Is SHA-1 conceptually sound?

Charanjit S. Jutla
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
csjutla@watson.ibm.com

Anindya C. Patthak*
University of Texas at Austin
Austin, TX 78712
anindya@cs.utexas.edu

Abstract

We argue that if the message expansion code of SHA-1 is replaced by a linear code with a better minimum distance, then the resulting hash function is collision resistant. To support this argument, we characterize the disturbance vectors which are used to build local collision attacks as a linear code. This linear code is the xor-sum of two codes, the message expansion code and a linear code representing the underlying block cipher in SHA-1.

We also show that the following constraint satisfaction problem is NP-hard. The constraints are restricted to being XOR constraints, or Majority constraints on at most three variables each. The instances are further restricted by requiring that the constraints can be listed in a sequence $C_1, C_2, \cdots, C_m$, such that for every constraint C_i, two of the variables in it occur only in constraints C_j, with $|j - i| < 48$. This problem is similar to the problem modeling the one-way function property of SHA-1.

1 Introduction

In this paper we address the security issues of the cryptographic hash function SHA-1 [Uni95] and its variants. Although, SHA-1 is one of the most popular substitute for Random Oracles, we address only the one-way-ness and collision resistance properties of SHA-1 (see [Gol01]).

Since SHA-1 is used as a primitive in the Merkle-Damgård [Dr89, Mer89] construction for hashing longer messages, and the security of the latter is well studied, we focus here on SHA-1 as the compression function. However, as a compression function primitive, SHA-1 takes a 512 bit input and outputs a 160 bit hash value, and hence is a constant length primitive, which makes it impossible to rigorously define its security.

*This work was done while the author was visiting IBM T.J. Watson Research Center, N.Y.

However, one can try to model SHA-1 as an asymptotic problem, and then address its cryptographic one-way function property. In this paper we show that the one-way property of SHA-1 can be modeled as a Constraint Satisfaction Problem (CSP), which in its general form is known to be NP-hard. However, the CSP which SHA-1 specifies is firstly a constant size CSP, and second has a specific structure. In particular, all the clauses are either XOR clauses or Majority function clauses. Further, for each clause three out of four variables occur only in neighboring clauses (in a ordered listing of the clauses). Although, it is impossible to generalize the SHA-1 structure to an asymptotic setting, we show in this paper that the CSP satisfaction problem restricted to XOR and MAJ clauses, and with the further restriction that each clause have two out of three of its variables in only neighboring clauses is NP-hard. This gives evidence that SHA-1 is a practically secure one-way function.

We next address the collision-resistance property of SHA-1 from a practical point of view. We argue that if a Wang, Chaboud, Joux (WCJ-) local collision [Wan97a, Wan97b, CJ98] based randomized strategy is not employed for finding collisions, then the problem is as hard as the one-way function property of SHA-1. For the WCJ-local collision strategy, we exactly characterize the disturbance vectors which can be used to build the differential path. This characterization was an open problem, as it allows one to find small disturbance vectors, or rule out small disturbance vectors. Small disturbance vectors are the key to the WCJ-local collision based collision attacks.

We will be dealing with two kinds of linear codes. One is the message expansion linear code, whose minimum distance for SHA-1 and SHA1-**IME** [JP05] is known. This code has dimension 16×32. The second code (*cipher code*) specifies the block cipher structure of the SHA-1 state update function. This code has dimension 5×32. In this paper we show that if a non-zero difference vector (which is a message expansion codeword) is obtained by putting together WCJ-local collisions then the disturbance vector is the xor-sum of two codewords, one a codeword of the message expansion code, and other a codeword of the cipher code. Moreover, the message expansion codeword is non-zero. Since the cipher code is not similar to the message expansion code, the best strategy to build a disturbance vector is to pick a small weight message expansion codeword, and a zero cipher-codeword. This along with analysis of the various probabilities involved in assuring local-collisions, leads us to conclude that differential attacks based on local collisions cannot succeed with probability better than $2^{-52\times 2}$ for SHA1-**IME**.

2 One-way Property of SHA-1

We briefly recall the message expansion code and the state update transforms of SHA-1 [Uni95]. Let $M_0, \cdots, M_{15}$ be the input message blocks. Then

<u>SHA-1</u> :
for $i \;=\; 0, 1, \cdots, 15, \quad W_i = M_i$ and

for $i = 16$ to 79

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) <<< 1 \quad (1)$$

where $<<< 1$ denotes a one bit rotation to left. The state update functions are given as follows: for $i = 0$ to 79:

$$\begin{aligned} A_{i+1} &= W_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + K_i, \\ B_{i+1} &= A_i, \\ C_{i+1} &= B_i <<< 30, \\ D_{i+1} &= C_i, \\ E_{i+1} &= D_i, \end{aligned}$$

Round	Step(i)	$f_i(X,Y,Z)$
1	0-19	$XY \vee \overline{X}Z$
2	20-39	$X \oplus Y \oplus Z$
3	40-59	$XY \oplus XZ \oplus YZ$
4	60-79	$X \oplus Y \oplus Z$

where '$+$' denotes the binary addition modulo 2^{32}. The output of the compression function is the string of five words $A_{80}, B_{80}, C_{80}, D_{80}$ and E_{80}.

Before we go into the collision resistance properties of SHA-1 and SHA1-**IME**[JP05], lets explore their properties as one-way functions. Specifically, it is desired that no program should be able to solve for m with high probability, given a random output $\langle \alpha, \beta\, \gamma, \mu, \nu \rangle$, i.e.

$$A_{80}(m) \;=\; \alpha \wedge B_{80}(m) \;=\; \beta \wedge C_{80}(m) \;=\; \gamma \wedge D_{80}(m) \;=\; \mu \wedge E_{80}(m) = \nu$$

It is widely believed that the above property of one-wayness holds for SHA-1. We must note that actually proving the above one-way claim, even in an asymptotic sense is an extremely difficult problem. One approach could be to show that (in an asymptotic version of the above problem) the problem can be framed as a Polynomial Constraint Satisfaction Problem over $\mathbb{F}_2$ (where each polynomial has degree at most two), which is known to be NP-hard ([GJ79]). However, the notion of cryptographic one-wayness requires showing the problem to be average-case hard for NP. Unfortunately, all advances in this direction have been stymied by a theorem of Impagliazzo ([Imp95]) that any such result must be non-relativizing. Further, it has been shown ([FF76, BT05]) that under non-adaptive reductions this reduction is not possible unless the polynomial hierarchy collapses to the third level.

The other problem is that even though general Polynomial Constraint Satisfaction Problem is known to be NP-hard, the instances involved in SHA-1 may have too much structure. To address this concern, we show that a restricted Constraint Satisfcation Problem, which is restricted similarly to SHA-1 is NP-hard.

Let us take a look at the actual equations or constraints involved in SHA-1. Recall, a program P must solve for m given $\alpha, \beta, \gamma, \mu, \nu$.

We denote the j^{th} bit of word X_i by $x_{i,j}$. Also, when we consider any addition of two variables we will introduce binary auxiliary variables to represent carries, and we will use Greek letters to denote them. Also, we break up any addition involving more than two words into several two word additions, thus ensuring that carry variables are binary.

$\forall i \in [1, 80]\ \forall j \in [0, 31]$

$$
\begin{aligned}
f_{i,j} &= f_i(a_{i-2,j}, a_{i-3,j+2}, a_{i-4,j+2}) \\
g_{i,j} &= (k_{i,j} + w_{i-1,j} + \sigma_{i,j-1}) \bmod 2 \\
\sigma_{i,j} &= (k_{i,j} + w_{i-1,j} + \sigma_{i,j-1}) \operatorname{div} 2 \\
h_{i,j} &= (g_{i,j} + f_{i-1,j} + \tau_{i,j-1}) \bmod 2 \\
\tau_{i,j} &= (g_{i,j} + f_{i-1,j} + \tau_{i,j-1}) \operatorname{div} 2 \\
m_{i,j} &= (h_{i,j} + a_{i-5,j+2} + \phi_{i,j-1}) \bmod 2 \\
\phi_{i,j} &= (h_{i,j} + a_{i-5,j+2} + \phi_{i,j-1}) \operatorname{div} 2 \\
a_{i,j} &= (m_{i,j} + a_{i-1,j-5} + \psi_{i,j-1}) \bmod 2 \\
\psi_{i,j} &= (m_{i,j} + a_{i-1,j-5} + \psi_{i,j-1}) \operatorname{div} 2
\end{aligned} \tag{2}
$$

with $\sigma_{i,-1} = \tau_{i,-1} = \phi_{i,-1} = \psi_{i,-1} = 0$ for all i.

$$
\begin{aligned}
\forall j \in [0, 31]\ a_{0,j} &= IV_{0,j} \\
\forall j \in [0, 31]\ b_{0,j} &= a_{-1,j} = IV_{1,j} \\
\forall j \in [0, 31]\ c_{0,j} &= a_{-2,j} = IV_{2,j} \\
\forall j \in [0, 31]\ d_{0,j} &= a_{-3,j} = IV_{3,j} \\
\forall j \in [0, 31]\ e_{0,j} &= a_{-4,j} = IV_{4,j}
\end{aligned} \tag{3}
$$

where IV_i ($i \in [0..4]$) is the initial vector specified in SHA-1.

We remark that three binary variables added and divided by two is just the majority function over GF2, and three binary variables added and reduced modulo two is just addition in GF2. Thus, all of the above equations, including the message expansion, and the computation of f, can be written as polynomial equations over GF2, with degree at most two. Although, this variant of the polynomial constraint satisfaction problem is known to be NP-hard ([GJ79]), the above equations have a much more restricted structure in the sense that three out of the four variables in each constraint occur only in neighboring constraints.

We now show that the following similarly restricted constraint satisfaction problem is NP-hard.

A *constraint* is a function $\{0,1\}^k \to \{0,1\}$. A constraint is said to be satisfied if it evaluates to 1. A constraint is called an XOR constraint, if it represents the XOR of the k variables. Similarly, a constraint is called a MAJ constraint if it is the Majority function of the k variables. Throughout this section we will restrict to $k \leq 3$.

Given a set of boolean variables $U = \{u_1, u_2, ..., u_m\}$, a clause over U is a set of literals over U, e.g., $\{u_1, \bar{u_4}, u_6\}$. Clauses will be restricted to have size at most three. In our instance of the satisfaction problem, we will only have XOR-clauses and MAJ-clauses. An XOR-clause is satisfied if the XOR-constraint applied to the clause is satisfied (similarly for a MAJ-clause).

Theorem 2.1 *Given an instance consisting of a set of MAJ and XOR-clauses C over a set of boolean variables U the problem of determining if the set of clauses is simultaneously satisfiable is* NP*-hard. Further, the following restriction of the problem remains* NP*-hard: there is an ordering of the n clauses $C_1, C_2, \cdots, C_n$, such that for each clause C_i, at most one variable u in C_i* does not *satisfy the requirement that u occur only in clauses C_j, $|j-i| \leq 48$.*

Proof: See appendix. ■

For example, if there are three literals $u_1, \bar{u_4}, u_6$ in a clause then two of the variables corresponding to these clauses must occur in clauses within ± 48 of this clause. The above theorem says that instances with such highly structured set of clauses remains NP-hard.

Can a similar heuristic argument be given for collision resistance? We address this question in the next section. We end this section by an attempt to solve the system of equations using Schoning's Algorithm for 3-SAT and CSPs [Sch99]. Recall that in Schoning's algorithm, a random initial assignment of the n variables is chosen, and if some clause is falsified, a random literal from the clause is picked and flipped. This process is continued for up to $3n$ steps, before a new random assignment is picked. The complexity of this randomized algorithm is within a polynomial factor of $(2(1-\frac{1}{k}))^n$, where k is the number of literals per clause.

In our case, we have written our system of equations or constraints as 4-constraints. However, the number of variables has shot up considerably from 160. Even, if we consider auxiliary variables other than W to be not a factor, we still have a total of 80×32 variables, out of which we may fix at most $512 - 160$ variables corresponding to M, thus leaving us with at least $n = 64 \times 32$ variables. If we get rid of the W variables as well, and directly write the clauses in terms of M, we find that each clause has on average 80 binary variables corresponding to M, rendering k to be large. Note that the same argument holds for SHA1-**IME**.

3 Collision Resistance Properties of SHA1-IME

The hash function SHA1-**IME** [JP05], is exactly the same as SHA-1 except that the message expansion code as in equation (1) is replaced by

$$W^i = \begin{cases} W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16} \oplus \left(W^{i-1} \oplus W^{i-2} \oplus W^{i-15}\right) <<< 13 & \text{if } 16 \leq i < 36 \\ W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16} \oplus \left(W^{i-1} \oplus W^{i-2} \oplus W^{i-15} \oplus W^{i-20}\right) <<< 13 & \text{if } 36 \leq i \leq 79 \end{cases} \tag{4}$$

It was shown that this code has minimum weight 80 in just the last 64 words. Further, the minimum weight restricted to the last 60 (48 steps) is at least 75 (52 respectively).

The attacker's task is to find two messages M and M' which hash to the same value, say $\alpha, \beta, \gamma, \mu$ and ν. If we use primed variables to write another set of equations as in the previous section for the second message M', then we can get rid of α, β etc. by equations of the form $a_{80,j} = a'_{80,j}$, $a_{79,j} = a'_{79,j}$ etc.

We will use the prefix Δ to denote the "xor" difference of a variable with its primed variable; thus $\Delta a_{65,12}$ denotes $a_{65,12} \oplus a'_{65,12}$. Thus, the equations in the previous paragraph are really

$$\Delta a_{80,j} = 0,\ \Delta a_{79,j} = 0, ...,\ \Delta a_{76,j} = 0 \tag{5}$$

Is it possible to write all the equations in terms of difference variables? For this to be true, exclusive-or has to distribute over the majority function which, although not always true, does happen with non-trivial probability. This leads to a trivial solution, i.e. $\Delta M = 0$, which is not very useful. However, many equations can be made to be trivially true (and with probability one) if the difference variables involved are zero. One then tries to focus on a non-zero ΔM which requires the least number of equations to have to go through the probabilistic distribution of "xor" over majority.

This is the idea behind local collisions [Wan97a, Wan97b, CJ98], [WYY05], as with local collisions one takes a non-zero ΔM and tries to set most of the difference variables to zero as quickly as possible. Notice that a non-zero ΔM leads to a non-zero ΔW, which keeps disturbing the equality of intermediate step variables. This disturbance is then offset as quickly as possible by additional differences coming from ΔW.

Other than this local collision strategy, or a method which makes many equations true with high probability, it is an open problem to find collisions in a way better than birthday attack. The only alternative seems to be to use general purpose randomized algorithms for satisfiability of CSPs like Schoning's algorithm mentioned in the previous section.

So, at the present state of knowledge about solving general CSPs, and without any further insights into any special structure these CSPs may have, an attacker is left with the option of trying to optimize the local collision based attacks.

In the next section we show that, if the linear message expansion code is good then either the disturbance vector itself has a large hamming weight, or finding a small hamming weight disturbance vector (if any such exists) is an instance of a NP-hard problem (which does not seem to have any special structure, e.g. sparsity, to reduce its complexity).

4 Are there better Disturbance vectors?

Recall that in [Wan97a, Wan97b, CJ98] a *disturbance vector* indicates where new disturbances start, and these individual disturbances are canceled in the next six rounds by additional differences in the expanded message; each of these events is called a *local collision*. The linear combination (xor) of all these disturbances and additional differences is called the *difference vector*. There can be many kinds of local collision strategies, and we explore all such possibilities in the next few sub-sections.

The one requirement on the difference vector is that it must be a codeword of the SHA-1 message expansion linear code.

To review recent attacks, in [Wan97a, Wan97b, CJ98] the collision attack on SHA-0 is carried out by explicitly constructing a difference vector out of a disturbance vector. Moreover, in there a disturbance vector is itself chosen to be a codeword of SHA-0. Later [BC04] and [RO05] extend their technique to cause collisions in reduced SHA-1. In [WYY05], this idea is further extended to attack the full SHA-1. However, the difference vector is still constructed out of a disturbance vector which is a codeword. One advance has been in not requiring local collisions at every disturbance – in particular, the first 16 to 20 rounds in [WYY05] are handled in a more complicated juxtaposition of local collisions, and the last 5 round disturbances are allowed to run loose. The inner round (i.e. from round 20 to round 75) disturbances however are still handled by local collisions.

Thus, since these attacks crucially depends on the weight of the disturbance vector (particularly in the inner rounds), the issue of obtaining a small weight disturbance vector is a crucial one.

4.1 Local Collision Based Strategies

In the following, we wlll be using a linear code which models the local collisions. We will call it the *cipher code* as it indirectly models the underlying block cipher structure of the SHA-1 state update function. We will define it more formally later.

In this sub-section, we prove that if the local collision is constructed as in [Wan97a, Wan97b, CJ98], then the disturbance vector must be the xor-sum of a non-zero message expansion codeword in SHA-1 (the same is also true for SHA1-**IME**, and any other linear message expansion code) and a codeword in the cipher code. It was an open problem whether one could consider a disturbance vector which is not a message expansion codeword and yet the difference vector is a codeword. We address the implications of this characterization at the end of this subsection

To be specific, we show that if the global collision arises as a result of local collisions from step t onwards (that is we allow one to manipulate the disturbances and additional differences in any arbitrary way till the first t steps) then the disturbance vector restricted to steps $t+5$ onwards must be the xor-sum of a non-zero SHA-1 message expansion codeword and a cipher codeword.

Consider the front-truncated SHA-1 code, i.e. restricted to the last 65 words. We assume that the parity check constraints of SHA-1 is denoted by R, i.e., for any 65×32-bit vector w, $R(w) = 0$ if and only if

$$\forall j \in \{1, \cdots, 32\}, \forall i \in 16, \cdots, 64 \;\; w_{i,j} \;=\; w_{i-3,j-1} \oplus w_{i-8,j-1} \oplus w_{i-14,j-1} \oplus w_{i-16,j-1}.$$

We also define similar set of constraints $\tilde{R}$ on any 60×32-bit vector $\tilde{w}$ such that

$$\forall j \in \{1, \cdots, 32\}, \forall i \in 16, \cdots, 60 \;\; \tilde{w}_{i,j} \;=\; \tilde{w}_{i-3,j-1} \oplus \tilde{w}_{i-8,j-1} \oplus \tilde{w}_{i-14,j-1} \oplus \tilde{w}_{i-16,j-1}.$$

We denote a vector w of dimension 65×32 by $w[0 \cdots 64]$. Then R can be thought as constraints on $w[0 \cdots 64]$ and $\tilde{R}$ on $w[5 \cdots 64]$. We will denote $w[5 \cdots 64]$ by $\tilde{w}$ i.e., $\tilde{w}[0] = w[5]$. Assume that the local collisions are desired from rounds 5 to 64. Also, we do not force that the last 5 words in the disturbance vectors be zero, thus allowing the possibility of near-collisions.

Define a map $\imath : \{0,1\}^{2^{65\times 32}} \to \{0,1\}^{2^{65\times 32}}$. Let $z \stackrel{def}{=} \imath(u)$. Then

$$\imath(u)_{i,j} \;=\; z_{i,j} \stackrel{def}{=} \begin{cases} u_{i,j} & \text{if } 0 \le i \le 4 \\ u_{i,j} \oplus u_{i-1,j-5} \oplus u_{i-2,j} \oplus u_{i-3,j+2} \oplus u_{i-4,j+2} \oplus u_{i-5,j+2} & \text{if } 5 \le i \le 64 \end{cases} \tag{6}$$

We also define a map $\tilde{\imath}$ which is similar to $\imath$. Specifically, $\tilde{\imath} : \{0,1\}^{2^{65\times 32}} \to \{0,1\}^{2^{60\times 32}}$ such that

$$\tilde{\imath}(u)_{i,j} \;=\; \tilde{z}_{i,j} \stackrel{def}{=} u_{i,j} \oplus u_{i-1,j-5} \oplus u_{i-2,j} \oplus u_{i-3,j+2} \oplus u_{i-4,j+2} \oplus u_{i-5,j+2} \quad 5 \le i \le 64 \tag{7}$$

Lemma 4.1 *Let $R(u) = 0$ for $u \in \{0,1\}^{65\times 32}$. Then $\tilde{R}(\tilde{\imath}u) = 0$.*

Proof: Let $U = \{u | R(u) = 0\}$ and $u \in U$. Then, rearranging and regrouping the terms and using the fact that $R(u) = 0$, all the parity check constraints of the code i.e., constraints of the form

$$u_{i+16,j} \oplus u_{i+13,j-1} \oplus u_{i+8,j-1} \oplus u_{i+2,j-1} \oplus u_{i,j-1} = 0$$

are satisfied for all $i \ge 5$. ■

Lemma 4.2 *$\imath$ is an injection.*

Proof: Let $z = \imath(u)$ and $z' = \imath(u')$. If u and u' differ in any j, for any $j \le 4$, then clearly the corresponding z and z' differs. Therefore assume $i^* \ge 5$ be the smallest i where u and u' differs, say in some j^* bit. From Equation 6, it is clear then that $z_{i^*,j^*} \oplus u_{i^*,j^*} = z'_{i^*,j^*} \oplus u'_{i^*,j^*}$ which implies $z \neq z'$, and hence $\imath$ is an injection. ■

Lemma 4.3 *$\tilde{\imath}$ is an injection on the set $U = \{u | R(u) = 0\}$.*

Proof: Let $u \neq u' \in U$. Note that by linearity $u \oplus u' \in U$. Let denote $\delta u = u' \oplus u$. Assume for contradiction that $\tilde{\imath}$ is not an injection. This implies that there are $z = \imath(u), z' = \imath(u')$ such that $\tilde{z} = \tilde{z'}$ i.e., $\tilde{\delta u} = 0$ as $\imath$ is linear. Moreover since $\imath$ is injective (Lemma 4.2), $z \oplus z' \neq 0$.

Denote the following set of constraints on $\tilde{z}$ and $\tilde{z'}$ by T

$$v_{i,j} \oplus v_{i-1,j-5} \oplus v_{i-2,j} \oplus v_{i-3,j+2} \oplus v_{i-4,j+2} \oplus v_{i-5,j+2} \text{ if } 5 \le i \le 64,$$

and $V = \{v | T(v) = 0\}$. Note that $\tilde{\imath}(\delta u) = 0$ implies that $\delta(u) \in V$. Therefore $\delta u \in U \cap V$. To complete the claim, it is therefore enough to show that the intersection is empty, which we do following a rank argument. ■

We now define a set of constraints E, arising from the the structure of the cipher, on any 65×32-bit vector w, $E(w) = 0$ if and only if

$$\forall j \in \{1, \cdots, 32\},\ \ u_{i,j} \oplus u_{i-1,j-5} \oplus u_{i-2,j} \oplus u_{i-3,j+2} \oplus u_{i-4,j+2} \oplus u_{i-5,j+2} = 0 \text{ if } 5 \leq i \leq 65,$$

where the addition in the second subscript is done modulo 32. We also define similar set of constraints $\tilde{E}$ on any 60×32-bit vector $\tilde{u}$ such that

$$\forall j \in \{1, \cdots, 32\},\ \ u_{i,j} \oplus u_{i-1,j-5} \oplus u_{i-2,j} \oplus u_{i-3,j+2} \oplus u_{i-4,j+2} \oplus u_{i-5,j+2} = 0 \text{ if } 5 \leq i \leq 65.$$

Theorem 4.4 *For a difference vector z (and since it is a codeword, $R(z) = 0$), then for every u such that $z = \imath(u)$, u restricted to the last 60 steps must be of the form $\tilde{s} \oplus \tilde{t}$, where $\tilde{R}(\tilde{s}) = 0$ and $\tilde{E}(\tilde{t}) = 0$ and s non-zero.*

Proof: First observe that $\tilde{E}(\tilde{t}) = 0$ is equivalent to saying $\tilde{\imath}(t) = 0$. Following this observation, it follows that if $z = \imath(u)$, then $z = \imath(u')$ for any $u' = u \oplus t$ such that $\tilde{E}(\tilde{t}) = 0$. Thus it is enough to prove that $\tilde{R}(\tilde{s}) = 0$.

Now by Lemma 4.1, if $R(z) = 0$, then it must hold that $\tilde{R}(\tilde{\imath}(u)) = 0$. This completes the proof. ■

Although this characterization of disturbance vectors shows that one can build disturbance vectors by taking the xor-sum of a non-zero message expansion codeword and a cipher codeword, using a non-zero cipher codeword is not expected to yield a small weight disturbance vector. For this to happen, the cipher codeword must cancel large number of bits which are ON in a message expansion codeword. However, the two codes are not similar, and hence we do not expect any such cancellations.

4.2 Mixed Local Collision Strategies

An adversary could try two different local collision strategies. For example, although the map $\imath$ (WCJ-local collision strategy in section 2.1) is the most effective strategy, there could be another slightly less effective (i.e. with a slightly lower probability of success) local collision strategy modeled by a map $\jmath$. Now, the adversary seeks two disturbance vectors u and $u1$, such that $\imath(u) \oplus \jmath(u1)$ is a codeword difference vector. Of course, the intent is to find small hamming weight u and $u1$, for instance $u \oplus u1$ which is not a codeword, and with much smaller weight than the min weight of the code.

Before we address this question, we have to first see if the map $\imath$ is indeed the best local collision map, i.e. one with the largest probability of success. By a local collision, we mean a differential characteristic with a single bit starting expanded message disturbance, which is offset by a string of additional differences in the expanded message so that the output difference of the characteristic is zero.

4.2.1 Local Collision Probabilities

Since, SHA-1 has different non-linear functions in the four different rounds, we expect the characteristics to have different probabilities in the different rounds, as well as in ones bordering on two rounds. Regardless, the initial disturbance (step 0), say $\Delta W^i_j = 1$, *always causes* $\Delta A^i_j = 1$. What is not certain is whether the carry bit(s) from this addition is non-zero.

Proceeding to the next step (step 1), $\Delta A^i_j = 1$ causes ΔA^{i+1}_{j+5} to be one, unless offset by a ΔW^{i+1}_{j+5}. Moreover, $\Delta B^{i+1}_j = 1$ *is automatic.*

In the next step (step 2), $\Delta B^{i+1}_j = 1$ causes $\Delta A^{i+2}_j = 1$, unless offset by ΔW^{i+2}_j, or if the $(i+2)$th step is in the IF and MAJ rounds. In the latter case, the probability of $\Delta A^{i+2}_j = 0$ is half. Moreover, $\Delta C^{i+2}_{j-2} = 1$ *is automatic.*

In step 3, $\Delta C^{i+2}_{j-2} = 1$ causes $\Delta A^{i+3}_{j-2} = 1$, unless offset by ΔW^{i+3}_{j-2}, or if the $(i+3)$th step is in the IF and MAJ rounds. In the latter case, the probability of $\Delta A^{i+3}_{j-2} = 0$ is half. Moreover, $\Delta D^{i+3}_{j-2} = 1$ *is automatic.*

In step 4, $\Delta D^{i+3}_{j-2} = 1$ causes $\Delta A^{i+4}_{j-2} = 1$, unless offset by ΔW^{i+4}_{j-2}, or if the $(i+4)$th step is in the IF and MAJ rounds. In the latter case, the probability of $\Delta A^{i+4}_{j-2} = 0$ is half. Moreover, $\Delta E^{i+4}_{j-2} = 1$ *is automatic.*

In step 5, $\Delta E^{i+4}_{j-2} = 1$ causes $\Delta A^{i+5}_{j-2} = 1$, unless offset by ΔW^{i+5}_{j-2}. This time however, there is no automatic propagation of difference.

If certain differences mentioned above are not offset as mentioned, then the differences propagate and fan out, causing additional offsets to be required later, which may or may not work with certainty. Also, since the IF round spans steps 1 to 20, and we allow the attacker complete success in the first 20 steps, we need not consider the IF round anymore. Thus, note that in the XOR rounds, apart from the initial disturbance in W, five additional differences are required in the subsequent steps. In the MAJ rounds, the only additional disturbances which are imperative are in steps 1 and 5, and the remaining three are optional. In this respect, the map $\imath$ is not unique in being the best probability local collision strategy.

The probability that there is no carry in step 0 is half, unless $j = 31$. But if $j = 31$, then in step 1, $j = 4$, and hence there is a carry there with probability half. *Thus a local collision, which has required offsets as above, cannot have probability better than half.* This includes the WCJ-local collision strategy. Heuristically, on average over all the steps, the probability of local collisions is about $2^{-2.5}$, even when additional conditions are imposed on the message bits (not the differences, but the actual bits). We give more details in the next sub- section.

4.2.2 Further Analysis

Let us assume that local collisions are as described in the previous section, with the required offsets in steps 1 to 5. An important observation made in [Wan97a, Wan97b, CJ98] is that if there is no carry difference in step 0, then if the propagation of ΔA^i_j to later steps is predictable, then one can impose conditions on the message (or W) bits, so that "no carry" in later steps is a certainty given other conditions which are required anyway. Unfortunately, for the attacker, the XOR function flips the difference (i.e. $+1$ to -1 and vice versa) with probability half. So, in XOR rounds, this feature is not applicable. On the other hand, in the MAJ rounds, where this is applicable, the MAJ function behaves linearly with probability only half. Since, there is another way to tackle carries, i.e. by requiring that the difference is in the 31st bit, it is best for the adversary to require that $j = 1$ (in step 0) for the XOR rounds.

In such a case for the XOR rounds, assuming that some message conditions can be imposed, the probability of local collision is 2^{-2}. This follows from the "no carry" in step 0, and the "no carry" in step 2, which involves the XOR function (again assuming $j = 1$). If $j \neq 1$, then the probability is at most 2^{-3}.

For the MAJ rounds, assuming that some message conditions can be imposed, the probability of local collision is at most 2^{-4}, regardless of j. In overlapping rounds, we can conclude that the probability is not better than 2^{-2}.

4.2.3 Highly Interacting Local Collisions

In the previous subsection, we dealt with local collisions in isolation. It is possible that two local collisions, or more accurately, two disturbance bits which are near each other, can have their local collisions share some of the offsets, and hence probabilities. The simplest such possibility [WYY05] is when the two disturbance bits are adjacent, say $\Delta W^i_j = 1$, and $\Delta W^i_{j+1} = 1$. Then, in step 0 we know that $\Delta A^i_j = 1$ is guaranteed. However, the carry from this may offset $\Delta W^i_{j+1} = 1$ to lead to $\Delta A^i_{j+1} = 0$.

With additional message conditions, the combined probability of $(\Delta A^i_{j+1} = 0)$ and higher carry bit differences being zero can be as as high as $1/2$. The probabilities in the remaining steps will be as in the previous subsection, i.e. for a single local collision. Thus, the probability of local collisions for these two adjacent disturbances combined can be as high as 2^{-2} in the XOR rounds.

This still gives an average of $1/2$ per disturbance bit. Further, as the code in SHA1-**IME** has a 13 bit rotation, the small weight codewords have disturbance bits widely spaced, and we do not expect this criteria to be applicable in SHA1-**IME**.

4.2.4 Mixed WCJ-like local collision strategies

In section 2.3.1 we saw that in the MAJ round, the local collision need not have offsets (from W) in steps 2, 3 and 4. This leads to a choice for the attacker in specifying the map $\imath$. Let us denote these variants of WCJ-local collision strategy maps by $\jmath$. For now, lets assume that we are dealing with only one variant, and thus $\jmath(e) = \imath(e) + e'$, where e has hamming weight one, and e' is just the required shift of e.

Given this choice, the attacker now seeks a disturbance vector $u = u1 \oplus u2$, such that $\imath(u1) \oplus \jmath(u2)$ is the difference vector, a codeword. However, by theorem 2.3, $\imath(u1) \oplus \jmath(u2) = \imath(s \oplus t)$ for some s, t, $R(s) = 0$, and $E(t) = 0$ Since the map $\imath$ is linear, we can write the above as $\imath(u1 \oplus s \oplus t) = \jmath(u2)$. Further, let $u3 = u1 \oplus s$. Then, $\imath(u3 + t) = \imath(u3) = \jmath(u2)$. Since $\jmath$ is only supposed to work in the MAJ rounds, we can assume that $\jmath(u2)$ is zero in the remaining steps. From this, one can calculate $u3$, given $u2$. Since the code specified by $\imath$ is not similar to the code specified by the SHA1-**IME** message expansion (or for that matter SHA-1), we do not expect any cancelations of $u3$ with s leading to a small hamming weight $u1$.

This is not a proof, but we give this heuristic argument to point out that there is no obvious way for the attacker to come up with a small hamming weight disturbance vector. We note that the general problem of finding low weight codewords is NP-hard [Var97]. We further note that, if we were to write down the equations in $R(\imath(u1) \oplus \jmath(u2)) = 0$, we will get equations with at least 50 terms in each equation. The more complicated that $\jmath$ gets, the more the number of terms in these equations. This then defines a parity check code, which can no longer be viewed as low-density.

5 Conclusion

To conclude, there are two extreme ways of trying to find collisions in SHA1-**IME** (or SHA-1)

1. Write down all the equations as in (2), (3) and (5), and try to solve for M by general purpose algorithms like Schoning's algorithm [Sch99] and variants, or just brute force search which includes the birthday attack.

2. Try to rewrite the equations in terms of difference variables, even if only probabilistically true, and use the fact that many equations are trivially true when the difference variables involved are zero. The extreme case here is the WCJ-local collision attack [Wan97a, Wan97b, CJ98], [WYY05].

The probability of success of the first approach is no better than 2^{-160} (with the birthday attack leading to a success in 2^{80} attempts). The probability of success in the second approach has been estimated to be at most $2^{-52\times 2.5}$ (assuming the first 32 rounds can be handled with probability one – an extremely generous assumption). Various intermediate, or mixed approaches were studied, and no approach seems to increase the probability of success.

It remains an open problem to find structure in the CSPs given by (2),(3), (4) and (5), so as to improve on the above techniques.

References

[BC04] E. Biham and R. Chen. New results on SHA-0 and SHA-1. In *Short talk presented at CRYPTO'04 Rump Session*, 2004.

[BT05] A. Bogdanov and L. Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. Electronic Colloquium on Computational Complexity, Report 2005/247, 2005. `http://eccc.uni-trier.de/eccc-reports/2005/TR05-017/index.htm`.

[CJ98] F. Chabaud and A. Joux. Differential collisions in SHA-0. In *Crypto*, 1998.

[Dr89] I. Damgå rd. A design principle for hash functions. In *in Proc. Crypto*, pages 416–427, 1989.

[FF76] J. Feigenbaum and L. Fortnow. On the randon-self-reducibility of complete sets. *SIAM Journal of computing*, pages 644–654, 22(6),1976.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP*-completeness.* W. H. Freeman, 1979.

[Gol01] O. Goldreich. *Foundations of Cryptography : Volume 1, Basic Tools.* Cambridge University Press, 2001.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[JP05] C.S. Jutla and A.C. Patthak. A simple and provably good code for SHA message expansion. In *IACR eprint archive 2005/247*, July 2005.

[Kar72] R. M. Karp. *Reducibility among combinatorial problems.* Plenum Press, New York, 1972.

[Mer89] R. C. Merkle. One way hash functions and des. In *in Proc. Crypto*, pages 428–446, 1989.

[RO05] V. Rijmen and E. Oswald. Update on SHA-1. In *Lecture Notes in Computer Science, Vol. 3376, Springer*, 2005.

[Sch99] U. Schoning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th FOCS*, pages 410–414, 1999.

[Uni93] United States Department of Commerce, National Institute of Standards and Technology, Federal Information Processing Standard Publication #180. *Secure Hash Standard*, 1993.

[Uni95] United States Department of Commerce, National Institute of Standards and Technology, Federal Information Processing Standard Publication #180-1 (addendum to [Uni93]). *Secure Hash Standard*, 1995.

[Var97] A. Vardy. The intractability of computing the minimum distance of a code. In *IEEE Transaction on Information Theory, 43(6)*, 1997.

[Wan97a] X. Y. Wang. The collision attack on SHA-0. In Chinese, 1997.

[Wan97b] X. Y. Wang. The Improved collision attack on SHA-0. In Chinese, 1997. `http://www.infosec.edu.cn/`.

[WYY05] X. Wang, H. Yu, and Y. L. Yin. Finding collisions in the full SHA-1. In *Crypto*, 2005.

A Sketch of NP-Hardness proof

A *constraint* is a function $\{0,1\}^k \to \{0,1\}$. A constraint is said to be satisfied if it evaluates to 1. A constraint is called an XOR constraint, if it represents the XOR of the k variables. Similarly, a constraint is called a MAJ constraint if it is the Majority function of the k variables. Throughout this section we will restrict to $k \leq 3$.

Given a set of boolean variables $U = \{u_1, u_2, \cdots, u_m\}$, a clause over U is a set of literals over U, e.g., $\{u_1, \bar{u_4}, u_6\}$. Clauses will be restricted to have size at most three. In our instance of the satisfaction problem, we will only have XOR-clauses and MAJ-clauses. An XOR-clause is satisfied if the XOR-constraint applied to the clause is satisfied (similarly for a MAJ-clause).

Theorem A.1 *(***Restatement of Theorem 2.1***) Given an instance consisting of a set of MAJ and XOR-clauses C over a set of boolean variables U the problem of determining if the set of clauses is simultaneously satisfiable is* NP*-hard. Further, the following restriction of the problem remains* NP*-hard: there is an ordering of the n clauses $C_1, C_2, \cdots, C_n$, such that for each clause C_i, at most one variable u in C_i* does not *satisfy the requirement that u occur only in clauses C_j, $|j - i| \leq 48$.*

For example, if there are three literals $u_1, \bar{u_4}, u_6$ in a clause then two of the variables corresponding to these clauses must occur in clauses within ± 48 of this clause. The above theorem says that instances with such highly structured set of clauses remains NP-hard.

The following theorem was proven by Karp [Kar72].

Theorem A.2 *(X3C : Exact Three Cover) Given a set X with $|X| = 3q$ and a collection T of 3-element subsets (***triples***) of X, deciding whether T contains an exact cover for X (i.e. a sub-collection $T' \subseteq T$ such that every element of X occurs in exactly one member of T') is* NP*-hard.*

We show that the problem remains NP-hard when restricted to the following collection of subsets of X: there is an ordering of the n triples $T_1, T_2, ..., T_n$ of X, such that for each T_i, two of its elements occur only in triples T_j, $|j-i| < 6$. Further, each element occurs in at most three triples.

Proof Sketch: We will reduce 3SAT-3 to this restriction of X3C. Recall that 3SAT-3 is the restriction of 3-SAT in which every variable occurs in at most three clauses. This problem is NP-hard as can easily be seen by the following reduction from 3SAT-5 [GJ79]: for each variable x_i, introduce new variables $x_i1, x_i2, ..., x_i5$,a and replace each occurence of x_i by these variables, thus assuring that each new variable occurs only once. For each i, introduce a ring of clauses $(x_1 \implies x_2)$, $(x_2 \implies x_3)$, ..., $(x_5 \implies x_1)$, and thus consistency is maintained.

Let U be the set of m variables in the 3SAT-3 instance, and C the set of n clauses.

The set X will consist of two kinds of elements: "internal" elements and "external" elements. Each 3-element subset (triple) from the collection will have two internal elements and one external element. While reducing a 3SAT-3 instance to an X3C instance, the internal elements will only occur in neighboring subsets (in a ordering of the subsets to be given), thus proving the theorem.

There will be three kinds of triples :

1. "truth setting",
2. "satisfaction testing", and
3. "garbage collecting".

For each variable $u_i \in U$, let $N \leq 3$ be the number of clauses in which u occurs. Let there be external elements named $u_i[j]$, and $\bar{u}_i[j]$, $1 \leq j \leq N$ in X. Let there also be internal elements named $a_i[j]$ and $b_i[j]$, $1 \leq j \leq N$.

Using these variables define a (ordered) collection of type 1 (truth setting) triples (which are to be viewed as forming a ring) as follows. The first triple is $(u_i[1], a_i[1], b_i[1])$. Next is the triple $(\bar{u}_i[1], b_i[1], a_i[2])$, and then $(u_i[2], a_i[2], b_i[2])$ and so forth till the last triple $(\bar{u}_i[N], b_i[N], a_i[1])$, thus completing the ring. Note that since $N \leq 3$, there are at most six triples in this ring. Moreover, each internal variable occurs in only neighboring triples (viewed as a ring). If the ring is opened up, the distance between two triples is at most ten. These internal variables will not occur in any other triples (of type 1, 2 or 3). Moreover, these external variables will not occur in any other type 1 triples.

Next we define the collection of type 2 triples (satisfaction testing). These will use external elements already defined above. However, new internal elements will be defined and included in X. For each clause C_j, let the clause have literals $u1$, $u2$, and $u3$. For each such clause three triples will be defined, all three of them having the same two new internal elements s_j and t_j. These internal elements will not occur in any other triples. Thus, at least one of these three triples (and exactly one) must be chosen in an exact cover. The external elements in the three triples will be

$u1[j]$, $u2[j]$, $u3[j]$ (which were already defined as part of type 1 triples). The intuition being that the one triple picked will signify the literal occurring in it as the one "satisfying" the clause C_j. Because of the ring structure of the type 1 triples, it cannot be the case that one clause is satisfied by a literal u, and another clause by the negation of that literal. To see this further, note that in the type 1 triples organized as a ring above, only alternate triples can be chosen in an exact cover – this alternation assures that either all $u_i[k]$ (for all k) containing triples are chosen or all $\bar{u}_i[k]$ containing triples are chosen. Thus consistency is maintained.

Finally, when the 3SAT-3 instance is satisfiable, to ensure an exact cover in our construction, we must throw in type 3 collection of triples (garbage collectors). Note that for each clause C_j one of the literals (say u) is chosen from the type 2 triples defined for this clause. Moreover, since this literal is already part of the exact cover, this literal must not be chosen in the type 1 ring corresponding to u. Hence, its negation must be chosen in the type 1 ring. Thus, both the literal $u[j]$ and its negation are accounted for. For the other two variables in the clause, only one of the two literals for these variables is accounted for.

Thus, for each clause C_j we define two collections of six garbage collecting triples. In both of these, all six triples will have the same two internal variables (which do not occur anywhere else). Hence, exactly one of these six triples must be chosen. The external variables in the six triples will just be $u1[j], \bar{u1}[j]$, $u2[j], \bar{u2}[j]$, $u3[j], \bar{u3}[j]$, where $u1, u2, u3$ are the variables occurring in C_j. This assures an exact cover. We have already proved soundness. This proves that the X3C is NP-hard. We next note that if the triples are listed in the order they were defined, and as internal variables occur only locally, it is the case that for each triple T_i two of its elements occur only in subsets T_j, $|j - i| < 10$.

The only problem with the above garbage collection scheme is that we are violating the requirement that each variable occur in only 3 triples. This is easily handled by the following modification to the garbage collection scheme. For each clause C_j, introduce three new external variables $w[j], y[j], z[j]$. Introduce two collections (A1, A2) of triples, each having three triples. The three triples in each collection will share the internal variables, hence forcing exactly one triple to be chosen per collection. The three external variables in both the collections will be $w[j], y[j], z[j]$. Thus, except for one of these, the other two will be accounted for. Also introduce three further collections (B1, B2, B3) of triples, each having three triples. Again, the three triples in each collection will share the internal variables, forcing exactly one triple to be chosen per collection. The external variables in B1 will be $w[j], u1[j], \bar{u1}[j]$. The external variables in B2 will be $y[j], u2[j], \bar{u2}[j]$. The external variables in B3 will be $z[j], u3[j], \bar{u3}[j]$. Since, only one of $w[j], y[j], z[j]$ was not accounted for, garbage collection is complete. Notice that internal variables continue to occur locally. ■

Proof Sketch (Theorem 2.1): Let X be a set with $|X| = 3q$ and a collection T of 3-element subsets (triples) of X. Further by Theorem A.2, we can assume that each element occurs in at most three triples. We reduce this X3C instance to the satisfiability problem of a set of MAJ and XOR-clauses.

For each pair of element x and triple T, such that x is in T, introduce a binary variable $\langle x, T\rangle$ which when set to 1 signifies that triple T has been chosen for the exact cover, and it covers x. Since, each element in X occurs in exactly three triples, for each element introduce an XOR-clause (if an element occurs in fewer than three triples, it is easier to handle and we ignore that case)

$$\langle x, T1\rangle \oplus \langle x, T2\rangle \oplus \langle x, T3\rangle$$

Since, if this is satisfiable all three of these variables maybe set to 1, we need a constraint forcing the majority of these three to 0, or the majority of the negation of these three to 1. Further, for all triples T, and the three elements x, y, z in T, we must require

$$\langle x, T\rangle = \langle y, T\rangle = \langle z, T\rangle$$

This completes the reduction. We now address the NP-hardness of the restricted instances. To start with we consider an instance of X3C where for each triple T_i two of its elements occur only in triples T_j, $|j - i| < 6$. As we saw in the proof of Theorem A.2, the internal variables occur only locally, and hence if the triples are listed so that triples sharing the internal variables are listed together, the above property hold.

So, consider a listing where first, for each clause the type 2 triples and the type 3 triples are listed. After these triples are listed for all the clauses, then the type 1 triples are listed. We make a list of XOR and MAJ clauses following the above list of triples in order as follows: For each triple T, we first list the clauses

$$\langle x, T\rangle = \langle y, T\rangle = \langle z, T\rangle \tag{8}$$

for the three elements x, y, z in T. For each of these three variables, i.e. $\langle x, T\rangle, \langle y, T\rangle, \langle z, T\rangle$, if they have not been previously listed in a clause, introduce in the list the XOR and MAJ clauses corresponding to these variables, i.e.,

$$\langle x, T\rangle \oplus \langle x, T2\rangle \oplus \langle x, T3\rangle \tag{9}$$

$$\text{MAJ}(\overline{\langle x, T\rangle}, \overline{\langle x, T2\rangle}, \overline{\langle x, T3\rangle}) \tag{10}$$

where $T2$ and $T3$ are the other two triples in which x occurs (same holds for y and z).

Note that the maximum number of clauses listed for each triple is at most $2 + 6 = 8$. For the variables occurring in clauses (Equation 8) above, if any of these three occurred in clauses above this clause, then for two of these the earlier triple in which they could have occurred are only six above, and hence the gap between the clauses is at most $6 * 8 = 48$. If any of these variables did not occur earlier, all the clauses in which they can occur are listed right below, i.e. in Equation 9.

For the clause in Equation 9, none of the variables have occurred before, otherwise this clause will not be listed here. One of the variables is listed right above in clause 8. Moreover, either this clause has only 2 variables, i.e. element x occurs in only two triples, or this triple T must correspond to type 2 triple, or element x is of the type $w[j], y[j], z[j]$ (see the collections A1, A2, B1,B2, B3 in the proof of theorem B). In the latter two cases, one of the triples $T2$ in which x occurs will be listed within the next 3 triples (one needs to carefully arrange the collection A1,A2, B1, B2, B3 for this), and hence the variable $\langle x, T2\rangle$ will occur within $3 * 8$ clauses. ∎