The Application Layer: Fighting Spam

COMPUTER NETWORKS A.A. 24/25

Leonardo Maccari, DAIS: Ca' Foscari University of Venice,
leonardo.maccari@unive.it

Venice, fall 2024

Preface: E-mail Signature

- Yesterday Someone asked about the format of my signature, starting with "--". Is it standard?
- Of course there is an RFC for it: `https://datatracker.ietf.org/doc/html/rfc2646#section-4.3`
- And another one that says to keep it short: `https://datatracker.ietf.org/doc/html/rfc1855#section-2.1.1`
- However, they are not rules, just recommendations.

Introduction

- Many of the protocols we study in this course have been upgraded since their first introduction
- However, many of them still work if you implement them like in the original RFC, e.g. TCP
- For e-mail this is not true.
- At the core of the service, you still find the protocols we studied, but if you set-up a server with just these protocols, you will probably not deliver one single e-mail.
- This is because there is a long list of added protocols that are intended to limit SPAM
- If you don't support them, the other SMTP servers will simply drop your e-mails.

Unpopular subject

- I have checked at least 5 books for the best practices to reduce spam, and books don't talk about it.
- This is primarily because in the years literally tens of RFCs have been produced for this topic, and it is hard to keep track of all of them
- So I decided to study the subject and prepare a lesson for you on this theme

Thanks

- Many thanks to Christopher R. Gabriel[1], that, with a 6 years experience in making e-mail work, reviewed the slides
- As a side-note: I met him at the Firenze Linux User Group (FLUG), where I learned a lot about networks and system administration
- A suggestion I give you, find your local LUG and join their activities: `https://lugmap.linux.it/`

[1] `https://it.linkedin.com/in/christophergabriel`

Sect. 1 Fighting SPAM

The problem with spam

- spam and phishing e-mails are a plague that we were not able to eradicate
- A spammer generally controls a number of zombie computers that connect to some SMTP server and try to massively send e-mails
- If the spammer is a MUA, it is easy to detect by its own MSA, because it will connect, authenticate and send a huge number of e-mails

The problem with spam

- However the spammer can pretend to be an MTA, and in this case for the receiving MTA it is harder to distinguish good traffic from bad one, as MTAs may normally receive thousands of e-mails from other MTAs
- good MTAs use several techniques to detect bad MTAs, and refuse/delay/limit e-mails from them.

Anti-spam techniques

- Several techniques have been proposed to reduce spam
- None of them solves the problem completely, but servers can use many of them to give a score to an e-mail and decide if it is spam or not, and possibly reject the e-mail.
- we focus on the nework-based ones, and ignore other ones (like filters based on text analysis).

Technique 0: closing Relays

- It is intuitive that having an open relay that anybody can use is a bad practice
- Because allowing relaying without authentication, your MTA will be used by spammers to send e-mail to all the world
- Plus, they will use your MTA to send e-mail with any source address, and *spoof* the `From:` field.
- So the basic technique that was introduced is the distinction between MTA and MSA that we already talked about.

Fighting SPAM

↳1.1 IP-based Anti-spam Techniques

FQDN check

- The `HELO` command must carry a domain name if the client has one (an MTA), or an address literal (something in the form `[123.255.37.2]`) if the client does not have a fully qualified domain name (a MUA).
- If `HELO` contains a domain, the server resolves the domain into an IP.
- MTAs can check that the domain used in the `HELO` command resolve to some existing IP (and more, see RFC 2505)

FQDN check

- Furthermore, they will perform a reverse DNS query for that IP, and check that it matches the provided domain name.
- Finally, they may check that the IP that initiated the connection is the same one.
- If one of these conditions is not met, the server MTA may drop the connection.

Another (edited) e-mail header

```
1  Delivered-To: leonardo.maccari@unive.it
2  Received: by 2002:a9a:69c5:0:b0:299:9b06:7765 with SMTP id c5csp1571665lkg;
3  Wed, 6 Nov 2024 01:21:37 -0800 (PST)
4  Received: from mail.ietf.org (mail.ietf.org. [50.223.129.194]) by mx.google.com ...
5  for <leonardo.maccari@unive.it> (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
6  Wed, 06 Nov 2024 01:21:37 -0800 (PST)
7  Received-SPF: pass (google.com: domain of forwardingalgorithm@ietf.org designates 50.223.129.194 as
       permitted sender) client-ip=50.223.129.194;
8    ...
9  Received: from ietfa.amsl.com (localhost [IPv6:::1]) by ietfa.amsl.com (Postfix) with ESMTP id 1895
       DC14F6A9 for <leonardo.maccari@unive.it>; Wed,
10 6 Nov 2024 01:21:35 -0800 (PST)
11 DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
       gKKOhIEVnQtATJMEQQoY5UY13/bV+mlSqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
       :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
       WyjSYJ1FL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
       OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
12 Received: from [192.168.178.37] (77-172-38-251.fixed.kpn.net [77.172.38.251]) by smtp.xs4all.nl (Halon)
       with ESMTPSA id 7bf838ad-9c20-11ef-891f-00505699d6e5; Wed, 06 Nov 2024 10:21:20 +0100 (CET)
13 Message-ID: <5eeeb3e9-942f-4716-90ed-b6b7fbf2bc6d@xs4all.nl>
14 Date: Wed, 6 Nov 2024 10:21:19 +0100
15 MIME-Version: 1.0
16 User-Agent: Mozilla Thunderbird
17 To: gaia <gaia@irtf.org>
18 Content-Language: en-US
19 From: Vesna Manojlovic <becha@xs4all.nl>
20   ...
```

Let's focus on line 4

FQDN check

`mx.google.com` received an SMTP connection from IP `50.223.129.194`. The MTA presented itself as `mail.ietf.org`

```
Received: from mail.ietf.org (mail.ietf.org. [50.223.129.194]) by mx.google.com ...
```

Let's resolve the name into an IP

```
$ dig mail.ietf.org
;; ANSWER SECTION:
mail.ietf.org.    60   IN   A 50.223.129.194
```

OK, IPs correspond. Now let's make a PTR resolution (`-x` with `dig`)

```
$ dig -x 50.223.129.194
;; ANSWER SECTION:
194.129.223.50.in-addr.arpa. 3024 IN   PTR mail.ietf.org.
```

We verified that the domain name corresponds to an IP that corresponds to the inverse domain resolution. This is what the MTA does all the times.

FQDN Check

- If you want to operate an MTA, you need a server with a public IP and a domain name, and you need a PTR record too
- This requires some effort and prevents spammer to create an MTA without at least owning a domain.
- It is almost impossible to operate an MTA if you don't own a public IP and an associated domain name.
- It also helps to make an MTA identifiable, in order to check its reputation.

IP blacklisting

- several organizations maintain updated lists of IP addresses that have been found to be generating spam messages, so servers test the IP against the lists.
- There is a conventional protocol based on DNS, that is named DNSBL (DNS Block List, (historically this was called Realtime Block List, RBL).

DNSBL

- Before allowing the submission of an e-mail, the MTA will query a DNSBL server to test if the IP is present in the database.
- If the IP is present, it will deny the e-mail.
- Spamhaus is one of the most known service providers for DNSBL.
- The process of entering (and exiting) a blacklist is not really transparent[2], and spammers can temporary lease and use large sets of IPs and then change them altogether.
- So this technique does not work alone.

[2] See the Amazon AWS guidelines on this topic

DNSBL, in practice

- a DNSBL query is a DNS query similar to a PTR query, but against an `A` record (see RFC 6471. `127.0.0.2` can be used for testing)

```
1 $ dig  2.0.0.127.zen.spamhaus.org
2 ;; ANSWER SECTION:
3 2.0.0.127.zen.spamhaus.org. 60  IN  A 127.0.0.4
4 2.0.0.127.zen.spamhaus.org. 60  IN  A 127.0.0.2
5 2.0.0.127.zen.spamhaus.org. 60  IN  A 127.0.0.10
```

- The `A` query is used in this case to verify the presence of the IP in the database
- If the response does not return a valid `A` record, the IP is OK
- In this case, we have instead 3 existing records.

DNSBL, in practice

- Then a `TXT` query will tell why it is there (listed in 3 blacklists):

```
dig  2.0.0.127.zen.spamhaus.org TXT
;; ANSWER SECTION:
2.0.0.127.zen.spamhaus.org. 60  IN  TXT "Listed by XBL, see https://check.spamhaus.org/query/ip/127.0.0.2"
2.0.0.127.zen.spamhaus.org. 60  IN  TXT "Listed by SBL, see https://check.spamhaus.org/sbl/query/SBL2"
2.0.0.127.zen.spamhaus.org. 60  IN  TXT "Listed by PBL, see https://check.spamhaus.org/query/ip/127.0.0.2"
```

- Let's verify `mail.irtf.org`

```
$ dig 194.129.223.50.zen.spamhaus.org
;; QUESTION SECTION:
;194.129.223.50.zen.spamhaus.org. IN  A

;; AUTHORITY SECTION:
zen.spamhaus.org. 10  IN  SOA need.to.know.only. hostmaster.spamhaus.org. 2411061106 3600 600 432000 10
```

- OK the answer is empty (there is only an informative `SOA` record, no `A` record)

Fighting SPAM

↳1.2 RFC Compliance

Strict RFC Compliance

- A well-behaving SMTP server adheres to the SMTP standard, while instead, a spammer tends to not comply to the part of the standard that may slow down its activity.
- For instance, the receiving MTA can delay the answer to some commands, or return a termporary error
- spammers will not wait for the reply, and they will drop the connection, simply moving forward to the next SMTP target
- This is a classical and inexpensive technique, but it imposes a delay also on legit traffic[3]

[3] See https://tldp.org/HOWTO/Spam-Filtering-for-MX/smtpdelays.html

Strict RFC Compliance: Graylisting (RFC 6647)

- When an MTAs receive a 4** error, it should try again after some time. Spammers will not try again.
- The server MTA that uses graylisting keeps a list of known IP addresses
- When a connection is received by a client MTA that is not in the list, the server MTA answers with a 4** error, and adds the client IP in the whitelist.

Strict RFC Compliance: Graylisting (RFC 6647)

- The legitimate client MTA will try again after some time, and then the server will allow the connection as the IP is in the whitelist.
- The spammer wil just give up and test another MTA
- This is called *graylisting*, it is inexpensive but introduces a delay in the first e-mail received by a client MTA.

Strict RFC Compliance: Graylisting (RFC 6647)

- How long will the client wait? it is not known, normally some minutes, but it could be more.
- So the server should accept the second connection if it arrives between one minute and 24h (see the RFC)
- After some time of inactivity, the IP should be removed from the whitelist.

Strict RFC Compliance: Graylisting (RFC 6647)

- This technique is easy and cheap, however, it delays the reception of the first e-mail
- For MTAs that serve many e-mails this is not a problem. For instance, google MTAs are constantly whitelisted, because they send e-mails all day long, 24h/day.
- However, it is particularly annoying for MTAs that are used rarely.
- Typical example is the MTA that a web server uses to send password reset e-mails
- That MTA may send one e-mail every now and then (depending on the size of the user base), and likely, the first one will be delayed.

Fighting SPAM

↳1.3 Sender Policy Framework SPF

Further checks on HELO and MAIL FROM

```
S: 220 smtp.example.com ESMTP MTA information
C: HELO mta.example.org
S: 250 Hello mta.example.org, glad to meet you
C: MAIL FROM:<alice@example.org>
S: 250 Ok
C: RCPT TO:<bob@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Alice Doe" <alice@example.org>
C: To: Bob Smith <bob@example.com>
C: Date: Mon, 9 Mar 2010 18:22:32 +0100
C: Subject: Hello
C:
C: Hello Bob
C: This is a small message containing 4 lines of
     text.
C: Best regards,
C: Alice
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
```

- the HELO command is followed by a domain name/IP address
- the MAIL FROM: is followed by an e-mail
- the From: header in the body also contains an e-mail
- What is their real meaning?

From fields

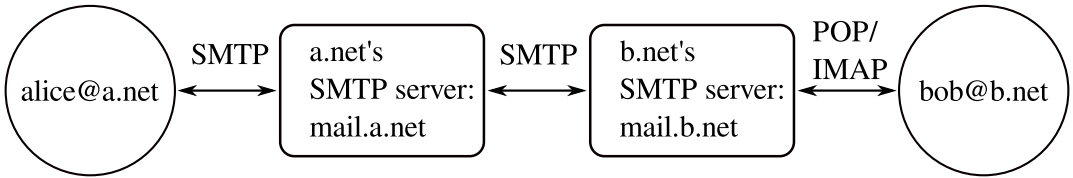
HELO: Identifies the domain name of the MTA, that is the SMTP server that is acting as a client

MAIL FROM: This is not necessarily the sender of the e-mail. This is where to send errors, that could be a different e-mail address than the one of the user who generated the e-mail.

From: This is what is shown to the receiver as the sender of the e-mail and the address that is used when using Reply-to

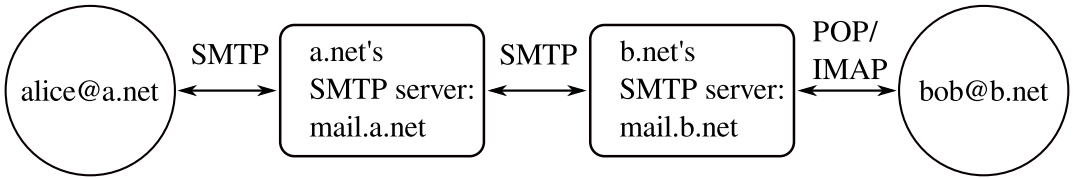
Sender spoofing

- The server MTA at `b.net` receives an `SMTP` connection by the client MTA at `a.net`. The client MTA is identified by the `HELO` command.
- However, in the connection the client can use different domains in the `HELO`, `MAIL FROM` or `From` fields
- A single `SMTP` client can legitimately send e-mail using `From` with several different domains.



Sender spoofing

- For instance, when you register a domain name, often the provider gives you also a set of e-mail addresses you can use.
- The MTA of the provider sends e-mails with the `From` field using your domain, and many other domains registered to the same provider.
- Then, the MTA at `b.net` can not drop an e-mail because the `From:` field does not match the domain used in the `HELO`



e-mail spoofing

- This is normally used by spammers/scammers to trick their victims, because an e-mail is more credible if it comes from a known domain
- Spam/scam e-mail in fact do not normally ask to answer to the e-mail, but to click on some link contained in the body of the e-mail
- There are several techniques to verify the `HELO`, `MAIL FROM`, `From` values.
- Let's start from `HELO` and `MAIL FROM`

Fighting SPAM

↳1.4 Sender Policy Framework

Fraudolent spammers

- Let's consider a (non-existing) subdomain of `unive.it`, for instance, `computer-networks.dais.unive.it`
- This domain can be delegated to a sub-part of an organization, for instance the professor of the computer networks course.
- That professor uses it to provide material to the students, however, he is lazy and does not update Wordpress. . . ;-)
- Some attacker takes over the server exploiting a vulnerability, he becomes *root*, the system administrator

Fraudolent spammers

- He can set-up an MTA: he installs all the needed software
- The MTA is legit, because it has a public IP and a FQDN
- plus, `unive.it` is a reputable organization, its IPs are not part of blacklists.
- The spammer now uses the fraudolent MTA to send spam to all the world.
- SPF tries to avoid this situation

SPF protects HELO and MAIL FROM

- SPF is a protocol that prevents a spammer to use a HELO or a MAIL FROM command using a domain that he doesn't own
- It works in conjunction with the DNS protocol
- It does not protect the From: field

SPF DNS entries

- An SPF entry is an entry associated to the domain name of one organization that specifies what are the IP addresses that are allowed to send e-mail
- It is a TXT entry. Recall TXT entries do not have a format, you can write anything on them
- When using SPF the entry has a format similar to this:

```
unive.it.  86400 IN TXT "v=spf1 ip4:17.18.7.120 -all"
```

- This means:
  - version of the protocol is version 1
  - one IP address is explicitly allowed to send e-mails
  - all others are forbidden

SPF operation

- When the MTA at `b.com` receives a connection by the MTA at `a.com`, it will do the following:
  - make a DNS resolution for the domain specified in the `HELO` and `MAIL FROM` address
  - verify if an `SPF` record exists for these domains
  - check that the IP address that the connection is coming from is allowed by the `SPF` syntax
- It will then take some decision (it could strictly ban the e-mail or apply some policy to it)

A base assumption

The base assumption of SPF and also the next methods we look at, is that an attacker that can take control of some infrastructure within an organization can not modify the DNS records. This is one more reason why the DNS server of an organization is extremely critical under the security point of view.

SPF Example

Let's test `mail.ietf.org`, that is found in the e-mail header (used with `HELO`)

```
$ dig mail.ietf.org TXT
;; AUTHORITY SECTION:
ietf.org.   1800  IN  SOA jill.ns.cloudflare.com. dns.cloudflare.com. 2356136896 10000 2400 604800 1800
```

No luck, let's test the `MAIL From`. It is not reported in the header, but most likely it is `ietf.org`

```
$ dig ietf.org TXT
;; ANSWER SECTION:
ietf.org.   300 IN  TXT "v=spf1 ip4:50.223.129.192/26 ip6:2001:559:c4c7::/48
        a:ietf.org mx:mail.ietf.org ip4:192.95.54.32/27 ip6:2607:5300:60:9ccf::/64
        ip4:54.240.73.154/31
        include:_spf.google.com include:spf.hostedrt.com include:amazonses.com ~all"

ietf.org.   300 IN  TXT "vs58md9pf8hu6knlglfda9lk6g"
ietf.org.   300 IN  TXT "ca3-5567e36d3f9947308ac2892e009840cc"
```

`50.233.192.192/26` is allowed. This is a syntax that means any address between `50.233.192.192`-`50.233.192.254`, including the one we resolved before.

SPF record syntax (essentials)

- `ipv4/ipv6`: follows an IP that can send e-mail, which means it can behave like a MTA using `HELO` or `MAIL FROM` with the `unive.it` domain
- `a`: same, but with a domain name. The sender MTA will resolve the domain and check the IP.
- `mx`: same, but with a MX record. The sender MTA will resolve the domain with an MX record and check the IP.
- `include`: include also the `SPF` record of some other domain, for instance `include: _spf.google.com` means that some of the e-mail will be sent on behalf of `unive.it` by google using gmail.
- `all` Everything. Used as the last token in the record to set a default
- qualifiers: "+", "-","∼", "?". Pass, fail, softfail, neutral. No qualifier means pass

Decision Delegation

- If the server MTA finds an SPF rule that ends with "`-all`", and the client MTA does not match the rule, the server will drop the e-mail
- In this sense, it is the sender domain that decides, not the server MTA
- If instead the rule ends with "`?all`" the sender domain does not give an explicit rule, the server MTA will decide on its own.

Back to the e-mail header

```
1  Delivered-To: leonardo.maccari@unive.it
2  Received: by 2002:a9a:69c5:0:b0:299:9b06:7765 with SMTP id c5csp1571665lkg;
3  Wed, 6 Nov 2024 01:21:37 -0800 (PST)
4  Received: from mail.ietf.org (mail.ietf.org. [50.223.129.194]) by mx.google.com ...
5  for <leonardo.maccari@unive.it> (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
6  Wed, 06 Nov 2024 01:21:37 -0800 (PST)
7  Received-SPF: pass (google.com: domain of forwardingalgorithm@ietf.org designates 50.223.129.194 as
       permitted sender) client-ip=50.223.129.194;
8  ...
9  Received: from ietfa.amsl.com (localhost [IPv6:::1]) by ietfa.amsl.com (Postfix) with ESMTP id 1895
       DC14F6A9 for <leonardo.maccari@unive.it>; Wed,
10 6 Nov 2024 01:21:35 -0800 (PST)
11 DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
       gKKOhIEVnQtATJMEQQoY5UY13/bV+mlSqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
       :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
       WyjSYJ1FL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
       OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
12 Received: from [192.168.178.37] (77-172-38-251.fixed.kpn.net [77.172.38.251]) by smtp.xs4all.nl (Halon)
       with ESMTPSA id 7bf838ad-9c20-11ef-891f-00505699d6e5; Wed, 06 Nov 2024 10:21:20 +0100 (CET)
13 Message-ID: <5eeeb3e9-942f-4716-90ed-b6b7fbf2bc6d@xs4all.nl>
14 Date: Wed, 6 Nov 2024 10:21:19 +0100
15 MIME-Version: 1.0
16 User-Agent: Mozilla Thunderbird
17 To: gaia <gaia@irtf.org>
18 Content-Language: en-US
19 From: Vesna Manojlovic <becha@xs4all.nl>
20 ...
```

Focus on line 7

Back to the e-mail header

```
1 Delivered-To: leonardo.maccari@unive.it
2 Received: by 2002:a9a:69c5:0:b0:299:9b06:7765 with SMTP id c5csp15716651kg;
3 Wed, 6 Nov 2024 01:21:37 -0800 (PST)
4 Received: from mail.ietf.org (mail.ietf.org. [50.223.129.194]) by mx.google.com ...
5 for <leonardo.maccari@unive.it> (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
6 Wed, 06 Nov 2024 01:21:37 -0800 (PST)
7 Received-SPF: pass (google.com: domain of forwardingalgorithm@ietf.org designates 50.223.129.194 as
      permitted sender) client-ip=50.223.129.194;
8   ...
```

This header was added by google MTA after checking the SPF rule of ietf.org. The MTA controlled that the client IP matched one of the IPs that is allowed by the SPF record.

Exercise

- that was not the real SPF record of `unive.it`, find it out and use the content in the extra slide to interpret it.
- Write an SPF rule that states that only the IPs `1.2.3.4` and `1.2.3.5`, and anything that is resolved by `example.com` is strictly allowed to send e-mail for a certain domain.

Fighting SPAM

↳1.5 DomainKeys Identified Mail (DKIM) Signatures

DKIM (RFC 6376)

- We are now able to identify an MTA by its FQDN, and associate the IP with the domain that can be used in the `HELO` command.
- We need a way to verify the other fields of the e-mail format.
- Recall that the e-mail fields are generated by the MUA, and are only forwarded by the MTAs.
- The only entity that can verify that these fields are legit is the MSA.
- In fact, the MSA authenticates the user, and can check the `From` field, and sign the whole e-mail.

DKIM (RFC 6376)

- DKIM is another protocol that focuses on authenticating the association between some of the e-mail headers and body, and one valid MTA
- The way DKIM works is similar to SPF, but with cryptography in the loop
- DKIM requires an MSA to own a public/private key pair, that the administrator must generate and configure in the MSA.

DKIM principles

- When the MUA contacts the MSA it authenticates
- This is the only moment in which the e-mail could be somehow *certified*, as the next MTA will not be able to ask for any authentication mechanism
- When using DKIM the MSA will select some fields of the e-mail, potentially including important headers (From:) and the body, and make a digital signature using its private key.
- The digital signature is applied to a hash, and it is included in the header fields of the message.
- The next MTA that receives the e-mail will be able to check the signature, assuming it knows the public key of the signing MTA.

DKIM header main fields

The DKIM header is made of several fields:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
    gKKOhIEVnQtATJMEQQoY5UY13/bV+mlSqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
    :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
    WyjSYJlFL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
    OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
```

- v: Version;

DKIM header main fields

The DKIM header is made of several fields:

```
1 DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
      gKKOhIEVnQtATJMEQQoY5UY13/bV+ml5qzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
      :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
      WyjSYJlFL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
      OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
```

- v: Version;
- a: Crypto algorithm used

DKIM header main fields

The DKIM header is made of several fields:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
    gKKOhIEVnQtATJMEQQoY5UY13/bV+mlSqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
    :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
    WyjSYJlFL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
    OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
```

- `v`: Version;
- `a`: Crypto algorithm used
- `h`: signed headers. Among them: `From`, `To`, `Subject`

DKIM header main fields

The DKIM header is made of several fields:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
    gKKOhIEVnQtATJMEQQoY5UY13/bV+mlSqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
    :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
    WyjSYJlFL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
    OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
```

- `v`: Version;
- `a`: Crypto algorithm used
- `h`: signed headers. Among them: `From, To, Subject`
- `bh`: body hash. The (in this case) `SHA256` hash of the whole body field

DKIM header main fields

The DKIM header is made of several fields:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
    gKKOhIEVnQtATJMEQQoY5UY13/bV+m1SqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
    :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
    WyjSYJ1FL9S28eiCiR/ybDhCL1BAoYzYhM1LaDQUPo4jYp6FwU82cSjaj7/
    OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
```

- v: Version;
- a: Crypto algorithm used
- h: signed headers. Among them: From, To, Subject
- bh: body hash. The (in this case) SHA256 hash of the whole body field
- b: signature. The (in this case) RSA signature of headers and body hash

DKIM header main fields

The DKIM header is made of several fields:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1; t=1730884895; bh=
    gKKOhIEVnQtATJMEQQoY5UY13/bV+mlSqzFkKJnzquw=; h=Date:To:From:Subject:List-Id:List-Archive:List-Help
    :List-Owner:List-Post:List-Subscribe:List-Unsubscribe; b=ILdywWIjDUwI3JoXDLHPdhOBf02HPj9+
    WyjSYJlFL9S28eiCiR/ybDhCL1BAoYzYhMlLaDQUPo4jYp6FwU82cSjaj7/
    OIr6JcS0wVcCYLJQaLhRfRUkEA7i9xilpNuBEIyax9Ltt1YHZNvXuZ9m8Bfuapxs1UKL3tW0luG8MU6E=
```

- `v`: Version;
- `a`: Crypto algorithm used
- `h`: signed headers. Among them: `From, To, Subject`
- `bh`: body hash. The (in this case) `SHA256` hash of the whole body field
- `b`: signature. The (in this case) `RSA` signature of headers and body hash
- `d,s`: domain and selector. Needed to retrieve the public key.

Key Publication

- The administrator of the signing MTA will have to publish the public key, again among the DNS information
- This will be included in another `TXT` record
- The record **must** be published under the domain: `selector._domainkey.domain` (`_domainkey` is a fixed string).
- For instance in the previous example it will be: `ietf1._domainkey.ietf.org`
- The receiving MTA will make a DNS query and retrieve the key, that can be used to verify the signature

DKIM: retrieving the Key

```
$ dig ietf1._domainkey.ietf.org TXT

;; ANSWER SECTION:
ietf1._domainkey.ietf.org. 300  IN  TXT ‘‘k=rsa; p=
     MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDNzNnjKTd5cczd2CDzHflCZuv1tMWYwd7zE+deoJ6s/fXR7/
     n9ZIBnDS5egt7HAHjNjZrmjcoRlfSsNxRJvUQFyYvaU1BT1s8R+mkPgSOqZ4t9HqAVjiczn2B9+dbjdNN+S/
     zvSyMMuSCSJDKKAXhBpDeQTpeY7/UdP9s6ws0yjQIDAQAB’’
```

FQDN + DKIM + SPF: results

- We have a way to check that a MTA IP corresponds to a domain (with FQDN check)
- We have a protocol that limits the IP addresses allowed to send e-mail using a certain `Mail From` domain (SPF)
- We have a protocol that allows to verify that the `From` field is aligned with a certain domain (DKIM).
- We have all the components to verify the chain of requisites:

  Valid IP/domain $\rightarrow$ MTA allowed by SPF $\rightarrow$ `DKIM`-valid `From` for a certain domain
- We still need a way to match the domain with the From and take a decision

Fighting SPAM

↳1.6 DMARC

DMARC

- Let's assume that MTA at b.com receives an e-mail, what happens in these cases:
  - Both the `SPF` or `DKIM` check fail
  - The `SPF` policy is OK, but the `HELO` domain is different from the one used in `From`
  - the DKIM signature is OK, however, the `From` domain differs from the `Mail From` or the `DKIM d=` domain?

DMARC alignment

- This kind of check is called *alignment check*: the basic alignment check is that `From:` header matches at least one of:
  - `d=` in the `DKIM` signature.
  - `SPF` verified domain
- What happens if the alignment of an email received by `b.com` fails?
- Should the MTA drop the e-mail?
- should it quarantine it? (that means adding a new header that signals to the MUA that the e-mail is probably spam)
- should it report back to the original MSA that something is wrong?

DMARC (RFC 7489)

- Whatever decision should be consistent to all MTAs, we don't want the same e-mail to be delivered by some MTAs and dropped by some other
- We want the owner of the sender domain to decide for all receiving MTAs
- DMARC is a protocol that allows the domain owner to set a policy that all the other MTAs should respect.
- the domain owner can use another TXT record in the DNS that is accessible at `_dmarc.domain.com`

DMARC (RFC 7489)

- the syntax of the TXT record is similar to SPF, important fields are:
  - p (policy): it's the action when alignment fails: *none* (do nothing), *quarantine* (flag the e-mail), *reject* (drop it)
  - rua: report aggregate failures to this e-mail
- Example: `ietf.org` does not want the MTA to take any action, if not send aggregated feedback on failures

```
$ dig _dmarc.ietf.org TXT
;; ANSWER SECTION:
_dmarc.ietf.org.  300 IN  TXT "v=DMARC1; p=none; rua=mailto:dmarc_agg@vali.email,mailto:dmarc-report@ietf.org"

```

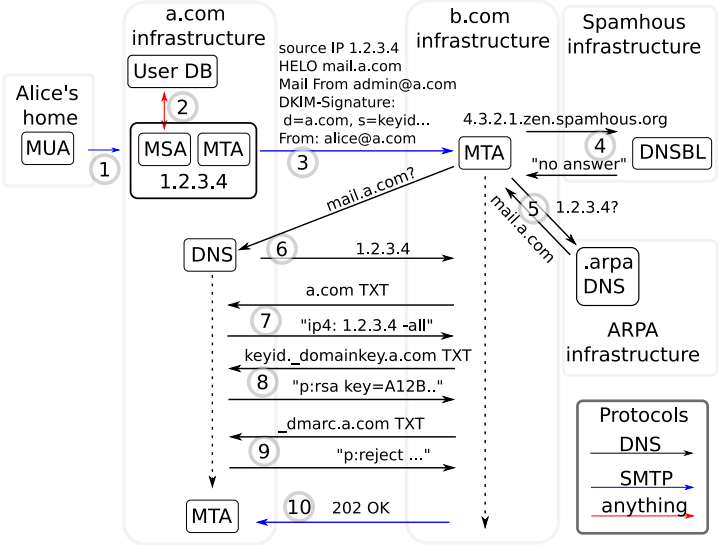
- Example: `whitehouse.gov` wants you to drop e-mail if they fail the check

```
$ dig _dmarc.whitehouse.gov TXT
;; ANSWER SECTION:
_dmarc.whitehouse.gov.  2012  IN  TXT "v=DMARC1; p=reject; rua=mailto:b1fabe8b7f3f41a181ecd1253a794edf@dmarc-reports.cloudflare.net,mailto:reports@dmarc.cyber.dhs.gov"

```

Sect. 2 Summary: the Big Picture

The Big Picture



Steps:

1. Alice uses `SMTP` with her local MSA, she wants to send an e-mail from alice@a.com to bob@b.com
2. The MSA authenticates Alice with some credentials
3. The MTA in a.com uses `SMTP` to deliver an e-mail, using appropriate `HELO`, `Mail From`, `From` fields and `DKIM-Signature`. The source IP is `1.2.3.4`.
4. `1.2.3.4` is tested against the Spamhaus DNSBL with a DNS query for `4.3.2.1.zen.spamhous.com` (or similar service)
5. The server MTA in b.com makes a reverse DNS query for the IP address of the client MTA

Steps:

6. The server MTA resolves `mail.a.com`, checks both resolutions are consistent
7. The server MTA queries `a.com` DNS for a `TXT` entry, checks that the SPF policy allows `1.2.3.4` to send e-mails on behalf of `a.com`
8. The server MTA queries `a.com` DNS for a `_domainkey` subdomain, receives the `DKIM` key, checks the e-mail signature
9. The server MTA queries `a.com` DNS for a `_dmarc` subdomain, checks what are the policies that must be respected.
10. If all policies are respected, the e-mail is accepted

Sect. 3 Extra Protocols

When you think it's over...

...you'r just scratching the surface. Some other protocols you will meet if you work in this field are:

- ARC
- BIMI (not enough space to mention it here)

Multiple Signatures: ARC

- When you are part of a Mailing List it is common that the mailing list MTA software will modify the e-mail, for instance, adding a subject tag or a signature
- This breaks the signature
- It is then allowed to have more than one signature in the headers, added by the MTAs that modified the message
- This requires another protocol (ARC RFC 8617) that we don't have time to look at.