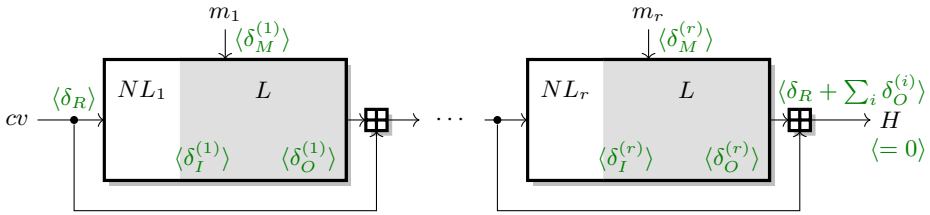

Fig. 3. Multi-block chosen-prefix collision attack. We assume that an arbitrary difference δ_R can be decomposed as $\delta_R = -\left(\delta_O^{(1)} + \delta_O^{(2)} + \cdots + \delta_O^{(r)}\right)$, where each $\delta_O^{(i)}$ can be reached as the output of a differential trail.

This technique can be seen as a generalization of an optimisation used for collision attacks with two blocks, where the first is less constrained and several output differences are allowed (for instance the SHA-1 collision attack of [26] allows 6 output differences, so that the first block is 6 times less expensive than the second).

Using these techniques, we obtain significant improvements to chosen-prefix collision attacks against MD5 and SHA-1.

Application to MD5. We use multiple near-collision blocks to improve the complexity of the chosen-prefix collision attack with a single near-collision block given in [30, Section 6]. We start with the same differential trail, and a set $\mathcal{D}$ of size $2^{25.2}$, built in the same way. Using two near-collision blocks, we can target the set $\mathcal{S} := \{\delta_1 + \delta_2 \mid \delta_1, \delta_2 \in \mathcal{D}\}$ which contains $2^{37.1}$ elements. This leads to an attack with complexity roughly $\sqrt{\pi \cdot 2^{128}/2^{37.1}} \approx 2^{46.3}$, while the best previous attack with two blocks or less required $2^{53.2}$ MD5 computations. However, the best chosen-prefix collision attack against MD5 is still the attack from [30] with complexity $2^{39.1}$ using 9 near-collision blocks.

Application to SHA-1. For SHA-1, we start with the attack of Stevens [26], and after using several improvements we obtain a chosen-prefix collision attack with estimated complexity between $2^{66.9}$ and $2^{69.4}$ SHA-1 computations. This is within a small factor of the complexity of a plain collision attack, estimated at $2^{64.7}$ on average [33, 27], and orders of magnitude better than the $2^{77.1}$ computations cost of the currently best known chosen-prefix collision attack [26] on SHA-1. We have conducted tests to check that our assumptions are indeed verified in practice.

First, we use a more relaxed version of the differential trail than used in [26], so that we have a set $\mathcal{S}$ of size 8768 rather than 192. This directly reduce the attack cost by a factor 6.75, down to $2^{74.3}$. Next, we use the multi-block technique to build a large set $\mathcal{S}$ and to reduce further the cost of the birthday stage. Using a set $\mathcal{S}$ of size $2^{29.4}$ with a near-collision cost at most $12 \times 2^{64.7}$, this reduces the cost of the attack down to $2^{68.6}$ (with an optimistic estimate). Finally, we use the clustering technique to reduce the near-collision cost. After optimization, we have a set $\mathcal{S}$ of $2^{32.67}$ differences that can be reached with a maximum cost of $3.5 \times 2^{64.7}$ (with an optimistic estimate), leading to a full attack with complexity $2^{66.9}$ — about five time more expensive than the collision attack.

Our result is surprising since we show that the cost to find a chosen-prefix collision for SHA-1 is not much more than a simple collision search. Moreover our work has a strong impact in practice as chosen-prefix collision attacks are much more dangerous than simple collisions (see for example the case of MD5 [30]). This is yet another warning that SHA-1 should be totally removed from any security product as soon as possible. The thinking "a collision attack is not directly exploitable, thus we are fine" is clearly wrong for SHA-1, and we give a proof here.

Function	Collision type		Complexity	Ref.
SHA-1	free-start collision		$2^{57.5}$	[28]
	collision		2^{69}	[33]
			$2^{64.7}$	[27]
	chosen-prefix collision		$2^{77.1}$	[26]
			$2^{66.9}$ — $2^{69.4}$	New
MD5	collision		2^{40}	[34]
			2^{16}	[30]
	chosen-prefix collision	(9 blocks)	$2^{39.1}$	[30]
		(3 blocks)	2^{49}	[30]
		(1 block)	$2^{53.2}$	[30]
		(2 blocks)	$2^{46.3}$	New

Table 1. Comparison of previous and new cryptanalysis results on MD5 and SHA-1. A free-start collision is a collision or the compression function only, where the attacker has full control on all the primitive's inputs. Complexities in the table are given in terms of SHA-1 equivalents on a GTX-970 GPU (when possible).

Our method is in essence quite generic, even though a lot of details have to be taken care of in order to make it work. Since most collision attacks on members of the MD-SHA family are built on the same principles as SHA-1 attacks, we believe similar ideas would apply and a collision attack can probably be transformed into a chosen-prefix collision attack for a reasonable extra cost factor. We do not foresee any reason why this technique would not apply to non MD-SHA hash functions as well (except wide-pipe hash functions which would make the birthday part too costly).

Outline. We first consider the impact of this result and give some recommendations in Section 2. Then, we describe SHA-1 and previous cryptanalysis works on this hash function in Section 3. The generic high-level description of our attack is given in Section 4, while the details regarding its application to MD5 and SHA-1 are provided in Sections 5 and 6, respectively. Eventually, we conclude and propose future works in Section 7.

2 Implications and Recommendations

Our work shows that finding a chosen-prefix collision is much easier than previously expected, and potentially not much harder than a normal collision search for SHA-1. As a real collision has already been computed for this hash function, one can now assume that chosen-prefix collisions are reachable even by medium funded organisations. Since a chosen-prefix collision attack can break many widely used protocols, we strongly urge users to migrate to SHA-2 or SHA-3, and to disable SHA-1 to avoid downgrade attacks.

Cost Estimation. We use the same estimation process as in [27]. With an optimistic spot-price scenario on `g2.8xlarge` instances of Amazon EC2, the authors estimated that the workload spent to find the `SHA-1` collision was equivalent to a cost of about US$ 110 K, with $2^{63.4}$ `SHA-1` equivalent calls on GTX-970 GPUs. We recall that they found the collision with less computations than expected. Using expected computational cost, the average workload required to find a `SHA-1` collision is equivalent to a cost of about US$ 275 K, or $2^{64.7}$ `SHA-1` calls. An optimistic analysis of our attack leads to a complexity of $2^{66.9}$ `SHA-1` equivalent calls on GTX-970 GPUs, corresponding to a cost of US$ 1.2 M, while a more conservative analysis yields a complexity of $2^{69.4}$, or a cost of US$ 7M.

Hardware will improve as well as cryptanalysis and we can expect that collision together with chosen-prefix collision attacks will get cheaper over the years. Migration from `SHA-1` to the secure `SHA-2` or `SHA-3` hash algorithms should now be done as soon as possible, if not already.

Impact of Chosen-prefix Collisions. Chosen-prefix collision attacks have been demonstrated already for `MD5`, and they are much more dangerous that identical-prefix collision attacks, with a strong impact in practice. For example, they have been shown to break important internet protocols, including TLS, IKE, and SSH by allowing the forgery of handshake messages. The SLOTH attacks [1] can break various security properties of these protocols using `MD5` chosen-prefix collisions, such as client impersonation in TLS 1.2. It was also shown [29] that one can generate colliding X.509 certificates and later a rogue certificate authority [30] from a chosen-prefix collision attack on `MD5`, undermining the security of websites. `MD5` has been removed from most security applications, but the very same threats are now a reality for `SHA-1`.

The SLOTH attacks with `SHA-1` chosen-prefix collisions allow client impersonation in TLS 1.2 and peer impersonation in IKEv2. In particular, IKEv2 can be broken with an *offline* chosen-prefix collision, which is now practical for a powerful adversary. On the other hand, the creation of a rogue CA requires to predict in advance all the fields of the signed certificate. Hopefully, this is not possible with current certificate authorities, because they should randomize the serial number field.

Usage of `SHA-1`. Even if practically broken only very recently, `SHA-1` has been theoretically broken since 2004. It is therefore surprising that `SHA-1` remains deployed in many security systems. In particular, as long as `SHA-1` is *allowed*, even if it is not used in normal operation, an attacker can use weaknesses in `SHA-1` to forge a signature, and the signature will be accepted.

First, `SHA-1` is still widely used to authenticate handshake messages in secure protocols such as TLS, SSH or IKE. As shown with the SLOTH attack [1], this allows various attacks using chosen-prefix collision, such as breaking authentication. These protocols have removed support for `MD5` after the SLOTH attack, but `SHA-1` is still widely supported. Actually, more than[4] 5% of the web servers

[4] https://censys.io/domain?q=tags:https+and+443.https.tls.signature.hash_algorithm:sha1

from Alexa's top 1M (including skype.com) *prefer* to use SHA-1 to authenticate TLS handshake messages.

An important effort is underway to remove SHA-1 certificates from the Web, and major browsers are now refusing to connect to servers still using SHA-1-based certificates. Yet SHA-1-based certificates remains present: according to scans of the top 1 million websites from Alexa by censys.io, there are still about 35 thousand[5] servers with SHA-1 certificates out of 1.2 million servers with HTTPS support. SHA-1-based certificates are also used with other protocols: for instance 700 thousand[6] out of 4.4 million mail servers (with IMAPS) use a SHA-1 certificate. Actually, it is still possible to buy a SHA-1 certificate from a trusted root[7]! Even though recent web browsers reject those certificates, they are accepted by older browsers and by many clients for other protocols. For instance, the "Mail" application included in Windows 10 still accepts SHA-1 certificates without warnings when opening an IMAPS connection.

Unfortunately, many industry players did not consider moving away from SHA-1 a priority, due to important costs and possible compatibility and bug issues induced by this move. An often-heard argument is that a simple collision attacks against a hash function is not very useful for an attacker, because he doesn't have much control over the colliding messages. Therefore, there seemed to be a long way to go before really useful collision attacks would be found for SHA-1, if ever. Indeed, the current best chosen-prefix collision attack against SHA-1 requires $2^{77.1}$ hash calls [26], thus orders of magnitude harder than the cost of finding a simple collision. Similarly, in the case of MD5, the cost goes from 2^{16} to 2^{39} for the currently best known collision and chosen-prefix collision attacks. However, this is a dangerous game to play as the history showed that cryptanalysis only keep improving, and attackers will eventually come up with ways to further improve their cryptanalysis techniques. For example, in the case of MD5, collisions for the compression function were found [9] in 1993, collisions for the whole hash function were found [34] in 2004, colliding X.509 MD5-based certificates were computed [29] in 2007, and rogue Certificate Authority certificate [30] was eventually created in 2009.

3 Preliminaries

3.1 Description of SHA-1

We describe here the SHA-1 hash function, but we refer to [19] for all the complete details.

[5] https://censys.io/domain?q=tags:https+and+443.https.tls.certificate.parsed.signature_algorithm.name:SHA1*

[6] https://censys.io/ipv4?q=tags:imaps+and+993.imaps.tls.tls.certificate.parsed.signature_algorithm.name:SHA1*

[7] https://www.secure128.com/online-security-solutions/products/ssl-certificate/symantec/sha-1-private-ssl/

SHA-1 is a 160-bit hash function based on the well-known Merkle-Damgård paradigm [6, 16]. The message input is first padded (with message length encoded) to a multiple of 512 bits, and divided into blocks m_i of 512 bits each. Then, each block is processed via the SHA-1 compression function h to update a 160-bit chaining variable cv_i that is initialised to a constant and public initial value (IV): $cv_0 = IV$. More precisely, we have $cv_{i+1} = h(cv_i, m_{i+1})$. When all blocks have been processed, the hash output is the last chaining variable.

The compression function is similar to other members of the MD-SHA family of hash functions. It is based on the Davies-Meyer construction, that turns a block cipher E into a compression function: $cv_{i+1} = E_{m_{i+1}}(cv_i) + cv_i$, where $E_k(y)$ is the encryption of the plaintext y with the key k, and $+$ is a word-wise modular addition.

The internal block cipher is composed of 80 steps (4 rounds of 20 steps each) processing a generalised Feistel network. More precisely, the state is divided into five registers $(A_i, B_i, C_i, D_i, E_i)$ of 32-bit each. At each step, an extended message word W_i updates the registers as follows:

$$\begin{cases} A_{i+1} = (A_i \lll 5) + f_i(B_i, C_i, D_i) + E_i + K_i + W_i \\ B_{i+1} = A_i \\ C_{i+1} = B_i \ggg 2 \\ D_{i+1} = C_i \\ E_{i+1} = D_i \end{cases}$$

where K_i are predetermined constants and f_i are boolean functions (in short: IF function for the first round, XOR for the second and fourth round, MAJ for the third round, see Table 2). Since only a single register value is updated (A_{i+1}), the other registers being only rotated copies, we can express the SHA-1 step function using a single variable:

$$\begin{aligned} A_{i+1} = (A_i \lll 5) + f_i(A_{i-1}, A_{i-2} \ggg 2, A_{i-3} \ggg 2) \\ + (A_{i-4} \ggg 2) + K_i + W_i. \end{aligned}$$

For this reason, the differential trails figures in this article will only represent A_i, the other register values at a certain point of time can be deduced directly.

step i	$f_i(B, C, D)$	K_i
$0 \leq i < 20$	$f_{IF} = (B \wedge C) \oplus (\overline{B} \wedge D)$	0x5a827999
$20 \leq i < 40$	$f_{XOR} = B \oplus C \oplus D$	0x6ed6eba1
$40 \leq i < 60$	$f_{MAJ} = (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D)$	0x8fabbcdc
$60 \leq i < 80$	$f_{XOR} = B \oplus C \oplus D$	0xca62c1d6

Table 2. Boolean functions and constants of SHA-1

The extended message words W_i are computed linearly from the incoming 512-bit message block m, the process being called message extension. One first

splits m into 16 32-bit words $M_0, \ldots, M_{15}$, and then the W_i's are computed as follows:

$$W_i = \begin{cases} M_i, & \text{for } 0 \leq i \leq 15 \\ (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1, & \text{for } 16 \leq i \leq 79 \end{cases}$$

3.2 Previous Works

Collision attacks on SHA-1. We quickly present here without details the previous advances on SHA-1 collision search. First results on SHA-0 and SHA-1 were obtained by linearizing the compression function and constructing differential trails based on the probabilistic event that difference spreads will indeed happen linearly. These linear trails are generated with a succession of so-called local collisions (small message disturbances whose influence is corrected with other message differences inserted in the subsequent SHA-1 steps) that follows the SHA-1 message expansion. However, with this linear construction, impossibilities might appear in the first 20 steps of SHA-1 (for example due to the f_{IF} boolean function that never behaves linearly in some specific situations) and the cheapest linear trail candidates might not be the ones that start and end with the same difference (which is a property required to obtain a collision after the compression function feed-forward). Thus, since [33], collision attacks on SHA-1 are performed using two blocks containing differences. The idea is to simply pick the cheapest linear trail from roughly step 20 to 80, without paying any attention to the f_{IF} constraints or to the input/output differences. Then, the attacker will generate a non-linear differential trail for the first 20 steps in order to connect the actual incoming input difference to the linear part difference at step 20. With two successive blocks using the same linear trail (just ensuring that the output difference of the two blocks have opposite signs), one can see in Figure 1 that a collision is obtained at the end of the second block.

Once the differential trail is set, the attacker can concentrate on finding a pair of messages that follows it for each successive block. For this, he will construct a large number of messages that follow the trail up to some predetermined step, and compute the remaining steps to test whether the output difference is the required one. The computational cost is minimized by using a simple early-abort strategy for the 16 first steps, but also more advanced amortization methods such as neutral bits [3], boomerangs [10] or message modification [33] for a few more steps. Usually, the first 20 or so steps do not contribute the complexity of the attack.

Chosen-prefix collision attacks. The first concrete application of a chosen-prefix collision attack was proposed in [29] for MD5. This work was also the first to introduce a birthday search phase in order to find such collisions. The idea is to process random message blocks after the challenged prefixes, until the chaining variable difference δ belongs to a large predetermined set $\mathcal{S}$. Since the message blocks after each prefix are chosen independently, this can be done with birthday complexity $\sqrt{\pi \cdot 2^n/|\mathcal{S}|}$. Then, from that difference δ, the authors

eventually manage to reach a collision by slowly erasing the unwanted difference bits by successfully applying some near-collision blocks. We note that the starting difference set $\mathcal{S}$ during the birthday phase must not be too small, otherwise this phase would be too costly. Moreover, the near-collisions blocks must not be too expensive either, and this will of course depend on the cryptanalysis advancements of the compression function being studied.

Finally, using also this two-phase strategy, in [26] is described a chosen-prefix collision attack against the full `SHA-1`, for a cost of $2^{77.1}$ hash calls. The improvement compared to the generic 2^{80} attack is not very large, due to the difficulty for an attacker to generate enough allowable differences that can later be erased efficiently with a near-collisions block. Indeed, the compression function of `SHA-1` being much stronger than that of `MD5`, few potential candidates are found. Actually, Stevens only considers one type of near collision block, following the best differential trail used for the collision attack. By varying the signs of the message and output differences, and by letting some uncontrolled differences spread during the very last steps of the compression function, a set $\mathcal{S}$ of 192 allowable differences is obtained. However, having such a small set makes the birthday part by far the most expensive phase of the attack.

In this article, we will use essentially the same strategy: a birthday phase to reach a set $\mathcal{S}$ of allowable differences and a near-collision phase to remove the remaining differences. We improve over previous works on several points. First, we further generalise for `SHA-1` the set of possible differences that can be obtained for a cheap cost with a single message block. Secondly, we propose a multi-block strategy for `SHA-1` that allows to greatly increase the size of the set $\mathcal{S}$. Finally, we study the clustering effect that appears when using a multi-block strategy. This can be utilised by the attacker to select dynamically the allowable differences at the output of each successive blocks, to further reduce the attack complexity. Notably, and in contrary to previous works, our set $\mathcal{S}$ is not the direct sum of independent subspaces corresponding to distinct trails. On the opposite, our applications use the same core differential trail for all the near-collision blocks. Overall, we improve substantially previous attack [26] from $2^{77.1}$ `SHA-1` calls to only $2^{66.9}$. Surprisingly, our attack is very close to some sort of optimal since its cost is not much greater than that of finding a simple collision. Our attack being rather generic, we believe that this might be the case for many hash functions, which contradicts the idea that chosen-prefix collisions are much harder to obtain than simple collisions.

One can mention other parallel researches conducted on finding chosen-prefix collision attacks for various hash functions. For example, in [22], the author explains how to compute collisions with random incoming differences in the internal state for the `GRINDAHL` hash function, the strategy being to slowly remove these differences thanks to the many degrees of freedom available every step. Such a divide-and-conquer technique is not applicable at all to `SHA-1` as the degrees of freedom are much fewer and only available at the beginning of the compression function. In [15], inspired by the second-preimage attack against `SMASH` [13], the authors proposed a chosen-prefix collision attack on a reduced version of the

GROSTL hash function. However, this attack strongly relies on the ability of the attacker to perform a rebound attack, which seems really hard to achieve in the case of SHA-1.

4 From Collision to Chosen-Prefix Collision

4.1 The Chosen-prefix Collision Attack

We assume that the hash function considered is an n-bit narrow pipe primitive, based on a Merkle-Damgård-like operating mode. In addition, we assume that the compression function is built upon a block cipher in a Davies-Meyer mode.

Preparing the attack. The attacker first builds a set $\mathcal{S}$ and a graph $\mathcal{G}$; $\mathcal{S}$ corresponds to a set of differences that can be cancelled with near-collision blocks, and $\mathcal{G}$ is used to find the sequence of blocks needed to cancel a difference in $\mathcal{S}$. We first explain how to execute the attack when $\mathcal{S}$ and $\mathcal{G}$ are given, and we will explain how to build them in Section 4.2.

The prefixes (stage 1 of Figure 4). The attacker receives the two challenge prefixes and pads them to two prefixes of the same length, to avoid differences in the final length padding. After processing the two padded prefixes starting from the IV, he reaches states cv/cv', and we denote the corresponding difference as δ_R.

The birthday search (stage 2 of Figure 4). The goal of the attacker is now to find one message block pair (u, u') to reach a chaining variable pair with a difference δ that belongs to $\mathcal{S}$, the set of acceptable chaining variable differences.

For this stage, we use the parallel collision search algorithm of van Oorschot and Wiener [31]. When a memory $M \gg C$ is available, this algorithm can find C collisions in a function $f : \{0,1\}^k \to \{0,1\}^k$ with complexity $\sqrt{\pi/2 \cdot 2^k \cdot C}$, and is efficiently parallelizable. It computes chains of iterates of the function f, and stops when the end point is a *distinguished point*, *i.e.* it satisfies some easy to detect property. In practice, we stop a chain when $x < 2^k \cdot \theta$, with $\theta \gg \sqrt{C/2^k}$, and we store on average the starting points and end points of $\theta \cdot \sqrt{\pi/2 \cdot 2^k \cdot C}$ chains (the expected length of a chain is $1/\theta$). When colliding end points are detected, we restart the corresponding chains to locate the collision point, with an expected cost of $2C/\theta$, which is small compared to the total complexity if $\theta \gg \sqrt{C/2^k}$.

In our case, we are looking for message blocks (u, u') such that $h(cv, u) - h(cv', u') \in \mathcal{S}$. Therefore, we need a function f such that a collision in f corresponds to good (u, u') with high probability. First, we consider a truncation function $\tau : \{0,1\}^n \to \{0,1\}^k$, so that pairs (x, x') with $x - x' \in \mathcal{S}$ have $\tau(x) = \tau(x')$ with high probability:

$$p = \Pr_{x,x'} \left[\tau(x) = \tau(x') \mid x - x' \in \mathcal{S}\right] \approx 1.$$

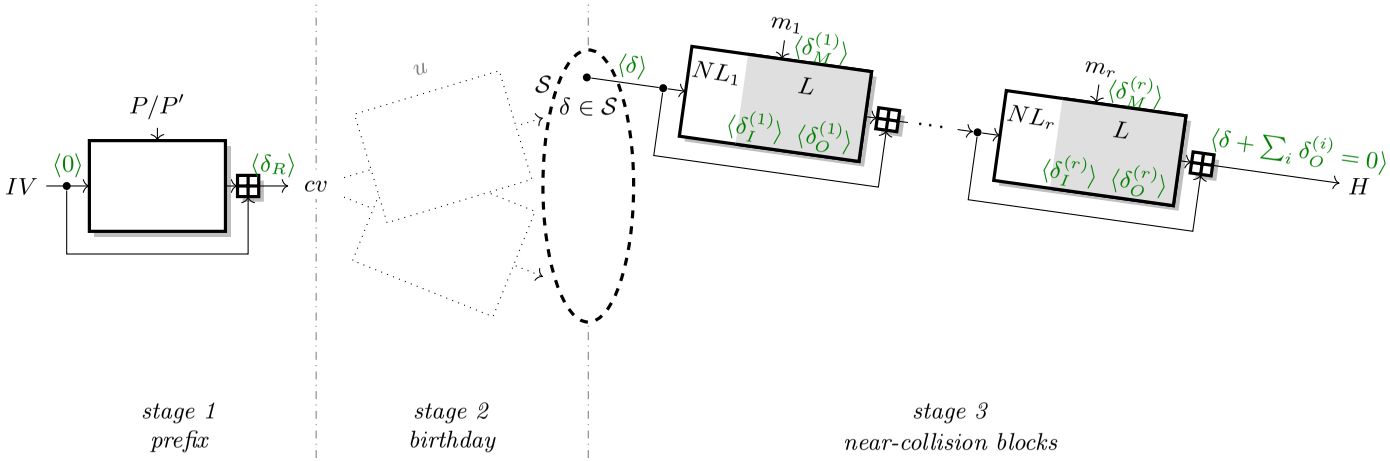

Fig. 4. High-level view of the chosen-prefix collision attack

For functions of the MD-SHA family, the group operation + is a word-wise modular addition, and we build τ by removing bits that are directly affected when adding a value in $\mathcal{S}$, and bits that are affected by a carry with a relatively high probability. This typically leads to p close to one (as seen in previous attacks [30, 26], and the new attacks in this paper). Then, we build f as:

$$f(u) := \begin{cases} \tau(h(cv\ , \text{pad}(u))) & \text{if } u[0] = 1; \\ \tau(h(cv', \text{pad}(u))) & \text{else.} \end{cases}$$

The probability that a collision in f corresponds to a pair (u, u') with $h(cv, u) - h(cv', u') \in \mathcal{S}$ can be estimated as:

$$p_f = \frac{1}{2} \cdot \Pr_{x,x'} \left[x - x' \in \mathcal{S} \mid \tau(x) = \tau(x')\right] = \frac{p}{2} \cdot \frac{|\mathcal{S}|/2^n}{2^{-k}}$$

Finally, we need $1/p_f$ collisions in f, and the total complexity of the birthday stage is on average:

$$\sqrt{\frac{\pi}{2} \cdot \frac{2^k}{p_f}} = \sqrt{\frac{\pi \cdot 2^n}{p \cdot |\mathcal{S}|}} \approx \sqrt{\frac{\pi \cdot 2^n}{|\mathcal{S}|}}.$$

The multi-block collision search (stage 3 of Figure 4). The attacker now uses the graph $\mathcal{G}$ to build a sequence of near-collision blocks that ends up with a collision. Each node of the graph represents one chaining variable difference in the set $\mathcal{S}$. To each node i of the graph is associated a cost value w_i that represents the cost an attacker will expect to pay from this particular chaining variable difference i in order to reach a colliding state (with one or multiple message blocks). Of course, a null cost will be associated with the zero difference ($w_0 = 0$). A directed edge from node i to node j represents a way for an attacker to reach chaining variable difference j from difference i with a single message block. Note that the graph is acyclic, as we will ensure that the edges will always go to strictly lower costs (i.e. an edge from i to j is only possible if $w_j < w_i$). To each edge is attached the details of the differential trail and message difference to use for that transition to happen. However, a *very* important point is that all edges going out of a node i will share the same core differential trail (by core differential trail, we mean the entire differential trail except the last steps for which one can usually accept a few divergences in the propagation of the differences). For example, during the attack, from a chaining difference i, an attacker can potentially reach difference j or difference k using the same core differential trail (and thus without having to commit in advance which of the two differences he would like to reach). Thus, in essence, the details of the differential trail and message difference to use can be directly attached to the source node.

Once the attacker hits a starting difference $\delta \in \mathcal{S}$ in the birthday phase, he will pick the corresponding node in $\mathcal{G}$, and use the differential trail and message difference attached to this node. He will use this differential trail until he reaches one of the target nodes (which has necessarily a lower expected cost attached

to it). As explained in the next section, targeting several nodes simultaneously reduces the cost of the attack, because it is easier to hit one node out of many than a fixed one. We call this the clustering effect, because we use a cluster of paths in the graph. When a new node is reached, the attacker repeats this process until he eventually reaches the colliding state (i.e. null difference). Overall, the expected computational cost for this phase is the cost attached to the node δ (in practice, when actually computing one collision, he might pay a slightly lower or higher computational cost as the w_i's are expected values).

We note that any suffix message blocks that do not contain differences can of course be added after this colliding state, as the Merkle-Damgård-like mode will maintain the collision throughout the subsequent compression function calls.

4.2 Building the Set $\mathcal{S}$ and the Graph $\mathcal{G}$

We now describe how the set $\mathcal{S}$ and the graph $\mathcal{G}$ can be built during the preparation of the attack. For that, we first need to describe what an adversary can do when he tries to attack the compression function. We consider that the attacker knows some good core differential trails for the internal block cipher E, that is differential trails that go from early steps to late steps of E. For each core differential trail $\texttt{CDT}_i$ there are several possible output differences δ_j^i for E. This is typically what happens in the chosen-prefix collision attack on `SHA-1` [26] where some differences in the very last steps can be allowed to spread differently than planned, thus generating new output differences. We denote the set of all possible output differences as $\mathcal{D}$ (in particular, we have $0 \in \mathcal{D}$, and $\delta \in \mathcal{D} \iff -\delta \in \mathcal{D}$ because of symmetries).

We finally assume that any input difference for E can be mapped to any of the core differential trails inside the primitive. In the case of a `SHA-1` attack, this is achieved with the non-linear part of the trail in the first steps of the function. As shown in previous works, it allows to map any input difference to any internal difference. The non-linear part has a low probability, but during the near-collision search this is solved using the many degrees of freedom available in the first steps of the function.

Building the graph $\mathcal{G}'$. The attacker will first build a graph $\mathcal{G}'$ and filter it later to create $\mathcal{G}$. The graph $\mathcal{G}'$ is similar to the graph $\mathcal{G}$: each node will represent a chaining variable difference. A directed edge from node i to node j represents again a way for an attacker to reach chaining variable difference j from difference i with a single message block, stored with the details of the differential trail attached to it, and the cost to find the corresponding block. The differences with $\mathcal{G}$ are that (see Figure 5):

- $\mathcal{G}'$ can potentially be cyclic, as we do not ensure that an edge goes from a higher to a strictly smaller cost;
- all outgoing edges from a node i will not necessarily share the same core differential trail;
- there can be several edges from i to j, with different core differential trails.

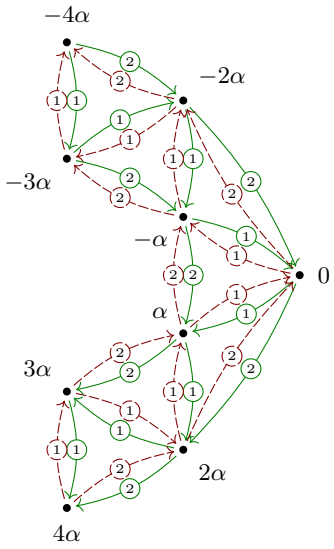

Fig. 5. Example of a graph $\mathcal{G}'$, with a bundle $\{\alpha, 2\alpha\}$ with costs 1 and 2 (solid green lines), and a bundle $\{-\alpha, -2\alpha\}$ with costs 1 and 2 (dashed red lines). The corresponding set $\mathcal{S}$ is $\{-4\alpha, -3\alpha, -2\alpha, -\alpha, 0, \alpha, 2\alpha, 3\alpha, 4\alpha\}$.

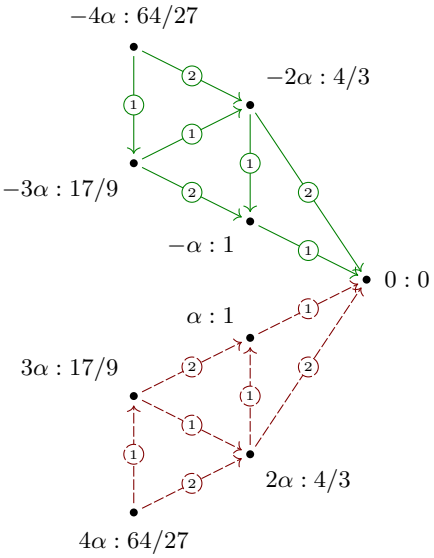

Fig. 6. The graph $\mathcal{G}$ corresponding to $\mathcal{G}'$. We show the cost of each edge and each node. In particular, note that use of clustering reduces the cost of node 4α from 4 to $64/27 \approx 2.37$

In order to build the graph $\mathcal{G}'$, starting from the colliding state $\delta = 0$, we will simply proceed backward. We go through all possible core differential trails for E and their possible output differences δ^i_j. Due to the feed-forward of the Davies-Meyer construction, all these differences can be used to reach the colliding state by simply forcing their respective opposite $-\delta^i_j$ as input difference of the cipher (since we assumed that any input difference for E can be mapped to any of the core differential trails inside the primitive, this is always possible). Thus, for each such difference δ^i_j coming from a core differential trail $\texttt{CDT}_i$, we will add a node $-\delta^i_j$ in the graph $\mathcal{G}'$, and an edge going from this new node to the colliding state. If a node with that difference already exists in the graph, we add the edge between this node and the colliding state. This means that nodes can have several incoming and outgoing edges.

We iteratively repeat this process again with all the newly created nodes as starting points (instead of the collision state). This will create a bigger and bigger graph as we keep iterating, and the attacker can simply stop when he believes that he has enough nodes in the graph (alternatively, he can set an upper limit on the cost of the nodes to consider, or on the depth of the search, which will naturally limit the size of the graph).

The clustering effect. A simple way to build a graph $\mathcal{G}$ for the attack is to keep only the minimal cost paths in the graph $\mathcal{G}'$ (the corresponding edges form a tree), and to set the cost of the nodes to cost of the minimal path. However, the attack cost can be greatly improved with the *clustering effect*: during the last phase of the attack, when the attacker is currently located at a node N, he does not necessarily need to choose in advance which outgoing edge of N he will try to follow. Indeed, the only commitment he needs to make at this point is which core differential trail he will use to go to the next node. Thus, he can simultaneously target several output differences corresponding to the same core differential trail, and the cost to reach one difference out of many is significantly lower than the cost of reaching any given output difference. For instance, when computing the first block of a `SHA-1` collision, Stevens [26] allows six output differences with a similar cost, so that the cost to reach one of them is one sixth of the cost to reach a predetermined one.

For a given node, we call *bundle* of a core differential trail $\texttt{CDT}_i$ the grouping of all outgoing edges of that node that use $\texttt{CDT}_i$ (see Figure 5). Let $\mathcal{B}_N$ stand for the set of all bundles of a node N, where each bundle $\beta \in \mathcal{B}_N$ corresponds to a distinct core differential trail $\texttt{CDT}_i$. Then, for each node of $\mathcal{G}'$, we compute its assigned cost as follows[8]:

$$w_N = \min_{\beta \in \mathcal{B}_N} \left\{ \frac{1 + \sum_{(N,j) \in \beta | w_j < w_N} \left(w_j / c_j^\beta \right)}{\sum_{(N,j) \in \beta | w_j < w_N} \left(1 / c_j^\beta \right)} \right\}, \tag{1}$$

where for an edge (N, j) of the bundle β, c_j^β represents the cost to find a conforming message pair with difference output $j - N$ for E, and w_j is the cost assigned to the node j.

We initialize the costs of the nodes in $\mathcal{G}'$ using the shortest path in the graph, and update them iteratively until we can't find any more opportunity for improvement.

Building $\mathcal{S}$ and $\mathcal{G}$. The graph $\mathcal{G}$ is obtained from $\mathcal{G}'$ by only keeping the edges that goes from a higher to a strictly lower cost (in order to render the graph

[8] In order to explain this formula, we consider that when the adversary uses a bundle β, he has to perform C_β operations to find a pair conforming to the core differential trail up to some intermediate step, and those pairs lead to an output difference $j - N$ (*i.e.* to node j) with probability p_j^β (with $p_j^\beta = C_\beta / c_j^\beta$). If none of the predetermined output differences is reached (or if the target node reached has a cost $w_j \geq w_N$), then he stays at node N and will have to still pay w_N to reach the colliding state. Thus, we have that:

$$w_N = C_\beta + \sum_{j \in \beta | w_j < w_N} \left(p_j^\beta \cdot w_j \right) + \left(1 - \sum_{j \in \beta | w_j < w_N} p_j^\beta \right) \cdot w_N$$

which leads to (1) with $c_j^\beta = C_\beta / p_j^\beta$.

acyclic), and by only keeping for each node the outgoing edges for the bundle that minimizes the cost w_N in (1).

The set $\mathcal{S}$ is then finally deduced by harnessing all the differences corresponding to every node in $\mathcal{G}$ (one node in $\mathcal{G}$ will correspond to one difference in $\mathcal{S}$). In particular, if $\mathcal{G}'$ includes all nodes at depth at most r, then $\mathcal{S} = \{\delta_1 + \delta_2 + \cdots + \delta_r \mid \delta_1, \delta_2, \ldots, \delta_r \in \mathcal{D}\}$.

5 Application to MD5

Our techniques can easily be applied to MD5, to build an alternative chosen-prefix collision attack. We can't reach an attack complexity as low as 2^{39} (the best attack from [30]), because this would require to build a set $\mathcal{S}$ and graph $\mathcal{G}$ of size roughly 2^{50}, which is impractical. However, when the number of blocks available for the chosen-prefix collision is limited, the complexity of the best-known attack grows; for instance, the chosen-prefix collision used to create a rogue certificate was limited to 3 blocks, and this increased the complexity to 2^{49}. In this scenario we can improve the currently best-known attack with our multi-block technique using a single differential trail.

We start from the single-block chosen-prefix collision attack given in [30, Section 6]: this attack uses a high probability trail for `MD5` collisions, where the last steps are relaxed to allow a set $\mathcal{D}$ of size $2^{23.3}$. Therefore the birthday stage has complexity roughly $\sqrt{\pi \cdot 2^{128}/2^{23.3}} \approx 2^{53.2}$, and the near-collision block is found with a complexity of $2^{40.8}$. In our analysis, we recomputed the set $\mathcal{D}$ used by Stevens *et al.*, but we actually found a set of size $2^{24.2}$ using the same trail, with a maximum cost of $2^{26} \cdot 2^{14.8}$ (following [30], we only consider output differences with $\delta a = -2^5, \delta d = -2^5 + 2^{25}, \delta c = -2^5 \bmod 2^{20}$). Then, we extend $\mathcal{D}$ by adding the zero value and the opposite of each value, to generate $\mathcal{D}' := \mathcal{D} \cup -\mathcal{D} \cup \{0\}$. Finally, we build the set $\mathcal{S}$ and the graph $\mathcal{G}'$ corresponding to an attack with at most 2 blocks, with $\mathcal{S} := \{\delta_1 + \delta_2 \mid \delta_1, \delta_2 \in \mathcal{D}'\}$. Since the cost of the near-collision stage is negligible (at most $2 \cdot 2^{40.8}$), we do not need to use clustering, and we can just use the minimal cost paths of $\mathcal{G}'$ as the graph $\mathcal{G}$.

We find that the set $\mathcal{S}$ contains $2^{37.1}$ elements, so that the birthday stage has a complexity of roughly $\sqrt{\pi \cdot 2^{128}/2^{37.1}} \approx 2^{46.3}$. Therefore, we have a simple chosen-prefix collision attack with two near-collision blocks with complexity $2^{46.3}$, while the best previous attack with two blocks or less requires $2^{53.2}$ MD5 computations, and even the best attack with three blocks requires 2^{49} MD5 computations.

6 Application to SHA-1

For the attack on `SHA-1`, we directly recycle the details of the collision attack from [27]: we will use the same linear part for our successive near-collision blocks (even though the very last steps might behave slightly differently as we will explain in this section). We assume that the attacker can generate non-linear parts on the fly and can apply amortization methods just like in [27]. In order

to validate this assumption, we have tried to generate a non-linear part with several random input differences from $\mathcal{S}$ and random input chaining values. In our experiments, we have successfully generated such non-linear part, and we could even make it limited to the very first `SHA-1` steps. We discuss this assumption and our experiments in more details in Section 6.3.

We now explain how to apply the framework of Section 4 to a chosen-prefix collision against `SHA-1`. As mentioned, our attack uses the best core trail known for attacks against `SHA-1`, as used in previous attacks [26, 25, 27]. This allows us to have a relatively good complexity estimation for the attack, because this trail has been well studied, and a full collision attack with this trail was recently implemented. In the following we denote the complexity to find a block conforming to the trail (with an optimal output difference) as C_{block}. In the case of the recent collision attack, this cost was estimated as $C_{\text{block}} = 2^{64.7}$ `SHA-1` evaluations on a GTX-970 GPU [27, Section 5.7]. In this work, we consider several hypothesis for the cost of finding near-collision blocks: an optimistic hypothesis with $C_{\text{block}} = 2^{64.7}$ (following [27]) and a conservative hypothesis with $C_{\text{block}} = 2^{67.7}$. This parameter depends on the difficulty of linking an arbitrary input difference to the core trail, and will be discussed in more detail below.

As in the previous chosen-prefix collision attack on `SHA-1` [26], we consider several variants of the core trail by changing some of the message constraints in the last steps (in particular, we flip the sign of some message bits), and we relax the last steps to reach a larger set of output difference. However, we do this more exhaustively than Stevens: he only describes a set $\mathcal{D}$ of size 192 with cost at most $1.15 \cdot C_{\text{block}}$, but we found a set of size 8768 with cost at most $8 \cdot C_{\text{block}}$, including 576 values with cost at most $1.15 \cdot C_{\text{block}}$. In particular, this directly leads to an improvement of the single-block chosen-prefix collision from [26], with complexity roughly $\sqrt{\pi \cdot 2^{160}/8768} \approx 2^{74.3}$, rather than $\sqrt{\pi \cdot 2^{160}/192} \approx 2^{77}$ (ignoring some technical details of the birthday step).

More precisely, we allow the signs of the message differences to not necessarily follow local collision patterns in the last steps. Instead, we consider variants of the trail where each of those constraints is either followed or not. In addition, we fix the sign of some additional state bits to reduce the cost to reach a given output difference. Table 3 compares our message constraints with those used for the second-block of the attack from [27]. This leads to 288 differences with optimal probability ($2^{-19.17}$ in steps 61 to 79), and 288 with almost optimal probability ($2^{-19.36}$ in steps 61 to 79), as listed in Table 4. Moreover, we consider output differences whose cost is higher than the optimal cost C_{block}, up to roughly $8 \cdot C_{\text{block}}$ (we allow a probability up to 2^{-22} in steps 61 to 79).

Instead of building the corresponding set of output differences and their probability analytically, we used a heuristic approach. For each choice of the message constraints z_i in Table 3 (up to some symmetries), we generated 2^{30} intermediate states at step 60, and we computed the corresponding output differences in order to identify high probability ones. We also keep track of the differences reached with the same constraints, to build the corresponding bundles of differences. Next, we used symmetries in the set of differences to verify the consistency

Stevens *et al.* constraints [27]	Our constraints
$W_{68}^{[5]} = W_{67}^{[0]} \oplus 1$	$W_{68}^{[5]} = W_{67}^{[0]} \oplus 1$
$W_{72}^{[30]} = W_{67}^{[0]} \oplus 1$	$W_{72}^{[30]} = W_{67}^{[0]} \oplus 1$
$W_{71}^{[6]} = W_{70}^{[1]} \oplus 1$	$W_{71}^{[6]} = W_{70}^{[1]} \oplus 1$
$W_{72}^{[5]} = W_{71}^{[0]} \oplus 1$	$W_{72}^{[5]} = W_{71}^{[0]} \oplus 1$
$W_{76}^{[30]} = W_{71}^{[0]} \oplus 1$	$W_{76}^{[30]} = W_{71}^{[0]} \oplus 1$
$W_{74}^{[7]} = W_{73}^{[2]} \oplus 1$	$W_{74}^{[7]} = W_{73}^{[2]} \oplus 1, W_{73}^{[2]} = z_1$
$W_{75}^{[6]} = W_{74}^{[1]} \oplus 1$	$W_{75}^{[6]} = W_{74}^{[1]} \oplus 1$
$W_{76}^{[6]} = W_{75}^{[1]} \oplus 1$	$W_{76}^{[6]} = W_{75}^{[1]} \oplus 1$
$W_{76}^{[1]} = W_{76}^{[0]} \oplus 1$	$W_{76}^{[1]} = z_2, \quad W_{76}^{[0]} = z_3$
$W_{77}^{[1]} = W_{77}^{[0]} \oplus z_1$	$W_{77}^{[1]} = z_4, \quad W_{77}^{[0]} = z_5$
$W_{77}^{[2]} = W_{77}^{[1]} \oplus 1$	$W_{77}^{[2]} = z_6$
$W_{77}^{[8]} = W_{76}^{[3]} \oplus 1, W_{76}^{[3]} = z_2$	$W_{76}^{[3]} = z_7, \quad W_{77}^{[8]} = z_8$
$W_{78}^{[0]} = W_{74}^{[7]}$	$W_{78}^{[0]} = z_9$
$W_{78}^{[3]} = z_3$	$W_{78}^{[3]} = z_{10}$
$W_{78}^{[7]} = z_4$	$W_{78}^{[7]} = z_{11}$
$W_{79}^{[2]} = z_5$	$W_{79}^{[2]} = z_{12}$
$W_{79}^{[4]} = z_6$	$W_{79}^{[4]} = z_{13}$

Table 3. Message constraints in the final steps. The z_i are fixed to `0` or `1` to define variant of the trail with distinct output differences. We use three more constraints than [27].

of the results, and to increase the precision of the heuristic probabilities. This strategy leads to a set of 8768 possible output differences, grouped in 2304 bundles. We list the output differences with (almost) optimal probability that we have identified in Table 4, with the corresponding bundles (we do not give the set with all considered differences due its large size).

Next, we build the set $\mathcal{S}$ of acceptable differences, and the graph $\mathcal{G}$ that indicates the sequences of near-collision blocks to use to cancel the differences in $\mathcal{S}$. We first build the graph $\mathcal{G}'$ as explained in Section 4.2. We use a limit on the cost of the nodes added to graph: we only consider nodes that have a path with cost at most $18 \cdot C_{\text{block}}$ in the graph $\mathcal{G}'$ (where this cost is computed with a single path, without using clustering). This yield a set of $2^{33.78}$ unique differences. After optimizing the cost with clustering, most of the nodes have a cost at most $4.5 \cdot C_{\text{block}}$, and we use a subset of the graph by bounding the cost of the near-collision stage. We describe various trade-offs in Table 5: a larger set reduces the cost of the birthday stage, but increase the cost of the near-collision stage.

We note that the memory requirements of our attack are rather limited: one just has to store the graph, and the chains for the birthday phase. With the parameters we propose, this represents less than 1TB.

Bundle	Output difference					Proba ($-\log$)
$\mathcal{B}_1$	0xffffedea	0xfffffff70	0x00000000	0x00000002	0x80000000	19.17
	0xffffedee	0xfffffff70	0x00000000	0x00000002	0x80000000	19.17
	0xffffefea	0xfffffff80	0x00000000	0x00000002	0x80000000	19.17
	0xffffefee	0xfffffff80	0x00000000	0x00000002	0x80000000	19.17
	0xfffffe5ec	0xfffffff30	0x80000000	0x00000002	0x80000000	19.36
	0xfffffe7ec	0xfffffff40	0x80000000	0x00000002	0x80000000	19.36
$\mathcal{B}_2$	0xffffedea	0xfffffff70	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffedee	0xfffffff70	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefea	0xfffffff80	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefee	0xfffffff80	0x00000000	0xfffffffe	0x80000000	19.17
	0xfffffe5ec	0xfffffff30	0x80000000	0xfffffffe	0x80000000	19.36
	0xfffffe7ec	0xfffffff40	0x80000000	0xfffffffe	0x80000000	19.36
$\mathcal{B}_3$	0xffffedea	0xfffffff70	0x00000000	0x00000002	0x80000000	19.17
	0xffffedee	0xfffffff70	0x00000000	0x00000002	0x80000000	19.17
	0xffffefea	0xfffffff80	0x00000000	0x00000002	0x80000000	19.17
	0xffffefee	0xfffffff80	0x00000000	0x00000002	0x80000000	19.17
	0xffffff5ec	0xffffffb0	0x80000000	0x00000002	0x80000000	19.36
	0xffffff7ec	0xffffffc0	0x80000000	0x00000002	0x80000000	19.36
$\mathcal{B}_4$	0xffffedea	0xfffffff70	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffedee	0xfffffff70	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefea	0xfffffff80	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefee	0xfffffff80	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffff5ec	0xffffffb0	0x80000000	0xfffffffe	0x80000000	19.36
	0xffffff7ec	0xffffffc0	0x80000000	0xfffffffe	0x80000000	19.36
$\mathcal{B}_5$	0xffffedaa	0xfffffff6e	0x00000000	0x00000002	0x80000000	19.17
	0xffffedae	0xfffffff6e	0x00000000	0x00000002	0x80000000	19.17
	0xffffefaa	0xfffffff7e	0x00000000	0x00000002	0x80000000	19.17
	0xffffefae	0xfffffff7e	0x00000000	0x00000002	0x80000000	19.17
	0xfffffe5ac	0xfffffff2e	0x80000000	0x00000002	0x80000000	19.36
	0xfffffe7ac	0xfffffff3e	0x80000000	0x00000002	0x80000000	19.36
$\mathcal{B}_6$	0xffffedaa	0xfffffff6e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffedae	0xfffffff6e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefaa	0xfffffff7e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefae	0xfffffff7e	0x00000000	0xfffffffe	0x80000000	19.17
	0xfffffe5ac	0xfffffff2e	0x80000000	0xfffffffe	0x80000000	19.36
	0xfffffe7ac	0xfffffff3e	0x80000000	0xfffffffe	0x80000000	19.36
$\mathcal{B}_7$	0xffffedaa	0xfffffff6e	0x00000000	0x00000002	0x80000000	19.17
	0xffffedae	0xfffffff6e	0x00000000	0x00000002	0x80000000	19.17
	0xffffefaa	0xfffffff7e	0x00000000	0x00000002	0x80000000	19.17
	0xffffefae	0xfffffff7e	0x00000000	0x00000002	0x80000000	19.17
	0xffffff5ac	0xffffffae	0x80000000	0x00000002	0x80000000	19.36
	0xffffff7ac	0xffffffbe	0x80000000	0x00000002	0x80000000	19.36
$\mathcal{B}_8$	0xffffedaa	0xfffffff6e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffedae	0xfffffff6e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefaa	0xfffffff7e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffefae	0xfffffff7e	0x00000000	0xfffffffe	0x80000000	19.17
	0xffffff5ac	0xffffffae	0x80000000	0xfffffffe	0x80000000	19.36
	0xffffff7ac	0xffffffbe	0x80000000	0xfffffffe	0x80000000	19.36

Table 4. Bundles of trails with (near) optimal cost and the corresponding probability for steps 61–79. For each bundle $\mathcal{B}_i$ in the table, there are 32 related bundles where we flip some of the messages bits, that can be constructed as:
$B_0 = \{\mathcal{B}_i\}$
$B_1 = \left\{\{\beta + (2^5, 0, 0, 0, 0) \mid \beta \in \mathcal{B}\} \;\middle|\; \mathcal{B} \in B_0\right\} \cup B_0$
$B_2 = \left\{\{\beta + (2^3, 0, 0, 0, 0) \mid \beta \in \mathcal{B}\} \;\middle|\; \mathcal{B} \in B_1\right\} \cup B_1$
$B_3 = \left\{\{\beta + (2^{13}, 2^8, 0, 0, 0) \mid \beta \in \mathcal{B}\} \;\middle|\; \mathcal{B} \in B_2\right\} \cup B_2$
$B_4 = \left\{\{\beta + (2^9, 2^4, 0, 0, 0) \mid \beta \in \mathcal{B}\} \;\middle|\; \mathcal{B} \in B_3\right\} \cup B_3$
$B_5 = \left\{\{\beta + (2^6, 2, 0, 0, 0) \mid \beta \in \mathcal{B}\} \;\middle|\; \mathcal{B} \in B_4\right\} \cup B_4$.
The set used in [26] corresponds to bundles $\mathcal{B}_1$ to $\mathcal{B}_4$, with extension rules B_1 to B_4. Note that most output differences appears in several bundles.

6.1 Limiting the number of near-collision blocks

The attack above is optimized to minimize the time complexity of the attack, but this can result in long paths in the graph. For instance, when starting from a random difference with cost at most $3.0 \cdot C_{\text{block}}$, a random path has on average 15.7 near-collision blocks, but the maximal length is 26 near-collision blocks. This might be impractical for some applications of chosen-prefix collision attacks, and the work needed to generate all the differential trails for the near-collision blocks might also be an issue.

Therefore, we propose an alternative attack where we limit the length of the paths in the graph $\mathcal{G}$. This result in a slightly higher complexity, but might be better in practice. More precisely, we first construct a graph with only paths of length 1, and we iteratively build graphs by increasing the length of allowed paths. Note that a given difference can often be reached by many paths of varying length, and the cost of a node decreases when allowing longer paths.

We have constructed exactly the graph with all paths of length at most 4, and all paths of length at most 8 and cost at most $3.5 \cdot C_{\text{block}}$; for larger parameters, we cannot build the full graph, but we can build an approximation by limiting the set of values as in the previous construction. We give the size of the corresponding sub-graphs in Table 6. As we can see, with 8 near-collision blocks we already have a set $\mathcal{S}$ almost as large as the set corresponding to the previous attack (cf. Table 5), so that limiting the attack to 8 blocks has a small impact on the complexity. We can even find chosen-prefix collisions with just 4 near-collision blocks with a small cost increase, using a larger threshold on the maximum cost per block. We evaluate the complexity of such attacks in detail in Table 7.

We can also study the sparseness of the values in $\mathcal{S}$ to better understand the difficulty of building the differential trails for the near collision blocks. Using the set of size $2^{29.71}$ with a limit of 8 near-collision blocks and a maximum cost of $3.0 \cdot C_{\text{block}}$, the maximum weight in the differences is 26, and the average is 15.4 (using the non-adjacent form — NAF).

6.2 Birthday Stage

For the birthday stage of the attack, we follow the approach given in [26]: we consider a truncation of the `SHA-1` state by keeping bits which are likely to contain a difference, and we use the distinguished points technique of [31]. Parameters for the birthday step with various choice of $\mathcal{G}$ are given in Table 5; we now explain in detail the case where the maximum cost of the near-collision stage is set to $3.0 \cdot C_{\text{block}}$. First, we truncate the state to 98 bits[9] so that for a random pair of values with their difference in $\mathcal{S}$, there is a probability 0.78 that the values collide on 98 bits (this probability has been computed with the tools from [14]). Reciprocally, if two truncated `SHA-1` outputs are equal, then their difference is

[9] Given by mask `0x7f800000`, `0xfffc0001`, `0x7ffff800`, `0x7fffff80`, `0x7fffffff`

in the set $\mathcal{S}$ with probability $2^{-31.97}$. Therefore, the birthday stage will require on average $2 \cdot 2^{31.97}$ collisions in the following function:

$$f(r) := \begin{cases} \tau(h(cv\ , \mathrm{pad}(u))) & \text{if } u[0] = 1; \\ \tau(h(cv', \mathrm{pad}(u))) & \text{else.} \end{cases}$$

In order to keep the cost of rerunning the trail low, we use chains of average length 2^{31} (*i.e.* a point u is distinguished when $u < 2^{98-31}$). Therefore, the expected complexity of the birthday stage is[10]:

$$T = \sqrt{\pi/2 \cdot 2^{98} \cdot 2^{32.97}} \approx 2^{65.81}\ \texttt{SHA-1}\ \text{computations}$$
$$M = 2^{65.81}/2^{31} \cdot 19\,\text{bytes} \approx 570\,\text{GB},$$

and the cost to re-run the chains to locate collisions is only $2^{32.97} \cdot 2 \cdot 2^{31} \approx 2^{64.97}$. Finally, we can evaluate the complexity of the full attacks as: $2^{65.81} + 2^{64.97} + 3.0 \cdot C_{\text{block}}$.

Set $\mathcal{S}$		Birthday parameters					
Max cost	Size	Mask	Proba	# coll.	Chain len.	# chain	Attack cost
$2.0 \cdot C_{\text{block}}$	$2^{24.66}$	106 bits	0.71	$2^{30.83}$	2^{34}	$2^{34.74}$	$2^{68.74} + 2^{65.83} + 2.0 \cdot C_{\text{block}}$
$2.5 \cdot C_{\text{block}}$	$2^{28.59}$	102 bits	0.65	$2^{31.03}$	2^{32}	$2^{34.84}$	$2^{66.84} + 2^{64.03} + 2.5 \cdot C_{\text{block}}$
$3.0 \cdot C_{\text{block}}$	$2^{30.95}$	98 bits	0.76	$2^{32.44}$	2^{31}	$2^{34.55}$	$2^{65.55} + 2^{64.44} + 3.0 \cdot C_{\text{block}}$
$3.5 \cdot C_{\text{block}}$	$2^{32.70}$	98 bits	0.76	$2^{30.70}$	2^{30}	$2^{34.68}$	$2^{64.68} + 2^{61.70} + 3.5 \cdot C_{\text{block}}$
$4.0 \cdot C_{\text{block}}$	$2^{33.48}$	98 bits	0.74	$2^{29.95}$	2^{30}	$2^{34.30}$	$2^{64.30} + 2^{60.95} + 4.0 \cdot C_{\text{block}}$
$4.5 \cdot C_{\text{block}}$	$2^{33.66}$	98 bits	0.74	$2^{29.77}$	2^{30}	$2^{34.21}$	$2^{64.21} + 2^{60.77} + 4.5 \cdot C_{\text{block}}$

Table 5. Trade-offs between the cost of birthday phase and the near-collision phase.

6.3 Near-Collision Stage

An important parameter to evaluate the cost of the attack is C_{block}, the complexity to find near-collision blocks. An optimistic hypothesis is that we can find them with same complexity as in the attack of [27], *i.e.* $C_{\text{block}} = 2^{64.7}$. As mentioned earlier, we have conducted tests to verify that one can easily find short non-linear differential paths, regardless of the input chaining difference and value, to allow for a good use of neutral bits (one path example is given in Table 8).

We note that our trails are somewhat more constrained than the trails used in the collision attack, because we have denser chaining value differences and we have a few more conditions in the last round, as seen in Table 3. This could lead to fewer degrees of freedom than in the collision attack of Stevens *et al.*, and

[10] To store a chain, we use 40 bits for the starting point, 40 bits for the length, and $98 - 31 = 67$ bits for the output, *i.e.* 19 bytes in total.

Max Cost	1 bl.	2 bl.	3 bl.	4 bl.	5 bl.	6 bl.	7 bl.	8 bl.
$2.0 \cdot C_{\text{block}}$	$2^{9.17}$	$2^{16.30}$	$2^{19.92}$	$2^{22.05}$	$2^{23.13}$	$2^{23.95}$	$2^{24.44}$	$2^{24.55}$
$2.5 \cdot C_{\text{block}}$	$2^{10.17}$	$2^{16.62}$	$2^{21.04}$	$2^{23.76}$	$2^{25.50}$	$2^{26.58}$	$2^{27.38}$	$2^{27.92}$
$3.0 \cdot C_{\text{block}}$	$2^{10.17}$	$2^{17.10}$	$2^{21.76}$	$2^{24.66}$	$2^{26.58}$	$2^{27.95}$	$2^{28.96}$	$2^{29.71}$
$3.5 \cdot C_{\text{block}}$	$2^{12.53}$	$2^{17.89}$	$2^{22.47}$	$2^{25.62}$	$2^{27.70}$	$2^{29.18}$	$2^{30.29}$	$2^{31.22}$
$4.0 \cdot C_{\text{block}}$	$2^{12.53}$	$2^{18.60}$	$2^{22.97}$	$2^{26.34}$	$\geq 2^{28.68}$	$\geq 2^{30.35}$	$\geq 2^{31.55}$	$\geq 2^{32.15}$
$5.0 \cdot C_{\text{block}}$	$2^{12.53}$	$2^{19.65}$	$2^{24.18}$	$2^{27.44}$	$\geq 2^{29.83}$	$\geq 2^{31.64}$	$\geq 2^{32.95}$	$\geq 2^{33.04}$
$6.0 \cdot C_{\text{block}}$	$2^{12.53}$	$2^{19.79}$	$2^{24.81}$	$2^{28.26}$	$\geq 2^{30.74}$	$\geq 2^{32.55}$	$\geq 2^{33.59}$	$\geq 2^{33.59}$
$7.0 \cdot C_{\text{block}}$	$2^{13.09}$	$2^{20.37}$	$2^{25.30}$	$2^{28.82}$	$\geq 2^{31.33}$	$\geq 2^{32.93}$	$\geq 2^{33.77}$	$\geq 2^{33.77}$
$8.0 \cdot C_{\text{block}}$	$2^{13.09}$	$2^{20.62}$	$2^{25.72}$	$2^{29.27}$	$\geq 2^{31.72}$	$\geq 2^{33.09}$	$\geq 2^{33.81}$	$\geq 2^{33.81}$

Table 6. Size of the set $\mathcal{S}$ with various limits on the maximum cost and on the number of near-collision blocks. We give a lower bound when we couldn't compute the full set.

Set $\mathcal{S}$			Birthday parameters					
Max bl.	Max cost	Size	Mask	Proba	# coll.	Chain len.	# chain	Attack cost
4	$4.0 \cdot C_{\text{block}}$	$2^{26.34}$	106 bits	0.48	$2^{29.70}$	2^{33}	$2^{35.18}$	$2^{68.18} + 2^{63.70} + 4.0 \cdot C_{\text{block}}$
4	$5.0 \cdot C_{\text{block}}$	$2^{27.44}$	102 bits	0.67	$2^{32.14}$	2^{32}	$2^{35.40}$	$2^{67.40} + 2^{65.14} + 5.0 \cdot C_{\text{block}}$
4	$6.0 \cdot C_{\text{block}}$	$2^{28.26}$	102 bits	0.65	$2^{31.35}$	2^{32}	$2^{35.00}$	$2^{67.00} + 2^{64.35} + 6.0 \cdot C_{\text{block}}$
4	$7.0 \cdot C_{\text{block}}$	$2^{28.82}$	102 bits	0.64	$2^{30.82}$	2^{32}	$2^{34.74}$	$2^{66.74} + 2^{63.82} + 7.0 \cdot C_{\text{block}}$
4	$8.0 \cdot C_{\text{block}}$	$2^{29.26}$	102 bits	0.63	$2^{30.39}$	2^{32}	$2^{34.52}$	$2^{66.52} + 2^{63.39} + 8.0 \cdot C_{\text{block}}$
8	$2.0 \cdot C_{\text{block}}$	$2^{24.55}$	106 bits	0.71	$2^{30.94}$	2^{34}	$2^{34.80}$	$2^{68.80} + 2^{65.94} + 2.0 \cdot C_{\text{block}}$
8	$2.5 \cdot C_{\text{block}}$	$2^{27.92}$	102 bits	0.63	$2^{31.75}$	2^{32}	$2^{35.20}$	$2^{67.20} + 2^{64.75} + 2.5 \cdot C_{\text{block}}$
8	$3.0 \cdot C_{\text{block}}$	$2^{29.71}$	98 bits	0.73	$2^{33.73}$	2^{31}	$2^{35.19}$	$2^{66.19} + 2^{65.73} + 3.0 \cdot C_{\text{block}}$
8	$3.5 \cdot C_{\text{block}}$	$2^{31.22}$	98 bits	0.72	$2^{32.23}$	2^{30}	$2^{35.44}$	$2^{65.44} + 2^{63.23} + 3.5 \cdot C_{\text{block}}$

Table 7. Trade-offs between the cost of birthday phase and the near-collision phase with a limited number of near-collision blocks (4 or 8).

i	A_i	W_i
-4:	10110011001011000101111011010101	
-3:	11011000110000100010011 1un01un01	
-2:	00101010001101011101 1unnn1011n11	
-1:	00101010111000011un1nn0010n0u111	
00:	1010111000111un1u0110n1u0nn0un1n	1011un01010001010101111110-10-u0
01:	10100u101u0u10101n1nu0111n-u1-1u	nu11100111011000111110-00-00n0110
02:	1u1u0nn010nuunnu11011uuuu0001uu1	u1nn0u00010001000101011 1---unn00
03:	u1un01uunn1u1010u0u101101nu11uu1	00uuun11110101111001110 10000-u1-
04:	n0110unnnnnnnnnnnnnnn11nu1000u1n1	n0nunu00----001----0---1000uu0u1
05:	un0n011100--11001--111-1u1uu1u11	10u-1--101110-000-1100-0110n-00-
06:	1101-0-101101011110101-10nun01uu	--u--u1---------------0101uun--
07:	0nuu-00----------------0100100uu	xun-nu------------1-1----11u0u--
08:	---u01----------------0--0n010-u	----un-----------------------u0
09:	0n--------------------0--1-1--0u	xn----------------------0--n-0--
10:	1--1-1------------------0---1---	x-nx-x-------------1-------uxx--
11:	-1n-----------------------0-----	--u0nn-------------------1-1-u--
12:	----0---------------------1-----	n-nxxu---------------------un---
13:	n---1----------------------------	x-uu-0----------------------u----
14:	--n----------------------------	-------------------------1-un--
15:	u-1-1--------------------------	x-nxn----------------------n----
16:	un0-0--------------------------	----u----------------------nu---

Table 8. Example of a `SHA-1` non-linear differential path generated for one of the differences in $\mathcal{S}$. Notations follow [8]. $\delta = [-2^{17} - 2^{15} + 2^{10} - 2^8 + 2^5 + 2^6 - 2^2 + 2^0, -2^{13} + 2^{11} + 2^{10} + 2^5 - 2^3, -2^5 + 2^0, -2^4 - 2^0, 0]$, with cost $2.954 \cdot C_{\text{block}}$.

increase the cost of finding a conforming block. In particular, this can affect the use of accelerating techinques such as neutral bits and boomerangs; boomerangs are the most powerful technique, but they require significant degrees of freedom in the path construction. Therefore, we also consider a conservative complexity estimate, where we assume that boomerangs are no longer available. Since there are three boomerangs in the trail of [27], this would give $C_{\text{block}} = 2^{67.7}$.

Our experiments show that that those assumptions are reasonable. The path given in Table 8 is about as constrained as the path used for the second block of the collision attack [27] in the first round. In particular, most condition are in the first 6 steps, and don't affect the use of neutral bits, and the same three boomerang are available. In general we expect similar results with a few boomerangs, but this might of course vary depending on the exact chaining input diffence/value.

Finally, with the optimistic hypothesis, the best trade-off is to use a limit of $3.5 \cdot C_{\text{block}}$, for a total complexity of

$$2^{64.68} + 2^{61.70} + 3.5 \cdot C_{\text{block}} \approx 2^{66.9} \qquad (\text{using } C_{\text{block}} = 2^{64.7})$$

With the conservative hypothesis, the best trade-off is to set the limit at $2.5 \cdot C_{\text{block}}$, for a total complexity of

$$2^{66.84} + 2^{64.03} + 2.5 \cdot C_{\text{block}} \approx 2^{69.35} \qquad (\text{using } C_{\text{block}} = 2^{67.7})$$

There are other trade-offs possible between the various parameters of attack. For instance, we discussed attacks with a limited number of near-collision blocks in Section 6.1; we can now evaluate the complexity of the resulting attacks. If we limit the attack to 8 near-collision blocks, the best trade-offs give the following complexities for the optimistic and conservative hypothesis respectively:

$$2^{65.44} + 2^{63.23} + 3.5 \cdot C_{\text{block}} \approx 2^{67.2} \qquad (\text{using } C_{\text{block}} = 2^{64.7})$$
$$2^{67.20} + 2^{64.75} + 2.5 \cdot C_{\text{block}} \approx 2^{69.5} \qquad (\text{using } C_{\text{block}} = 2^{67.7})$$

Even with a limit of only 4 near-collision blocks, we have a relatively small increase of the complexity, with the following trade-offs:

$$2^{66.74} + 2^{63.82} + 7.0 \cdot C_{\text{block}} \approx 2^{68.3} \qquad (\text{using } C_{\text{block}} = 2^{64.7})$$
$$2^{68.18} + 2^{63.70} + 4.0 \cdot C_{\text{block}} \approx 2^{70.2} \qquad (\text{using } C_{\text{block}} = 2^{67.7})$$

7 Conclusion and Future Works

This work puts another nail in the `SHA-1` coffin, with almost practical chosen-prefix collisions, between five and twenty-six times more expensive than the identical-prefix collisions recently demonstrated. This shows that continued usage of `SHA-1` for certificates or for authentication of handshake messages in TLS, SSH or IKE is dangerous, and could already be abused today by a well-motivated adversary. `SHA-1` has been broken since 2004, but it is still used in many security

systems; we strongly advise users to remove SHA-1 support to avoid downgrade attacks.

More generally, our results show that, for some hash functions, chosen-prefix collision attacks are much easier than previously expected, and potentially not much harder than a normal collision search.

Our research opens several new directions. Obviously, future work will have to implement this attack to demonstrate a real chosen-prefix collision for SHA-1. While the computation cost of our attack is somewhat practical, SHA-1 attacks still require a huge computation power (thousands of GPUs in order to obtain the chosen-prefix collision in a reasonable time) and a large implementation effort. For a concrete demonstration, a good target would be to break a protocol such as TLS or IKE, or to build a rogue certificate authority.

Another research direction is to study how one can improve SHA-1 collision attacks, not only for minimising the cost of finding a simple collision, but to improve our chosen-prefix collision search complexity. In particular, our attack requires the ability to reach many distinct output differences for the compression function. In this paper, to simplify our analysis, we only considered the differential trail from [27] because a real collision was found with this trail, and a precise complexity evaluation was conducted. However, it should be possible to increase the pool of available differences, and further reduce the total complexity, by using other (slightly more costly) differential trails.

Finally, a last direction is to evaluate how our strategy actually applies to other hash functions, such as RIPEMD, (reduced-round) SHA-2, or even others. Again, this will require a deep knowledge of the functions studied, as many details might impact the overall complexity. We can however expect that our attack strategy will be applicable mostly on classical Davies-Meier constructions inside a single-pipe Merkle-Damgård operating mode.

In order to make our easier to verify, we are publishing some additional data and code online at: https://github.com/Cryptosaurus/sha1-cp.

Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments. The second author is supported by Temasek Laboratories, Singapore.

References

[1] Bhargavan, K., Leurent, G.: Transcript collision attacks: Breaking authentication in TLS, IKE and SSH. In: NDSS 2016, The Internet Society (February 2016)

[2] Biham, E., Chen, R.: Near-collisions of SHA-0. In Franklin, M., ed.: CRYPTO 2004. Volume 3152 of LNCS., Springer, Heidelberg (August 2004) 290–305

[3] Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of LNCS., Springer, Heidelberg (May 2005) 36–57

[4] Brassard, G., ed.: CRYPTO. Volume 435 of Lecture Notes in Computer Science., Springer (1990)
[5] Chabaud, F., Joux, A.: Differential collisions in SHA-0. In Krawczyk, H., ed.: CRYPTO'98. Volume 1462 of LNCS., Springer, Heidelberg (August 1998) 56–71
[6] Damgård, I.: A Design Principle for Hash Functions. [4] 416–427
[7] Damgård, I.: A design principle for hash functions. In Brassard, G., ed.: CRYPTO'89. Volume 435 of LNCS., Springer, Heidelberg (August 1990) 416–427
[8] De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-step SHA-1: On the full cost of collision search. In Adams, C.M., Miri, A., Wiener, M.J., eds.: SAC 2007. Volume 4876 of LNCS., Springer, Heidelberg (August 2007) 56–73
[9] den Boer, B., Bosselaers, A.: Collisions for the compressin function of MD5. In Helleseth, T., ed.: EUROCRYPT'93. Volume 765 of LNCS., Springer, Heidelberg (May 1994) 293–304
[10] Joux, A., Peyrin, T.: Hash functions and the (amplified) boomerang attack. In Menezes, A., ed.: CRYPTO 2007. Volume 4622 of LNCS., Springer, Heidelberg (August 2007) 244–263
[11] Karpman, P., Peyrin, T., Stevens, M.: Practical free-start collision attacks on 76-step SHA-1. In Gennaro, R., Robshaw, M.J.B., eds.: CRYPTO 2015, Part I. Volume 9215 of LNCS., Springer, Heidelberg (August 2015) 623–642
[12] Klima, V.: Tunnels in hash functions: MD5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105 (2006) http://eprint.iacr.org/2006/105.
[13] Lamberger, M., Pramstaller, N., Rechberger, C., Rijmen, V.: Second preimages for SMASH. In Abe, M., ed.: CT-RSA 2007. Volume 4377 of LNCS., Springer, Heidelberg (February 2007) 101–111
[14] Leurent, G.: Analysis of differential attacks in ARX constructions. In Wang, X., Sako, K., eds.: ASIACRYPT 2012. Volume 7658 of LNCS., Springer, Heidelberg (December 2012) 226–243
[15] Mendel, F., Rijmen, V., Schläffer, M.: Collision attack on 5 rounds of Grøstl. In Cid, C., Rechberger, C., eds.: FSE 2014. Volume 8540 of LNCS., Springer, Heidelberg (March 2015) 509–521
[16] Merkle, R.C.: One Way Hash Functions and DES. [4] 428–446
[17] Merkle, R.C.: One way hash functions and DES. In Brassard, G., ed.: CRYPTO'89. Volume 435 of LNCS., Springer, Heidelberg (August 1990) 428–446
[18] National Institute of Standards and Technology: FIPS 180: Secure Hash Standard (May 1993)
[19] National Institute of Standards and Technology: FIPS 180-1: Secure Hash Standard (April 1995)
[20] National Institute of Standards and Technology: FIPS 180-2: Secure Hash Standard (August 2002)
[21] National Institute of Standards and Technology: FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (August 2015)
[22] Peyrin, T.: Cryptanalysis of Grindahl. In Kurosawa, K., ed.: ASIACRYPT 2007. Volume 4833 of LNCS., Springer, Heidelberg (December 2007) 551–567
[23] Rivest, R.L.: The MD4 message digest algorithm. In Menezes, A.J., Vanstone, S.A., eds.: CRYPTO'90. Volume 537 of LNCS., Springer, Heidelberg (August 1991) 303–311
[24] Rivest, R.L.: RFC 1321: The MD5 Message-Digest Algorithm. Internet Activities Board. (April 1992)

[25] Stevens, M.: Attacks on Hash Functions and Applications. PhD thesis, Leiden University (June 2012)
[26] Stevens, M.: New collision attacks on SHA-1 based on optimal joint local-collision analysis. In Johansson, T., Nguyen, P.Q., eds.: EUROCRYPT 2013. Volume 7881 of LNCS., Springer, Heidelberg (May 2013) 245–261
[27] Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In Katz, J., Shacham, H., eds.: CRYPTO 2017, Part I. Volume 10401 of LNCS., Springer, Heidelberg (August 2017) 570–596
[28] Stevens, M., Karpman, P., Peyrin, T.: Freestart collision for full SHA-1. In Fischlin, M., Coron, J.S., eds.: EUROCRYPT 2016, Part I. Volume 9665 of LNCS., Springer, Heidelberg (May 2016) 459–483
[29] Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In Naor, M., ed.: EUROCRYPT 2007. Volume 4515 of LNCS., Springer, Heidelberg (May 2007) 1–22
[30] Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In Halevi, S., ed.: CRYPTO 2009. Volume 5677 of LNCS., Springer, Heidelberg (August 2009) 55–69
[31] van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology **12**(1) (1999) 1–28
[32] Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of LNCS., Springer, Heidelberg (May 2005) 1–18
[33] Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In Shoup, V., ed.: CRYPTO 2005. Volume 3621 of LNCS., Springer, Heidelberg (August 2005) 17–36
[34] Wang, X., Yu, H.: How to break MD5 and other hash functions. In Cramer, R., ed.: EUROCRYPT 2005. Volume 3494 of LNCS., Springer, Heidelberg (May 2005) 19–35
[35] Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In Shoup, V., ed.: CRYPTO 2005. Volume 3621 of LNCS., Springer, Heidelberg (August 2005) 1–16