1 Introduction

The *LSGA Encryption Working Group* is considering how to encourage financial institutions, amongst others, to make encryption a routine procedure so we feel safe dealing with them by email. Some of them have figured out ways of letting you view your information *via* `https` without sending any easily-compromised details with the email, but most are not anywhere near that yet.

You would expect that the simplest and ideal way is to write your message and add attachments using your email program, press a button labelled `encrypt`, and the whole email is encrypted using the public key of the recipient, and sent. End of story. Why is this impossible?

There are over 50 email programs: only a handful have integrated encryption.

Changing to use one of this handful just for occasional encryption is going to be resisted.

The recipient often has multiple public keys and some might not contain the email address you are using to send.

People are good at their email program but cannot be asked to both learn encryption and learn the email integration techniques at the same time.

Encryption itself is a black box and people easily get confused using it.

One of the studies we consulted was *Why Johnny Can't Encrypt*[1], which showed us that *O'Toole is everywhere* when encryption is concerned, and that a new paradigm is needed. Google spent four years trying this and failed; Yahoo failed. Digital certificates are unbelievably complictated to configure. Thunderbird crashes when there are no public keys ...

We drew these preliminary conclusions from the survey that we made.

- Leave people with their (unreconstructed) email program of choice.
- Let them deal with encryption as a separate program (this wizard).
- Hold their hand, and act as someone who is 'always at their elbow' to prompt them at each stage of the protection process. We believe this means using text-to-speech and animations.

The new paradigm shaping our heuristic `Cryptowiz` program trial comprises: (1) Send this wizard program to our correspondent to let them get up to speed. (2) Protect files by encryption separately from email, and then attach the files to email. (3) Be a constant helper at each stage to re-enforce the encryption process. (4) Emphasize continuous learning of the steps by explaining them, in speech and by simple animations.

We are experimenting with `dialog`-style programs, combined with *text-to-speech* and ASCII+GIF animations to explain and illustrate encryption concepts. These are much easier to write than `GUI` style programs and add novelty for Mac and Windows users — we expect that to be enjoyable. We want to keep this program small – no videos – so we can send it over email to our recipient and they can execute it on their system. We can leverage our grasp of `bash` and `powershell` to write simple dialog emulators for them.

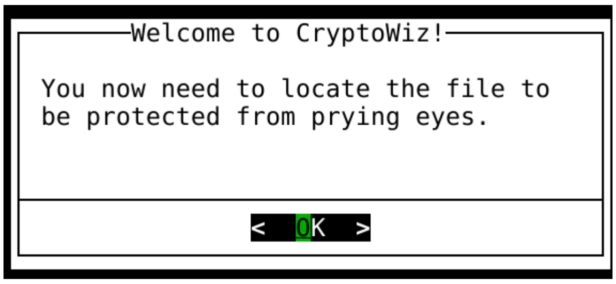
LSGA shall be a Digital Certificate Authority

Finally, we believe that we shall be more credible if we practice what we preach. To this end, not only will we sign future newsletters, but also some of our web content. In addition, because some email programs natively can deal with encryption *via* digital certificates, the LSGA shall become a *Digital Certificate Authority* at our meeting on Wednesday the 7th of February 2018. Come along to this historic *key-signing* occasion and be amongst the first to receive a certificate!

[1]ftp://ftp.iks-jena.de/mitarb/lutz/crypt/software/pgp/PGP5:Evaluation-study.pdf

2 Running the CryptoWiz Program

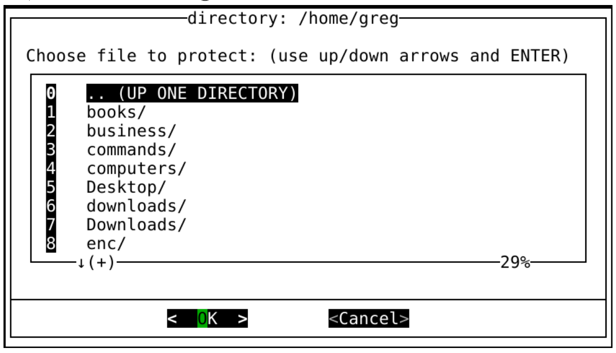
Text-to-Speech is also used. Below left: greets you at first. Below right: starts in your home directory (printed at top of window), gives brief navigation advice, with a listing of the contents.



```
Welcome to CryptoWiz!

You now need to locate the file to
be protected from prying eyes.

< OK >
```

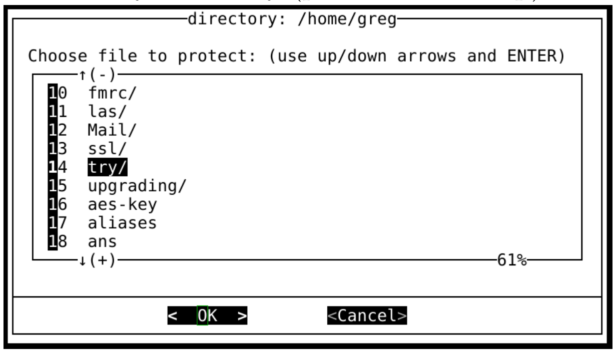


```
directory: /home/greg

Choose file to protect: (use up/down arrows and ENTER)
0   .. (UP ONE DIRECTORY)
1   books/
2   business/
3   commands/
4   computers/
5   Desktop/
6   downloads/
7   Downloads/
8   enc/
  ↓(+)                                      29%

< OK >          <Cancel>
```

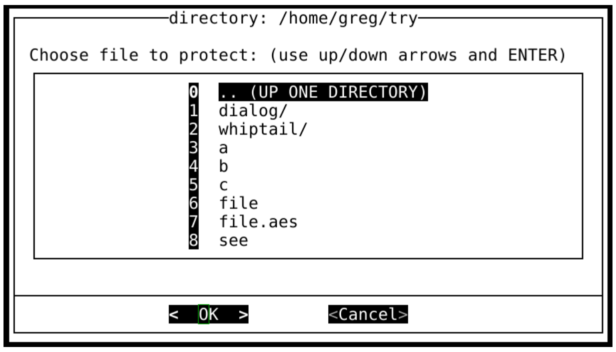
Below left: you have scrolled down to the directory 'try'. Below right: you pressed ENTER and are now in the 'try' directory (printed at the top) with a listing of it.



```
directory: /home/greg

Choose file to protect: (use up/down arrows and ENTER)
  ↑(-)
10  fmrc/
11  las/
12  Mail/
13  ssl/
14  try/
15  upgrading/
16  aes-key
17  aliases
18  ans
  ↓(+)                                      61%

< OK >          <Cancel>
```

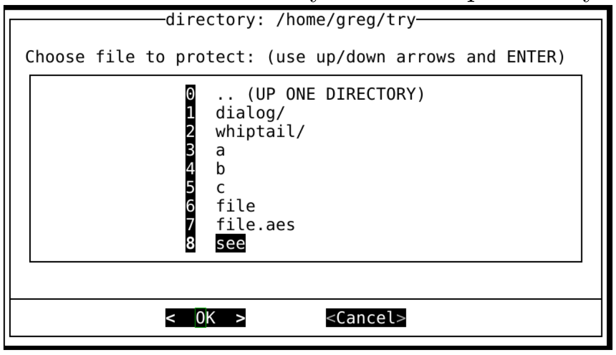


```
directory: /home/greg/try

Choose file to protect: (use up/down arrows and ENTER)
                  0   .. (UP ONE DIRECTORY)
                  1   dialog/
                  2   whiptail/
                  3   a
                  4   b
                  5   c
                  6   file
                  7   file.aes
                  8   see

< OK >          <Cancel>
```

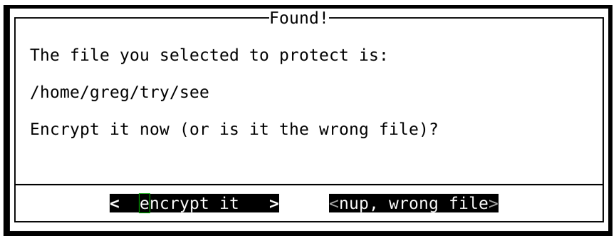
Below left: you have scrolled down to the file you seek, named 'see'. Below right: gives you a chance to confirm that this is the file you wish to protect by encryption.



```
directory: /home/greg/try

Choose file to protect: (use up/down arrows and ENTER)
                  0   .. (UP ONE DIRECTORY)
                  1   dialog/
                  2   whiptail/
                  3   a
                  4   b
                  5   c
                  6   file
                  7   file.aes
                  8   see

< OK >          <Cancel>
```



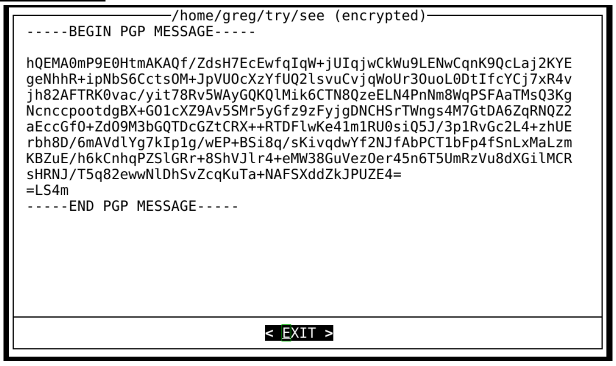
```
Found!
The file you selected to protect is:

/home/greg/try/see

Encrypt it now (or is it the wrong file)?

< encrypt it >     <nup, wrong file>
```

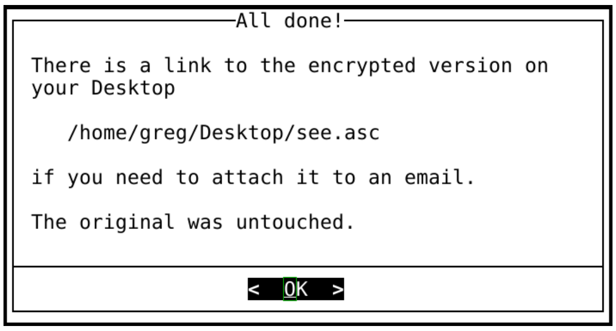
Below left: encrypts that file and the encrypted contents are shown here as confirmation of that. Below right: tells you where this encrypted version of your file has been stored.



```
/home/greg/try/see (encrypted)
-----BEGIN PGP MESSAGE-----

hQEMA0mP9E0HtmAKAQf/ZdsH7EcEwfqIqW+jUIqjwCkWu9LENwCqnK9QcLaj2KYE
geNhhR+ipNbS6CctsOM+JpVUOcXzYfUQ2lsvuCvjqWoUr3OuoL0DtIfcYCj7xR4v
jh82AFTRK0vac/yit78Rv5WAyGQKQlMik6CTN8QzeELN4PnNm8WqPSFAaTMsQ3Kg
NcnccpootdgBX+GO1cXZ9Av5SMr5yGfz9zFyjgDNCHSrTWngs4M7GtDA6ZqRNQZ2
aEccGfO+ZdO9M3bGQTDcGZtCRX++RTDFlwKe41m1RU0siQ5J/3p1RvGc2L4+zhUE
rbh8D/6mAVdlYg7kIp1g/wEP+BSi8q/sKivqdwYf2NJfAbPCT1bFp4fSnLxMaLzm
KBZuE/h6kCnhqPZSlGRr+8ShVJlr4+eMW38GuVezOer45n6T5UmRzVu8dXGilMCR
sHRNJ/T5q82ewwNlDhSvZcqKuTa+NAFSXddZkJPUZE4=
=LS4m
-----END PGP MESSAGE-----

< EXIT >
```



```
All done!
There is a link to the encrypted version on
your Desktop

   /home/greg/Desktop/see.asc

if you need to attach it to an email.

The original was untouched.

< OK >
```

3 Analysis of the CRYPTOWIZ Program

We have implemented our ideas in the program listed on the last page. We analyse that program, demonstrated in this talk, and you can follow the rationale for the program algorithm.

line 1 specifies `bash` as the application to process this file.

line 4 uses the `espeak` program to make it seem as if we are at your elbow.

line 5 presents the first `dialog` box, suggesting *protect from prying eyes* rather than just *encryption.*

line 6 saves the current directory so that you can return here when done; after all, you'll be jumping around a bit looking for files shortly.

line 7 sets the filename you seek to your home directory, for the purpose of the following algorithm.

lines 8 to 21 are executed whilst you are popping in and out of directories looking for your file.

line 10 enters the directory you have selected and then makes a listing of the directory, with directories first (each with a trailing slash) followed by all the files.

line 11 tells you that you can use the left and right arrows to answer the question.

line 12 presents a menu of that directory listing. The directory is printed at the top. There is brief advice on navigation. Then a list of tag1 item1 tag2 item2 ... tagN itemN, with the ability to go up one directory as item 0 and each file or folder in the directory in listing order by number 1..N (for purposes of selection the item). The selection is written to file '2' which we send to a temporary file.

line 14 extracts the number of the item in the list that you selected.

lines 15 to 21 extracts the file or folder name itself from the listing (it is item 'n'), unless it was the top one (item '0'), in which case you just want to go up one directory (..).

line 22 is executed when you eventually select a file, not a folder. The full path to the file is computed.

line 23 reminds you to use the left and right arrows, in case you forgot.

line 24 presents a dialog asking you to confirm the selected file's identity.

lines 25 to 36 deal with your response to this confirmation. If it is the wrong file it exits (the response on the right-hand button returns a '1'). If it is the correct file it receives a '0', the left button.

line 26 forces removal of a temporary file so no distracting messages get printed.

line 27 constructs the default recipient ID for encryption (you@your-computer)

line 28 encrypts selected file to your public key and saves encrypted version in a temporary file.

line 29 tells you that it has been encrypted.

line 30 proves that this encryption has been done by showing you the result.

line 31 tells you where the encrypted file has been saved in case you want to grab it and attach it to an email.

line 34 tells you that its no problem that you made a mistake and selected the wrong file.

line 35 reminds you that nothing was done, and to re-run the program and try again.

lines 38 to 40 return to your starting directory, clear the screen and quit this program successfully.

4 The Program Listing

This was developed in two days: one playing with `dialog`; one creating a file-finder+encrypter.

```
#!/bin/bash
# CryptoWiz: exploring 'leading the user by the hand' for encryption
# this subprogram is for trying out how to select a file and encrypt it
espeak "OK, lets find that file" &
dialog --title 'Welcome to CryptoWiz!' --msgbox '\nYou now need to locate
    the file you want to protect from prying eyes.' 9 40
pushd $(pwd) 1>/dev/null
filename=$HOME
while [ -d $filename ]
do
    cd $filename;  /bin/ls --group-directories-first -p > /tmp/listing
    espeak "use arrows, then enter" &
    dialog --title "directory: $(pwd)" --menu "\nChoose file to protect:
        (use up/down arrows and ENTER)" 18 60 9   "0" ".. (UP ONE DIRECTORY)"
    $(cat /tmp/listing|nl|tr '\n' ' ')   2>/tmp/num
    n=$(cat /tmp/num)
    if [ "$n" == 0 ]
    then
        filename=".."
    else
        filename=$(cat /tmp/listing|head -$n|tail -1)
    fi
done
selected="$(pwd)/$filename";    name=$(basename $selected)
espeak "got it! use left right arrows to check it" &
dialog --title "Found!" --yes-button "encrypt it" --no-button "nup, wrong file"
    --yesno "\nThe file you selected to protect is:\n\n$selected\n\n
    Encrypt it now (or is it the wrong file)?" 12 60
case $? in
     0) /bin/rm -f /tmp/$name.asc
        gpg_id=$(whoami)@$(hostname)
        gpg --encrypt --armour -r $gpg_id -o /tmp/$name.asc $selected
        espeak "its encrypted, as you see" &
        dialog --title "$selected (encrypted)" --textbox /tmp/$name.asc 22 70
        espeak "Its ready to attach to email" &
        dialog --title "All done!" --msgbox "\nEncrypted version is at
            /tmp/$name.asc\nwhile the original was untouched." 12 50
        ;;
    1) espeak "No probs, try again" &
       dialog --title "Oops!" --msgbox "\nNothing done; rerun this program to
           try again." 12 50
        ;;
esac
popd 1>/dev/null
clear
exit 0
```