A Note on the Practical Value of Single Hash Collisions for Special File Formats

Max Gebhardt, Georg Illies, Werner Schindler
Bundesamt für Sicherheit in der Informationstechnik (BSI)
Godesberger Allee 185–189
53175 Bonn, Germany
{Maximilian.Gebhardt,Georg.Illies,Werner.Schindler}@bsi.bund.de

Abstract: We investigate Merkle-Damgard hash functions and different file formats. Our goal is to construct many meaningful hash collisions with given semantic contents from one single abstract collision. We show that this is not only possible for PostScript ([DL1]) but also for PDF, TIFF and MS Word 97. Our results suggest that this property might be typical for 'higher' file formats.

1 Introduction

In the last two years for several hash functions tremendous progress has been achieved in the construction of collisions ([WLFCY], [WY], [J2], [WYuY], [WYiY]). In particular, concrete collisions for MD5 and SHA-0 were generated and a (theoretical) break of SHA-1 (more precisely: a clear improvement compared to the birthday attack) was described. These findings make it necessary to reconsider the actual security level of crypto systems built upon such hash functions.

The 'classical' threat induced by hash collisions is the following: If someone was able to create two documents having identical hash values, and if he could persuade a person to sign one of these documents digitally (employing that hash function) he would at the same time obtain a valid signature for the second document. Clearly, this could cause a serious problem for the signer. Similarly, a dishonest signer could create two such documents, sign the first and later claim that he signed the second one, e.g. in case of liability, to discredit a signature system. Of course, signature schemes should exclude such attacks regardless whether they are viewed as realistic under real-world conditions, i.e. with respect to actual signature laws and practice. That is, only collision-resistant hash functions should be used.

Two standard arguments are often given to relativize this requirement: The first argument says that it is much more difficult to construct collisions for meaningful documents and having sufficient control over the content of those documents than to construct collisions for meaningless abstract strings of characters. It is well known that this argument is not true for generic attacks (Yuval's birthday attack, see [MOV] Section 9.7.1). Recent papers explain how abstract collisions can be used to obtain meaningful collisions of files: [Mi] (and more rudimentary [Ka][1]) for executables, [LW], [KL] for certificates and [DL1], [DL2] for PostScript documents. If, for instance, the method described in [WYiY] indeed allows the generation of SHA-1 collisions with an effort of no more than 2^{69} SHA-1 computations finding meaningful SHA-1 format file collisions (to be defined below) needs the same complexity. (Recently Wang et al. claimed that even 2^{63} hash operations were sufficient for an improved attack.)

A further argument says that a collision was worth much less than finding a secret key since the knowledge of a secret key allows to decrypt or to sign any further message without additional efforts, so that even high costs might pay off. Forging signatures without knowing the key yet needed one collision per forged signature - presumably at a high cost, e.g. for SHA-1. However, the construction in [DL1] has the interesting property that starting with a specific single MD5-collision one may construct a number of meaningful MD5 collisions of PostScript documents almost for free: In fact, using this 'basic' collision one can transform any two given Postscript documents $M_1.ps$ and $M_2.ps$ into two others with identical hash values. More precisely,

$$(M_1.ps, M_2.ps) \longmapsto (\widetilde{M_1}.ps, \widetilde{M_2}.ps)$$

such that

$$md5(\widetilde{M_1}.ps) = md5(\widetilde{M_2}.ps)$$

and such that the output (of PostScript viewers or printers) for $\widetilde{M_1}.ps$ and $\widetilde{M_2}.ps$ is the same as for $M_1.ps$ and $M_2.ps$, respectively. There is no cryptographic workload for that transformation, it only needs PostScript programming (compare Section 3 for details). This paper shows that also other file formats are vulnerable to similar attacks.

In this paper we use the following intuitive terminology for hash collisions:

- *Abstract collisions*: Two arbitrary strings having identical hash values or identical values of the compression function.
- *Meaningful format file collisions*: The two strings represent syntactically correct files for a certain format with different (partially predefined) message meanings.
- *Reusable collisions*: An abstract collision which can be used to easily construct many meaningful format file collisions.
- *Universal collisions*: An abstract collision which can be used to easily construct meaningful format file collisions with (almost) arbitrary predefined message meanings.

The construction and the properties of such 'reusable' and 'universal' collisions (also implicitly in [Mi] and [Ka] for executables) is the main topic of this short note. Clearly, the

[1] [Ka] also describes other situations in which such collisions could be harmful, e.g. Digital Right Management.

requirements for universal collisions are much stronger than for reusable collisions. We point out that for document file formats the term 'message meaning' means the 'human message' encoded in the file, e.g. graphics and texts displayed or printed from electronic document files. However, it is hardly possible to give a precise definition. We point out that it is a fundamental problem in the context of electronic signatures to specify file formats with reasonably unambiguous translations into 'human messages'.

In Section 2 we briefly describe the structure of Merkle-Damgard hash functions, basic properties and the general strategy for the construction of meaningful collisions provided that one is able to generate abstract collisions of the compression function. We examined several document file formats in Section 3. It will be shown that PDF and TIFF share the 'reusability' property with PostScript at least to a certain extent, although PostScript seems to be particularly well suited in that direction. Also the MS Word 97 file format allows the construction of meaningful format file collisions at least if macros are employed. In Section 4 and Section 5 we sketch results from [Mi] for executables and from [LW] and [KL] for certificates. We also have a look at RPM files (Red Hat Package Manager), a widespread format for distributing Linux program packages.

We consider the reusability of abstract collisions, and whether such constructions can be detected by an expert in case of a trial at court. We generated concrete collisions of PDF, TIFF and Word 97 files for MD5. To avoid a possible misuse by free-riders who could easily modify the example files we restrict ourselves to the description of the central aspects but do not publish the files themselves.

We point out that this note is a byproduct of ongoing work on hash functions at the Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany. According to the German Signature Act the BSI has to annually publish a list of recommended algorithms and parameters for qualified electronic signatures. The authors of this note are no experts for file formats. We just wanted to get an idea of the relevance of the 'trick' employed in [DL1] and [DL2] for digital signatures. The full version of this paper containing a few more technical details of the attacks described below can be found at [GIS].

2 Merkle-Damgard hash functions and meaningful collisions

A compression function is a function

$$f : \{0,1\}^n \times \{0,1\}^m \longrightarrow \{0,1\}^n$$

with some $m, n \geq 1$. Given the compression function f and an $IV \in \{0,1\}^n$ and $k \geq 1$ we define the k-block compression function

$$f_{k,IV} : \{0,1\}^{km} \longrightarrow \{0,1\}^n$$

in the following way: Let $x = x_1||x_2||\ldots||x_k$ with m-bit blocks x_i and let for $i = 1, \ldots, k$ the values h_i be defined recursively by

$$h_0 := IV, \quad h_i := f(h_{i-1}, x_i) \text{ for } i = 1, \ldots, k.$$

then

$$f_{k,IV}(x) := h_k$$

From the compression function f and a (fixed) initialization vector IV one obtains a Merkle-Damgard hash function

$$h : \{0,1\}^* \longrightarrow \{0,1\}^n$$

as follows: First, the bit string $x \in \{0,1\}^*$ is padded in a pre-defined way, typically by appending a '1' (bit), then a sequence of '0' bits and an t-bit value denoting the length of x. The number of zeroes is chosen minimal so that the length of the whole string $\widetilde{x}$ is a multiple of the block length m, let's say lm. Finally, the hash value of x is given by

$$h(x) := f_{l,IV}(\widetilde{x})$$

All the hash functions commonly in use (in particular MD5, SHA-1, RIPEMD160) are Merkle-Damgard hash functions.

We are concerned with the following situation: There is an integer $k \geq 1$ (depending on the concrete hash function) and a method that for any given IV delivers pairs of different strings $x, x' \in \{0,1\}^{km}$ with $f_{k,IV}(x) = f_{k,IV}(x')$, i.e. abstract k-block collisions of the compression function with arbitrary IV. The attacks on specific hash functions cited above actually provide such methods to construct abstract k-block collisions ($k = 2$ for MD5) of the compression functions (only theoretical for SHA-1 with $k = 1$).

A first simple observation is that for a given string a with a bitlength $k_1 m$ finding two different strings b, b' with equal bitlengths $k_2 m$ satisfying $h(a||b) = h(a||b')$ is equivalent to finding such a collision of the (multiblock) collision function described above:

$$f_{k_2,IV'}(b) = f_{k_2,IV'}(b'), \quad IV' := f_{k_1,IV}(a).$$

The second observation is that then also

$$h(a||b||c) = h(a||b'||c)$$

holds for *every* string c.

A straightforward strategy to construct meaningful collisions to a given file format works as follows: Determine a suitable string a so that for a reasonable portion of pairs b, b' which are delivered by the abstract collision search there exist strings c such that

$$M = a||b||c, \quad M' = a||b'||c$$

represent meaningful messages for that specific file format. We get a universal collision if for the fixed strings a, b, b' it is always possible to choose a string c so that M and M' have any predetermined meaning. Daum and Lucks used the term 'poisoned messages' for such constructions.

3 Document file formats

We picked out PDF, TIFF and the MS Word 97 file format (.doc) to examine whether there exist reusable or even universal collisions for these file formats. PDF and TIFF are sometimes mentioned in discussions about suitable document file formats for electronic signatures. Although the Word file format is not a candidate for such applications — it is a format for word processing not for the exact description of a document — it is perhaps the most commonly used document file format for the average PC user.

3.1 PostScript

We will briefly describe the method of [DL1]. The 'poisoned messages' have the following shape:

$$\begin{aligned}\widetilde{M}_1.ps &= \text{PS-preamble}||(S_1)(S_1)\text{eq}\{T_1\}\{T_2\}\\ \widetilde{M}_2.ps &= \text{PS-preamble}||(S_2)(S_1)\text{eq}\{T_1\}\{T_2\}\end{aligned}$$

with some strings S_1 and S_2 and two sets of further PostScript-commands T_1 and T_2. This is an 'if-then-else'-construction: A PostScript-viewer or printer checks if the two strings in parentheses (i.e. S_1, S_1 for $\widetilde{M}_1.ps$ and S_2, S_1 for $\widetilde{M}_2.ps$) are equal or not. In the first case the commands of T_1 are executed, while in the second the commands of T_2 are performed. Thus for $\widetilde{M}_1.ps$ one will get a display or printer output determined by T_1 and for $\widetilde{M}_2.ps$ an output determined by T_2.

So the general strategy described in the last section applied to this situation would be to fix a =PS-preamble||(with the preamble filled by comments such that a has a suitable blocklength and then to find b, b' as above with $h(a||b) = h(a||b')$. Setting $S_1 := b$ and $S_2 := b'$ yields $h(\widetilde{M}_1.ps) = h(\widetilde{M}_2.ps)$ for any T_1 and T_2.

Thus the pair $a||b, a||b'$ is a universal collision in the sense of Section 1. It is described in [DL1] how a concrete universal collision for MD5 can be constructed. A concrete collision is given in [DL2]. Similar constructions are possible for any file format that allows such if-then-else conditions on strings within the file.

3.2 PDF

PDF (Portable Document Format) is a widely used document format for platform independent distribution of documents. Version 1.3 is specified in [PDF]. If certain interactive features of PDF (e.g. forms, web capture) are disabled the format is basically suitable to precisely describe the appearance of a document (at least if viewers and printing software obey the PDF specification). In contrast to PostScript PDF has no programming language features such as procedures, variables and control constructs (except for the above excluded possibility to have JavaScript actions for forms). Because of that it is somewhat more difficult to find something like a reusable or universal collision.

In PDF files arbitrary strings (i.e. without restrictions on the allowed characters) can occur only in streams and as *color strings*. The latter delivers a possibility to construct poisoned black/white messages. Each byte of such a color string represents a referenceable gray scale value. In our construction we use *transfer functions* (comp. [PDF]): The actual gray scale value for a piece of text is calculated by applying such a transfer function to a byte of the color string as input. The following few lines of our PDF files (with comments after the %s) contain the basic method:

```
%PDF-1.3
  .
  .
      /HelpGS1 gs            %use the first graphic state
  .
      21 sc                  %use byte no. 21 of the color string
      (interest rate 16.00%) %first text
  .
      /HelpGS2 gs            %use the second graphic state
  .
      21 sc                  %use byte no. 21 of the color string
      (interest rate 4.50%)  %second text
  .
  .
```

To perform the general strategy of Section 2 one would let b and b' be part of the color string (placed behind the preamble). Once these two different strings have been found one chooses a color number (byte position) with different values in b and b' (21 in the above example). The rest of the file has to be modified accordingly: The chosen color number 21 has to be employed as above. The graphic states `HelpGS1` and `HelpGS2` to be defined behind the color string have to use two different transfer functions which must be chosen such that the results are 0 and 1, respectively, for byte 21 in the first file but 1 and 0 for byte no. 21 in the second file. So only one of the two texts is actually visible (printed in black) while the other is invisible (printed in white).

Remark: In our examples we printed the two texts at different positions. 'White on Black' is 'White' in PDF unless the 'overprint mode' is active. With a correctly working overprint mode the above reusable collisions would be nearly universal. But we were not able to activate this feature and it is said in [PDF] that it is device dependent. So we only claim that basically it should be possible to construct nearly universal collisions for black/white texts if all the viewers and printers strictly obey the specification. (A possible solution to overcome the overprint problem is to print the overlap of the two texts in black at the end. This basically works but the results we obtained were not perfect, the shape of the resulting characters was not really smooth, some information about the hidden text leaked out.)

3.3 TIFF

TIFF (Tagged Image File Format) is a standard image file format described in [TIFF] which in particular is used for scanning paper documents:

A TIFF file has a maximal length of 2^{32} bytes. The pages of a TIFF document are described by IFDs (image file directory). At the beginning of the TIFF file there is a header which at its end contains a 4-byte offset that specifies the start address of the first IFD (offsets in TIFF always give the position relative to the 1. byte of the file). Every IFD contains several 12-byte directory entries and a 4-byte value at the end, the latter being either the offset to specify the start of the next IFD if there is any or the value 0x00000000 if there is no further IFD.

Obviously the offsets to the next IFD could be used to construct poisoned messages: Two documents only differing in one of these offsets could be very different. But this does not work, for example, for the collision attack on MD5 described in [WY] as the hash input strings do not differ at the first 32 bytes for their collisions. One would have to find an attack with differences also for the first 4 bytes. As the offset to the first IFD is located quite at the beginning of the file even then this first IFD probably cannot be used for the attack and only the second and the following IFDs could be different.

A closer look at the directory entries in an IFD delivers a method that leads to a more practical 'poisoned message' attack. One of the 12-byte directory entries could look as follows:

tag	type	count	value
0x0111	0x0003	0x00000003	0x000014e0

The 2-byte tag 0x0111 indicates that the entry describes the offsets of the strips of the IFD: Every page can be split up in strips which can be located at different positions in the file. The 2-byte type 0x0003 indicates that the offsets will be given as 'shorts' i.e. 2-byte values. The 4-byte count 0x00000003 indicates that there are 3 strips. The 4-byte value gives the offset of the location in the TIFF file where three 2-byte offsets for the three strips can be found stored consecutively namely at 0x000014e0. (If the count was 0x00000001 or 0x00000002 then 'value' would contain the 1 or 2 short offsets themselves not their address.)

To perform the general procedure described in Section 2 one would let the 6 bytes at offset 0x000014e0 be part of the strings b, b'. And if one of the three 2-byte offsets for the strips differs for b and b' to display the corresponding strip the viewer will look at different locations in the TIFF file.

Of course there must not be overlaps between the resulting storage area of the strip data and other data in the TIFF file which leads to additional constraints on b and b' which will be satisfied with good probability if the difference $b - b'$ is random and the TIFF pages contain significantly less than 2^{15} byte information (for example fax quality text pages). Thus one can get very well reusable collisions[2] for TIFF files.

[2] There is also an index entry for the lengths of the strips which is a further constraint: The exchanged

If one tries to produce MD5 collisions for TIFF using the abstract collision attack described in [WY] one is restricted by the actual differences between b and b' that can occur. As can be seen from the TIFF reference the strips of the page must have the same length except for the last which can be significantly smaller. So the maximum that can be expected is a reusable collision that allows to produce colliding TIFF pages with basically arbitrary given contents but such that the first halfs (or the second halfs) of the pages are equal.

3.4 Word 97

In this subsection we treat meaningful file format MD5-Word 97 collisions. Roughly speaking, we wrote a Word macro that performs a specific action depending on the status of a particular if-condition. In our concrete example the last decimal digit of a purchase price, a fee or something like that should be displayed, resp. printed, in white instead of in black. In other words, the final digit should be invisible for the potential victim on the screen, resp. on the printout.

We point out that we did not study the Word format in detail. Instead, we followed a purely empirical approach where our only tool was an ordinary hex editor. The key observation was the following: After some meaningful bytes the first sector of a Word 97 file (512 Bytes) merely contains fillers ('FF'). Practical experiments underlined that Word neglects the values of these fillers.

Hence an adversary may overwrite a subsequence of fillers (e.g. Bytes 0x80 to 0xFF in our example; the numbering starts with 0x00) with a hex editor by strings that lead to abstract collisions. These strings are used as b and b' and result in two different Word files having the same hash value.

The macro scans this area and evaluates a simple arithmetic expression that is different for both files. In case of MD5, for instance, the most significiant bit of byte 0x80 + 0x13 = 0x93 equals 0 for one file and 1 for the other if we use the construction from [WY] (keep in mind the 'little endian' convention for MD5). Or equivalently, interpreted as a character byte 0x93 is < 128 in the first case but ≥ 128 in the second. Depending on the value of this character the macro either has no effect or it searches the \$-sign in the displayed text and changes the color of the last digit of the corresponding price from black to white which obviously changes the price or fee.

We point out that the general procedure is the same for any hash function provided that an adversary is able to generate abstract collisions to given IVs.

Remark a) This method seems to deliver a collision which is reusable to a certain extent: Experiments indicate that exchanging characters of the text (keeping the length constant) does not change the bytes in the file located prior to the b, b' positions.
b) We already pointed out that we followed an empirical approach. The Word format is yet very complicated and contains hundreds of parameters like offsets and modifiers. So it might not be too surprising if someone was able to evaluate more sophisticated methods

graphical data basically should have the same lengths. But practical experiments show that viewers are quite tolerant with respect to that, we didn't need to care about this.

to produce poisoned messages without using macros, similarly to the constructions for PostScript, PDF and TIFF. However, the straight-forward idea to use the offset contained in the FIB (File Information Block) of a Word 97 document, which determines the beginning of the proper text does not seem to work. As the Word format is less interesting with respect to digital signatures than PDF and TIFF we did not go into details.

3.5 Concrete examples for MD5

We constructed MD5 format file collisions for PDF, TIFF and Word 97 applying the attack of [WY]. The concrete MD5-PDF-collisions seem to work for Acrobat Readers from version 4 to 8. The concrete MD5-TIFF-collisions consist of two one-page-documents that differ at 25 % (instead of 50 %). This is due to the shape of the files produced by our scanning software: The files contained four strips of almost equal size, and for convenience we did not change this structure.

4 Executables and packages

In this section we observe that universal collisions can be constructed for binary executables and reusable collisions for RPM. These are probably less interesting in practice than the results for document file formats: The signing party should really understand and accept the code it signs otherwise there will arise problems also without hash collisions.

4.1 Plain binaries

Obviously assembler languages for common processors allow if-then-else constructions and offset constructions of poisoned programs. Also reusable and even universal collisions are possible. [Mi] describes a way to obtain such executables by writing suitable C-code instead of assembler[3]: After compilation one inspects the resulting executable to find out where the strings b, b' are located. This method could also be suitable to construct universal collisions - of course this depends on the assumption that the compiler keeps some 'natural' order of the relevant data and commands of the C-program in the executable file.

4.2 RPM

In the Linux world executables normally are distributed as packages. One of the most popular package formats is RPM (Red Hat Package Management) which is used by many Linux distributions and described for example in [Ba].

[3] Actually [Mi] describes how to produce colliding selfextracting executables common to MS Windows users.

An RPM package consists of four successive parts: The lead, the signature, the header and the archive. The lead is some sort of exterior badge whose integrity is not secured, the signature contains the size, the MD5 value and optionally a PGP-signature of the rest of the file (header+archive). We now assume that the PGP-signature employs MD5 as hash function and ask for some reusable collisions for (header+archive) to forge the RPM-signature.

The header itself contains another header and after that several 16-bytes index entries and a store at the end. The index entries look as follows:

tag	type	offset	count
4 bytes	4bytes	4 bytes	4bytes

A useful index entry for constructing reusable hash collisions has tag 0x000003ff which corresponds to RPMTAG_PREIN specifying the pre-installation script. Let the type for that entry be 0x0000006 corresponding to STRING type and the count be 0x00000001 then offset would give the location of a string inside the store: This is a shell script to be executed before installation. But Linux shells allow if-then-else constructions and allow to compare strings. So ’poisoned packages’ should be possible: b and b' are strings contained in the pre-installation script. As the archive which contains the binaries is located behind the header and thus much of the pre-installation script and at least the binaries and also some information contained in the store (for example the post-installation script which on the other side could probably as well be used for the ’poisoning’) do not need to be known when the collision is constructed.

Remark: The other index entries in the header restrict the reusableness of the collision to a certain extent.

5 Certificates

In [KL] and [LW] the possibility of constructing X.509 certificates for DSA with different primes p (one of them for example weak against known algorithms for the discrete logarithm!) but the same MD5 hash value was observed. [LW] also explicitely deliver two ASN.1 DER encoded X.509 certificates for RSA with two different RSA-moduli but the same MD5 hash value.

These examples show that collisions for X.509 certificates can be constructed. The question we are dealing with here is whether reusable collisions can be constructed. This should also be the case — in the trivial sense of extending collisions in different ways — as there are fields in X.509 certificates which contain strings of different lengths for example the ’subject organization’ field located ahead of the public key. But this possibility does not seem to have much value in practice as for example the field ’serial number’ determined by the CA is located quite at the beginning of the ASN.1 DER encoded X.509 file. So if the CA works correctly and gives different serial numbers to different certificates such a hash collision will only be usable for one pair of X.509 certificates.

6 Discussion

We picked out several file formats and searched for ways to construct reusable or universal collisions similar to that in [DL1] for PostScript. It turned out that this was in fact possible for all examined document file formats although to a different extent. If there were no explicit program language control constructions (e.g. for PDF and TIFF) these could be faked by some tricks. Maybe this is typical for many common file formats except for primitive ones like ASCII text files.

We believe that these observations are relevant with respect to electronic signatures and at least can be used as good examples to contradict the wide-spread opinion that abstract collisions of the compression function do not cause any threat in real life. Hash functions have to be collision resistant.

Two arguments could naturally arise:

1. The phenomenon is not new: If file formats allow program structures (macros, scripting) in documents it is possible to fool people even without hash collisions. The displayed content just has to differ in different environments (time, hardware).

This argument is not really true. The unambiguity of the 'human message' (cf. the introduction) is a solvable problem for digital signatures. However, some of the examples mentioned in this paper show that also file formats that are suitable with respect to that and excluding such kind of tricks, e.g. TIFF, are vulnerable against 'poisoned message' collisions, and this makes a difference.

2. A quick look at the document with a hex editor reveals the 'trick' of the forger even if only one message still exists since the victim has not stored his version. (This is an important difference to meaningful collisions of ASCII files, for instance.)

This argument is completely true except for the adjective 'quick'. Examining a file in a certain format that uses all kinds of different compressions and encodings can be quite complicated. However, although an expert should be able to figure out what has been done (provided that he knows such constructions) even if all sort of obfuscating was applied the following 'signature phishing' scenario seems to be possible: An attacker uses reusable collisions to generate 1000 collisions in a specific file format. He sends one meassage of each pair to a potential victim, hoping to get at least some valid signatures. He uses these signatures for fraud and disappears before his attack will be recognized. In this specific scenario reusable hash collisions may turn out to be quite valuable for the crook.

Once a concrete format file collision $(a||b||c)$ and $(a||b'||c)$ is known the string c may be replaced by c' keeping the collision property but changing the meaning of the encoded human message. Hence we do not publish concrete examples to avoid misuse by free-riders. Instead, we described the central aspects to sensitize the community and to put experts into the position to detect such constructions.

The examples given in this paper seem to indicate a practical basic weakness of the Merkle-Damgard construction in addition to those mentioned in [MOV] Section 9.7, [J1] and [KS].

References

[Ba] E.C. Bailey, *Maximum RPM*, Red Hat, 2000, online at http://www.rpm.org/max-rpm/

[DL1] M. Daum, S. Lucks, *The Story of Alice and Bob*, Presented at the rump session of Eurocrypt '05, May 2005, online at http://www.cits.rub.de/imperia/md/content/magnus/rump_ec05.pdf

[DL2] M. Daum, S. Lucks, concrete postscript collisions, online at http://www.cits.rub.de/MD5Collisions/

[GIS] M. Gebhardt, G. Illies, W. Schindler, *A Note on the Practical Value of Single Hash Collisions for Special File Formats*, extended version, NIST Cryptographic Hash Workshop 2005, online at http://www.csrc.nist.gov/pki/Hashworkshop/2005/Oct31_Presentations/Illies_NIST_05.pdf

[J1] A. Joux, *Multicollisions in Iterated Hash Functions*, Crypto 2004, LNCS 3152 (2004), 306-316.

[J2] A. Joux, *Collisions for SHA-0*, Presented at the rump seesion of Crypto '04, August 2004.

[Ka] D. Kaminsky, *MD5 to be considered harmful someday*, preprint, December 2004, online at http://www.doxpara.com/md5 someday.pdf.

[KL] J. Kelsey, B. Laurie, Contributions to the mailing list cryptography@metzdowd.com, December 22, 2004, online at http://diswww.mit.edu/bloom-picayune/crypto/16587.

[KS] J. Kelsey, B. Schneier, *Second preimages on n-bit hash functions for much less than* 2^n *work*, Cryptology ePrint Archive, Report 2004/304, http://eprint.iacr.org/2004/304.

[LW] A. Lenstra, B. de Weger, *On the possibility of constructing meaningful hash collisions for public keys*, In ACISP 2005, Springer LNCS 3574 (2005), 267-279. Full version at http://www.win.tue.nl/ bdeweger/CollidingCertificates/ddl-full.pdf.

[Mi] O. Mikle, *Practical Attacks on Digital Signatures Using MD5 Message Digest*, Cryptology ePrint Archive, Report 2004/356, http://eprint.iacr.org/2004/356.

[MOV] A. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997.

[PDF] Adobe Systems Incorporated, *PDF Reference, 2nd edition*, Addison Wesley, online at http://partners.adobe.com/public/developer/pdf/index_reference.html

[TIFF] TIFF Revision 6.0, Adobe Developers Association, online at http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

[WLFCY] X. Wang, X. Lai, D. Feng, H. Chen and X. Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, EuroCrypt 2005, Springer LNCS 3494 (2005), 1-18.

[WY] X. Wang and H. Yu , *How to Break MD5 and Other Hash Functions*, EuroCrypt 2005, Springer LNCS 3494 (2005), 19-35.

[WYiY] X. Wang, Y. L. Yin, H. Yu, *Collision Search Attacks on SHA-1*, Crypto 2005, Springer LNCS 3621 (2005), 17-36.

[WYuY] X. Wang, H. Yu, Y. L. Yin *Efficient Collision Search Attacks on SHA-0*, Crypto 2005, Springer LNCS 3621 (2005), 1-16.