# Email Authentication for Receivers

Sebastiaan de Vos – sebastiaan@inboxsys.com · Patrick Ben Koetter – p@sys4.de – Version 0.6, 30.05.2022

## Table of contents

# Document history

| Title | Date | Authored by | Abbreviation |
|---|---|---|---|
| Email Authentication for Receivers | 30.05.2022 | Sebastiaan de Vos | SdV |

| Version | Date | Description | Abbreviation |
|---|---|---|---|
| 0.6 | 30.05.2022 | Migration from old repo after review of MW | PBK |
| 0.5 | 15.04.2022 | Terminologies (From-Header / Envelope) | SdV |
| 0.4 | 14.04.2022 | Detail correction | MK, SdV |
| 0.3 | 31.03.2022 | Notes formulated in more detail | PBK |
| 0.2 | 15.03.2022 | Migration to Asciidoc | PBK |
| 0.1 | 14.03.2022 | First Draft | SdV |

Falsifying sender addresses and thus pretending to have a false identity is one of the most common forms of Internet email fraud. By impersonating someone else, attackers aim to elicit information from their victim (e.g. phishing) or persuade them to commit an act that is useful to the attackers (e.g. CEO fraud). This leads to mistrust in email in general on the part of the victims and it causes great economic damage to private individuals as well as to companies. In recent years, however, email experts have developed several methods to curb this form of abuse.

The three main methods are used in combination to a) legitimise systems sending for an envelope sender domain (SPF), b) verify the identity of a domain (DKIM) and c) to set a policy (DMARC) on how to deal with messages that do not meet SPF and DKIM, as well as to receive reports on the current status of possible identity abuse. These three methods can be summarised with the term "email authentication".

*Email Authentication*

Email authentication combines the methods of SPF, DKIM and DMARC into a mechanism with which incoming messages can be checked for authenticity. The methods in themselves provide the following possibilities:

**SPF**

SPF allows the identification of whether systems that want to send are legitimated to send on behalf of an envelope sender domain or not and also specifies how to deal with those that are not legitimated.

**DKIM**

DKIM allows the detection whether a message has been signed by the domain specified in the `DKIM Signature: header` and whether the `body` or selected headers of the message have been modified.

**DMARC**

DMARC requires successful authentication via SPF or DKIM of the `From: header` domain for a message. In addition, DMARC specifies which policy should be applied in the event of SPF and DKIM violations **and** enables the receipt of so-called feedback reports on the authentication results of a domain by storing a contact address.

This guide looks at email authentication from the perspective of a receiving mail system. It suggests software and gives configuration examples for SPF, DKIM and DMARC so that email can be authenticated before acceptance, identity abuse can be detected, and receivers can be protected from abusive messages. The aim is to only deliver messages that meet the sender-side guidelines for SPF, DKIM and DMARC.

*Email authentication for senders?*

It is equally important to legitimise one's own sender domain(s) with the methods of SPF, DKIM and DMARC and to make them verifiable for others. Already today, but even more so in the future, the deliverability of one's own messages to external systems will essentially depend on correct email authentication.

However, what needs to be done to achieve this is not covered in this document. Information on this can be found, for example, at https://certified-senders.org or https://dmarc.org. They are aimed particularly at senders and are mainly limited to how DMARC should be configured.

The following sections show how to identify, evaluate and apply SPF, DKIM and DMARC policies using various open-source software.

*Terminology*

| Letter | Email Part | Designation according to RFC | Designation in this document |
|---|---|---|---|
| Sender on the envelope | Message Envelope | RFC5321.MailFrom | Envelope Sender |
| Receiver on the envelope | Message Envelope | RFC5321.RcptTo | Receiver |
| Sender on letter | Message Header | RFC5322.From | From Header |

## 1. Risk assessment

This section covers the risk of SPF, DKIM and DMARC for delayed delivery and loss of legitimate messages.

All three methods take some time to complete their task, but the delay involved is in the range of milliseconds. The biggest time factor is (potentially sequential) DNS queries for SPF and a DNS query and signature verification for DKIM. DMARC consists of only one DNS query and has been designed to minimise the impact on delivery:

> *Scalability is a major issue for systems that need to operate in a system as widely deployed as current SMTP email. For this reason, DMARC seeks to avoid the need for third parties or pre-sending agreements between senders and receivers. This preserves the positive aspects of the current email infrastructure.*
>
> — *RFC 7489*
> *Section 2.3*

When considering the risk of message loss, it is important to understand the basic principle of DMARC: DMARC publishes guidelines for handling breaches of SPF and DKIM. DMARC requires that an email conforms to at least one of the two methods, SPF or DMARC. If both methods fail, an email is considered inauthentic.

If an attacker misuses a foreign domain to send an illegitimate message, this will result in the message failing both SPF and DKIM checks on the shelf. The question on the receiving end is now how to deal with these "errors".

This is where the p-tag policy comes into play. Here DMARC publishes the policy in the DMARC record in the DNS of the from-header domain with the help of the `p-tag` . Three values are permitted for the `p-tag` :

`none`

If `none` is set, the sender domain specified in the `From:` header requests that no action be taken when SPF and DKIM violations occur.

`quarantine`

If `quarantine` is set, the sender domain specified in the `From:` header requests that the message be accepted but not delivered directly to the mailbox, but placed in quarantine, e.g. the SPAM folder.

`reject`

If `reject` is set, the sender domain specified in the `From: header` requests that acceptance of the message be refused and that the message not be delivered.

This mechanism is simple and it works quickly and reliably. However, it may result in legitimate messages also being rejected if a sender domain (temporarily) fails to maintain its own DMARC policy.

*Friendly fire?*

- Particularly in larger organisations, it happens time and again that mail systems legitimately send emails on behalf of the organisation, but these have not been legitimised by SPF. This is the case if the IP addresses of the mail system used do not appear in the SPF entry of the envelope sender domain. [1].
- Sometimes the DKIM public key material was entered or transmitted incorrectly in the DNS of the DKIM-signing domain (usually corresponds to the From: header domain) and as a result, although the private signing key is valid, the verification of the DKIM signature fails.
- Your mail system sends a legitimate message to a mailing list that redistributes the message in a way that is not compatible with DMARC. A typical problem is when the mailing list keeps the From: header unchanged, but in doing so either removes the DKIM signature or invalidates it by modifying the message.

These three exemplary scenarios show how a DMARC‘reject’policy jeopardises the delivery of legitimate messages.

In the first two cases, it would be the sender domain's responsibility to correct the SPF and DKIM configuration and to use DMARC mounting to ensure that any own configuration errors do not cause delivery problems. For the latter case, it is the task of the mailing list to carry out the message dispatch in a DMARC-compliant manner. Furthermore, a procedure called `ARC` was developed to transport the authenticity of a message across several instances. However, `ARC` is at an experimental stage and has therefore not yet become established.

As a receiver, you can optionally configure exceptions to exclude what you consider legitimate and trusted mail systems from the SPF, DKIM and DMARC checks. Furthermore, you should send DMARC reports so that senders whose messages violate their own DMARC policy can find out about it and correct it.

## 2. The right software

Just to set the record straight: A single software product that can do everything perfectly does not exist. Depending on the use case, however, one or the other software product fits better into your service architecture.

If you prefer a modular architecture or run a mail service that is distributed across multiple instances or even machines, software that is limited to one aspect such as verifying SPF, authenticating DKIM, applying DMARC policies and generating feedback reports is better suited than a monolithic application. The following software products are suitable for this use case:

*Modular software*

- OpenSPF (http://www.open-spf.org/Software/)
- OpenDKIM (http://www.opendkim.org/)
- OpenDMARC (http://www.trusteddomain.org/opendmarc/)

If, on the other hand, you want an all-in-one solution or operate a mail service that combines everything in one server, a monolith is more suitable. The following software products are available for this application:

*Monolithic software*

- rspamd (https://rspamd.com/)
- Authentication Milter (https://github.com/fastmail/authentication_milter/)

All of the software applications mentioned are subject to an open-source licence and require a Linux or other Unix-like operating system to run. The following sections show how to configure them for the different tasks.

# 3. Check DMARC

Checking an incoming message against a DMARC policy consists of the following steps:

1. Has the sending system been legitimised by the SPF policy of the envelope sender domain?
2. Does the message include a DKIM signature, and can this can be successfully verified?
3. Has the From: header domain published a DMARC policy and was the SPF or DKIM check for the From: header domain successful?

If this is the case, from the point of view of the DMARC policy there is nothing to prevent the message from being accepted. If, on the other hand, the sending system or the message violates the DMARC policy, the receiving system should apply the DMARC policy specifications to any violations.

These three task complexes – SPF, DKIM and DMARC – can be processed successively by software modules or within a software. Regardless of which architecture you choose, the programmes will put anAuthentication-Results: header in the checked message – it lists the various check results:

*mx.example.com dokumentiert die Prüfergebnisse von sender@example.net*

RAW

```
Authentication-Results: mx.example.com;
    dkim=pass header.d=example.net header.s=202203-example.net header.b=dhpvJqM6;
    dmarc=pass (policy=reject) header.from=example.net;
    spf=pass (mx.example.com: domain of sender@example.net designates 192.2.0.1 as permitted sender)
smtp.mailfrom=sender@example.net
```

1. + Falsify examination results

An attacker who deliberately circulates emails with spoofed sender addresses to commit fraud will also attempt to subvert SPF, DKIM and DMARC by injecting spoofed "verification results" into the emails themselves, in the form of an `Authentication-Results:` header.

These fraud attempts can be prevented by specifying for your own programmes which `Authentication-Results:` headers they should trust and which they should ignore. This is done by naming a host or domain that the programmes should trust.

In the following examples, `example.com` or a subdomain of this domain is always used. Adjust the domain according to the domain of your own mail platform.

## 3.1. Modular processing

For modular processing, the OpenDKIM and OpenDMARC software are successively integrated into the processing of incoming messages via the MILTER interface of the MTA. OpenDMARC takes on two tasks – that of SPF legitimisation and that of DMARC policy checking.

The DMARC standard only makes SPF checking mandatory and considers the application of valid DKIM signatures as optional. Therefore, it is practical to have both tasks handled by one application (here: OpenDMARC).

But the importance of IP addresses in reputation systems is steadily declining, not least because of cloud-based mail services and their changing IP addresses, and so the E-Mail Competence Group of the eco Association recommends always attaching DKIM signatures as a "second pledge" and checking for them when accepting messages, in this case with OpenDKIM.

### 3.1.1. OpenDKIM

OpenDKIM (https://github.com/trusteddomainproject/OpenDKIM) can verify the DKIM signatures of incoming messages and it can apply DKIM signatures to outgoing messages. The programme is available in all common Linux distributions and its behaviour is usually controlled via the file `/etc/opendkim.conf`.

The following example configures OpenDKIM to bind locally to the IP address `127.0.0.1` on TCP port `8892` using the`socket parameter`to wait there for incoming messages and to verify their DKIM signatures, if available.

*/etc/opendkim.conf*

```
Syslog          true
Socket          inet:8892@127.0.0.1
AuthservID      mx.example.com          1
Mode            v                       2
```

1 The `AuthservID` parameter determines which identity OpenDMARC uses, when it enters its check results in an `Authentication-Results:` header.

2 The `Mode` parameter specifies with the `v` option that OpenDKIM should verify emails.

### 3.1.2. OpenDMARC

OpenDMARC (https://github.com/trusteddomainproject/OpenDMARC) checks whether incoming messages conform to the DMARC policy of the From-Header Domain, optionally generates DMARC reports and (also optionally) checks whether sending servers are authorised to send on behalf of the domain according to the SPF policy of the Envelope Sender Domain. A detailed functional description is provided by the Trusted Domain Project (http://www.trusteddomain.org/opendmarc), which develops and publishes OpenDMARC.

OpenDMARC Milter must be included in the MTA as MILTER according to OpenDKIM. The DKIM check must be completed and an `Authentication-Results:` header entered when OpenDMARC starts checking for SPF and DMARC.

The behaviour of the application is usually controlled with the help of the configuration file `/etc/opendmarc.conf`. In the following example, OpenDMARC is configured to bind locally to the IP address `127.0.0.1` on TCP port `8893`, wait there for incoming messages, perform an SPF check, evaluate the `Authentication-Results:` header, which the preceding OpenDKIM has already inserted, and then check the DMARC policy of the From-Header Domain – if available.

*/etc/openmarc.conf*

CONF

```
Syslog                true
Socket                inet:8893@127.0.0.1          1
AuthservID            mx.example.com               2
TrustedAuthservIDs    mx.example.com               3
SPFSelfValidate       true                         4
RejectFailure         yes                          5
```

1. OpenDMARC can be accessed via an `inet socket` or via a `local socket` from the MTA.
2. The `AuthservID` option specifies which identity OpenDMARC uses when it enters its check results in an `Authentication-Results:` header.
3. This option determines which existing `Authentication-Results:` headers OpenDMARC trusts and which it ignores. The identity of the upstream OpenDKIM in this section must be listed here so that OpenDMARC includes its check results in its DMARC policy check.
4. The SPF check is always done by OpenDMARC with `SPFIgnoreResults true` With `SPFSelfValidate true`, SPF is only checked by OpenDMARC if no SPF check is otherwise present in the `Authentication-Result:` header.
5. OpenDMARC does not reject any emails without being configured to do so. To do this, one must explicitly set `RejectFailure yes`.

## 3.2. Monolithic processing

Monolithic processing means all processing steps for SPF, DKIM and DMARC take place in one application. The most popular software for this is the programme `rspamd`.

### 3.2.1. rspamd

rspamd (https://rspamd.com/downloads.html) is more than just a programme to perform email authentication. While it started as a high-performance replacement for the `SpamAssassin` software, `rspamd` has been expanded over time into an all-in-one solution. It offers many filter functions (https://rspamd.com/doc/) and is constantly updated and expanded.

As delivered, rspamd already checks whether a From header domain has a DMARC policy and notes the check results in the header of the corresponding email, **but** it does not perform the actions associated with the DMARC policy. Activate these actions with the following configuration:

*/etc/rspamd/local.d/dmarc.conf*

CONF

```
actions = {
    quarantine = "add_header";
    reject = "reject";
}
```

# 4. Sending DMARC feedback reports

By sending DMARC feedback reports, email receivers are acting for the benefit of all participants of the DMARC ecosystem by providing feedback on the SPF / DKIM / DMARC compliance of their mail domain. Feedback reports report possible identity misuse as well as possible DMARC configuration errors, which benefits the stability of the entire ecosystem.

The goal of the DMARC policy of a from-header domain is ultimately always to set the policy level to either `quarantine` or `reject`. Only then there any kind of protection! The lowest policy level `none` is reserved for harmless testing of the SPF and DKIM settings. It is usually set until the From header domain has reached a "strict alignment" of its emails and then iteratively made more restrictive.

Especially in this trial phase, it is important for senders to receive DMARC feedback reports. The external view of the transmission behaviour of their domain makes it visible to them whether adjustments to their SPF policy are still necessary and/or whether DKIM signatures validate cleanly.

DMARC feedback reports are sent from the receiving mail platform to one or more email addresses noted in the DMARC policy of the From header domain. The DMARC standard distinguishes between two types of DMARC feedback reports:

**"Forensic" / "Failure" Reports**

"Forensic reports" are detailed reports that are sent for each violation of a DMARC policy. From a data protection perspective, it can happen that such a report passes on personal and thus sensitive information, which is why forensic reports are controversial. This type of report is written in the so-called `AFRF` format and is sent to the address(es) specified with the `ruf` tag in the DMARC DNS TXT record.

**"Aggregated" Reports**

An "aggregated report" summarises the events that affect a sender domain with DMARC policy in aggregated form at a certain time interval (daily is recommended). The report is created as a compressed XML file and sent to the address(es) specified in the `rua` tag in the DMARC DNS TXT record.

Sending aggregated reports is a two-part process: First, the DMARC test results are collected and then, at a certain point in time, reports are generated from them and sent. The following "Best Practices" for sending are based on recommendations and experiences of the E-Mail CG.

## 4.1. Best practices for sending

If your report service sends notifications about possible abuse of a sender domain, receivers must be able to trust it and their service must handle the perhaps personal data with confidence. We, therefore, recommend the following measures:

1. Avoid sending failure reports. Under certain circumstances, the sending of failure reports can even be legally problematic, as personal data, such as the receiver's email address or the subject, is mentioned in failure reports. The Report on the compliance of DMARC with the EU GDPR (https://certified-senders.org/wp-content/uploads/2018/08/Report_DMARC_and_GDPR.pdf)https://certified-senders.org/wp-content/uploads/2018/08/Report_DMARC_and_GDPR.pdf, which the eco E-Mail Competence Group commissioned as a legal opinion, shows in detail when failure reports are GDPR-compliant and when they violate the GDPR.
2. Always use a separate domain or subdomain as the sending domain for reports so that the sending behaviour of this (sub)domain does not negatively influence the reputation of your main domain (because reports contain IP addresses – or in the case of failure reports, even content – of spammers/phishers).
3. Use your own dedicated IP address for sending the reports, because this IP address could also get a bad reputation.
4. Add a DKIM signature to reports so that receivers can verify beyond doubt that the report originates from your service and that it can thus build up a good reputation over time.
5. Create a separate DMARC policy for the report (sub)domain and do not request failure reports or aggregated reports for this (sub)domain so that no infinite report loop can occur between your report domain and other mail platforms.
6. If your mail service consists of several mail servers that should all send reports, then only use one report service for all servers. Collect the report data centrally in a database and let the report service generate the aggregated reports based on all the information stored there.
7. Have your report service send aggregated reports only once a day. Do not send at exactly 00:00, because the volume of reports generated worldwide would otherwise be tantamount to a DDoS attack for the receiving platform.

8. Count on backscatter! Some receivers accept reports and respond to them just a few seconds later with a bounce email or a delivery notification (DSN). Think about where all these mails should end up.
9. A noticeable proportion of receiver addresses named in the `rua` tag name mailboxes for which the target system does not accept messages. The reports will remain in the mail queue of your MTA until they bounce. You should delete reports that cannot be delivered and possibly exclude the receiver addresses from delivery.
10. The sending own mailbox should have enough rate limit so that the reports can all be sent. Often thousands of reports are sent within seconds.

## 4.2. DMARC feedback reports with OpenDMARC

OpenDMARC collects data that it will use for reports in a file. The path to this file is specified with the help of the `HistoryFile` parameter. The file itself must be readable and writable for the user with whom OpenDMARC is operated.

Failure reports are best redirected to yourself:

*/etc/opendmarc.conf*

```CONF
CopyFailuresTo        postmaster@mx.example.com
FailureReportsSentBy  postmaster@mx.example.com
FailureReports        yes
FailureReportsOnNone  true
ReportCommand         /usr/sbin/sendmail userpart@sub.domain.tld     1
FailureReportsBcc     localpart@example.net                          2
```

1 Do not specify `sendmail -t` here

2 If you (still want to) send failure reports, you can use `FailureReportsBcc` to always send them to themselves and thus follow what is being reported.

*OpenDMARC Multi-Host Reporting*

If your mail platform uses several servers, each of which integrates separate OpenDMARC instances, you should collect their data in a central location and have the reports generated centrally. This facilitates the evaluation for the recipient of the report.

To do this, create a cronjob / systemd timer and let theòpendmarc-import ùprogramme periodically write the data of each OpenDMARC instance to a central database:

```SH
% opendmarc-import --dbhost=hostname --dbname=name --dbpasswd=password --dbport=port
```

Then create a second cronjob / systemd timer and let the `opendmarc-reports` programme generate and send reports centrally. You control the interval at which reports are generated and sent withînterval=secs. If you do not want to send the reports locally, but via a specific other server, specify this with the parameters `smtp-host`and `smtp-port`.

## 4.3. DMARC feedback reports with rspamd

rspamd collects data, which it will use for reports, in one or more redis databases. You must explicitly enable the generation and sending of reports by activating and configuring the `reporting` section in the file `/etc/rspamd/local.d/dmarc.conf`:

*/etc/rspamd/local.d/dmarc.conf*

CONF

```
servers = "192.2.0.1:6379";                 1
reporting {
    enabled = true;
    email = 'dmarc_reports@sub.example.com';    2
    domain = 'example.com';                 3
    org_name = 'Example organisation';      4
    bcc_addrs = ["postmaster@example.com"]; 5
    smtp = '127.0.0.1';                     6
    smtp_port = 25;                         7
    from_name = 'example.com DMARC report'; 8
    helo = 'example.com';                   9
    msgid_from = 'example.com';             10
}
```

1. Here you can, if required, deviate from the central redis configuration for rspamd and specify into which redis-DB DMARC report data should be written.
2. Here you define the envelope sender address that rspamd will use to send the reports.
3. The receiver domain in whose name the reports are generated.
4. Enter the name of your organisation here.
5. If you always want to send the reports to other addresses as well, you can specify a list of addresses here.
6. This parameter specifies the SMTP server to be contacted for dispatch
7. This parameter specifies the TCP port of the SMTP server to be contacted for sending.
8. This parameter sets the display name of the sender (default is: "Rspamd")
9. HELO in SMTP dialogue
10. Message-Id Format

Once rspamd has collected DMARC check results, you can start sending them in reports. To do this, create a cronjob / systemd timer that periodically evaluates the data of the previous day (!) and sends it as reports:

SH

```
% 27 3 * * * rspamadm dmarc_report
```

> *rspamd Multi-Host Reporting*
>
> If your mail platform uses several servers, each with its own rspamd instances, you should collect their data in a central location and generate the reports centrally. This facilitates the evaluation for the recipient of the report.
>
> Configure all rspamd instances to send their report data to the central redis database and have only one host run the cronjob / systemd timer that periodically generates and sends reports.

---

1. An expert email service provider will check whether SPF, DKIM and DMARC are correct **before** sending out the newsletter, draw attention to possible problems and point out possible solutions