Counter-Cryptanalysis

Marc Stevens

CWI, Amsterdam, The Netherlands
marc@marc-stevens.nl

Abstract. We introduce *counter-cryptanalysis* as a new paradigm for strengthening weak cryptographic primitives against cryptanalytic attacks. Redesigning a weak primitive to more strongly resist cryptanalytic techniques will unavoidably break backwards compatibility. Instead, counter-cryptanalysis exploits unavoidable anomalies introduced by cryptanalytic attacks to detect and block cryptanalytic attacks while maintaining full backwards compatibility. Counter-cryptanalysis in principle enables the continued secure use of weak cryptographic primitives.

Furthermore, we present the first example of counter-cryptanalysis, namely the efficient detection whether any given single message has been constructed – together with an *unknown* sibling message – using a cryptanalytic collision attack on MD5 or SHA-1.

An immediate application is in digital signature verification software to ensure that an (older) MD5 or SHA-1 based digital signature is not a forgery using a collision attack. This would certainly be desirable for two reasons. Firstly, it might still be possible to generate malicious forgeries using collision attacks as too many parties still sign using MD5 (or SHA-1) based signature schemes. Secondly, any such forgeries are currently accepted nearly everywhere due to the ubiquitous support of MD5 and SHA-1 based signature schemes. Despite the academic push to use more secure hash functions over the last decade, these two real-world arguments (arguably) will remain valid for many more years.

Only due to counter-cryptanalysis were we able to discover that Flame, a highly advanced malware for cyberwarfare uncovered in May 2012, employed an as of yet unknown variant of our chosen-prefix collision attack on MD5 [SLdW07, SSA+09]. In this paper we disect the revealed cryptanalytic details and work towards the reconstruction of the algorithms underlying Flame's new variant attack. Finally, we make a preliminary comparision between Flame's attack and our chosen-prefix collision attack.

1 Introduction

1.1 Weak Cryptographic Primitives

Cryptographic primitives that are broken or weak due to the existence of cryptanalytic attacks should be retired in favor of a more secure one. However, in practice, widely used cryptographic primitives that are broken continue to be

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 129–146, 2013.

used long after their expiration date. This phenomenom is caused by many reasons among which are cost and/or risk considerations, unconvincing real-world abuse scenarios and even laxness.

However, in the case of weak digital signature schemes there is also the issue of supporting old signatures. It may well be impossible to replace all old weak signatures with more secure ones, as signatures tend to proliferate beyond the control of the original signer. It seems that therefore signature verifiers will continue to accept weak – and possibly malicious – signatures for a long time to come. Unfortunately, signature verifiers have no way of knowing whether all signers have actually retired the weak scheme and whether an 'old' weak signature is really an old one or just forged to look like one.

This is exactly what we're currently seeing for MD5-based signatures in practice. MD5 was first proven to be broken in 2004 by Wang et al.[WY05], however the first truly convincing attack scenario using MD5 collisions was our construction of a rogue Certification Authority from 2008 using a more powerful attack called the chosen-prefix collision attack [SSA+09]. MD5 has been explicitly disallowed for digital signatures for Certification Authorities ever since, but it's still used by some and still supported nearly everywhere.

1.2 Flame

An example showing that the continued support for weak signature schemes leaves one vulnerable is Flame [Cry12, Kas12]. Flame is a highly advanced malware for cyberwarfare discovered in May 2012, which spread itself locally by impersonating as a properly, but illegitimately, signed Windows Update security patch. Flame's code-signing certificate was obtained by fooling Microsoft into signing an colliding and innocuous-looking certificate using an MD5-based signature algorithm. As the to-be-signed part of both certificates were carefully crafted to result in the same MD5-hash using a chosen-prefix collision attack, the MD5-based signature is valid for both certificates.

Even though Microsoft was fully aware of these severe weaknesses of MD5 and spent great effort in migrating to more secure hash functions for *new* digital signatures at least since 2008, their software continued to accept (old) MD5-based digital signatures. Also, in their efforts they overlooked their use of MD5-based signatures for licensing purposes in their Terminal Server Licensing Service up to the discovery of Flame in 2012. This, together with other unforeseen circumstances, allowed the creation of Flame's properly, but illegitimately, signed security patch that was trusted by *all* versions of the Windows [MS12a].[1]

1.3 Counter-Cryptanalysis

We introduce *counter-cryptanalysis* as a new paradigm for strengthening weak cryptographic primitives against cryptanalytic attacks by exploiting subtle, unavoidable anomalies introduced by the cryptanalytic attack. This might seem to

[1] Any license certificate produced by the Terminal Server Licensing Service could directly be used to attack Windows Vista and earlier versions, but not later versions.

be impossible for, e.g., passive cryptanalytic attacks on public and/or private key encryption schemes. But any active cryptanalytic attack that feeds carefully crafted inputs to the cryptographic primitive may thereby introduce subtle unavoidable anomalies that can be exploited to detect such attacks. In effect, counter-cryptanalysis protects against cryptanalytic attacks and thereby may prevent significant leaks or damages.

Note that in contrast to a strengthened redesign of the cryptographic primitive, applying counter-cryptanalysis does not alter the cryptographic primitive intrinsically. Thus counter-cryptanalysis can be applied transparantly in the cryptographic primitive, only altering its behaviour when a cryptanalytic attack is detected. Thereby in principle enabling the continued secure use of a weak primitive for full backwards compatibility.

1.4 Collision Attack Detection

We also introduce the first example of counter-cryptanalysis, namely the efficient detection whether any given single message has been constructed using a cryptanalytic collision attack on MD5 or SHA-1. In particular our novel techniques solves the above verifiers problem as he can now assess whether a message having a MD5-based or SHA-1-based signature is part of a forgery attack using a cryptanalytic collision attack.

Although one way to use our novel technique is to obtain, together with the MD5 or SHA-1 hash, an auxilary boolean output indicating whether a cryptanalytic attack has been detected. This auxilary boolean output can then be used by the application to decide to invalidate the signature as well as informing the user of a forged signature. Another possibility to effectively invalidate a forgery (attempt) that does not require changes at the application level is to ensure that the two colliding messages result in different outputs, e.g., by outputting the truncated SHA-256 hash or outputting a random hash value instead.

1.5 Overview

The rest of this paper is split into two parts. In Sect. 2, we first introduce our novel collision detection algorithm and apply it to both MD5 and SHA-1. Next in Sect. 3, we discuss the discoveries made by analyzing Flame's malicious certificate using our counter-cryptanalysis technique and our work towards the reconstruction of the underlying algorithms and our preliminary conclusions.

2 Detection of Cryptanalytic Collision Attacks

2.1 Brief Background on Collision Attacks

MD5 and SHA-1 are cryptographic hash functions that use the Merkle-Damgård construction in which the security of the hash function is reduced to that of a compression function that takes as input an Intermediate Hash Value IHV and

512-bit message block B. The compression starts with a working state WS_0 initialized with IHV and goes through 64 (MD5) or 80 (SHA-1) steps $t = 0, \ldots$ computing state WS_{t+1} from WS_t. Finally it outputs the sum of IHV and the last working state. A collision for a hash function is a pair of messages (M, M') that have the same hash. For any named variable X related to M, we denote by X' the same variable for M'.

The first collision attack on MD5 is due to Wang et al.[WY05] and is constructed from two sequential near-collision attacks on the compression function. Each near-collision attack starts with a given (IHV, IHV')-pair with a known difference denoted by δIHV and uses specific message block differences denoted by δB. It is based on a differential path that describes exactly how the input differences δIHV and δB propagate through the compression function, for which then a solution (B, B') with $B' = B + \delta B$ is found. As Wang et al.'s attack requires a zero δIHV before the two near-collision blocks, this type of collision attack is called an identical-prefix collision attack.

The more powerful chosen-prefix collision attack [SLdW07] can start from an arbitrary (IHV, IHV') pair. It first uses a birthday search to obtain a new (IHV_b, IHV_b') pair whose difference δIHV_b has a specific form. Then it employs a series of near-collision attacks that iteratively reduces δIHV_b to zero and thereby results in a collision.

2.2 Exploiting Cryptanalytic Necessities

The main principle of our novel technique of detecting collision attacks is to detect the last near-collision block of a collision attack and uses two key observations on the literature on MD5 and SHA-1 cryptanalysis:

- There are only a small number of possible message block differences that may lead to feasible near-collision attacks;
- All published MD5 and SHA-1 collision attacks use differential paths that at some step have no differences at all in the working state, or – in the case of MD5 – the differences $(2^{31}, 2^{31}, 2^{31}, 2^{31})$ (see [dBB93]).[2]

Due to these observations it is possible to check for collision attacks given only one message of a colliding pair of messages. First we present our basic algorithm and then prove its correctness if the message was actually constructed using a collision attack. Then we argue that the probability of a false positive is practically negligible. Lastly, we apply our algorithm to MD5 and SHA-1 specifically.

Algorithm. We present our collision detection algorithm in Alg. 2-1 that should work for any Merkle-Damgård hash function with a MD4-style compression function and in particular for MD5 and SHA-1. Our algorithm depends on a list of triples $(\delta B, i, \delta WS_i)$ for which there may exist a feasible collision attack that uses

[2] The reason for this is simple: these working state differences can be maintained at every step of the 64 steps of MD5Compress with probability at least 1/2 if not 1.

Algorithm 2-1. Last near-collision block detection

This algorithm returns `True` if a near-collision attack is detected and `False` otherwise. For a given message M, let $M_0, \ldots, M_{N-1}$ be the N message blocks that result from the padding and the splitting of M by the hash function. For $k \in \{0, \ldots, N-1\}$ do the following:

1. Let IHV_k be the intermediate hash value before the message block M_k is processed.
2. Initialize the working state WS_0 with IHV_k, compute steps $0, \ldots, S-1$ resulting in working states $WS_1, \ldots, WS_S$ and determine IHV_{k+1} using IHV_k and WS_S.
3. For each possible combination of values for message block differences δB, step i and working state differences δWS_i belonging to a feasible near-collision attack do the following:
   (a) Apply the message block differences δB to M_k to obtain M'_k.
   (b) Apply the working state differences δWS_i to WS_i to obtain WS'_i.
   (c) Compute steps $i-1, \ldots, 0$ backwards to obtain the working states $WS'_{i-1}, \ldots, WS'_0$.
   (d) Compute steps $i, \ldots, S-1$ to obtain the working states $WS'_{i+1}, \ldots, WS'_S$.
   (e) Determine IHV'_k from WS'_0 and IHV'_{k+1} from IHV'_k and WS'_S.
   (f) If $IHV'_{k+1} = IHV_{k+1}$ then (M_k, M'_k) is a near-collision block pair: return `True`
4. Return `False`

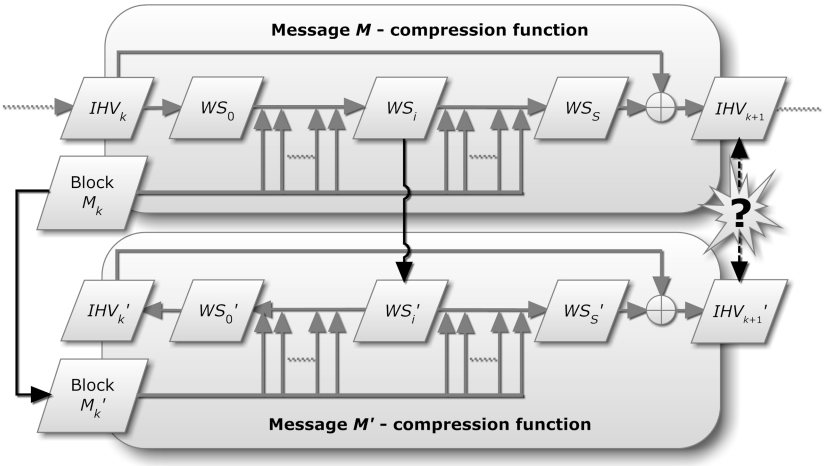
message block differences δB and always uses working state differences δWS_i at step i. For a total of C triples, the runtime-complexity of Alg. 2-1 for a message M is approximately $C+1$ times the runtime-complexity of computing the hash value of M.

Correctness. First we assume that our list of triples $(\delta B, i, \delta WS_i)$ is exhaustive for all possible feasible collision attack. For two colliding messages M and M' constructed with a feasible collision attack, let (M_k, M'_k) be the last near-collision block pair of a collision attack. Let IHV_{k+1} and IHV'_{k+1} be the intermediate hash values just after applying the compression function to M_k and M'_k in the hash value computation of M and M', respectively. Evidently, it follows that

$$IHV_{k+1} = IHV'_{k+1},$$

however as only the message M is given, the values of M', M'_k and IHV'_{k+1} are not directly known. Since M and M' were constructed with a feasible collision attack, there exists a triple $(\delta B, i, \delta WS_i)$ in our list such that $\delta B = \delta M_k$ and this last near-collision attack uses working state differences δWS_i at step i. As illustrated in Fig. 1, step 3 of Alg. 2-1 computes IHV_{k+1} and IHV'_{k+1} and tests for the telltale condition $IHV_{k+1} = IHV'_{k+1}$ in the following manner.

The hash value computation of M gives us values for the input IHV_k, output IHV_{k+1} and intermediate state WS_i (the working state before step i) of the compression function applied to IHV_k and M_k. Since we know the message block differences and the working state differences by assumption, we can determine the message block M'_k and the working state WS'_i associated with the message



If the message (upper half) was constructed using a collision attack and we correctly guess both the working state differences at a certain step and the used message block differences, then we obtain values of the internal computation of its sibling message (lower half). These are sufficient to reconstruct the entire compression function computation for this sibling block and verify whether there is a collision: $IHV'_{k+1} = IHV_{k+1}$.

Fig. 1. Detection of near-collisions

M' that collides with M. Computing steps $i+1, \ldots, S$ (where $S = 64$ for MD5 and $S = 80$ for SHA-1) of the compression function using M'_k and WS'_i, we obtain working states $WS'_{i+1}, \ldots, WS'_S$. As the step functions of MD5 and SHA-1 are reversible we can also compute working states $WS'_{i-1}, \ldots, WS'_0$. The value of IHV'_k can be derived from WS'_0 and the value of IHV'_{k+1} can be computed from IHV'_k and WS'_S.

It is clear that Alg. 2-1 on input M for the value k in step 3 for the triple $(\delta B_i, i, \delta WS_i)$ will determine the correct value IHV'_{k+1}, verify that indeed $IHV'_{k+1} = IHV_{k+1}$ and therefore return `True`. What remains is to argue that the probability of a false positive, i.e., it returns `True` for a given message M which was *not* constructed using a cryptanalytic collision attack, is negligible.

False Positives. If the given message block is not part of a near-collision block pair then the guessed WS'_i is passed through all 64 or 80 steps of the compression function to determine IHV'_{k+1}. Therefore, we argue that if there was no cryptanalytic attack then the distribution of the resulting value IHV'_{k+1} is close to the uniform distribution. Hence, the probability of a false positive, namely that $IHV'_{k+1} = IHV_{k+1}$, is thereby approximately $C \cdot 2^{-L}$ where L is the bit length of the hash value and C is the number of triples attempted in step 3 of Alg. 2-1 as before.

Interestingly, the false positive probability may be prove to be higher when there exists a differential path compatible with one of the combinations of δB, i and δWS_i that holds with probability higher than 2^{-L}. So far only one non-zero differential path is known with probability higher than 2^{-128} for MD5 (and none for SHA-1), namely the differential path consisting of differences in the most significant bit [dBB93], and this differential path is treated as a special case below and also checks for the necessary second-last near-collision block. So if nevertheless the false positive probability proves to be higher than we conjecture then this may well point towards interesting unknown cryptanalytic weaknesses.

2.3 Application to MD5

Alg. 2-1 can be directly applied to MD5. What remains is to determine possible combinations of values for message block differences δB, step i and working state differences δWS_i that belong to a feasible near-collision attack. The message block differences are additive in $\mathbb{Z}/2^{32}\mathbb{Z}$ and for each message block M_k the message block M'_k can be either $M_k + \delta B$ or $M_k - \delta B$. There are two trivial different working state differences δWS_i that can be used for MD5, namely $(0, 0, 0, 0)$ and $(2^{31}, 2^{31}, 2^{31}, 2^{31})$, written more compactly as $\underline{0}$ and $\underline{2^{31}}$.

We refer to Sect. A for a list of 222 triples $(\delta B, i, \delta WS_i)$ derived from the literature. We do not guarantee that this list forms the exhaustive list of all combinations that lead to feasible near-collision attacks. However, it should be noted that interesting message block differences have been studied extensively for nearly a decade, which has resulted in the above mentioned list. Nevertheless, other combinations from future collision attacks can easily be added to this list.

All published near-collision attacks require complex differential steps in the first round, thereby requiring a high number of bitconditions, say at least 200. E.g., the differential paths by Wang et al. require roughly 300 bitconditions [WY05]. This implies that the probability of a false positive is dominated by the general $C \cdot 2^{-L}$ term explained earlier. Hence, the probability of a false positive is estimated as $222 \cdot 2^{-128}$ and thus negligible.

However, there is a special case. Due to the *pseudo-collision* attack against MD5's compression function by den Boer and Bosselaers [dBB93], there is also a special near-collision attack not yet included in the above list. It uses zero message block differences and $\delta WS_i = \underline{2^{31}}$ for all $i \in \{0, \ldots, 64\}$. One can test for this pseudo-collision attack using $\delta B = 0$, $i = 32$ and $\delta WS_{32} = \underline{2^{31}}$. The probability of a false positive is 2^{-48} which is not negligible. However, since it requires $\delta WS_0 = \underline{2^{31}}$ and thus $IHV_{\text{in}} = \underline{2^{31}}$, this pseudo-collision attack requires at least one preceding near-collision block to form a collision attack against MD5.

This observation calls for the following modification of Alg. 2-1 for MD5 to reduce the chance of a false positive to $222 \cdot 2^{-128} \cdot 2^{-48}$ for the case $\delta B = 0$. Whenever a near-collision block is detected in step 3.(f) for the combination $\delta B = 0$, $i = 32$ and $\delta WS_{32} = \underline{2^{31}}$ and before returning `True`, perform steps 1–4 of Alg. 2-1 on the previous message block M_{k-1} using all combinations that have $\delta B \neq 0$ and using the condition $IHV'_k = IHV_k + \underline{2^{31}}$ instead of

the condition $IHV'_k = IHV_k$. If this sub-instance returns `False` then the main instance continues with the next combination of δB, i and δWS_i. Otherwise, the main instance returns `True`.

Given a message M, the average complexity to detect whether M is constructed by a collision attack against MD5 using one of the given message differences is about $222 + 1 + 1 = 224$ times the complexity of computing the MD5 hash of M. It has a conjectured false positive probability of about $222 \cdot 2^{-128}$.

2.4 Application to SHA-1

Alg. 2-1 can be directly applied to SHA-1. Note that this is possible even though no actual colliding messages for SHA-1 are known yet. What remains is to determine possible combinations of values for message block differences δB, step i and working state differences δWS_i that belong to a feasible near-collision attack.

All known attempts at a SHA-1 collision attack are based on combining local collisions according to a disturbance vector $(DV_i)_{i=0}^{79} \in (\mathbb{Z}/2^{32}\mathbb{Z})^{80}$. Furthermore, Manuel [Man11] has found that all proposed disturbance vectors can be categorized into two classes. A disturbance vector from the first class denoted by $\mathrm{I}(j,b)$ is defined by $DV_j = \ldots = DV_{j+14} = 0$ and $DV_{j+15} = 2^b$. Similarly, a disturbance vector from the second class denoted by $\mathrm{II}(j,b)$ is defined by $DV_{j+1} = DV_{j+3} = RL(2^{31}, b)$ and $DV_{j+15} = 2^b$ and $DV_{j+i} = 0$ for $i \in \{0,2,4,5,\ldots,14\}$. For both classes, the remaining $DV_0,\ldots,DV_{j-1}$ and $DV_{j+16},\ldots,DV_{79}$ are determined through the message expansion relation.

For a given disturbance vector $(DV_i)_{i=0}^{79}$, the necessary message block differences are the XOR differences $(DW_i)_{i=0}^{15} = M_k \oplus M'_k$ determined as:

$$DW_i := \bigoplus_{(j,r)\in\mathcal{R}} RL(DV_{i-j}, r), \quad \mathcal{R} = \{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\},$$

where $DV_{-1},\ldots,DV_{-5}$ are given by the reversed message expansion relation:

$$DV_i = RR(DV_{i+16}, 1) \oplus DV_{i+2} \oplus DV_{i+8} \oplus DV_{i+13}, \quad i = -1,\ldots,-5.$$

For both disturbance vector $\mathrm{I}(j,b)$ and $\mathrm{II}(j,b)$ there are no differences at step $j+8$, hence to test for near-collision block pair using either disturbance vector we use Alg. 2-1 with the combination $(DW_t)_{t=0}^{15}$, $i = j+8$ and $\delta WS_i = (0,0,0,0,0)$.

Given the fact that no actual collisions are known yet, it is somewhat difficult to decide which triples to include. For this we refer to our recent analysis [Ste13] that seems to use the most appropriate cost function, namely one that is exact, exhaustive and takes the dependence of local collisions fully into account. However, due to the complex nature of constructing a collision attack, this cost function is not perfect as it does not accurately predict the final attack complexity. Nevertheless, we propose (a bit arbitrarily) to limit ourselves to the following 14 best disturbance vectors:

I(46,0), I(48,0), I(49,0), I(50,0), I(51,0), I(48,2), I(49,2),
II(46,0), II(50,0), II(51,0), II(52,0), II(53,0), II(54,0), II(56,0).

Similar to the case of MD5, it is always possible to add extra disturbance vectors to the above list in the future whenever it is believed it can lead to a feasible collision attack. Ignoring the first round, each disturbance vector has a probability in the order of 2^{-70} that a false positive occurs. Taking into account the complex differential steps necessary in the first round, we can safely assume that the probability of a false positive is negligible.

Given a message M, the average complexity to detect whether M is constructed by one of the above possibly feasible collision attacks against SHA-1 is about $14 + 1 = 15$ times the complexity of computing the SHA-1 hash of M. It has a conjectured false positive probability of about $14 \cdot 2^{-160}$.

3 Analyzing Flame's Chosen-Prefix Collision Attack

3.1 Background on Flame

Flame is a highly advanced malware for cyberwarfare and was discovered in May 2012 by the Iranian CERT, Kaspersky Lab and CrySyS Lab. It seemed to have targeted the Middle-East, with the most infections in Iran. We refer to the analysis of Kaspersky Lab and CrySyS Lab for more details on the functionality, purpose and origin of Flame. Here, we will focus on Flame's advanced propagation.

For a malware, it has a number of quite uncharacteristic features [Kas12, Cry12]. It has a modular design with up to 20 different plugins with different specific roles, each of which can be carefully selected prior infection. Flame is about 20MB in size as it also includes many different libraries such as for compression (zlib, bz2, ppmd), database (sqlite) and even a Lua virtual machine. Infections seem to occur with surgical precision with each target carefully selected instead of wildly spreading, which may be one of the reasons it has evaded discovery since about 2007 when Flame's main file was first seen. It spread itself locally as a valid, but illegitimate, Microsoft Windows security patch by impersonating Windows Update. Flame seems to be the first to use a chosen-prefix collision attack maliciously in the wild. Lastly, as we've discovered, it employed a yet unknown variant chosen-prefix collision attack.

3.2 Applying Counter-Cryptanalysis

On the 3rd of June 2012, Microsoft blogged that in their initial analysis of Flame they "*identified that an older cryptography algorithm could be exploited and then be used to sign code as if it originated from Microsoft*" [MS12b]. An immediate guess for this cryptically worded attack was a chosen-prefix collision attack on MD5 due to our construction of a rogue Certification Authority [SSA+09]. However, only the certificates in the chain leading to the forged signature on Flame's executable were circulating on the Internet. In particular the sibling innocuous-looking certificate actually used to obtain the forged signature on the malicious certificate was not available to directly verify a collision attack.

We were asked by enthousiasts if we could indeed verify whether the malicious certificate named 'MS' was constructed using a collision attack. We ran a proof-of-concept implementation of our technique from Sect. 2 dating from 2008 on a privately-obtained copy of the 'MS' certificate. In 0.03 seconds, it detected 4 sequential near-collision blocks and reconstructed the underlying differential paths that are given in the full version of this paper (the first differential path is also given in Sect. B). These differential paths indeed indicate a chosen-prefix collision attack that starts with a δIHV containing many bit differences that is gradually reduced to zero by the four near-collision blocks. However, very surprisingly, we discovered that these differential paths are not of the same family we used and also show characteristics that do not match those from known differential path construction methods for MD5. In the following sections we first describe the observed characteristics and then analyze and compare them.

On a historic note, the validity period of the 'MS' certificate started February 19 of 2010. Although the date can be faked, it can be argued that it does not make sense to craft this special code-signing certificate that only becomes valid in the (far) future. This puts Flame's attack years after the first identical-prefix and chosen-prefix collision attack. Also, Project HashClash [HC] released a chosen-prefix collision toolkit in 2009, which was generally expected to have been used before our discovery. Hence, it is our guess that the development of Flame's attack started before this release, but after the publication of the first chosen-prefix collision attack.

3.3 Observed Characteristics of Flame's Differential Paths

1. **Wang et al.'s message block differences.** All four differential paths are based on the same message block differences that were used by Wang et al. for the first MD5 collision [WY05]: $\delta m_4 = \delta m_{14} = 2^{31}$ and $\delta m_{11} = \pm 2^{15}$. The first and third path use $\delta m_{11} = +2^{15}$ and the second and third path use the negated form $\delta m_{11} = -2^{15}$.
2. δIHV **corrections.** The four differential paths are used in a step-wise manner to eliminate the differences in $\delta IHV = (\delta a, \delta b, \delta c, \delta d)$ resulting from the birthday search in their chosen-prefix collision. The corrections each path made are:

Block	$\delta a = \delta Q_{61}$	$\delta b = \delta Q_{64}$	$\delta c = \delta Q_{63}$	$\delta d = \delta Q_{62}$
1	[31]	[31,25,-18,-15,-12,9,1]	[31,25,-14,-12,9]	[31,25]
2	[31,5]	[-26,24,21,-14,-9,5,0]	[31,26,24,20,-9,5]	[31,-25,-9,5]
3	[31]	[30,26,-24,20,-17,15,9,-3]	[31,26,-24,-14,9]	[31,25,9]
4	[31]	[-25,14,-9,-5,3,0]	[31,-25,14,-9]	[31,-25,-9]
1+2	[5]	[31,-24,21,-18,-16,14,-12,5,2,-0]	[27,-24,20,-14,-12,5]	[-9,5]
3+4	[]	[30,24,20,-16,-14,-5,0]	[24]	[]
1+4	[]	[31,-18,-14,-12,-4,-2,-0]	[-12]	[-9]
2+3	[5]	[30,22,-20,-17,14,5,-3,0]	[27,20,-14,5]	[5]
all	[5]	[-30,21,19,17,-12,2]	[27,20,-14,-12,5]	[-9,5]

Note: we use the compact notation $[b_1, \ldots, b_n]$ for $\sum_{i=0}^{n} 2^{|b_i|}\text{sign}(b_i)$.

In comparison, the first collision attack by Wang et al. was based upon the δIHV 'correction' ([31],[31,25],[31,25],[31,25]) used in two sequential near-collision attacks, where the second uses the negated 'correction' such that the two 'corrections' cancel out.

3. bit differences in all bits of ΔQ_6, identical for blocks 1&3 and 2&4

Block	$\mathfrak{q}_6[31] \ldots \mathfrak{q}_6[0]$			
1	++----+-	---+----	-----+++	++++++++
2	+-++++++	+++-----	------+-	--+-----
3	++----+-	---+----	-----+++	++++++++
4	+-++++++	+++-----	------+-	--+-----

4. highest density of bitconditions found on $Q_4, \ldots, Q_8$. The four differential paths have, respectively, only 8, 4, 6 and 5 bits of freedom left out of those 160 bits of $Q_4, \ldots, Q_8$.

5. fixed differences $\delta Q_6, \ldots, Q_{60}$. The differential paths from the first and third block (that use the same message block differences) use the same differences $\delta Q_6, \ldots, \delta Q_{60}$. Similarly, the differential paths from the second and fourth block (that also use the same message block differences) use the same differences $\delta Q_6, \ldots, \delta Q_{60}$.

6. advanced message modification not maximized. One of the key message modifications to speed up to collision search are tunnels [Kli06]. The best and most important tunnel allows a simple message modification that does not affect all bitconditions on $Q_1, \ldots, Q_{24}$. For Flame's differential paths, this tunnel can maxime the time spent on steps 24 and onwards. This tunnel is based on flipping a bit $Q_9[b]$ with no bit condition and requires that $Q_{10}[b] = Q'_{10}[b] = 0$ and $Q_{11}[b] = Q'_{11}[b] = 1$. As shown in the table below, the near-collision blocks show a significantly lower tunnel strength than the maximal strength possible based on just the differential paths.[3]

Block	strength	max. strength	avg. strength
1	7	17	4.25
2	13	18	4.5
3	10	17	4.25
4	9	18	4.5

3.4 Differential Path Construction Analysis

So far there are two known methods for constructing a differential path for MD5. One is the our method [SLdW07] that uses a meet-in-the-middle approach. The second one is due to Mendel et al. [MRS09] that works similar to a probabilistic algorithm from coding-theory that searches for low weight code words.

The fact that all bit positions of ΔQ_6 have non-zero differences for all differential paths and that this does not help the collision search itself, indicates

[3] The 'avg. strength' is the average strength that would be observed if the extra conditions on Q_{10} and Q_{11} are each fulfilled randomly and the tunnel is *not* used.

that this choice was made with a specific purpose for the differential path construction. This choice seems to be a very bad one in combination with a method similar to Mendel et al.'s, as it is unnecessary and leads to significant increases in computational cost and number of differential path conditions. Hence, also given the uncharacteristicly high amount of differences and conditions in the first few steps, we argue that a meet-in-the-middle approach was used with a random starting path and a fixed ending path. However, from Observation 3 it is also clear that it does not use our method to construct a full differential path from the starting and ending paths: none of our differential paths have this characteristic. Evidently, it uses a yet unknown meet-in-the-middle method to construct full differential paths.

However, using the four differential paths, we can make an educated guess on how their method works. In any meet-in-the-middle approach, the lower and upper partial differential paths can be constructed independently except for four differential steps. It appears that Flame's uses a fixed differential path over steps $9, \ldots, 59$, then the meet-in-the-middle steps are $5, 6, 7, 8$. Our educated guess is that they first completed step 5 and then used an exhaustive search over steps 6, 7 and 8. With step 5 completed, the boolean function outcome modular differences δF_t for steps 6, 7 and 8 are completely determined. To complete steps 6, 7 and 8, such an exhaustive search only needs to find bit conditions that achieves these three modular differences simultaneously. The choice for non-zero differences in ΔQ_6 makes a lot of sense in this scenario, as it almost maximizes the number of choices for each $\Delta F_t[b]$ ($t = 6, 7, 8$, $b = 0, \ldots, 31$) and thus results in a higher success probability. Completing step 5 and the exhaustive search can either be done efficiently in a bit-wise approach, e.g., an adaptation of our method, or simply with a brute-force search. So far we were not able to distinguish between these two very different approaches from these differential paths.

Unfortunately, the choice to use non-zero differences in all bit positions of ΔQ_6 strictly reduces the solution space over steps $5, 6, 7, 8$ in comparison to our method and thus requires more lower/upper differential path pairs to succeed. Moreover, the choice to use a fixed upper differential path over steps $9, \ldots, 59$ implies that Flame's method requires many more lower differential paths to obtain the required amount of lower/upper path pairs. Overall, this would imply that Flame's method has higher complexity and results in differential paths with fewer degrees of freedom.

We were able to perform a somewhat simple quantitive comparison of Flame's differential paths with differential paths constructed using our publicly available HashClash toolkit [HC]. In an experiment we tried to find a replacement path for Flame's first differential path with as few bitconditions as possible. The resulting differential path is given in Tbl. C-1 and has only 266 bit conditions over $Q_1, \ldots, Q_{24}$ which are 62 fewer than the 328 bit conditions of Tbl. B-1. In another experiment we tried to construct a differential path with the HashClash toolkit in a very short amount of time, the result was an average runtime of only 15 seconds on an Intel i7-2600 CPU leading to differential paths with about 276 bit conditions, which is still 52 bit conditions fewer than Flame's path. This

experiment used only 20,000 lower and 20,000 upper partial paths leading to a total of 400,000,000 pairs. Future research might provide insights in the minimum complexity of constructing differential paths with the same characteristics of Sect. 3.3, however we have no results in this direction at this point of time.

3.5 Near-Collision Block Search

Though Observation 6 indicates the best tunnel strength is not maximized, it is also clear that this tunnel (or a slightly weaker version) is actually used as the observed tunnel strength is significantly higher than what would be observed if this tunnel was not used (cf. 'avg. strength' at Obs. 6). A reasonable guess is that they used tunnels in a dynamic manner depending on whether the necessary conditions on Q_{10} and Q_{11} were fulfilled.

Given the low number of bitconditions on $Q_{18}, \ldots, Q_{24}$ and sufficiently high tunnel strengths, we can reasonably say that the near-collision block search complexity is dominated by the cost of steps 24 up to 63. We have experimentally determined the success probability over steps 24 up to 63 for each of the near-collision blocks and these are given in Tbl. 3-1 together with lower-bounds for the average complexity in MD5 compression function calls. Note that because the inner-most loop computes at least 9 steps of the compression function, this search is well suited for massively parallel architectures in contrast to our chosen-prefix collision attack.

Table 3-1. Near-collision blocks: complexity lower-bounds

Block	estimated probability of steps 24-63	average complexity lower-bound
1	$2^{-38.8}$	$2^{36.0}$
2	$2^{-46.8}$	$2^{44.0}$
3	$2^{-33.6}$	$2^{30.8}$
4	$2^{-33.3}$	$2^{30.5}$
[WY05]	$2^{-20.5}$	$2^{17.7}$

3.6 Birthday and Reduction Procedures

The δIHV resulting from the birthday procedure can be observed as the differences for $t = -3, -2, -1, 0$ of the first differential path Tbl. B-1:

$$\delta IHV = (-2^5,\ -2^2+2^{12}-2^{17}-2^{19}-2^{21}+2^{30},\ -2^5+2^{12}+2^{14}-2^{20}-2^{27},\ -2^5+2^9).$$

Based on the available space in the certificate, our initial guess is that Flame uses 64 birthday bits over the first and last word of the IHV (matching $t = -3$ and $t = -2$ of the first path). However, this does not immediately imply that Flame's birtday search has complexity $\sqrt{\pi} \cdot 2^{32}$ MD5 compressions, as not every birthday collision is usable. In fact, the two random-looking differences have very low weights of 6 and 5 bit differences, where an uniform distribution that might be expected from an arbitrary birthday collision would actually lead to an average of about 11 bit differences each. Just aiming at such a low weight distribution

would result in a birthday complexity of about 2^{42} MD5 compressions. However, lacking a systematic family of differential path like that of [SSA+09], it is almost certain that the positions of the bit differences are also important, which further increases the birthday complexity.

Further research may provide more insights in which δIHV corrections are possible within the observed near-collision block complexities and the effect thereof on the birthday search and its complexity.

3.7 Preliminary Conclusions

Firstly, Flame's method to construct differential paths seems to be sub-optimal compared to those obtained with our public HashClash toolkit [HC].

Secondly, so far we have been able to provide a weak lower-bound for the birthday search and good lower-bounds for the near-collision block search complexities. These lower-bounds together indicate that Flame's new variant chosen-prefix collision attack likely costs more than $2^{44.3}$ MD5 compressions. How much more remains an open question as the birthday search complexity is inaccurate and it does not yet include the cost of the differential path construction. Also note that we have only one instance of a chosen-prefix collision from Flame's new variant attack, making it uncertain how close the observed near-collision block search complexities are to what can be expected *on average* with Flame's attack.

In comparison, the average complexity of our 2009 chosen-prefix collision attack for four near-collision blocks appears to be dominated by the birthday search complexity of $2^{44.55}$ MD5 compression function calls (cf. [SSA+09, Table 2] using $r = 4$ and $w = 5$). Comparing the weak lower-bound with this cost, the theoretical complexity of Flame's attack is not significantly lower than that of our attack. Nevertheless, Flame's attack might be more cost effective due to the suitabilitiy of the collision search for massively parallel architectures.

4 Conclusion

We have introduced counter-cryptanalysis as a new paradigm for strengthening weak cryptographic primitives. Also, we have presented the first example thereof, namely the efficient detection whether a given message was constructed using a cryptanalytic collision attack on MD5 and/or SHA-1. A reference implementation will be made available on project HashClash [HC].

Using our novel technique, we have analyzed Flame's malicious certificate and exposed its chosen-prefix collision attack. Our proof-of-concept collision detection implementation also reconstructed the four underlying differential paths. These differential paths reveal that Flame used a yet unknown variant chosen-prefix collision attack on MD5. We have analyzed these differential paths, working towards a reconstruction of the underlying algorithm, and found a preliminary *weak lower-bound* of $2^{44.3}$ MD5 compressions for Flame's new variant chosen-prefix collision attack.

Acknowledgements. I'm indepted to Arjen Lenstra for our insightful discussions which have led to the idea of counter-cryptanalysis.

References

[Cry12] CrySyS Lab, sKyWIper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks, Laboratory of Cryptography and System Security, Budapest University of Technology and Economics (May 31, 2012)

[dBB93] den Boer, B., Bosselaers, A.: Collisions for the Compressin Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)

[HC] HashClash project webpage, `http://code.google.com/p/hashclash`

[Kas12] Kaspersky Lab, The Flame: Questions and Answers, Securelist blog (May 28, 2012)

[Kli06] Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105 (2006)

[Man11] Manuel, S.: Classification and generation of disturbance vectors for collision attacks against SHA-1. Des. Codes Cryptography 59(1-3), 247–263 (2011)

[MRS09] Mendel, F., Rechberger, C., Schläffer, M.: MD5 Is Weaker Than Weak: Attacks on Concatenated Combiners. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 144–161. Springer, Heidelberg (2009)

[MS12a] Microsoft, Flame malware collision attack explained, Security Research & Defense, Microsoft TechNet Blog (June 6, 2012)

[MS12b] Microsoft, Microsoft certification authority signing certificates added to the Untrusted Certificate Store, Security Research & Defense, Microsoft TechNet Blog (June 3, 2012)

[SLdW07] 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. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)

[SSA+09] Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., 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. LNCS, vol. 5677, pp. 55–69. Springer, Heidelberg (2009)

[Ste13] Stevens, M.: New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 245–261. Springer, Heidelberg (2013)

[VJBT08] Vábek, J., Joščák, D., Boháček, M., Tůma, J.: A New Type of 2-Block Collisions in MD5. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 78–90. Springer, Heidelberg (2008)

[WY05] Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

[XF09] Xie, T., Feng, D.: How To Find Weak Input Differences For MD5 Collision Attacks. Cryptology ePrint Archive, Report 2009/223 (2009)

[XF10] Xie, T., Feng, D.: Construct MD5 Collisions Using Just A Single Block of Message. Cryptology ePrint Archive, Report 2010/643 (2010)

[XFL08] Xie, T., Feng, D., Liu, F.: A New Collision Differential for MD5 With Its Full Differential Path. Cryptology ePrint Archive, Report 2008/230 (2008)

[XLF08] Xie, T., Liu, F., Feng, D.: Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5? Cryptology ePrint Archive, Report 2008/391 (2008)

A List of Possible Feasible MD5 Near-Collision Attacks

Used non-zero message block differences in published near-collision attacks are:

- $\delta B = \pm(\delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$ [WY05]: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_2 = 2^8, \delta m_{11} = 2^{15}, \delta m_4 = \delta m_{14} = 2^{31})$ [SSA+09]: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_{11} = 2^b)$ for $b \in \{0, \ldots, 30\}$ [SLdW07]: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_{11} = 2^{31})$ [SLdW07]: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_5 = 2^{10}, \delta m_{10} = 2^{31})$ [XF10]: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_8 = 2^{31})$ [XLF08]: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_6 = 2^8, \delta m_9 = \delta m_{15} = 2^{31})$ [XFL08]: $i = 37$, $\delta WS_{37} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_9 = 2^{27}, \delta m_2 = \delta m_{12} = 2^{31})$ [VJBT08]: $i = 37$, $\delta WS_{37} \in \{\underline{0}, \underline{2^{31}}\}$.

Other non-zero message block differences taken from [XF09] and [XLF08] are:

- $\delta B = \pm(\delta m_4 = 2^{20}, \delta m_7 = \delta m_{13} = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_2 = 2^8)$: $i = 37$, $\delta WS_{37} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_5 = 2^{10}, \delta m_{11} = 2^{21})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_5 = 2^{10}, \delta m_{11} = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_5 = 2^{31}, \delta m_8 = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_2 = 2^8, \delta m_{14} = 2^{31})$: $i = 37$, $\delta WS_{37} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_4 = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_5 = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_{14} = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_4 = 2^{25})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_5 = 2^{10})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_8 = 2^{25})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_{11} = 2^{21})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_{14} = 2^{16})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_4 = 2^{20})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_6 = 2^8)$: $i = 50$, $\delta WS_{50} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_9 = 2^{27})$: $i = 50$, $\delta WS_{50} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_5 = 2^{10}, \delta m_9 = 2^{27})$: $i = 37$, $\delta WS_{37} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = (\delta m_5 = 2^{31}, \delta m_{11} = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_8 = 2^{31}, \delta m_{11} = 2^{21})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$;
- $\delta B = \pm(\delta m_8 = 2^{25}, \delta m_{13} = 2^{31})$: $i = 44$, $\delta WS_{44} \in \{\underline{0}, \underline{2^{31}}\}$.

B Flame's Differential Paths

Table B-1. Differential path of near-collision block 1

t	Bitconditions: $\mathtt{q}_t[31]\ldots\mathtt{q}_t[0]$
-3	....-.....
-2	00...... .1.1.01. ...1..+. ..-.10..
-1	110-+..1 .1.-.00. .+.+.... ..-110..
0	+-100..0 .-0+^++1 .0.+0.11 .110-+..
1	0+-++..- .-0++-+0 011-0..1 110+++..
2	+0-0-.00 .-++00+- 0-1-+.1+ 1+-0++^.
3	+010-000 .-+++0+1 +--.+^1+ -+-+++-.
4	-00-10+. .11-+-0+ +++11--0 -101-+0.
5	0-+-++-^ ^0110+1- -110+0-0 -0001+1^
6	++----+- ---+---- -----+++ ++++++++
7	111.-111 1101011. 110-1001 +0100.00
8	00+0.111 10111101 -1101100 .1110011
9	..0.1... .....-.. 0.10+... 0-....0.
10	..0^...1 ^....0.. 0^0-1... .1....+.
11	..0-...1 +....-.. .+-01... .0..^.1.
12	.1-1..^+ 1....+.. .0+0.... ....+.1.
13	.0+1..-+ 1....0.. 100....1 ....0...
14	.-+...1. .....1.. 1.+....1 ....1...
15	.0+...10 .... -.0....- ....-...
16	.1+..... .0...... ..^..... ....
17	..1..... .1....0. ^......^ ....^...
18	..0..... .+....1. ....
19	....-. ....
20	0....... .^...... ....
21	0....... ......^. ....
22	-....... ....
23	....
24	^....... ....
25–32	....
33	0....... ....
34	1....... ....
35–59	X....... ....
60	X.11110. ....
61	X.11000. .... .001.00. ....
62	X.+----. ....0.... ....
63	X.?0??+. ....--+.+-. ....
64	X......+ ++++++.. -..-.+-. .....+-.

$$\delta m_4 = \delta m_{14} = 2^{31}, \quad \delta m_{11} = 2^{15}$$

C Replacement Differential Path 1

Table C-1. Replacement differential path for near-collision block 1

t	Bitconditions: $\mathfrak{q}_t[31] \dots \mathfrak{q}_t[0]$			
-3	....	....	....	..-.....
-2	00......	.1.1.01.	...1..+.	..-.10..
-1	110-+..1	.1.-.00.	.+.+....	..-110..
0	+-100..0	.-0+^++1	.0.+0011	.110-+..
1	1+00-..-	.-.+.++.	.1.-11..	.0+10-..
2	1--.-..1	...-.+0.	...1--.^	.-0-1-..
3	.10.0.11	.1.+1+10	1.1+101+	.+0-+.^.
4	.00^+^0.	0..+00+1	1^0+-000	0-1.-1+^
5	^++++-+^	0.^+--0+	----1.+1	10-01.1+
6	-001+1-+	+.+.0-++	+1-++.0-	++0.0.+0
7	100--001	+.001+0.	-1+11.01	010...11
8	1.+00.10	-.0..010	-.0+-..0	1-....1.
9	..0-0...	0....1..	..11-...	.0....1.
10	..-1...1	+....-..	0..+....	.1....+.
11	..++..00	+....-..	...00...	......1.
12	..+1..1+	.....+..	..01....	......1.
13	..-1..-+	0....1..	1.1....1	....1...
14	..-...1.	.....1..	1.+....1	....1...
15	..+...10	....	-.1....-	....-...
16	..+.....	.0......	..1.....	....
17	..1.....	.1....0.	^......^	....^...
18	..0.....	.+....1.	....	....
19	....	......-.	....	....
20	0.......	.^......	....	....
21	0.......	......^.	....	....
22	-.......	....	....	....
23	....	....	....	....
24	^.......	....	....	....

$$\delta m_4 = \delta m_{14} = 2^{31}, \quad \delta m_{11} = -2^{15}$$