Mail Server Instructions

Aaron M. Schinder
aschinder@amssolarempire.com
4 June 2020
Version 0.5

Introduction

E-mail is one of the most widely used communication modes of the internet. Many people use the mailservers provided by their internet hosting providers or other platforms such as gmail. However, in recent years, concentration of the users of the internet onto a few centralized platforms has lead to many problems. The internet was originally conceived as a network of peer computers talking to each other. (As opposed to clients of centralized servers that can only consume the provided services on the providers terms.) It is out of such a free and open environment that the central services we all rely on today initially grew.

I believe that it is important for people to be able to run their own servers, set up their own websites, and serve their own content. This will allow us to recover the promise of this original vision of a peer internet.

This tutorial (still in a very terse draft stage), covers how to set up an email server that avoids some security pitfalls (I am no means an expert, however, please let me know if there is a serious flaw). That also uses some authentication software (openDKIM, and domain records) which enable the server to authenticate itself to other widely used email servers (so you don’t end up in the spam trap.)

Instructions that work on Ubuntu 19.04, and hopefully Raspbian 9.

Note: This tutorial is a mashup of several tutorials which I read when teaching myself the subject. They’re credited in the “links to other tutorials” section. I assembled the pieces which worked for me on Ubuntu 19.04 and Raspbian, setting up Postfix/Dovecot servers. I have Dovecot configured to use virtual users (mailboxes set up according to a database instead of according to linux accounts.)

The Pieces

1. Certbot – Electronic Frontier Foundation’s SSL certificate issuing program. This verifies that you own the server in question and registers an SSL public key with EFF, and creates a public/private key pair on your server. This allows you to use SSL encryption (https) on your site, and is necessary for secure email.

2. Postfix – the MTA (Mail Transfer Agent) (handles communication with outside world)

3. Dovecot – the MDA (Mail Delivery Agent) (handles authentication and the mail store directories)

4. MySQL – database backend that Dovecot uses to manage mail accounts

5. OpenDKIM – a program that signs outgoing emails with an RSA key
6. DNS Records for the Domain Name

- SPF
- DMARC
- DKIM

## Installing the Pieces

### DNS Config

Example configuration from anexamplesite.com

<table>
<thead>
<tr>
<th>Type</th>
<th>Host</th>
<th>Value</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Record</td>
<td>@</td>
<td>76.78.219.25 (not real!)</td>
<td>Automatic</td>
</tr>
<tr>
<td>AAAA Record</td>
<td>@</td>
<td>2611:3d01::f03c:91ff:fe3a:7bfa</td>
<td>Automatic</td>
</tr>
<tr>
<td>CNAME Record</td>
<td>mail</td>
<td>anexamplesite.com</td>
<td>Automatic</td>
</tr>
<tr>
<td>CNAME Record</td>
<td>www</td>
<td>anexamplesite.com</td>
<td>Automatic</td>
</tr>
<tr>
<td>TXT Record</td>
<td>@</td>
<td>v=spf1 mx -all</td>
<td>Automatic</td>
</tr>
<tr>
<td>TXT Record</td>
<td>mail._domainkey</td>
<td>v=DKIM1; h=sha256; k=rsa; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKg==</td>
<td>60 min</td>
</tr>
<tr>
<td>TXT Record</td>
<td>_dmarch</td>
<td>v=DMARC1;p=reject; pct=100; rua=mailto:post <a href="mailto:master@anexamplesite.com">master@anexamplesite.com</a></td>
<td>60 min</td>
</tr>
<tr>
<td>MX Record</td>
<td>@</td>
<td>anexamplesite.com</td>
<td>0</td>
</tr>
</tbody>
</table>
This contains an MX record, an SPF record, a DMARC record, and the OpenDKIM key to verify mail is from this domain. (Mail will be signed with private key and confirmed with public key.)

These keys must be added to the domain records as each piece is created and tested.

Start by making sure the MX record is set up.

**Install Initial Pieces**

```bash
sudo apt-install certbot
sudo apt-get install postfix postfix-mysql dovecot-core dovecot-imapd dovecot-pop3d dovecot-lmtpd dovecot-mysql mysql-server
run mysql_secure_installation
```

*Note: mysql has recently “upgraded” in a way that breaks a lot of encryption utilities used by Dovecot. You may want to install mariadb instead.*

```bash
sudo apt-get install mariadb-server
run mysql_secure_installation
```

**Configure /etc/hosts**

```
76.78.219.25 anexamplesite.com
```

**Getting the SSL Certificates**

You will need an SSL certificate for Dovecot and Postfix.

```bash
sudo certbot certonly --standalone
```

Some options

```bash
sudo certbot certonly --cert-name example.com
```

```bash
sudo certbot --expand -d existing.com, example.com, newdomain.com – expands a certificate to include other domains.
```

Cancel a certificate

```bash
sudo certbot revoke --cert-path /etc/letsencrypr/live/CERTNAME/cert.pem
```

Set RSA key size

```bash
sudo certbot certonly --rsa-key-size 4096 -d a.example.com
```

Test automatic renewal

```bash
sudo certbot renew --dry-run
```

To renew a certificate, type certbot renew
certbot certonly --rsa-key-size 4096 -d anexamplesite.com -d mail.anexamplesite.com --cert-name anexamplesite.com

Spin up a temporary webserver to verify (option 1). Make sure apache2 is off during this process, or certbot won’t be able to bind to port 80.

Location of the generated certificates:

/etc/letsencrypt/live/anexamplesite.com

fullchain.pem – the cert file used in most server software.

Create Mail User

Show current users:
select host,user,password from mysql.user;
create user ‘testuser’@’localhost’ identified by ‘password’;
drop user ‘testuser’@’localhost’ identified by ‘password’;

---

Create MySQL Mail User

sudo mysql_secure_installation

Remove anonymous users

Disallow root login remotely

Remove test database and access to it

Reload privileges table

Create the mail server database

sudo mysqladmin -u root -p create maildb

Create a mail user, and replace

CREATE USER ‘mailguy’@’localhost’ IDENTIFIED BY ‘MmMm11111’

GRANT SELECT ON maildb.* TO ‘mailguy’@’localhost’;

FLUSH PRIVILEGES;
USE maildb;

Now create a bunch of tables that will be used by Dovecot to route messages and authenticate mailbox users.

Create a table for the domains that will receive mail on the Mailserver:
CREATE TABLE `virtual_domains` (  
`id` int(11) NOT NULL auto_increment,  
`name` varchar(50) NOT NULL,  
PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Create a table for all of the email addresses and passwords:
CREATE TABLE `virtual_users` (  
`id` int(11) NOT NULL auto_increment,  
`domain_id` int(11) NOT NULL,  
`password` varchar(106) NOT NULL,  
`email` varchar(100) NOT NULL,  
PRIMARY KEY (`id`),  
UNIQUE KEY `email` (`email`),  
FOREIGN KEY (domain_id) REFERENCES virtual_domains(id) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Create a table for the email aliases:
CREATE TABLE `virtual_aliases` (  
`id` int(11) NOT NULL auto_increment,  
`domain_id` int(11) NOT NULL,  
`source` varchar(100) NOT NULL,  
`destination` varchar(100) NOT NULL,  
PRIMARY KEY (`id`),  
FOREIGN KEY (domain_id) REFERENCES virtual_domains(id) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Add the domains to the `virtual_domains` table. Replace the values for example.com and hostname with your own settings:
INSERT INTO `maildb`.`virtual_domains`  
(`id` ,`name`)  
VALUES  
('1', 'anexamplesite.com'),  
('2', 'mail.anexamplesite.com');

Adding Email Users

Note
Note which id corresponds to which domain, the id value is necessary for the next two steps.
Add email addresses to the `virtual_users` table. The domain_id value references the
**virtual_domain** table’s id value. Replace the email address values with the addresses that you wish to configure on the mailserver. Replace the password values with strong passwords.

```sql
INSERT INTO `mailserver`.`virtual_users`
  (`id`, `domain_id`, `password`, `email`) VALUES
  ('1', '1', ENCRYPT('password', CONCAT('$6$', SUBSTRING(SHA(RAND())), -16)), 'email1@example.com'),
  ('2', '1', ENCRYPT('password', CONCAT('$6$', SUBSTRING(SHA(RAND())), -16)), 'email2@example.com');

INSERT INTO `maildb`.`virtual_users`
  (`id`, `domain_id`, `password`, `email`) VALUES
  ('1', '1', ENCRYPT('MyMailPassword', CONCAT('$6$', SUBSTRING(SHA(RAND())), -16)), 'user1@anexamplesite.com');

INSERT INTO `maildb`.`virtual_users`
  (`id`, `domain_id`, `password`, `email`) VALUES
  ('2', '1', ENCRYPT('PostmasterPassword', CONCAT('$6$', SUBSTRING(SHA(RAND())), -16)), 'postmaster@anexamplesite.com');
```

Note: If you try to use Mysql 8, this doesn’t work. They’ve “helpfully” deprecated encrypt. Instead you must do the following:

```sql
INSERT INTO `maildb`.`virtual_users`
  (`id`, `domain_id`, `password`, `email`) VALUES
  ('1', '1', TO_BASE64(UNHEX(SHA2('MyMailPassword',512))), 'user1@anexamplesite.com');
```

Then, in `dovecot-sql.conf.ext`
you must use:

```plaintext
default_pass_scheme = SHA512
```

instead of

```plaintext
default_pass_scheme = SHA512-crypt
```

(infrastructure maintainers: let’s break even more things people have come to depend on! Yaay!)

An email alias will forward all email from one email address to another. To set up an email alias, add it to the `virtual_aliases` table:

```sql
INSERT INTO `mailserver`.`virtual_aliases`
  (`id`, `domain_id`, `source`, `destination`) VALUES
  ('1', '1', 'alias@example.com', 'email1@example.com');
```
Configure Postfix

Files to Edit:
1. main.cf
2. master.cf

Main.cf

```plain
# See /usr/share/postfix/main.cf.dist for a commented, more complete version

# Debian specific: Specifying a file name will cause the first
# line of that file to be used as the name. The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# TLS parameters
smtpd_tls_cert_file=/etc/letsencrypt/live/www.anexamplesite.com/fullchain.pem   #loc of fullchain.pem
smtpd_tls_key_file=/etc/letsencrypt/live/www.anexamplesite.com/privkey.pem
smtpd_use_tls=yes
smtpd_tls_auth_only = yes
smtpd_tls_security_level = may
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous

# Authentication
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

# Restrictions
smtpd_helo_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_invalid_helo_hostname,
    reject_non_fqdn_helo_hostname

smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_recipient,
    reject_unknown_recipient_domain,
    reject_unlisted_recipient,
    reject_unauth_destination

smtpd_sender_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_sender,
    reject_unknown_sender_domain

smtpd_relay_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
```
The main.cf file declares the location of virtual_mailbox_domains, virtual_mailbox_maps, and virtual_alias_maps files. These files contain the connection information for the MySQL lookup tables created in the MySQL section of this guide. Postfix will use this data to identify all domains, corresponding mailboxes, and valid users.

/etc/postfix/mysql-virtual-mailbox-domains.cf
user = mailguy
password = MmMm1111
hosts = 127.0.0.1
dbname = maildb
query = SELECT 1 FROM virtualDomains WHERE name='%s'

/etc/postfix/mysql-virtual-mailbox-maps.cf

user = mailguy
password = MmMm1111
hosts = 127.0.0.1
dbname = maildb
query = SELECT 1 FROM virtualUsers WHERE email='%s'

/etc/postfix/mysql-virtual-alias-maps.cf

user = mailguy
password = MmMm1111
hosts = 127.0.0.1
dbname = maildb
query = SELECT destination FROM virtualAliases WHERE source = '%s'

/etc/postfix/mysql-virtual-email2email.cf

user = mailguy
password = MmMm1111
hosts = 127.0.0.1
dbname = maildb
query = SELECT email FROM virtualUsers WHERE email = '%s'

sudo systemctl restart postfix
(or sudo service postfix restart?)

The following commands test that postfix sees the email addresses and domains. Should return 1.
sudo postmap -q amscyberpunkatlanta.com mysql:/etc/postfix/mysql-virtual-mailbox-domains.cf
sudo postmap -q aschinde@amscyberpunkatlanta.com mysql:/etc/postfix/mysql-virtual-mailbox-maps.cf

Master.cf

# Postfix master process configuration file. For details on the format
# of the file, see the master(5) manual page (command: "man 5 master" or
# on-line: http://www.postfix.org/master.5.html).
#
# Do not forget to execute "postfix reload" after editing this file.
#
# service type  private unpriv chroot  wakeup  maxproc command + args

#               (yes)   (yes)   (no)    (never) (100)
# ==================================================================
smtp      inet  n       -       y       -       -       smtpd
#smtp      inet  n       -       y       -       1       postscreen
#smtpd     pass  -       -       y       -       -       smtpd
dnsblog   unix  -       -       y       -       0       dnsblog
tlsproxy  unix  -       -       y       -       0       tlsproxy
submission inet n       y       -       -       smtpd
    -o syslog_name= postfix/submission
    -o smtpd_sasl_auth_enable=yes
    -o smtpd_sasl_type=dovecot
    -o smtpd_sasl_path=private/auth
    -o smtpd_reject_unlisted_recipient=no
    -o smtpd_client_restrictions=permit_sasl_authenticated,reject
    -o milter_macro_daemon_name=ORIGINATING
smtps     inet  n       -       y       -       -       smtpd
    -o syslog_name= postfix/smtps
    -o smtpd_sasl_auth_enable=yes
    -o smtpd_sasl_type=dovecot
    -o smtpd_sasl_path=private/auth
    -o smtpd_client_restrictions=permit_sasl_authenticated,reject
    -o milter_macro_daemon_name=ORIGINATING
#628       inet  n       -       y       -       -       qmqpd
pickup    unix  n       -       y       60      1       pickup
cleanup   unix  n       -       y       -       0       cleanup
qmgr      unix  n       -       n       300     1       qmgr
#qmgr     unix  n       -       n       300     1       qmgr
#tlsmgr    unix  -       -       y       1000?    1       tlsmgr
rewrite   unix  -       -       y       -       -       trivial-rewrite
bounce    unix  -       -       y       -       0       bounce
deferr     unix  -       -       y       -       0       bounce
trace     unix  -       -       y       -       0       bounce
verify     unix  -       -       y       -       1       verify
#628       inet  n       -       y       -       -       qmqpd
#628       inet  n       -       y       -       -       qmqpd
#628       inet  n       -       y       -       -       qmqpd
#628       inet  n       -       y       -       -       qmqpd
#628       inet  n       -       y       -       -       qmqpd
showq     unix  n       -       y       -       -       showq
error     unix  -       -       y       -       -       error
retry     unix  -       -       y       -       -       error
discard   unix  -       -       y       -       -       discard
local     unix  -       n       n       -       -       local

interfaces to non-Postfix software. Be sure to examine the manual
# pages of the non-Postfix software to find out what options it wants.
# Many of the following services use the Postfix pipe(8) delivery
# agent. See the pipe(8) man page for information about ${recipient}
# and other message envelope options.
# maildrop. See the Postfix MAILDROP_README file for details.
# Also specify in main.cf: maildrop_destination_recipient_limit=1
maildrop  unix  -       n       n       -       -       pipe
flags=DRhu user=vmail argv=/usr/bin/maildrop -d ${recipient}
#
# Recent Cyrus versions can use the existing "lmtp" master.cf entry.
# Specify in cyrus.conf:
#   lmtp    cmd="lmtpd -a" listen="localhost:lmtp" proto=tcp4
# Specify in main.cf one or more of the following:
# mailbox_transport = lmtp:inet:localhost
# virtual_transport = lmtp:inet:localhost
#
# Cyrus 2.1.5 (Amos Gouaux)
# Also specify in main.cf: cyrus_destination_recipient_limit=1
# cyrus     unix  -       n       n       -       -       pipe
# user=cyrus argv=/cyrus/bin/deliver -e -r ${sender} -m ${extension} ${user}
#
# Old example of delivery via Cyrus.
# old-cyrus unix  -       n       n       -       -       pipe
# flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
#
# See the Postfix UUCP_README file for configuration details.
# uucp      unix  -       n       n       -       -       pipe
# flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
#
# Other external delivery methods.
# ifmail    unix  -       n       n       -       -       pipe
# flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp     unix  -       n       n       -       -       pipe
flags=Fq. user=bsmtp argv=/usr/lib/bsmtp/bsmtp -t$nexthop -f$sender $recipient
scalemail-backend unix  -       n       n       -       2       pipe
Clean Up Permissions

chmod -R o-rwx /etc/postfix

sudo systemctl restart postfix

Configure Dovecot

We will be editing several config files: Back them up

```
sudo cp /etc/dovecot/dovecot.conf /etc/dovecot/dovecot.conf.orig
sudo cp /etc/dovecot/conf.d/10-mail.conf /etc/dovecot/conf.d/10-mail.conf.orig
sudo cp /etc/dovecot/conf.d/10-auth.conf /etc/dovecot/conf.d/10-auth.conf.orig
sudo cp /etc/dovecot/dovecot-sql.conf.ext /etc/dovecot/dovecot-sql.conf.ext.orig
sudo cp /etc/dovecot/conf.d/10-master.conf /etc/dovecot/conf.d/10-master.conf.orig
sudo cp /etc/dovecot/conf.d/10-ssl.conf /etc/dovecot/conf.d/10-ssl.conf.orig
```

Edit the `/etc/dovecot/dovecot.conf` file. Add `protocols = imap pop3 lmtp` to the `# Enable installed protocols` section of the file:

```
## Dovecot configuration file
...
# Enable installed protocols
!include_try /usr/share/dovecot/protocols.d/*.protocol
protocols = imap pop3 lmtp
...
postmaster_address=postmaster@example.com
```

Edit the `/etc/dovecot/conf.d/10-mail.conf` file. This file controls how Dovecot interacts with the server’s file system to store and retrieve messages:

Modify the following variables within the configuration file

```
... mail_location = maildir:/var/mail/vhosts/%d/%n/ ...
mail_privileged_group = mail ...
```

Create the `/var/mail/vhosts/` directory and a subdirectory for your domain. Replace `example.com` with your domain name:

```
sudo mkdir -p /var/mail/vhosts/example.com
```
Create the vmail group with ID 5000. Add a new user vmail to the vmail group. This system user will read mail from the server.

```
sudo groupadd -g 5000 vmail
sudo useradd -g vmail -u 5000 vmail -d /var/mail
```

Change the owner of the /var/mail/ folder and its contents to belong to vmail:

```
sudo chown -R vmail:vmail /var/mail
```

**Digression: user and group stuff in linux**

- `useradd -m -g accounting user2` (create user2 in group accounting)
- `usermod -a -G sudo geek` (add geek to sudo group (or rather, add the sudo group to user geek)).
- `usermod -g groupname username` (change a user’s primary group)

Note the `-g` here. When you use a lowercase `g`, you assign a primary group. When you use an uppercase `-G`, as above, you assign a new secondary group.

To view the groups the current user account is assigned to, run the `groups` command. You’ll see a list of groups.

```
groups
id
```

Edit the user authentication file, located in `/etc/dovecot/conf.d/10-auth.conf`.

Uncomment the following variables and replace with the file excerpt’s example values:

```
10-auth.conf
...
disable_plaintext_auth = yes
...  
auth_mechanisms = plain login
...  
!include auth-system.conf.ext
...  
!include auth-sql.conf.ext
...
```

Edit the `/etc/dovecot/conf.d/auth-sql.conf.ext` file with authentication and storage information. Ensure your file contains the following lines. Make sure the `passdb` section is uncommented, that the `userdb` section that uses the `static` driver is uncommented and update with the right argument, and comment out the `userdb` section that uses the `sql` driver:

```
auth-sql.conf.ext
...
passdb {
    driver = sql
    args = /etc/dovecot/dovecot-sql.conf.ext
}
...  
#userdb {
  #  driver = sql
```
Update the `/etc/dovecot/dovecot-sql.conf.ext` file with your MySQL connection information. Uncomment the following variables and replace the values with the excerpt example. Replace `dbname`, `user` and `password` with your own MySQL database values:

```
dovecot-sql.conf.ext

... driver = mysql
... connect = host=127.0.0.1 dbname=maildb user=mailguy password=MuMu$%11235
... default_pass_scheme = SHA512-CRYPT
... password_query = SELECT email as user, password FROM virtual_users WHERE email='%u';
...
```

The `password_query` variable uses email addresses listed in the `virtual_users` table as the username credential for an email account.

To use an alias as the username:

1. Add the alias as the source and destination email address to the `virtual_aliases` table.
2. Change the `/etc/dovecot/dovecot-sql.conf.ext` file's `password_query` value to:
   ```
   password_query = SELECT email as user, password FROM virtual_users WHERE email=(SELECT destination FROM virtual_aliases WHERE source = '%u');
   ```

**Note**
For reference, view a complete `dovecot-sql.conf.ext` file.

- Change the owner and group of the `/etc/dovecot/` directory to `vmail` and `dovecot`:
  ```
  sudo chown -R vmail:dovecot /etc/dovecot
  ```

- Change the permissions on the `/etc/dovecot/` directory to be recursively read, write, and execute for the owner of the directory:
  ```
  sudo chmod -R o-rwx /etc/dovecot
  ```

- Edit the service settings file `/etc/dovecot/conf.d/10-master.conf:`
Note

When editing the file, be careful not to remove any opening or closing curly braces. If there’s a syntax error, Dovecot will crash silently. You can check /var/log/upstart/dovecot.log to debug the error.

Here is an example of a complete 10-master.conf file.

Disable unencrypted IMAP and POP3 by setting the protocols’ ports to 0. Uncomment the port and ssl variables:

10-master.conf

```conf
... service imap-login {
  inet_listener imap {
    port = 0
  }
  inet_listener imaps {
    port = 993
    ssl = yes
  }
}...
... service pop3-login {
  inet_listener pop3 {
    port = 0
  }
  inet_listener pop3s {
    port = 995
    ssl = yes
  }
}...
```

Find the service lmtp section of the file and use the configuration shown below:

10-master.conf

```conf
... service lmtp {
  unix_listener /var/spool/postfix/private/dovecot-lmtp {
    #mode = 0666i
    mode = 0600
    user = postfix
    group = postfix
  }
}...
```
Locate `service auth` and configure it as shown below:

```
10-master.conf
...
service auth {
    ...
    unix_listener /var/spool/postfix/private/auth {
        mode = 0660
        user = postfix
        group = postfix
    }
    unix_listener auth-userdb {
        mode = 0600
        user = vmail
    }
    ...
    user = dovecot
}
...
```

In the `service auth-worker` section, uncomment the `user` line and set it to `vmail`:

```
10-master.conf
...
service auth-worker {
    ...
    user = vmail
}
```

Save the changes to the `/etc/dovecot/conf.d/10-master.conf` file.

- Edit `/etc/dovecot/conf.d/10-ssl.conf` file to require SSL and to add the location of your domain’s SSL certificate and key. Replace `example.com` with your domain:

```
10-ssl.conf
...
# SSL/TLS support: yes, no, required. <doc/wiki/SSL.txt>
ssl = required
...
ssl_cert = </etc/letsencrypt/live/anexamplesite.com/fullchain.pem
ssl_key = </etc/letsencrypt/live/anexamplesite.com/privkey.pem
```

Restart Dovecot to enable all configurations:

```
sudo systemctl restart dovecot
```
Configure OpenDKIM

DKIM stands for “Domain Keys Identified Mail”. OpenDKIM signs outgoing email from your server with an RSA key. The public key is posted in the domain records of your server, and confirms that received email is actually coming from your server (as opposed to some other server that is spoofing your headers). Most mailservers on the internet are set up to reject mail that can’t authenticate it originates from the sender.


https://www.linuxbabe.com/mail-server/setting-up-dkim-and-spf

(This may be fragmented/incomplete. For this section, I recommend doublechecking with one of my other linked tutorials)

```
sudo apt install opendkim opendkim-tools
sudo nano /etc/opendkim.conf
```

This creates and opendkim user and group (if it doesn’t, create the user and group)

Set up OpenDKIM

```
/etc/opendkim.conf

# This is a basic configuration that can easily be adapted to suit a standard
# installation. For more advanced options, see opendkim.conf(5) and/or
# /usr/share/doc/opendkim/examples/opendkim.conf.sample.

# Log to syslog
Syslog                   yes
# Required to use local socket with MTAs that access the socket as a non-
# privileged user (e.g. Postfix)
UMask                   002

# Sign for example.com with key in /etc/mail/dkim.key using
# selector ’2007’ (e.g. 2007._domainkey.example.com)
#Domain                 example.com
#KeyFile                /etc/mail/dkim.key
#Selector               2007

# Commonly-used options; the commented-out versions show the defaults.
Canonicalization        relaxed/simple
Mode                    sv
SubDomains              no
#ADSPAction             continue
AutoRestart             yes
AutoRestartRate         10/1M
Background              yes
DNSTimeout              5
SignatureAlgorithm      rsa-sha256

# Always oversign From (sign using actual From and a null From to prevent
# malicious signatures header fields (From and/or others) between the signer
# and the verifier. From is oversigned by default in the Debian package
```
# because it is often the identity key used by reputation systems and thus somewhat security sensitive.

OversignHeaders From

# List domains to use for RFC 6541 DKIM Authorized Third-Party Signatures (ATPS) (experimental)

#ATPSDomains example.com

#OpenDKIM user
# Remember to add user postfix to group opendkim
UserID opendkim

# Map domains in From addresses to keys used to sign messages
KeyTable refile:/etc/opendkim/key.table
SigningTable refile:/etc/opendkim/signing.table

# Hosts to ignore when verifying signatures
ExternalIgnoreList /etc/opendkim/trusted.hosts

# A set of internal hosts whose mail should be signed
InternalHosts /etc/opendkim/trusted.hosts

---

**Generate OpenDKIM Directory**

```
sudo mkdir /etc/opendkim
sudo mkdir /etc/opendkim/keys
sudo chown -R opendkim:opendkim /etc/opendkim
sudo chmod go-rw /etc/opendkim/keys
sudo nano /etc/opendkim/signing.table
```

Add this line to the file. This tells OpenDKIM that if a sender on your server is using a `@your-domain.com` address, then it should be signed with the private key identified by `default._domainkey.your-domain.com`.

```
/etc/opendkim/signing.table
*@your-domain.com default._domainkey.your-domain.com
```

Save and close the file. Then create the key table.

```
/etc/opendkim/key.table
default._domainkey.your-domain.com your-domain.com:default:/etc/opendkim/keys/your-domain.com/default.private
```

Save and close the file. Next, create the trusted hosts file.

Add the following lines to the newly created file. This tells OpenDKIM that if an email is coming from localhost or from the same domain, then OpenDKIM should not perform DKIM verification on the
email.

<table>
<thead>
<tr>
<th>/etc/opendkim/trusted.hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>localhost</td>
</tr>
<tr>
<td>*.your-domain.com</td>
</tr>
</tbody>
</table>

Generate Private/Public Keypair

sudo mkdir /etc/opendkim/keys/

sudo opendkim-genkey -b 2048 -d your-domain.com -D /etc/opendkim/keys/your-domain.com -s default -v

The above command will create 2048 bits keys. -d (domain) specifies the domain. -D (directory) specifies the directory where the keys will be stored and we use default as the selector (-s), also known as the name. Once the command is executed, the private key will be written to default.private file and the public key will be written to default.txt file.

Make opendkim as the owner of the private key.

Sudo

sudo chown opendkim:opendkim /etc/opendkim/keys/your-domain.com/default.private

sudo cat /etc/opendkim/keys/your-domain.com/default.txt

The string after the p parameter is the public key.

In you DNS manager, create a TXT record, enter default._domainkey in the name field. Then go back to the terminal window, copy everything in the parentheses and paste it into the value field of the DNS record. You need to delete all double quotes and white spaces in the value field. If you don’t delete them, then key test in the next step will fail.

sudo opendkim-testkey -d your-domain.com -s default -vvv

If everything is OK, you will see

opendkim-testkey: using default configfile /etc/opendkim.conf
opendkim-testkey: checking key 'default._domainkey.your-domain.com'
opendkim-testkey: key secure
opendkim-testkey: key OK

If you see “Key not secure”, don’t panic. This is because DNSSEC isn’t enabled on your domain name. DNSSEC is a security standard for secure DNS query. Most domain names haven’t enabled DNSSEC. You can continue to follow this guide.

**Connect OpenDKIM to Postfix**

Postfix can talk to OpenDKIM via a Unix socket file. The default socket file used by OpenDKIM is /var/run/opendkim/opendkim.sock, as shown in /etc/opendkim.conf file. But the postfix SMTP daemon shipped with Ubuntu runs in a chroot jail, which means the SMTP daemon resolves all filenames relative to the Postfix queue directory (/var/spool/postfix). So we need
to change the OpenDKIM Unix socket file.

Create a directory to hold the OpenDKIM socket file and allow only opendkim user and postfix group to access it.

```
sudo mkdir /var/spool/postfix/opendkim
sudo chown opendkim:postfix /var/spool/postfix/opendkim
```

```
/etc/opendkim.conf
#Socket local:/var/run/opendkim/opendkim.sock
Socket local:/var/spool/postfix/opendkim/opendkim.sock
```

```
/etc/default/opendkim
#SOCKET="local:/var/run/opendkim/opendkim.sock"
#SOCKET=local:$RUNDIR/opendkim.sock
SOCKET="local:/var/spool/postfix/opendkim/opendkim.sock"
```

```
/etc/postfix/mail.conf
# Milter configuration
milter_default_action = accept
milter_protocol = 6
smtpd_milters = local:opendkim/opendkim.sock
non_smtpd_milters = $smtpd_milters
```

```
sudo systemctl restart opendkim postfix
```

**Configure SPF Record**

Add the following to your DNS records

```
TXT @ v=spf1 mx ~all
```

**Configure DMARC Record**

A DMARC record basically lets another mailserver know who to send automated complaints to if your mail fails some anti-spam test, or if there are spam or compromised emails coming from your server. This record tells the receiving server to reject all emails that don’t pass tests and to notify the provided email address.

```
TXT Record _dmarc v=DMARC1;p=reject; pct=100; rua=mailto:post master@anexamplesite.com 60 min
```
Links To Other Tutorials

My main tutorial:


Others:

Taking e-mail back - part 1
Taking e-mail back - part 2
Taking e-mail back - part 3
Taking e-mail back - part 4


https://www.linuxbabe.com/mail-server/setting-up-dkim-and-spf

Additional Admin Stuff

creating a new subfolder

doveadm mailbox create -u username@example.com -s INBOX.newfolder

doveadm mailbox create -u aschinde@example.com -s INBOX._Househunting2019