TOPAL

GPG/GNUPG AND ALPINE/PINE INTEGRATION

Phil Brooke

Release 82
2022-08-27T150430BST

Contents

Chapter 1

Introduction & features

Topal is a 'glue' program that links GnuPG and Pine/Alpine/"Re-Alpine" (from Sourceforge). It offers facilities to encrypt, decrypt, sign and verify emails, including inline OpenPGP, MIME/OpenPGP and S/MIME. It can also be used directly from the command-line.

1.1 Features

- Multiple inline PGP blocks can be processed in display filters.
- Decryption and verification output can be cached to reduce the number of times a passphrase is entered[1]. This also helps when secret keys aren't always available, at the expense of storing decrypted output.
- MIME/OpenPGP (RFC2015/RFC3156) multipart messages can be sent and received. Depending on configuration, this might involve procmail, using `sendmail-path` or patching Alpine.
- The deprecated application/pgp content-type can be sent and received.
- S/MIME messages can be sent and received if `gpgsm` is available. (`openssl` is also used in some circumstances, but `gpgsm` is still required.)
- Topal can be used as Alpine's `sendmail-path` command.
- Topal has a remote sending mode (a server and a means of accessing the server) for reading email on a distant computer via SSH with secret keys on the local computer.

[1] Note that you should also consider the use of `gpg-agent`.

- A range of mechanisms for selecting keys for both self and recipients.
- There is a high level of configurability (although the configuration interface does not expose all of it; you may have to edit `.topal/config`).

1.2 Terminology

We use Pine and Alpine interchangably in this manual. The earliest version of Pine that Topal supported was 4.44. When referring to Thunderbird and Enigmail together, we sometimes just say Thunderbird.

Similarly, we often use GnuPG, GPG and sometimes OpenPGP and PGP interchangably.

1.3 Version numbering

Recent stable releases have been numbered 55, 56, ... Prior to release 55, the stable releases were 0.7.2, 0.7.8, 0.7.9 and 0.7.13.6.

Chapter 2

Important changes from previous stable versions

2.1 Important changes in release 70

The `use-agent` option sets GPG's `--use-agent` option as needed. Don't set it in any other way or it might be confusing. . . .

2.2 Important changes in release 68

- The sending interface has changed. Selecting (for example) 'e' for encrypt no longer immediately operates. Instead, the desired operation is chosen, and then 'g' for 'Go!' is used. This allows defaults to be associated with keys and email addresses in a sensible way.

- The MIME viewer setting `mime-viewer` has been replaced by two new settings, one for decrypting and one for verifying. You should set these to suit your preferences. (The redundant line will be dropped the next time you save your configuration inside Topal.)

2.3 Important changes in release 65

An additional patch has been added. This should make the procmail recipe redundant. More testing is required: it may have broken other Alpine features.

2.4 Important changes in release 60

MIME sending now requires MIME-tool; mime-construct is no longer used. See the compilation and installation instructions.

2.5 Important changes in release 58

The default configuration no longer uses absolute paths.

2.6 Important changes in release 55

- If you use a non-English locale, please check that Topal still works as expected (I replaced code that fixed some locale problems).
- The Alpine patch is based off my old Pine patches, but does a little more. You will need to set the

  ```
  Enable Topal hack for OpenPGP/MIME messages
  ```

  option in the hidden configuration list. Bug reports welcome.
- The `--fix-email` wrapper no longer creates a multipart/alternative: it creates a multipart/misc wrapper instead. Please check that your procmail recipe includes a suitable backup in case this doesn't work for you.

2.7 Important changes in version 0.7.10

The recommended procmail recipe has been changed.

2.8 Important changes in version 0.7.8

`topal-fix-email` and `topal-fix-folder` have been replaced by the main topal binary. Change `topal-fix-email` in your .procmailrc to be `topal --fix-email`. (Or add symlinks: the binary checks what it has been called as.)

New in version 0.7.8.

You *must* clear your cache otherwise the changes made for `inline-separate-output` will break (this occurs regardless of whether the option is on or off). This new feature shows the GnuPG/Topal output separately, then hands back the decrypted or verified output without any wrappers.

This makes it more suitable for dealing with attachments (but you need to set it manually via `topal -config`).

Finally, the send menu has a new option: ‘Pass through unchanged’. This does nothing to the message so, you can always have Topal invoked as a filter for sending.

Chapter 3

Installation and configuration

3.1 Options

As of version 76, the two Alpine patches are deprecated.

Topal supports a range of options depending on your preferences; some have particular requirements: the following table summarises them.

	Sending			Receiving			
	Configure Alpine's `sending-filters`	Configure Alpine's `sendmail-path`	Apply patch `-1` to Alpine	Configure Alpine's `display-filters`	Pipe entire message to `topal -raw`	Configure `.procmail` or run `topal -fix-email`	Apply patch `-2` to Alpine and configure `.mailcap`
Encrypt/sign inline OpenPGP	★	★					
Decrypt/verify inline OpenPGP				★	★		
Encrypt/sign MIME/OpenPGP		★	↓				
Decrypt/verify MIME/OpenPGP					★	○	↓
Encrypt/sign S/MIME		★	↓				
Decrypt/verify S/MIME					★	○	↓

Key:

★	This is a preferred option, but there may be an alternative (○).
○	This is a possible option, but there is a better option (★).
↓	Deprecated.

Inline OpenPGP can be processed via Alpine's sending and display filters. So you need to configure this within Alpine.

MIME messages are more complicated. MIME/OpenPGP messages can be sent if the deprecated patch 1 is applied to Alpine. However, although the corresponding S/MIME messages appear to be correctly formed, use of patch 1 requires that the message is embedded within a multipart/mixed message — and other mail clients (such as Mozilla Thunderbird and MS Outlook) will not handle them properly. So for MIME sending, you are strongly recommended to use Topal as a `sendmail-path` for Alpine.

Receipt of MIME messages (both OpenPGP and S/MIME) is best handled via piping the whole message to `topal -raw`; make sure you set `Raw text` (`^W`) and `Free output` (`^Y`). This also works for inline messages.

An alternative is to invoke Topal's `--fix-email` mode via `.procmail` which rewrites the top-level cryptographic message into a multipart/mixed message that can be more easily processed by Alpine.

The final option is to use Alpine with patch 2; this is deprecated. This will cause Alpine to crash if your `.mailcap` is not properly configured to handled the cryptography multipart/* types.

3.2 Compilation and installation

To compile Topal, you need

- a working C compiler,
- the GNU Ada Compiler (GNAT), and
- some common libraries, including `readline` (e.g., the package `libreadline6-dev` on Debian).

There is a makefile: simply type `make`.

My normal environment is Debian; I've put some code in `Buildflags.mk` to handle missing `dpkg-buildflags`, but it may not work. Bug reports and patches are welcome.

Type `make install` to actually install. The default location is `/usr`, so you'll need to be root to install. Alternatively, use `make install INSTALLPATH=/usr/local` to install into `/usr/local`. (Or use the more specific variables `INSTALLPATHBIN`, `INSTALLPATHMAN`, `INSTALLPATHDOC` and `INSTALLPATHPATCHES`.)

3.2.1 Cygwin

Before using the makefile, please apply the patch `cygwin.patch`. I haven't recently tested this. . . .

3.2.2 MIME-tool

MIME sending requires the Topal version of Jeffrey S. Dutky's `mime-tool`. This is included with the Topal sources, and is compiled and installed at the same time using the Makefile.

3.2.3 MIME viewing

MIME viewing can be handled via

1. metamail,
2. run-mailcap, or
3. by saving to a file (mbox format) in the `~/.topal` directory and viewing with Alpine.

If you are using patch 2 (see "Alpine patches" below), then depending on your Alpine configuration, Alpine might say something like

```
Attachment SIGNED unrecognized. Try opening anyway?
```

— you should say `yes` in that case.

3.2.4 GNU sed

GNU sed is required. I only discovered this when using Topal on Mac OS X. . . .

3.3 Alpine patches

There are two patches, for a range of versions of Pine and Alpine. Patch 1 deals with sending MIME messages (`-sendmime`); patch 2 with receiving them.

	Patch 1	Patch 2
Pine 4.44	✓	
Pine 4.50	✓	
Pine 4.53	✓	
Pine 4.58	✓	
Pine 4.60	✓	
Pine 4.64	✓	
Alpine 1.00	✓	
Alpine 2.00	✓	✓
Alpine 2.02	✓	✓

To apply these patches, `cd` into the Pine or Alpine source directory and use the `patch` command, *e.g.*, `patch -p1 < -2.02.patch-1`.

Please note that the Alpine patches also modify Alpine's configuration. There is a hidden preference 'enable Topal hack' (`enable-topal-hack`) that you need to enable.

3.4 Alpine filter configuration

Assuming that the topal binary is installed in `/usr/bin`, set up the Alpine sending and display filters as follows:

```
display-filters=_BEGINNING("-----BEGIN PGP ")_ /usr/bin/topal
                  -display _TMPFILE_ _RESULTFILE_

sending-filters=/usr/bin/topal --read-from _INCLUDEALLHDRS_
                  -send _TMPFILE_ _RESULTFILE_ _RECIPIENTS_,
                /usr/bin/topal --read-from _INCLUDEALLHDRS_
                  -sendmime _TMPFILE_ _RESULTFILE_ _MIMETYPE_
                  _RECIPIENTS_
```

You can choose either or both of the sending filters. The `-sendmime` option allows the user to choose the MIME type of the outbound email. (Legacy fixes are in place that make `-decrypt` and `-verify` behave the same as `-display`.) Note that `_RECIPIENTS_` must be last.

The `--read-from _INCLUDEALLHDRS_` text makes Topal attempt to guess a suitable key for signing and self-encryption. If multiple possible keys match, then you'll be offered a menu of the keys. Use of this option is strongly recommended (but can be omitted).

3.5 Alpine sendmail-path configuration

As an alternative (or addition!) to setting filters (above), you can set

```
sendmail-path="/usr/bin/topal -asend"
```

Also see section 5.4 for more details on configuring this mode.

This option allows Topal to handle attachments added within Alpine, whereas sending using patch 1 requires that attachments are added via Topal's interface (otherwise the attachments aren't covered by encryption or signing).

3.6 Alpine fcc — warning

When sending encrypted messages, then —depending on how you invoke Topal— the message saved by Alpine into your file carbon copy folder may not be encrypted. This applies primarily to the `sendmail-path` usage, where from Alpine's viewpoint, the message is sent by handing it to Topal. In this mode, I set `fcc` to be empty (for Alpine) and ensure that Topal uses `msmtp` to save an extra bcc copy to me (see section 5.4.1).

Also see the remarks on save-on-send in section 5.11.

3.7 Mailcap configuration

To decode MIME RFC2015/3156 multipart/signed and /encrypted messages requires help via the "mailcap" system. Add in either the user mailcap configuration (`.mailcap`) or the system configuration (`/etc/mailcap`) the lines

```
multipart/signed; /usr/bin/topal -mime '%s' '%t'; needsterminal
multipart/encrypted; /usr/bin/topal -mime '%s' '%t'; needsterminal
application/pgp; /usr/bin/topal -mimeapgp '%s' '%t'; needsterminal
```

Note that Alpine has been known to crash if you compile it with the second patch, but do not set up your mailcap file to handle these multipart types.

3.8 Procmail configuration

From Topal version 65, this recipe is no longer needed if you are using patch 2.

If you prefer to modify inbound multipart messages rather than use patch 2, you should add this recipe to your `.procmailrc`:

```
:0fw
| /usr/bin/topal --fix-email
```

This causes Topal to examine all inbound emails. Those with top-level multipart/signed or multipart/encrypted MIME types are modified to add a multipart/misc wrapper so that Alpine can hand it off to Topal. All other emails are left unchanged.

I strongly advise that you also use one of the backup recipes from the procmail manual.

3.9 Topal configuration

Create a directory called '`${HOME}/.topal`'. This is currently hard-coded into Topal. Create the basic configuration file by running topal with the `-dump` or `-default` options. This file should be named '`config`'.

All `.topal` files are silently ignored if they cannot be found. Comments begin with a # in the first column, and run to the end of a line. They are totally ignored and are not currently preserved. Parsing errors cause an exception.

If you want to include strings with spaces, you'll need to quote them with double-quotes ("). Double-quotes themselves can be included by 'stuffing' ("").

Chapter 4

Usage

4.1 Help!

`-help` as the first argument dumps a help message.

The help message is derived from the `help.txt` file in the source distribution.

See `help.txt` for information on non-Pine use of Topal (section 4.5).

Send email to me if you're really stuck.

4.2 Configuration

`-config` as the first argument brings up the configuration menu. This menu is also available when sending (so that the signing key can be changed).

However, not all features can be configured via the menus. In some cases, you'll need to edit `.topal/config`. Later sections in this manual give specific instructions for particular features.

If you want to change the comment mentioning Topal when sending messages (*i.e.*, the bit saying `Topal (https://zircon.org.uk/topal/)`) then modify `sending-options` (via the configuration menu or the `config` file).

Colours were introduced in release 71. If you want to turn them off, set `ansi-terminal=off` in the `config` file. You can change the colours by setting `colour-menu-title`, `colour-menu-key`, `colour-menu-choice`, `colour-important`, `colour-banner` and `colour-info`. The codes are the escape codes (see, for example, colour table in the `http://en.wikipedia.org/wiki/ANSI_escape_code` Wikipedia article on ANSI escape codes).

4.3 Decryption/verification

The recommended invocations for Topal from inside Alpine are

1. via Alpine's `display-filters` (see section 3.4) for inline messages; and

2. for any message type, via piping *the whole message* (Alpine command `|`) to `topal -raw` with the `Raw text` (`^W`) and `Free output` (`^Y`) options set.

Depending on configuration, Topal will either ignore the file altogether, ask you what you want to do with it, or proceed to process the file automatically.

GPG will ask you for your passphrase when it needs it.

Caching is in place; the results of decryption and verification are (subject to configuration) saved in `~/.topal/cache`. The results of caching mean that you won't be repeatedly asked for your passphrase, at the expense of storing decrypts in the clear.

Be warned: Topal often invokes `less` to view something. So you'll need to use `q` to get out of it. `metamail` or `run-mailcap` may be called for anything after MIME processing.

New in version 0.7.8.

A new option called `inline-separate-output` concerns inlined (*i.e.*, not MIME) messages. If the option is on, then the Topal/GnuPG output will be shown to you by `less`. Then the decrypted or verified output will be handed back to Pine/Alpine. This is the way to approach attachments. However, you will normally want to keep this option off, because if you're reading (for example) BugTraq mailings, then it will want you to hit `q` an awful lot. . . .

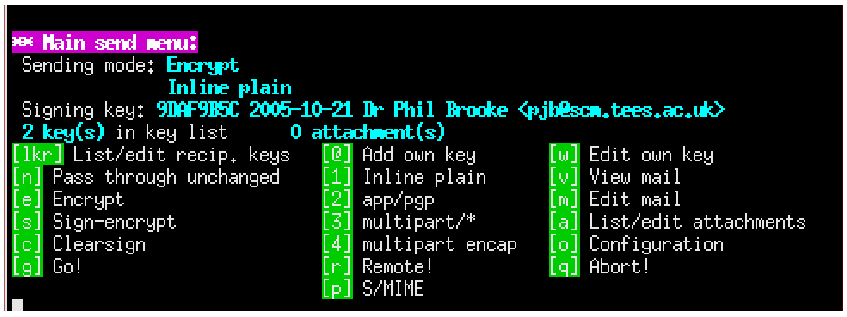
4.4 Sending

The recommended invocation for Topal from inside Alpine is via `sendmail-path` (see section 3.5).

4.4.1 Main send menu

Topal initially displays some chatter about selecting keys, including (if possible) a key for the sending user. After that, you will be presented with the sending main menu (figure 4.1). Some options (*e.g.*, those relating to MIME features) will not be offered unless you use the `-sendmime` option.

'Abort' tells Pine/Alpine you don't want Topal to process the email anymore.



```
*** Main send menu:
 Sending mode: Encrypt
               Inline plain
 Signing key: 9DAF9B5C 2005-10-21 Dr Phil Brooke <pjb@scm.tees.ac.uk>
 2 key(s) in key list      0 attachment(s)
[lkr] List/edit recip. keys   [@] Add own key       [w] Edit own key
[n] Pass through unchanged    [1] Inline plain      [v] View mail
[e] Encrypt                   [2] app/pgp           [m] Edit mail
[s] Sign-encrypt              [3] multipart/*       [a] List/edit attachments
[c] Clearsign                 [4] multipart encap   [o] Configuration
[g] Go!                       [r] Remote!           [q] Abort!
                              [p] S/MIME
```

Figure 4.1: Main sending menu

‘Pass through unchanged’ does nothing to the message. This means that you can always have Topal invoked for sending.

‘Add own key’ adds an ‘encrypt to self’ key. (It is added by default, but if you remove it, this is a quick way to restore it.)

When you select ‘Go!’ you will be asked to confirm the command-line, and after processing, `less` is invoked to visually check that the desired result has been achieved. Again, a confirmation is asked for.

Topal can offer a choice of three MIME types. Don’t use (2 — app/pgp) unless you really know what you’re doing. (4 — multipart encap) is only relevant if you are signing and encrypting: this encapsulates a MIME signed message inside an encrypted message (but Thunderbird doesn’t seem to process the signature on these). Otherwise, we do both operations at once. (If you choose ‘clearsign’ and ‘multipart/*’, then all trailing blank lines will be deleted. Note also that Pine/Alpine appears to delete trailing whitespace in trailing blank lines.)

‘Configuration’ offers the same menu that is available from the `-config` option.

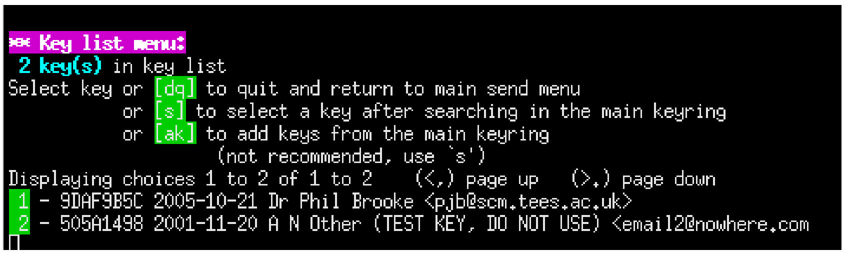
4.4.2 List current recipient keys

‘List current recipient keys’ offers a list of recipients (figure 4.2).

‘Quit and return to main send menu’ sends you back to the first menu.

‘Add key from main keyring’ prompts you for a search pattern. It will do a general search on your GPG keyring *and add* all matching keys. Beware of just pressing enter — it will select *all* keys on your keyring.

A better alternative is to use the ‘select after search’ option. This also does a search on your GPG keyring, but then you must select one key to be added to your list of recipients.

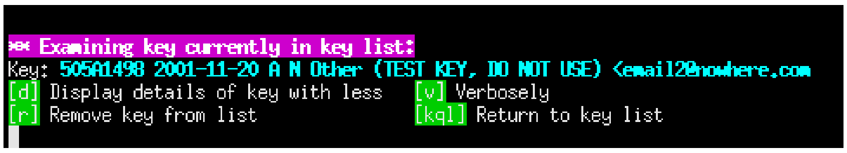


```
** Key list menu:
 2 key(s) in key list
Select key or [dq] to quit and return to main send menu
           or [s] to select a key after searching in the main keyring
           or [ak] to add keys from the main keyring
                    (not recommended, use `s')
Displaying choices 1 to 2 of 1 to 2    (<,) page up   (>.) page down
1 - 9DAF9B5C 2005-10-21 Dr Phil Brooke <pjb@scm.tees.ac.uk>
2 - 505A1498 2001-11-20 A N Other (TEST KEY, DO NOT USE) <email2@nowhere.com
```

Figure 4.2: Key list menu

4.4.3 Examine key menu

Selecting a key will offer a third menu (a similar menu is offered when selecting a single key) (figure 4.3).



```
** Examining key currently in key list:
Key: 505A1498 2001-11-20 A N Other (TEST KEY, DO NOT USE) <email2@nowhere.com
[d] Display details of key with less   [v] Verbosely
[r] Remove key from list               [kql] Return to key list
```

Figure 4.3: Single key menu

'Return to key list' takes you back to the second menu.

'Display details of key (less)' simply uses GPG to list the key details via `less`. You'll need to use q to get out of `less`.

'Verbose details of key (less)' pipes verbose output from GPG for this key into gpg. Again, you'll need to use q to get out of `less`.

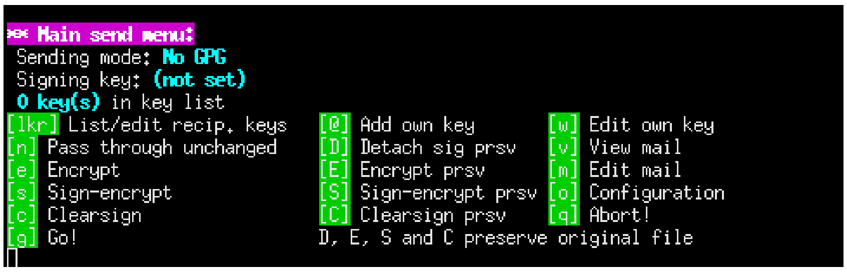
'Remove key from list' removes the key from this recipient list.

4.5 Command-line usage

4.5.1 Sending (-nps)

If you invoke Topal on the command-line with a filename as an argument, it will offer the sending functions on that file (figure 4.4). It doesn't actually send anything: instead it allows you to encrypt, sign, *etc.* the message. *You have a choice of overwriting or preserving the original file (this bit is case-sensitive).* So `e`, `s` and `c` are different from `E`, `S` and `C` respectively.

The main purpose of this mode is for encrypting or signing attachments before they are attached to the message in Pine/Alpine. Beware that Pine/Alpine



```
** Main send menu:
 Sending mode: No GPG
 Signing key: (not set)
 0 key(s) in key list
[lkr] List/edit recip. keys    [@] Add own key        [w] Edit own key
[n] Pass through unchanged     [D] Detach sig prsv    [v] View mail
[e] Encrypt                    [E] Encrypt prsv       [m] Edit mail
[s] Sign-encrypt               [S] Sign-encrypt prsv  [o] Configuration
[c] Clearsign                  [C] Clearsign prsv     [q] Abort!
[g] Go!                        D, E, S and C preserve original file
```

Figure 4.4: Non-pine sending menu

does not feed the attachments to a sending filter. Or you could use the attachments option when using Pine with MIME emails.

MIME functions are not available in this mode.

4.5.2 Receiving (-pd)

New in release 71.

Invoking Topal with only `-pd` as an argument causes it to read a message from standard input and attempt to decrypt/verify it. This is useful for piping MIME messages from other applications.

4.6 Remote and server mode

Suppose you are reading your email on a remote host via `ssh` (as I often do). You now want to compose an email and sign it, but your secret key is only accessible on the local computer. Topal has rudimentary support for this (primarily to support my style of working). This comes in two parts: a 'server' mode to run on the local computer (with access to the secret key) and a remote option in the sending menu.

The server mode (on the local host) is started by running `topal -server`. This is where GPG requests for signing are made.

When sending, you can choose 'remote'. This prompts for the host to connect to using `ssh`/`scp`: this host should be running the 'server'. The files are sent to the local server, processed by the server, then the results are copied back. `ssh` and `scp` are both used: because they're used repeatedly, you might want to use key-based authentication and have the key added to a current `ssh-agent`.

There is also a remote mode for receiving, with a similar behaviour as for sending. Alternatively could use something like unison (or some other file synchroniser or a simple scp) to move the email(s) concerned, then view them on the local computer.

A generally better alternative is to process your email locally by using IMAPS and SMTPS to remotely access the mail server.

4.7 Fixing multipart emails

Two scripts used to be included with topal (long ago): `topal-fix-email` and `topal-fix-folder`. They have been replaced by the `--fix-email` and `--fix-folder` command-line options to the main binary.

`topal --fix-email` modifies any email that is (at the top level) a multipart/signed or multipart/encrypted message. It creates a multipart/misc message instead: this revised message is simply a wrapped version of the original message so that Pine/Alpine can pass the signed or encrypted part to Topal.

Usage:

- `topal --fix-folder <folder> ...` fixes the named email folders.
- `topal --fix-email` takes no arguments; it accepts a single email on stdin. Ideally, it should be invoked by procmail (see section 3.8).

`topal --fix-email` has a simpler mode (`--simple`) where it pretends that there are two MIME content types: 'application/x-topal-encrypted' and 'application/x-topal-signed'. You might prefer using this. These effectively rename the multipart email instead of wrapping it.

Why do we need this? If we just set the `.mailcap` file for, say, multipart/signed, then Alpine (at least version 1.00) is unable to handle a top-level multipart/signed email: an error message starting

```
Can't find body for requested message
```

is seen. But multipart/signed inside a multipart/mixed (or multipart/alternative, *etc.*) can be successfully handed-off to Topal.

Replying to such messages is a pain: you'll have to save off the actual message and read it in. Suggestions on fixing this are welcome....

See `Workaround.Fix_Email` in the sources for more details.

Chapter 5

Notes

This chapter includes a number of notes, explanations and further details.

5.1 The Pine/Alpine patches, and sending other attachments

What does the (first) patch to Alpine do? It removes some of the safety checking when changing the content-type (`_MIMETYPE_`) in a filter. Normally, if the returned content-type is not text/*, then the entire content-type is dropped by Alpine.

The patch instead adds a flag, '`topal_hack`', and sets this if the returned content-type is not text. From time-to-time, we pretend that the body is normal text. We take a little care to check if this message is already a multipart message, so hopefully, the normal sending of attachments still works. However, these attachments are *not* encrypted or signed by Topal. You will need to separately process them before attaching them in Alpine (*e.g.*, using the command-line mode described in section 4.5) or add them using Topal's attachment menu (if you are using Topal's MIME sending features).

The second patch file slightly modifies some of the mail reading code to allow .mailcap settings to act directly on top-level multipart messages. But see section 5.15 if you find Alpine crashes when this patch is used.

New from Topal release 65 for Alpine 2.00.

5.2 Key IDs and keylists

Topal internally lists keys by their fingerprint. It uses GPG to look up key fingerprints by using whatever GPG can cope with.

Duplicate keys are silently suppressed. Removing a key only removes one instance, so if somehow you've coerced Topal to list duplicates (which is quite easy, since adding a key with its short key ID, and the same key with its fingerprint will add two identical keys).

The way that Topal chooses the keys is as follows:

- For each recipient email address (supplied by Pine)
  1. For each matching line in keylist, use the key ID to get a fingerprint, and add the key to the list.
  2. If there are no matching lines in keylist, try to get a fingerprint via just that email address (but exclude `xk` configuration entries).

The keylist is a way to say, 'for this particular email address, use this particular key'. In your `config` file, include lines such as

```
ake=50973B91,philb@soc.plym.ac.uk
ake=50973B91,p.j.brooke@bcs.org.uk
```

These mean 'use key 50973B91 for the given email addresses'. Similarly,

```
xk=50973B91
```

means 'don't use key 50973B91'. There are also similar `sake` and `sxk` options for selection of secret keys via the (recommended) `--read-from` option (page 10).

For S/MIME, the same lists are used. If a key isn't relevant (an S/MIME key when using OpenPGP or *vice versa*) it is ignored. Importantly, you need to use `sake`/`sxk` to give the signing key for S/MIME, as `my-key` only applies to OpenPGP usage.

5.3 Sending defaults

New in release 68.

`sd` lines in your `config` file can assign a default to a key ID or email address. The special string @ANY@ matches any email address (or key). The *last matching* `sd` expression applies, so use @ANY@ first.

The individual letters mean

n	no GPG	I	inline plain
e	encrypt	A	app pgp
s	sign and encrypt	M	multipart
c	clearsign	E	multipart encapsulated
		P	S/MIME (not OpenPGP)

Example:

```
sd=p.j.brooke@bcs.org.uk,eI
```

means that the default settings for the given address are encryption and inline plain.

Topal will warn you if you are sending to multiple recipients, `sd` has selected encryption, and not all of the recipients have keys. However, it won't stop you continuing and sending an email that some recipients can't read.

5.4 Sendmail-path configuration

New in release 73.

`sc` lines in your `config` file assign a `sendmail-path` command to the 'from' (sending) email address. Again, the special expression @ANY@ matches any email address (but not key id's) and the last matching `sd` expression applies.

Example:

```
sc=p.j.brooke@bcs.org.uk,mstmp -a default
```

means that emails sent using that address will actually be sent using the (external) `msmtp` program. If you are using Topal as a `sendmail-path`, you must ensure that there is a suitable `sc` line, as Topal does not actually know how to send email.

5.4.1 Additional (bcc) recipients

Note that `msmtp` can add additional recipients, *e.g.*,

```
mstmp -a default -- p.j.brooke@bcs.org.uk
```

5.4.2 Message-ID munging

The `sendmail-path` mode can also modify Message-IDs and Content-IDs. The configuration option `replace-ids` controls this. If set to 0, nothing is changed; 1 causes the Message-ID to be altered; and 2 (the default) alters both Message-ID and Content-IDs (of attachments).

The Message-ID is altered from the standard Alpine form of

```
<alpine.DEB.2.02.1103063481092.1140@somewhere.com>
```

to

```
<1103063481092.1140.NCYBIHRU%user@somehost.org>
```

The email address used in the replacement is based on the From line.

5.4.3 Send tokens

The `sendmail-path` mode also enables the `st` (send token) lines in the `config` file. These lines have the form

```
    st=user@example.org,somepasswordstring
```

If the Message-ID is altered and there is a matching send token, the original Message-ID is encrypted and stored in the headers. Additionally, a header is added based on this token, the Message-ID and From line. This can be used to match inbound emails (self blind copies) via procmail, *e.g.*,

```
:0fw
* ^From: .*\<user@example.org\>$
| /usr/local/bin/topal --check-send-token somepasswordstring

:0:
* ^X-Topal-Check-Send-Token: yes$
.topal-cc/
```

This causes emails apparently from `user@example.org` to be passed to Topal, which checks the headers. If the header token matches for that password, a new header, `Topal-Check-Send-Token: yes` is added. Then procmail can filter on the basis of this header.

5.4.4 Missing attachments trap

New in Topal 73.

If set, the configuration line `attachment-trap=on` causes Topal to complain if the message contains the string “attach” but it does not have any attachments.

This is only effective if Topal is running as a `sendmail-path`.

5.5 Errors

Bad things happening should result in Topal setting its exit status to ‘failed’, so Pine should detect this and not send your email.

Bug reports are welcome: send them by email to me (contact details in section 6).

5.6 Saving encrypted attachments

If an attachment is a plaintext PGP ASCII-armoured message, then Topal will be invoked by Pine. You probably want to say ‘no’ when asked here (beware of your configuration options here). Otherwise, you’ll get a decrypted file with the original attachment filename, plus the various Topal headers.

5.7 Locale problems

GPG does not do any encoding of input data. This means that the encoding is dependent on Pine/Alpine and Topal. If a message is sent with one encoding and received by a user running in a different locale, then we might end up with a good message not verifying (*i.e.*, bad signature).

I currently have no way to automatically fix this. However, the `--ask-charset` option will ask if you want to change the encoding. If you know that the message was written by a UTF-8 user (and you’re in a different locale), this might help. (This only happens if a bad signature is returned.)

I know it’s a kludge. I’d be interested to hear success and failure reports.

5.8 “`Cancelled by user`” and `GPG_TTY`

When reading an email from stdin, Topal sometimes confuses subsequent calls to GPG/Pinentry. Make sure that your environment (*e.g.*, `.bashrc`) includes something like

```
export GPG_TTY=$(tty)
```

5.9 “`Couldn’t find certificate needed to sign.`”

A regular query (for both Topal and other OpenPGP filters) concerns the message “`Couldn’t find certificate needed to sign.`” This message is part of Alpine’s S/MIME support: it is nothing to do with Topal. One option is to turn off the internal S/ MIME support via the (hidden) config option ("`S/MIME -- Turn off S/ MIME`").

5.10 Cleaning up the cache

You might want to run something like

```
find ${HOME}/.topal/cache -mtime +7  xargs rm|
```

to remove all the cache files that are a bit old (in this example, 7 days old or older).

5.11 Save-on-send

If `save-on-send` is set, then the last email sent will be written into `lastmail` in your `.topal` directory. If the email was to be encrypted, then so will the contents of `lastmail`.

This setting can be adjusted via `topal -config`.

5.12 Remote and server mode

When remote is invoked in a sending menu:

- The host has to be chosen for `ssh`/`scp`.
- Because Topal might be outside the normal path, you'll be asked for that too.
- The sender `scp`s the relevant files into `.topal/server`.
- The sender calls

  ```
  ssh (server) -remotesend ...
  ```

  or

  ```
  ssh (server) -remotesendmime ...
  ```

  .
- The invocation of `-remotesend` or `-remotesendmime` triggers the server to run a new instance of Topal on the local computer.
- When that instance is finished, the relevant files are copied back, along with the return value.

For receiving: if decrypt-not-cached/decrypt-cached are set to 2 (ask), then as well as offering to decrypt (yes or no), it will also offer remote decryption. The caching settings are irrelevant at this point.

5.13 Working with GPG Agent

The `use-agent` configuration option has three values: (1) never use an agent, (2) only use it for decryption and (3) always use it. Don't put GPG's `--[no-]use-agent` options in any other configuration options.

Notes:

1. `--no-use-agent` is deprecated in some versions of GPG.

2. If you change `trustlist.txt` (when using `gpgsm`, *i.e.*, GnuPG for S/MIME) then remember to send the HUP signal to your current GPG agent.

5.14 Decryption prerequisite

This relates to the configuration option `decrypt-prereq`.

If empty (the default), this is ignored. Otherwise, if decryption is required (*i.e.*, a non-cached encrypted block is found) it will run the command stated (this should be just the command name, with no arguments). If the command's exit status is 0, then decryption continues. If not, then no decryption is attempted.

I've introduced this for the case where decryption keys are not always available. It is a nuisance for Topal to offer to decrypt when it cannot. Example: the secret key ring is kept on removable media. Then set

```
decrypt-prereq=/path/to/topal-decrypt-prereq
```

The executable file `/path/to/topal-decrypt-prereq` contains something like

```
mount grep /path/of/keyring > /dev/null
```

This returns 0 if that path is found, and 1 otherwise.

5.15 Crash with second patch when reading multipart messages

Some Alpine crashes have been traced to the second Topal patch.

This occurs when you have compiled Alpine using this patch, but not set up your `.mailcap` configuration appropriately.

5.16 Threats

I should document a proper threat model for Topal.

For now, I make the following remarks. I assume that the local machine is secure, *i.e.*, it is safe to store decrypted content on the machine. For my purposes, this is a reasonable assumption: if my machine's compromised, then I have other, bigger problems — and the attacker can read my plaintext messages as I write or view them. However, it does increase the time of exposure, for example, caching plaintexts means that later compromise could reveal them. This is a trade-off against repeatedly decrypting my GPG keys. If that doesn't fit your threat model, then Topal's probably not the tool for you.

5.17 New releases

To be notified of new releases of Topal, send an email to me.

5.18 Interoperability

I have tested Topal's interoperability with MS Outlook (both IMAP and Exchange) and Thunderbird. It works in all tested cases, except as follows.

- At least one instance of MS Exchange rewrites the content-* headers of inbound clearsigned messages with attachments, which results in the signature failing to verify. This also affects messages sent from Thunderbird and MS Outlook. However, it does not apply to all MS Exchange systems.
- MS Outlook and Thunderbird do not handle clearsigned messages properly when the clearsigned message is itself part of a multipart/mixed message. The only way to reliably send messages to users of these systems is via the `sendmail-path` mode.

5.19 Hints

- Consider using GPG's `--trusted-key` option in Topal's `gpg-options` configuration setting. This is useful if you keep your secret keys offline.
- Use the `--read-from` option (page 10), especially if you use multiple roles rather than setting `my-key`.

Chapter 6

Author

Phil Brooke wrote this, partially out of boredom, but mostly because he wanted a GPG/Pine add-on to do exactly what he wants. There are many similar programs.

If you like this program, please tell me. If you'd like it better with changes, please tell me what changes you want. If particular items on the 'To do' list are important to you, let me know. In particular, if you find bugs, feel free to tell me the details by email.

I can be emailed on `p.j.brooke@bcs.org.uk`.

My key ID is 0x47DC67A1; the key is available from web pages and public key servers.

If you want to send snailmail to me, email me for my (physical) address.

Chapter 7

Licence

Chapter 8

To do

- Planned releases:
  - Improve attachments code (and add some documentation).
- Better error handling, particularly when missing dependencies such as mime-construct or metamail.
- Add signal handlers.
- Catch GPG keyboard interrupt.
- Should we check that the infile matches the cache file even if the MD5 hash matches? (We'd need to store the infile in the cache as well.)
- Check through code: all external calls should check return values.
- Refactor code.
- Add interrupt option at very beginning of execution? (which would bring up the configuration menu?)
- Associate extra options with particular keys?
- Configuration routine for managing keys/config/keylist?
- Implement rest of configuration menu.
- Make a much nicer interface all round....
- Separate out all the constant strings – so that we can have internationalization.

- Context-sensitive help throughout (modify mkhelp to create multiple procedures, or do it by number?); add COPYING option?
- More receiving/decrypt options: include both plaintext and ciphertext.
- Add periodic cache cleanup when Topal is invoked?
- Add logging for workaround mode (report time of email processing (include PID); indicate if the file was changed or not)?

Chapter 9

Change log

The change log is kept in a separate file, `Changelog.html`.