# Notes on Email Postmarking

Jeff Goldberg
Cranfield University Computer Centre

J.Goldberg@Cranfield.ac.uk

**Abstract**

This paper outlines a scheme by which it should be possible to prove that a particular message, with particular content, actually passed through a mailhub. This problem is related to, but distinct from, both timestamping and authenticating (digital signatures). Timestamping only proves that a document existed at a certain time, while signing proves "authorship" in an unrepudiatable manner. Neither of these, on their own, can be used to prove that a particular message was actually sent.

The general proposal is for mailhubs to not only log the information that they already do log, but also a "message summary" of the body and possibly some headers of the message. Logging such information with transaction material (as logged by the MTA) can be used to demonstrate that a particular message was sent from a particular address to a particular address.

This could be used (a) to make one type of Email forgery easier to trace; (b) allow resolution of conflicting claims about the contents of a message sent; (c) demonstrate that something really had been sent to someone who might claim failure to receive.

Although the current proposal is too vague to provide a basis for a standard, the hope is that after sufficient discussion, it will evolve into something that could become a standard which mailhub managers will choose to implement.

## 1 Preamble

Any seemingly obvious "really neat" new idea that you have while going for a walk, watching a sunset or taking a long hot shower is probably subject to one of two problems. Either you have not been reading the right things and there is nothing new about the idea at all, or the idea is deeply flawed and you haven't seen the flaw yet. The idea discussed in this essay may easily fall into one of these two categories. If so, my apologies in advance.

## 2 Goals and assumptions

I want an easy to implement system that would allow us to verify that a given Email messages passed through a particular mail hub at a particular time. The notion is related to one of timestamping[1], but is intended to be complimentary. A timestamper can be used to prove that a document existed at a particular time, but it can not prove that the document was actually sent to anyone other than the timestamper.

Imagine any of the following problems:

1. Alice is expected to send a document to Bob by a particular date. If Alice fails to send the document on time the consequences are serious, while if there were a delivery delay the consequences are less serious. Imagine for example that Alice was expected to make a payment at a particular time. The document fails to arrive. Bob suspects that Alice didn't send it. Alice produces (a possibly time-stamped) copy and says that she did. Either Bob is lying, Alice is lying, or there was a transport error. A postmarking system would allow Alice or Bob to prove whether Alice was lying. It may also be able to verify or rule out some classes of transport error. I will call this the "cheque is in the mail" problem.

2. Alice sends an unsigned[2] message to Bob. Bob later complains that Alice sent something that she shouldn't have (say, for example, something that would violate the terms of the law or constitute "hate mail"). Alice denies the accusation and produced a copy of a message asserting that it is what she sent. Bob may provide another message claiming that it is what he received. For simplicity, let's suppose that each agree that only one message was sent from Alice to Bob. A postmarking system should allow us to determine whether Alice is telling the truth (and possibly whether Bob is as well, but that is more difficult).

3. A suspected mail forgery has occurred and its header indicates a large number of sites it has passed through. For any postmarking site it claims to have passed through it should be easy to verify whether it has. This may make some types of mail forgery cases easier to investigate.

There are several desiderata of a perfect scheme, only some of which are met by the current proposal.

- It should be easy to implement

[1] `http://www.inconsult.co.uk/stamper.htm`

[2] Here "unsigned" means not containing an unrepudiatable digital signature. It does *not* mean "anonymous"

- It should have no noticeable effects on users who do not wish to use its facilities
- It should place a minimum of extra work on postmasters
- It should not be resource intensive
- It should not reduce existing Email privacy. In particular it should not make traffic analysis much easier nor should it damage the work of anonymous remailers.

Throughout I will assume that the reader is familiar with the notion of a cryptographically strong one-way hash or summary. See both Schneier's book [Schneier, 1996] and appendix A for more information. I will use MD5 as my example throughout, although nothing depends on this particular summary algorithm.[3] I will also assume that the reader is familiar with the basic workings of mail transport and particularly with SMTP [Postal, 1982].

## 3 A first *simple* suggestion

In this section I discuss a simple proposal. It can be elaborated on in a number of ways discussed later.

Each message passing through a postmarking mailhub would have an MD5 summary of the body of the message with selected headers line (in particular the `To:`, `From:`, and `Subject:` lines). The MD5 summary can be logged with other information in whatever mail transfer logs are kept. If, later, there is a dispute about whether a particular message was sent (say of the "cheque is in the mail" type. The sender can produce a copy; the MD5 summary can be constructed for it, and that MD5 summary can be compared with what has been logged.

Likewise with a dispute of someone denying they sent a particular message through a hub, the recipient can supply the message that he alleges to have received, and that can be checked against the MD5 summary that has been logged. It must be noted that this is far from secure, since Alice may have sent the message to Bob, but in a manner which didn't pass through mailhubs that perform this sort of checking and so could still deny sending it. By the same techniques that allow for forgery, it will *always* be possible for people to send messages which they can later deny. Basically, they only need to forge a message from themselves.

Thirdly, if forged message is created with bogus `Received-by:` fields, than the logs on each site with that the message claims to have passed

[3] A very recent report [Dobbertin, 1996] suggests that in future it may be possible to generate multiple texts which produce identical MD5 summaries. Other one-way hash algorithms may be more appropriate.

through can be checked. In some cases this may help trace the source of some email forgeries.

### 3.1 Some details

A large number of questions remain to be answered. A small number of this are discussed below.

1. Which header fields should be included for summary?

   Clearly the `To:`, `Cc:`, and `Subject:`fields should be. As should the `From:`fields. Certain fields shouldn't be. Any field which is not available to the recipient shouldn't be, this includes the `bcc:` field (if it becomes available to the MTA at all). Also fields that are added by MTAs should not be included. In particular the `Received-by:` fields.

   The `Date:` is debatable. Because the date information is actually logged it is not essential that this field be included. Because some MUAs to not produce dates and some MTAs do not insist on them (or will insert them themselves), it may be simplest not to include dates.

2. Should summary be calculated before or after any header rewriting?

   Some MTAs may rewrite the `From:`, and even the `To:` lines that appear in messages passing through. Any such rewriting will affect the summary. Because several different inputs to this rewriting can lead to the same output[4], it makes the most sense to perform the calculation *after* any such rewriting. This way the summary is most likely to correspond to what is actually delivered. In the case where the sender has a copy of a message to verify, it can always be cycled through the rewriting operation, while the reverse would not always be possible.

3. What order should the headers be calculated in.

   Some MTAs rewrite the header information to reorder the fields. Since this can occur after the message has passed through a mailhub, we must take care to always process the headers in a consistent order, e.g., `From:`, `To:`, `Subject:`, irrespective of the order of headers of the actual messages or the sites on reordering policy.

4. Will different ways of representing line terminators affect summaries?

   Yes. Two messages identical except that one uses Unix newline conventions and another VMS conventions would normally have differing MD5 summaries. To cope with this we will need to translate local conventions to some agreed upon form for the purposes of calculating summaries.

[4]Indeed, that is often the intention, so that *user@station1.site.ac.uk* and *user@station2.site.ac.uk* will be rewritten in `From:` lines to *user@site.ac.uk* .

5. Will trailing whitespace lead to similar problems?

   Some MTAs strip trailing whitespace and others don't. It would probably be safest to strip trailing whitespace when calculating sums.

6. Tabs and spaces?

   Some MTAs may replace tab characters with spaces. I suspect that such MTAs are so rare at the moment that we can ignore them, and document this is a potential problem.

7. ASCII and EBCDIC

   Surely you're joking!

8. Interaction with anonymous remailers

   If Alice sends a message to a public forum (say a newsgroup) via an anonymous remailer[5], and Bob suspects that Alice is the author of the message, Bob could reconstruct what the headers must have looked like and construct an MD5 summary which he could then check at the particular site, and thus subvert the function on anonymous remailers.

   There are several schemes which could be used to avoid this problem. Many anonymous remailers have a password scheme. If this is handled in the body of the message (which the remailer than removes), Bob will not be able to construct an MD5 summary of an anonymous message – even if he correctly guesses its author – unless he knows her password on the remailer. Other remailers (or the same ones, using multiple systems) look for a header `Anon-Password:` which it strips off before sending the mail on. The summary operation could include such headers. Again, only if Bob knows Alice's password for the remailer will he be able to use postmarking to confirm his suspicions.

9. Implementation

   There are many questions about implementation. Most MTAs should provide hooks so that at some stage a message could be passed temporarily to some other process, which might be able to write to the log files in question. However, if we wish to make this operation as costless as possible, it would be better to not have to carry the overhead of starting a separate process for each message. Thus such a summary should be built into the MTA. Libraries for performing MD5 are widely available, so it should not be much work for someone who understands the internals of the MTA to modify the source for doing that, but such people are still somewhat of a rarity.

[5] See For a description of anonymous remailers and their many legitimate uses, see `http://www.stack.urc.tue.nl/~galactus/remailers/` .

We must note that the above scheme will work, but doesn't satisfy all desiderata. The logs which this information is kept in (with other email logs) is usually confidential. Thus it would probably take manual intervention to confirm whether a given message (or summary) occurs in the log. This is not a minimal effort for postmasters.

We also must not forget that the address in the `To:` field does not have to correspond to that actual recipient address. Proof that a particular message with a `To:` field of *J.Goldberg@Cranfield.ac.uk* passed through a hub, does not prove that the message was actually sent to that address.[6] However since SMTP logs usually list the actual recipients and actual hosts the message is from (up to the limits of the authenticity of IP addresses) we will have that information available in the logs along side the MD5 summary.

### 3.2 Automating look-up

It would be possible to take the above scheme and automate the lookup. That is, any user (whether at the particular site or not) could automatically submit an MD5 summary (or a mail message) to some designated address (or port) using some specially designed protocol and receive an answer as to whether such a summary has been logged.

There are several reasons why automating such a scheme will not work. Almost all of the failures depend on the fact that the real recipients (and senders) may differ from what is in the `To:` and `From:` fields. The central problem is that the summary does not include the real recipient address, but only the `To:` field.

Imagine if Alice wants to send hate mail to Bob and be able to deny it later. She could send an innocuous letter to herself, but with Bob listed in the `To:` field. The real letter would be sent by another route, forged to appear as if it had gone through the particular hub. Alice would retain a copy of the outgoing innocuous letter, which would also appear as logged on the mail hub. The same scheme could be done with the "cheque in the mail" problem.

#### 3.2.1 Including real recipient list in summary

One's first temptation (well, mine at least) is to include a list of recipients as the MTA sees it in the summary. This won't work. It won't work because the users don't have access to the actual recipient lists. This means that neither Alice nor Bob would be able to generate a message summary from the information they have about a legitimate message which will match something in the postmarking log.

[6] Please consult the SMTP definition [Postal, 1982] if you are not familiar with this fact.

In general the recipient lists are not reconstructible from the message headers. There are (at least) three reasons why this can be the case for legitimate messages

1. Blind carbon copies will add addresses to the recipient list but omit them from headers.

2. Alias expansion will change the form of many addresses at various stages.

3. Elimination of duplication. Most (all?) MTAs will eliminate duplicate recipients after alias expansion.

This leaves us with a problem. If we include recipient lists in the material to be summarised it would be difficult, and often impossible, to check whether a given message passed through a system, while if we don't include them, unsent messages can be given postmarks.

### 3.3 A leaky alternative

One possibility would be to have an automatic lookup on the MD5 summary, and upon success return all the information logged about that message, including actual recipients. The most sever problem with this, is that it entirely obviates `bcc:` operations. Although the loss of `bcc:` may not be so great, there are other problems that arise with such an option. These mostly surround what is meant by "recipient information" which, like various header lines, can be rewritten by the mailhub. In the end, the many minor technical problems that would need to be solved, coupled with the loss of `bcc:` in an overall scheme designed to enhance security, privacy and integrity, makes this approach untenable.

### 3.4 Conclusions

Once we have decided to follow an approach that logs the information in a confidential log, which is in no way searchable by non-privileged users, we are left with the simplest of the various proposals. A secure summary of the body and some header fields of a message (in some canonical form) should be added to the logs along with other information to be logged. The information in those logs, can, and will continue, be revealed only at the discretion of those who currently have access to those systems.

The real question – for which I provide no answer – is whether such a system is worth the effort of constructing. The number of actual cases in which it would prove useful is small, and the additional security it gives could be evaded by someone who wishes to do so. On the other hand, if new releases of MTA were to include the necessary material, than we would probably be pleased to have it. At either rate, if we choose to reject a

system like this, it is best to do so explicitly after evaluating its benifits and the difficulties of running and implementing instead of implicitly by not considering it.

## References


[Dobbertin, 1996] Dobbertin, H. (1996). Cryptanalysis of MD5 compress. Technical report, German Information Security Agency.

[Postal, 1982] Postal, J. (1982). A simple mail transport protocol, RFC 821. Technical report.

[Schneier, 1996] Schneier, B. (1996). *Applied Cryptography.* John Wiley & Sons, New York, 2 edition.


# A MD5 Summary

A cryptographically strong summary is like a checksum except that it is practically impossible to take a given summary construct a message with that summary. The MD5 algorithm produces a 128 bit summary. Since most messages will be longer than their summaries, there will be collisions (multiple messages with the same summary), but they will be rare and not practical to create. A good summary system should have at least the following two properties:

1. Given a particular file (or summary) it is practically impossible to create a distinct file which will produce the same summary.

2. Given a summary alone it is practically impossible to reconstruct the document it was derived from.

The MD of MD5 stands for message digest. MD5 was developed by Ron Rivest, and is an improvement on MD4. It has been recently popular not least because of the reputation of its creator and its use in the popular program PGP for providing digital signatures.

## A.1 Digital Signature

A digital signature is usually a "publicly encrypted" message digest. The encryption method is asymmetric, which means – in this case – that only the holder of some secret key could have encrypted the summary, but anyone with access to the public key that corresponds to to private key can decrypt it. Only someone with access to the secret key could have created the digital signature and since the signature depends on the contents of the message signed as well it can't simply be lifted and placed on another text.

# Notes on Email Postmarking

Jeff Goldberg
Cranfield University Computer Centre

J.Goldberg@Cranfield.ac.uk

**Abstract**

This paper outlines a scheme by which it should be possible to prove that a particular message, with particular content, actually passed through a mailhub. This problem is related to, but distinct from, both timestamping and authenticating (digital signatures). Timestamping only proves that a document existed at a certain time, while signing proves "authorship" in an unrepudiatable manner. Neither of these, on their own, can be used to prove that a particular message was actually sent.

The general proposal is for mailhubs to not only log the information that they already do log, but also a "message summary" of the body and possibly some headers of the message. Logging such information with transaction material (as logged by the MTA) can be used to demonstrate that a particular message was sent from a particular address to a particular address.

This could be used (a) to make one type of Email forgery easier to trace; (b) allow resolution of conflicting claims about the contents of a message sent; (c) demonstrate that something really had been sent to someone who might claim failure to receive.

Although the current proposal is too vague to provide a basis for a standard, the hope is that after sufficient discussion, it will evolve into something that could become a standard which mailhub managers will choose to implement.

## 1 Preamble

Any seemingly obvious "really neat" new idea that you have while going for a walk, watching a sunset or taking a long hot shower is probably subject to one of two problems. Either you have not been reading the right things and there is nothing new about the idea at all, or the idea is deeply flawed and you haven't seen the flaw yet. The idea discussed in this essay may easily fall into one of these two categories. If so, my apologies in advance.

## 2 Goals and assumptions

I want an easy to implement system that would allow us to verify that a given Email messages passed through a particular mail hub at a particular time. The notion is related to one of timestamping[1], but is intended to be complimentary. A timestamper can be used to prove that a document existed at a particular time, but it can not prove that the document was actually sent to anyone other than the timestamper.

Imagine any of the following problems:

1. Alice is expected to send a document to Bob by a particular date. If Alice fails to send the document on time the consequences are serious, while if there were a delivery delay the consequences are less serious. Imagine for example that Alice was expected to make a payment at a particular time. The document fails to arrive. Bob suspects that Alice didn't send it. Alice produces (a possibly time-stamped) copy and says that she did. Either Bob is lying, Alice is lying, or there was a transport error. A postmarking system would allow Alice or Bob to prove whether Alice was lying. It may also be able to verify or rule out some classes of transport error. I will call this the "cheque is in the mail" problem.

2. Alice sends an unsigned[2] message to Bob. Bob later complains that Alice sent something that she shouldn't have (say, for example, something that would violate the terms of the law or constitute "hate mail"). Alice denies the accusation and produced a copy of a message asserting that it is what she sent. Bob may provide another message claiming that it is what he received. For simplicity, let's suppose that each agree that only one message was sent from Alice to Bob. A postmarking system should allow us to determine whether Alice is telling the truth (and possibly whether Bob is as well, but that is more difficult).

3. A suspected mail forgery has occurred and its header indicates a large number of sites it has passed through. For any postmarking site it claims to have passed through it should be easy to verify whether it has. This may make some types of mail forgery cases easier to investigate.

There are several desiderata of a perfect scheme, only some of which are met by the current proposal.

- It should be easy to implement

[1] `http://www.inconsult.co.uk/stamper.htm`

[2] Here "unsigned" means not containing an unrepudiatable digital signature. It does *not* mean "anonymous"

- It should have no noticeable effects on users who do not wish to use its facilities
- It should place a minimum of extra work on postmasters
- It should not be resource intensive
- It should not reduce existing Email privacy. In particular it should not make traffic analysis much easier nor should it damage the work of anonymous remailers.

Throughout I will assume that the reader is familiar with the notion of a cryptographically strong one-way hash or summary. See both Schneier's book [Schneier, 1996] and appendix A for more information. I will use MD5 as my example throughout, although nothing depends on this particular summary algorithm.[3] I will also assume that the reader is familiar with the basic workings of mail transport and particularly with SMTP [Postal, 1982].

## 3 A first *simple* suggestion

In this section I discuss a simple proposal. It can be elaborated on in a number of ways discussed later.

Each message passing through a postmarking mailhub would have an MD5 summary of the body of the message with selected headers line (in particular the `To:`, `From:`, and `Subject:` lines). The MD5 summary can be logged with other information in whatever mail transfer logs are kept. If, later, there is a dispute about whether a particular message was sent (say of the "cheque is in the mail" type. The sender can produce a copy; the MD5 summary can be constructed for it, and that MD5 summary can be compared with what has been logged.

Likewise with a dispute of someone denying they sent a particular message through a hub, the recipient can supply the message that he alleges to have received, and that can be checked against the MD5 summary that has been logged. It must be noted that this is far from secure, since Alice may have sent the message to Bob, but in a manner which didn't pass through mailhubs that perform this sort of checking and so could still deny sending it. By the same techniques that allow for forgery, it will *always* be possible for people to send messages which they can later deny. Basically, they only need to forge a message from themselves.

Thirdly, if forged message is created with bogus `Received-by:` fields, than the logs on each site with that the message claims to have passed

[3] A very recent report [Dobbertin, 1996] suggests that in future it may be possible to generate multiple texts which produce identical MD5 summaries. Other one-way hash algorithms may be more appropriate.

through can be checked. In some cases this may help trace the source of some email forgeries.

### 3.1 Some details

A large number of questions remain to be answered. A small number of this are discussed below.

1. Which header fields should be included for summary?

   Clearly the `To:`, `Cc:`, and `Subject:`fields should be. As should the `From:`fields. Certain fields shouldn't be. Any field which is not available to the recipient shouldn't be, this includes the `bcc:` field (if it becomes available to the MTA at all). Also fields that are added by MTAs should not be included. In particular the `Received-by:` fields.

   The `Date:` is debatable. Because the date information is actually logged it is not essential that this field be included. Because some MUAs to not produce dates and some MTAs do not insist on them (or will insert them themselves), it may be simplest not to include dates.

2. Should summary be calculated before or after any header rewriting?

   Some MTAs may rewrite the `From:`, and even the `To:` lines that appear in messages passing through. Any such rewriting will affect the summary. Because several different inputs to this rewriting can lead to the same output[4], it makes the most sense to perform the calculation *after* any such rewriting. This way the summary is most likely to correspond to what is actually delivered. In the case where the sender has a copy of a message to verify, it can always be cycled through the rewriting operation, while the reverse would not always be possible.

3. What order should the headers be calculated in.

   Some MTAs rewrite the header information to reorder the fields. Since this can occur after the message has passed through a mailhub, we must take care to always process the headers in a consistent order, e.g., `From:`, `To:`, `Subject:`, irrespective of the order of headers of the actual messages or the sites on reordering policy.

4. Will different ways of representing line terminators affect summaries?

   Yes. Two messages identical except that one uses Unix newline conventions and another VMS conventions would normally have differing MD5 summaries. To cope with this we will need to translate local conventions to some agreed upon form for the purposes of calculating summaries.

[4]Indeed, that is often the intention, so that *user@station1.site.ac.uk* and *user@station2.site.ac.uk* will be rewritten in `From:` lines to *user@site.ac.uk* .

5. Will trailing whitespace lead to similar problems?

   Some MTAs strip trailing whitespace and others don't. It would probably be safest to strip trailing whitespace when calculating sums.

6. Tabs and spaces?

   Some MTAs may replace tab characters with spaces. I suspect that such MTAs are so rare at the moment that we can ignore them, and document this is a potential problem.

7. ASCII and EBCDIC

   Surely you're joking!

8. Interaction with anonymous remailers

   If Alice sends a message to a public forum (say a newsgroup) via an anonymous remailer[5], and Bob suspects that Alice is the author of the message, Bob could reconstruct what the headers must have looked like and construct an MD5 summary which he could then check at the particular site, and thus subvert the function on anonymous remailers.

   There are several schemes which could be used to avoid this problem. Many anonymous remailers have a password scheme. If this is handled in the body of the message (which the remailer than removes), Bob will not be able to construct an MD5 summary of an anonymous message – even if he correctly guesses its author – unless he knows her password on the remailer. Other remailers (or the same ones, using multiple systems) look for a header `Anon-Password:` which it strips off before sending the mail on. The summary operation could include such headers. Again, only if Bob knows Alice's password for the remailer will he be able to use postmarking to confirm his suspicions.

9. Implementation

   There are many questions about implementation. Most MTAs should provide hooks so that at some stage a message could be passed temporarily to some other process, which might be able to write to the log files in question. However, if we wish to make this operation as costless as possible, it would be better to not have to carry the overhead of starting a separate process for each message. Thus such a summary should be built into the MTA. Libraries for performing MD5 are widely available, so it should not be much work for someone who understands the internals of the MTA to modify the source for doing that, but such people are still somewhat of a rarity.

[5] See For a description of anonymous remailers and their many legitimate uses, see `http://www.stack.urc.tue.nl/~galactus/remailers/` .

We must note that the above scheme will work, but doesn't satisfy all desiderata. The logs which this information is kept in (with other email logs) is usually confidential. Thus it would probably take manual intervention to confirm whether a given message (or summary) occurs in the log. This is not a minimal effort for postmasters.

We also must not forget that the address in the `To:` field does not have to correspond to that actual recipient address. Proof that a particular message with a `To:` field of *J.Goldberg@Cranfield.ac.uk* passed through a hub, does not prove that the message was actually sent to that address.[6] However since SMTP logs usually list the actual recipients and actual hosts the message is from (up to the limits of the authenticity of IP addresses) we will have that information available in the logs along side the MD5 summary.

### 3.2 Automating look-up

It would be possible to take the above scheme and automate the lookup. That is, any user (whether at the particular site or not) could automatically submit an MD5 summary (or a mail message) to some designated address (or port) using some specially designed protocol and receive an answer as to whether such a summary has been logged.

There are several reasons why automating such a scheme will not work. Almost all of the failures depend on the fact that the real recipients (and senders) may differ from what is in the `To:` and `From:` fields. The central problem is that the summary does not include the real recipient address, but only the `To:` field.

Imagine if Alice wants to send hate mail to Bob and be able to deny it later. She could send an innocuous letter to herself, but with Bob listed in the `To:` field. The real letter would be sent by another route, forged to appear as if it had gone through the particular hub. Alice would retain a copy of the outgoing innocuous letter, which would also appear as logged on the mail hub. The same scheme could be done with the "cheque in the mail" problem.

#### 3.2.1 Including real recipient list in summary

One's first temptation (well, mine at least) is to include a list of recipients as the MTA sees it in the summary. This won't work. It won't work because the users don't have access to the actual recipient lists. This means that neither Alice nor Bob would be able to generate a message summary from the information they have about a legitimate message which will match something in the postmarking log.

[6] Please consult the SMTP definition [Postal, 1982] if you are not familiar with this fact.

In general the recipient lists are not reconstructible from the message headers. There are (at least) three reasons why this can be the case for legitimate messages

1. Blind carbon copies will add addresses to the recipient list but omit them from headers.

2. Alias expansion will change the form of many addresses at various stages.

3. Elimination of duplication. Most (all?) MTAs will eliminate duplicate recipients after alias expansion.

This leaves us with a problem. If we include recipient lists in the material to be summarised it would be difficult, and often impossible, to check whether a given message passed through a system, while if we don't include them, unsent messages can be given postmarks.

### 3.3 A leaky alternative

One possibility would be to have an automatic lookup on the MD5 summary, and upon success return all the information logged about that message, including actual recipients. The most sever problem with this, is that it entirely obviates `bcc:` operations. Although the loss of `bcc:` may not be so great, there are other problems that arise with such an option. These mostly surround what is meant by "recipient information" which, like various header lines, can be rewritten by the mailhub. In the end, the many minor technical problems that would need to be solved, coupled with the loss of `bcc:` in an overall scheme designed to enhance security, privacy and integrity, makes this approach untenable.

### 3.4 Conclusions

Once we have decided to follow an approach that logs the information in a confidential log, which is in no way searchable by non-privileged users, we are left with the simplest of the various proposals. A secure summary of the body and some header fields of a message (in some canonical form) should be added to the logs along with other information to be logged. The information in those logs, can, and will continue, be revealed only at the discretion of those who currently have access to those systems.

The real question – for which I provide no answer – is whether such a system is worth the effort of constructing. The number of actual cases in which it would prove useful is small, and the additional security it gives could be evaded by someone who wishes to do so. On the other hand, if new releases of MTA were to include the necessary material, than we would probably be pleased to have it. At either rate, if we choose to reject a

system like this, it is best to do so explicitly after evaluating its benifits and the difficulties of running and implementing instead of implicitly by not considering it.

## References


[Dobbertin, 1996] Dobbertin, H. (1996). Cryptanalysis of MD5 compress. Technical report, German Information Security Agency.

[Postal, 1982] Postal, J. (1982). A simple mail transport protocol, RFC 821. Technical report.

[Schneier, 1996] Schneier, B. (1996). *Applied Cryptography.* John Wiley & Sons, New York, 2 edition.


# A MD5 Summary

A cryptographically strong summary is like a checksum except that it is practically impossible to take a given summary construct a message with that summary. The MD5 algorithm produces a 128 bit summary. Since most messages will be longer than their summaries, there will be collisions (multiple messages with the same summary), but they will be rare and not practical to create. A good summary system should have at least the following two properties:

1. Given a particular file (or summary) it is practically impossible to create a distinct file which will produce the same summary.

2. Given a summary alone it is practically impossible to reconstruct the document it was derived from.

The MD of MD5 stands for message digest. MD5 was developed by Ron Rivest, and is an improvement on MD4. It has been recently popular not least because of the reputation of its creator and its use in the popular program PGP for providing digital signatures.

## A.1 Digital Signature

A digital signature is usually a "publicly encrypted" message digest. The encryption method is asymmetric, which means – in this case – that only the holder of some secret key could have encrypted the summary, but anyone with access to the public key that corresponds to to private key can decrypt it. Only someone with access to the secret key could have created the digital signature and since the signature depends on the contents of the message signed as well it can't simply be lifted and placed on another text.

# Notes on Email Postmarking

Jeff Goldberg
Cranfield University Computer Centre

J.Goldberg@Cranfield.ac.uk

**Abstract**

This paper outlines a scheme by which it should be possible to prove that a particular message, with particular content, actually passed through a mailhub. This problem is related to, but distinct from, both timestamping and authenticating (digital signatures). Timestamping only proves that a document existed at a certain time, while signing proves "authorship" in an unrepudiatable manner. Neither of these, on their own, can be used to prove that a particular message was actually sent.

The general proposal is for mailhubs to not only log the information that they already do log, but also a "message summary" of the body and possibly some headers of the message. Logging such information with transaction material (as logged by the MTA) can be used to demonstrate that a particular message was sent from a particular address to a particular address.

This could be used (a) to make one type of Email forgery easier to trace; (b) allow resolution of conflicting claims about the contents of a message sent; (c) demonstrate that something really had been sent to someone who might claim failure to receive.

Although the current proposal is too vague to provide a basis for a standard, the hope is that after sufficient discussion, it will evolve into something that could become a standard which mailhub managers will choose to implement.

## 1 Preamble

Any seemingly obvious "really neat" new idea that you have while going for a walk, watching a sunset or taking a long hot shower is probably subject to one of two problems. Either you have not been reading the right things and there is nothing new about the idea at all, or the idea is deeply flawed and you haven't seen the flaw yet. The idea discussed in this essay may easily fall into one of these two categories. If so, my apologies in advance.

## 2 Goals and assumptions

I want an easy to implement system that would allow us to verify that a given Email messages passed through a particular mail hub at a particular time. The notion is related to one of timestamping[1], but is intended to be complimentary. A timestamper can be used to prove that a document existed at a particular time, but it can not prove that the document was actually sent to anyone other than the timestamper.

Imagine any of the following problems:

1. Alice is expected to send a document to Bob by a particular date. If Alice fails to send the document on time the consequences are serious, while if there were a delivery delay the consequences are less serious. Imagine for example that Alice was expected to make a payment at a particular time. The document fails to arrive. Bob suspects that Alice didn't send it. Alice produces (a possibly time-stamped) copy and says that she did. Either Bob is lying, Alice is lying, or there was a transport error. A postmarking system would allow Alice or Bob to prove whether Alice was lying. It may also be able to verify or rule out some classes of transport error. I will call this the "cheque is in the mail" problem.

2. Alice sends an unsigned[2] message to Bob. Bob later complains that Alice sent something that she shouldn't have (say, for example, something that would violate the terms of the law or constitute "hate mail"). Alice denies the accusation and produced a copy of a message asserting that it is what she sent. Bob may provide another message claiming that it is what he received. For simplicity, let's suppose that each agree that only one message was sent from Alice to Bob. A postmarking system should allow us to determine whether Alice is telling the truth (and possibly whether Bob is as well, but that is more difficult).

3. A suspected mail forgery has occurred and its header indicates a large number of sites it has passed through. For any postmarking site it claims to have passed through it should be easy to verify whether it has. This may make some types of mail forgery cases easier to investigate.

There are several desiderata of a perfect scheme, only some of which are met by the current proposal.

- It should be easy to implement

[1] `http://www.inconsult.co.uk/stamper.htm`

[2] Here "unsigned" means not containing an unrepudiatable digital signature. It does *not* mean "anonymous"

- It should have no noticeable effects on users who do not wish to use its facilities
- It should place a minimum of extra work on postmasters
- It should not be resource intensive
- It should not reduce existing Email privacy. In particular it should not make traffic analysis much easier nor should it damage the work of anonymous remailers.

Throughout I will assume that the reader is familiar with the notion of a cryptographically strong one-way hash or summary. See both Schneier's book [Schneier, 1996] and appendix A for more information. I will use MD5 as my example throughout, although nothing depends on this particular summary algorithm.[3] I will also assume that the reader is familiar with the basic workings of mail transport and particularly with SMTP [Postal, 1982].

## 3 A first *simple* suggestion

In this section I discuss a simple proposal. It can be elaborated on in a number of ways discussed later.

Each message passing through a postmarking mailhub would have an MD5 summary of the body of the message with selected headers line (in particular the `To:`, `From:`, and `Subject:` lines). The MD5 summary can be logged with other information in whatever mail transfer logs are kept. If, later, there is a dispute about whether a particular message was sent (say of the "cheque is in the mail" type. The sender can produce a copy; the MD5 summary can be constructed for it, and that MD5 summary can be compared with what has been logged.

Likewise with a dispute of someone denying they sent a particular message through a hub, the recipient can supply the message that he alleges to have received, and that can be checked against the MD5 summary that has been logged. It must be noted that this is far from secure, since Alice may have sent the message to Bob, but in a manner which didn't pass through mailhubs that perform this sort of checking and so could still deny sending it. By the same techniques that allow for forgery, it will *always* be possible for people to send messages which they can later deny. Basically, they only need to forge a message from themselves.

Thirdly, if forged message is created with bogus `Received-by:` fields, than the logs on each site with that the message claims to have passed

[3] A very recent report [Dobbertin, 1996] suggests that in future it may be possible to generate multiple texts which produce identical MD5 summaries. Other one-way hash algorithms may be more appropriate.

through can be checked. In some cases this may help trace the source of some email forgeries.

## 3.1 Some details

A large number of questions remain to be answered. A small number of this are discussed below.

1. Which header fields should be included for summary?

   Clearly the `To:`, `Cc:`, and `Subject:`fields should be. As should the `From:`fields. Certain fields shouldn't be. Any field which is not available to the recipient shouldn't be, this includes the `bcc:` field (if it becomes available to the MTA at all). Also fields that are added by MTAs should not be included. In particular the `Received-by:` fields.

   The `Date:` is debatable. Because the date information is actually logged it is not essential that this field be included. Because some MUAs to not produce dates and some MTAs do not insist on them (or will insert them themselves), it may be simplest not to include dates.

2. Should summary be calculated before or after any header rewriting?

   Some MTAs may rewrite the `From:`, and even the `To:` lines that appear in messages passing through. Any such rewriting will affect the summary. Because several different inputs to this rewriting can lead to the same output[4], it makes the most sense to perform the calculation *after* any such rewriting. This way the summary is most likely to correspond to what is actually delivered. In the case where the sender has a copy of a message to verify, it can always be cycled through the rewriting operation, while the reverse would not always be possible.

3. What order should the headers be calculated in.

   Some MTAs rewrite the header information to reorder the fields. Since this can occur after the message has passed through a mailhub, we must take care to always process the headers in a consistent order, e.g., `From:`, `To:`, `Subject:`, irrespective of the order of headers of the actual messages or the sites on reordering policy.

4. Will different ways of representing line terminators affect summaries?

   Yes. Two messages identical except that one uses Unix newline conventions and another VMS conventions would normally have differing MD5 summaries. To cope with this we will need to translate local conventions to some agreed upon form for the purposes of calculating summaries.

[4] Indeed, that is often the intention, so that *user@station1.site.ac.uk* and *user@station2.site.ac.uk* will be rewritten in `From:` lines to *user@site.ac.uk* .

5. Will trailing whitespace lead to similar problems?

   Some MTAs strip trailing whitespace and others don't. It would probably be safest to strip trailing whitespace when calculating sums.

6. Tabs and spaces?

   Some MTAs may replace tab characters with spaces. I suspect that such MTAs are so rare at the moment that we can ignore them, and document this is a potential problem.

7. ASCII and EBCDIC

   Surely you're joking!

8. Interaction with anonymous remailers

   If Alice sends a message to a public forum (say a newsgroup) via an anonymous remailer[5], and Bob suspects that Alice is the author of the message, Bob could reconstruct what the headers must have looked like and construct an MD5 summary which he could then check at the particular site, and thus subvert the function on anonymous remailers.

   There are several schemes which could be used to avoid this problem. Many anonymous remailers have a password scheme. If this is handled in the body of the message (which the remailer than removes), Bob will not be able to construct an MD5 summary of an anonymous message – even if he correctly guesses its author – unless he knows her password on the remailer. Other remailers (or the same ones, using multiple systems) look for a header `Anon-Password:` which it strips off before sending the mail on. The summary operation could include such headers. Again, only if Bob knows Alice's password for the remailer will he be able to use postmarking to confirm his suspicions.

9. Implementation

   There are many questions about implementation. Most MTAs should provide hooks so that at some stage a message could be passed temporarily to some other process, which might be able to write to the log files in question. However, if we wish to make this operation as costless as possible, it would be better to not have to carry the overhead of starting a separate process for each message. Thus such a summary should be built into the MTA. Libraries for performing MD5 are widely available, so it should not be much work for someone who understands the internals of the MTA to modify the source for doing that, but such people are still somewhat of a rarity.

[5] See For a description of anonymous remailers and their many legitimate uses, see `http://www.stack.urc.tue.nl/~galactus/remailers/`.

We must note that the above scheme will work, but doesn't satisfy all desiderata. The logs which this information is kept in (with other email logs) is usually confidential. Thus it would probably take manual intervention to confirm whether a given message (or summary) occurs in the log. This is not a minimal effort for postmasters.

We also must not forget that the address in the `To:` field does not have to correspond to that actual recipient address. Proof that a particular message with a `To:` field of *J.Goldberg@Cranfield.ac.uk* passed through a hub, does not prove that the message was actually sent to that address.[6] However since SMTP logs usually list the actual recipients and actual hosts the message is from (up to the limits of the authenticity of IP addresses) we will have that information available in the logs along side the MD5 summary.

### 3.2 Automating look-up

It would be possible to take the above scheme and automate the lookup. That is, any user (whether at the particular site or not) could automatically submit an MD5 summary (or a mail message) to some designated address (or port) using some specially designed protocol and receive an answer as to whether such a summary has been logged.

There are several reasons why automating such a scheme will not work. Almost all of the failures depend on the fact that the real recipients (and senders) may differ from what is in the `To:` and `From:` fields. The central problem is that the summary does not include the real recipient address, but only the `To:` field.

Imagine if Alice wants to send hate mail to Bob and be able to deny it later. She could send an innocuous letter to herself, but with Bob listed in the `To:` field. The real letter would be sent by another route, forged to appear as if it had gone through the particular hub. Alice would retain a copy of the outgoing innocuous letter, which would also appear as logged on the mail hub. The same scheme could be done with the "cheque in the mail" problem.

#### 3.2.1 Including real recipient list in summary

One's first temptation (well, mine at least) is to include a list of recipients as the MTA sees it in the summary. This won't work. It won't work because the users don't have access to the actual recipient lists. This means that neither Alice nor Bob would be able to generate a message summary from the information they have about a legitimate message which will match something in the postmarking log.

[6] Please consult the SMTP definition [Postal, 1982] if you are not familiar with this fact.

In general the recipient lists are not reconstructible from the message headers. There are (at least) three reasons why this can be the case for legitimate messages

1. Blind carbon copies will add addresses to the recipient list but omit them from headers.
2. Alias expansion will change the form of many addresses at various stages.
3. Elimination of duplication. Most (all?) MTAs will eliminate duplicate recipients after alias expansion.

This leaves us with a problem. If we include recipient lists in the material to be summarised it would be difficult, and often impossible, to check whether a given message passed through a system, while if we don't include them, unsent messages can be given postmarks.

### 3.3 A leaky alternative

One possibility would be to have an automatic lookup on the MD5 summary, and upon success return all the information logged about that message, including actual recipients. The most sever problem with this, is that it entirely obviates `bcc:` operations. Although the loss of `bcc:` may not be so great, there are other problems that arise with such an option. These mostly surround what is meant by "recipient information" which, like various header lines, can be rewritten by the mailhub. In the end, the many minor technical problems that would need to be solved, coupled with the loss of `bcc:` in an overall scheme designed to enhance security, privacy and integrity, makes this approach untenable.

### 3.4 Conclusions

Once we have decided to follow an approach that logs the information in a confidential log, which is in no way searchable by non-privileged users, we are left with the simplest of the various proposals. A secure summary of the body and some header fields of a message (in some canonical form) should be added to the logs along with other information to be logged. The information in those logs, can, and will continue, be revealed only at the discretion of those who currently have access to those systems.

The real question – for which I provide no answer – is whether such a system is worth the effort of constructing. The number of actual cases in which it would prove useful is small, and the additional security it gives could be evaded by someone who wishes to do so. On the other hand, if new releases of MTA were to include the necessary material, than we would probably be pleased to have it. At either rate, if we choose to reject a

system like this, it is best to do so explicitly after evaluating its benifits and the difficulties of running and implementing instead of implicitly by not considering it.

## References


[Dobbertin, 1996] Dobbertin, H. (1996). Cryptanalysis of MD5 compress. Technical report, German Information Security Agency.

[Postal, 1982] Postal, J. (1982). A simple mail transport protocol, RFC 821. Technical report.

[Schneier, 1996] Schneier, B. (1996). *Applied Cryptography.* John Wiley & Sons, New York, 2 edition.


# A MD5 Summary

A cryptographically strong summary is like a checksum except that it is practically impossible to take a given summary construct a message with that summary. The MD5 algorithm produces a 128 bit summary. Since most messages will be longer than their summaries, there will be collisions (multiple messages with the same summary), but they will be rare and not practical to create. A good summary system should have at least the following two properties:

1. Given a particular file (or summary) it is practically impossible to create a distinct file which will produce the same summary.

2. Given a summary alone it is practically impossible to reconstruct the document it was derived from.

The MD of MD5 stands for message digest. MD5 was developed by Ron Rivest, and is an improvement on MD4. It has been recently popular not least because of the reputation of its creator and its use in the popular program PGP for providing digital signatures.

## A.1 Digital Signature

A digital signature is usually a "publicly encrypted" message digest. The encryption method is asymmetric, which means – in this case – that only the holder of some secret key could have encrypted the summary, but anyone with access to the public key that corresponds to to private key can decrypt it. Only someone with access to the secret key could have created the digital signature and since the signature depends on the contents of the message signed as well it can't simply be lifted and placed on another text.

# Notes on Email Postmarking

Jeff Goldberg
Cranfield University Computer Centre

J.Goldberg@Cranfield.ac.uk

**Abstract**

This paper outlines a scheme by which it should be possible to prove that a particular message, with particular content, actually passed through a mailhub. This problem is related to, but distinct from, both timestamping and authenticating (digital signatures). Timestamping only proves that a document existed at a certain time, while signing proves "authorship" in an unrepudiatable manner. Neither of these, on their own, can be used to prove that a particular message was actually sent.

The general proposal is for mailhubs to not only log the information that they already do log, but also a "message summary" of the body and possibly some headers of the message. Logging such information with transaction material (as logged by the MTA) can be used to demonstrate that a particular message was sent from a particular address to a particular address.

This could be used (a) to make one type of Email forgery easier to trace; (b) allow resolution of conflicting claims about the contents of a message sent; (c) demonstrate that something really had been sent to someone who might claim failure to receive.

Although the current proposal is too vague to provide a basis for a standard, the hope is that after sufficient discussion, it will evolve into something that could become a standard which mailhub managers will choose to implement.

## 1 Preamble

Any seemingly obvious "really neat" new idea that you have while going for a walk, watching a sunset or taking a long hot shower is probably subject to one of two problems. Either you have not been reading the right things and there is nothing new about the idea at all, or the idea is deeply flawed and you haven't seen the flaw yet. The idea discussed in this essay may easily fall into one of these two categories. If so, my apologies in advance.

## 2 Goals and assumptions

I want an easy to implement system that would allow us to verify that a given Email messages passed through a particular mail hub at a particular time. The notion is related to one of timestamping[1], but is intended to be complimentary. A timestamper can be used to prove that a document existed at a particular time, but it can not prove that the document was actually sent to anyone other than the timestamper.

Imagine any of the following problems:

1. Alice is expected to send a document to Bob by a particular date. If Alice fails to send the document on time the consequences are serious, while if there were a delivery delay the consequences are less serious. Imagine for example that Alice was expected to make a payment at a particular time. The document fails to arrive. Bob suspects that Alice didn't send it. Alice produces (a possibly time-stamped) copy and says that she did. Either Bob is lying, Alice is lying, or there was a transport error. A postmarking system would allow Alice or Bob to prove whether Alice was lying. It may also be able to verify or rule out some classes of transport error. I will call this the "cheque is in the mail" problem.

2. Alice sends an unsigned[2] message to Bob. Bob later complains that Alice sent something that she shouldn't have (say, for example, something that would violate the terms of the law or constitute "hate mail"). Alice denies the accusation and produced a copy of a message asserting that it is what she sent. Bob may provide another message claiming that it is what he received. For simplicity, let's suppose that each agree that only one message was sent from Alice to Bob. A postmarking system should allow us to determine whether Alice is telling the truth (and possibly whether Bob is as well, but that is more difficult).

3. A suspected mail forgery has occurred and its header indicates a large number of sites it has passed through. For any postmarking site it claims to have passed through it should be easy to verify whether it has. This may make some types of mail forgery cases easier to investigate.

There are several desiderata of a perfect scheme, only some of which are met by the current proposal.

- It should be easy to implement

[1] `http://www.inconsult.co.uk/stamper.htm`

[2] Here "unsigned" means not containing an unrepudiatable digital signature. It does *not* mean "anonymous"

- It should have no noticeable effects on users who do not wish to use its facilities
- It should place a minimum of extra work on postmasters
- It should not be resource intensive
- It should not reduce existing Email privacy. In particular it should not make traffic analysis much easier nor should it damage the work of anonymous remailers.

Throughout I will assume that the reader is familiar with the notion of a cryptographically strong one-way hash or summary. See both Schneier's book [Schneier, 1996] and appendix A for more information. I will use MD5 as my example throughout, although nothing depends on this particular summary algorithm.[3] I will also assume that the reader is familiar with the basic workings of mail transport and particularly with SMTP [Postal, 1982].

## 3 A first *simple* suggestion

In this section I discuss a simple proposal. It can be elaborated on in a number of ways discussed later.

Each message passing through a postmarking mailhub would have an MD5 summary of the body of the message with selected headers line (in particular the `To:`, `From:`, and `Subject:` lines). The MD5 summary can be logged with other information in whatever mail transfer logs are kept. If, later, there is a dispute about whether a particular message was sent (say of the "cheque is in the mail" type. The sender can produce a copy; the MD5 summary can be constructed for it, and that MD5 summary can be compared with what has been logged.

Likewise with a dispute of someone denying they sent a particular message through a hub, the recipient can supply the message that he alleges to have received, and that can be checked against the MD5 summary that has been logged. It must be noted that this is far from secure, since Alice may have sent the message to Bob, but in a manner which didn't pass through mailhubs that perform this sort of checking and so could still deny sending it. By the same techniques that allow for forgery, it will *always* be possible for people to send messages which they can later deny. Basically, they only need to forge a message from themselves.

Thirdly, if forged message is created with bogus `Received-by:` fields, than the logs on each site with that the message claims to have passed

[3] A very recent report [Dobbertin, 1996] suggests that in future it may be possible to generate multiple texts which produce identical MD5 summaries. Other one-way hash algorithms may be more appropriate.

through can be checked. In some cases this may help trace the source of some email forgeries.

### 3.1 Some details

A large number of questions remain to be answered. A small number of this are discussed below.

1. Which header fields should be included for summary?

   Clearly the `To:`, `Cc:`, and `Subject:`fields should be. As should the `From:`fields. Certain fields shouldn't be. Any field which is not available to the recipient shouldn't be, this includes the `bcc:` field (if it becomes available to the MTA at all). Also fields that are added by MTAs should not be included. In particular the `Received-by:` fields.

   The `Date:` is debatable. Because the date information is actually logged it is not essential that this field be included. Because some MUAs to not produce dates and some MTAs do not insist on them (or will insert them themselves), it may be simplest not to include dates.

2. Should summary be calculated before or after any header rewriting?

   Some MTAs may rewrite the `From:`, and even the `To:` lines that appear in messages passing through. Any such rewriting will affect the summary. Because several different inputs to this rewriting can lead to the same output[4], it makes the most sense to perform the calculation *after* any such rewriting. This way the summary is most likely to correspond to what is actually delivered. In the case where the sender has a copy of a message to verify, it can always be cycled through the rewriting operation, while the reverse would not always be possible.

3. What order should the headers be calculated in.

   Some MTAs rewrite the header information to reorder the fields. Since this can occur after the message has passed through a mailhub, we must take care to always process the headers in a consistent order, e.g., `From:`, `To:`, `Subject:`, irrespective of the order of headers of the actual messages or the sites on reordering policy.

4. Will different ways of representing line terminators affect summaries?

   Yes. Two messages identical except that one uses Unix newline conventions and another VMS conventions would normally have differing MD5 summaries. To cope with this we will need to translate local conventions to some agreed upon form for the purposes of calculating summaries.

[4]Indeed, that is often the intention, so that *user@station1.site.ac.uk* and *user@station2.site.ac.uk* will be rewritten in `From:` lines to *user@site.ac.uk* .

5. Will trailing whitespace lead to similar problems?

   Some MTAs strip trailing whitespace and others don't. It would probably be safest to strip trailing whitespace when calculating sums.

6. Tabs and spaces?

   Some MTAs may replace tab characters with spaces. I suspect that such MTAs are so rare at the moment that we can ignore them, and document this is a potential problem.

7. ASCII and EBCDIC

   Surely you're joking!

8. Interaction with anonymous remailers

   If Alice sends a message to a public forum (say a newsgroup) via an anonymous remailer[5], and Bob suspects that Alice is the author of the message, Bob could reconstruct what the headers must have looked like and construct an MD5 summary which he could then check at the particular site, and thus subvert the function on anonymous remailers.

   There are several schemes which could be used to avoid this problem. Many anonymous remailers have a password scheme. If this is handled in the body of the message (which the remailer than removes), Bob will not be able to construct an MD5 summary of an anonymous message – even if he correctly guesses its author – unless he knows her password on the remailer. Other remailers (or the same ones, using multiple systems) look for a header `Anon-Password:` which it strips off before sending the mail on. The summary operation could include such headers. Again, only if Bob knows Alice's password for the remailer will he be able to use postmarking to confirm his suspicions.

9. Implementation

   There are many questions about implementation. Most MTAs should provide hooks so that at some stage a message could be passed temporarily to some other process, which might be able to write to the log files in question. However, if we wish to make this operation as costless as possible, it would be better to not have to carry the overhead of starting a separate process for each message. Thus such a summary should be built into the MTA. Libraries for performing MD5 are widely available, so it should not be much work for someone who understands the internals of the MTA to modify the source for doing that, but such people are still somewhat of a rarity.

---

[5]See For a description of anonymous remailers and their many legitimate uses, see `http://www.stack.urc.tue.nl/~galactus/remailers/` .

We must note that the above scheme will work, but doesn't satisfy all desiderata. The logs which this information is kept in (with other email logs) is usually confidential. Thus it would probably take manual intervention to confirm whether a given message (or summary) occurs in the log. This is not a minimal effort for postmasters.

We also must not forget that the address in the `To:` field does not have to correspond to that actual recipient address. Proof that a particular message with a `To:` field of *J.Goldberg@Cranfield.ac.uk* passed through a hub, does not prove that the message was actually sent to that address.[6] However since SMTP logs usually list the actual recipients and actual hosts the message is from (up to the limits of the authenticity of IP addresses) we will have that information available in the logs along side the MD5 summary.

### 3.2 Automating look-up

It would be possible to take the above scheme and automate the lookup. That is, any user (whether at the particular site or not) could automatically submit an MD5 summary (or a mail message) to some designated address (or port) using some specially designed protocol and receive an answer as to whether such a summary has been logged.

There are several reasons why automating such a scheme will not work. Almost all of the failures depend on the fact that the real recipients (and senders) may differ from what is in the `To:` and `From:` fields. The central problem is that the summary does not include the real recipient address, but only the `To:` field.

Imagine if Alice wants to send hate mail to Bob and be able to deny it later. She could send an innocuous letter to herself, but with Bob listed in the `To:` field. The real letter would be sent by another route, forged to appear as if it had gone through the particular hub. Alice would retain a copy of the outgoing innocuous letter, which would also appear as logged on the mail hub. The same scheme could be done with the "cheque in the mail" problem.

#### 3.2.1 Including real recipient list in summary

One's first temptation (well, mine at least) is to include a list of recipients as the MTA sees it in the summary. This won't work. It won't work because the users don't have access to the actual recipient lists. This means that neither Alice nor Bob would be able to generate a message summary from the information they have about a legitimate message which will match something in the postmarking log.

[6] Please consult the SMTP definition [Postal, 1982] if you are not familiar with this fact.

In general the recipient lists are not reconstructible from the message headers. There are (at least) three reasons why this can be the case for legitimate messages

1. Blind carbon copies will add addresses to the recipient list but omit them from headers.
2. Alias expansion will change the form of many addresses at various stages.
3. Elimination of duplication. Most (all?) MTAs will eliminate duplicate recipients after alias expansion.

This leaves us with a problem. If we include recipient lists in the material to be summarised it would be difficult, and often impossible, to check whether a given message passed through a system, while if we don't include them, unsent messages can be given postmarks.

### 3.3 A leaky alternative

One possibility would be to have an automatic lookup on the MD5 summary, and upon success return all the information logged about that message, including actual recipients. The most sever problem with this, is that it entirely obviates `bcc:` operations. Although the loss of `bcc:` may not be so great, there are other problems that arise with such an option. These mostly surround what is meant by "recipient information" which, like various header lines, can be rewritten by the mailhub. In the end, the many minor technical problems that would need to be solved, coupled with the loss of `bcc:` in an overall scheme designed to enhance security, privacy and integrity, makes this approach untenable.

### 3.4 Conclusions

Once we have decided to follow an approach that logs the information in a confidential log, which is in no way searchable by non-privileged users, we are left with the simplest of the various proposals. A secure summary of the body and some header fields of a message (in some canonical form) should be added to the logs along with other information to be logged. The information in those logs, can, and will continue, be revealed only at the discretion of those who currently have access to those systems.

The real question – for which I provide no answer – is whether such a system is worth the effort of constructing. The number of actual cases in which it would prove useful is small, and the additional security it gives could be evaded by someone who wishes to do so. On the other hand, if new releases of MTA were to include the necessary material, than we would probably be pleased to have it. At either rate, if we choose to reject a

system like this, it is best to do so explicitly after evaluating its benifits and the difficulties of running and implementing instead of implicitly by not considering it.

## References


[Dobbertin, 1996] Dobbertin, H. (1996). Cryptanalysis of MD5 compress. Technical report, German Information Security Agency.

[Postal, 1982] Postal, J. (1982). A simple mail transport protocol, RFC 821. Technical report.

[Schneier, 1996] Schneier, B. (1996). *Applied Cryptography.* John Wiley & Sons, New York, 2 edition.


# A MD5 Summary

A cryptographically strong summary is like a checksum except that it is practically impossible to take a given summary construct a message with that summary. The MD5 algorithm produces a 128 bit summary. Since most messages will be longer than their summaries, there will be collisions (multiple messages with the same summary), but they will be rare and not practical to create. A good summary system should have at least the following two properties:

1. Given a particular file (or summary) it is practically impossible to create a distinct file which will produce the same summary.

2. Given a summary alone it is practically impossible to reconstruct the document it was derived from.

The MD of MD5 stands for message digest. MD5 was developed by Ron Rivest, and is an improvement on MD4. It has been recently popular not least because of the reputation of its creator and its use in the popular program PGP for providing digital signatures.

## A.1 Digital Signature

A digital signature is usually a "publicly encrypted" message digest. The encryption method is asymmetric, which means – in this case – that only the holder of some secret key could have encrypted the summary, but anyone with access to the public key that corresponds to to private key can decrypt it. Only someone with access to the secret key could have created the digital signature and since the signature depends on the contents of the message signed as well it can't simply be lifted and placed on another text.