key2paper - Saving a cryptographic key to paper

I tried multiple solutions to save a private key to paper:

- using paperkey to reduce the filesize of the key
    - you need to have the public key in order to restore the private key --> the total size of stored data is not really smaller
- storing the resulting binary (full private key or paperkey) into a datamatrix code (dmtx-utils)
    - datamatrix is not supported well (anymore?) dmtx-utlis still work, but there is less wide support than for QR Codes
- storing the resulting binary directly into a set of structured (qrencode -S) qrcodes
    - reading structured (what does this mean?) QR codes is not widely available in common qr reader libraries (or needs to be configured). Additionally, some QR code readers don't support binary payloads

After trying aboves methods, I settled with following scheme to store data on paper:

- export the payload into a file
- encode the file into a base64 encoded string (to circumvent the limited support of reading binary QR data)
- split the base64 string into chunks of 140 characters
- for each chunk, create a payload for the QR code with 3 space separated components:
    - start with the number of the code and the total number of codes for this file: `03/42`
    - then the actual 140 characters of the base64 encoded payload
    - finally the first 6 characters of the md5sum as a checksum

Using this schema, one creates text chunks which look like this:

```
04/47 +t3J9YCmUc7r6Tk5phjGnaEvPo+mMczs...aT987Lhje53Z6fm45hnRFxwMlzxGpwag9sBd+XC3QAR 9bd63c
05/47 AQABAA/9H9iQl7qW5VqHlv/FTDYkDtNq...2SliwEFe+EbIV0KeoE4406BRJduOYkEgQtGhHQlyhEi 4238ba
06/47 /ySwcg+J3Ipr+xk0keyfm365+JXAIUOL...7JV+WU5ptfISwRUIpwGDgwQ/R8eceNqrBsyGmnEgCaR 8a5450
```

Each single line then is encoded into a QR code, e.g. using `qrencode`.

Install

- `cat`, `echo`, `split`, `wc`
- `base64`
- `qrencode`: Package `qrencode`
- `imagemagick` (`convert`, `montage`): Package `imagemagick`
- `a2ps`: base installation. Also here
- `ps2pdf`: Package `ghostscript`
- `pdfunite`: Package `poppler-utils`

Export to paper

Show your private keys:

```
> gpg -K
----------------------------
sec   rsa4096 2019-12-32 [SC]
      679E7032588A0E7525CBBEE44E53BF76569ACE7F
uid        [uneingeschränkt] Jan Martin <>
ssb   rsa4096 2019-12-32 [E]

```

You may export your key to paper in an unprotected (=without a password) manner. Therefore, optionally, remove the password from the key temporarily:

```
gpg --textmode --change-passphrase <KeyID>
```

Run the `create_paperkey.sh` script to create a pdf you need to print. For documentation, how the script exactly works, check out the source of the script itself:

```
./create_paperkey.sh <KeyID>
```

After that, go back and add a password to your key again: `gpg --textmode --change-passphrase <KeyID>`

Recreate private key using zbar:

```
zbarimg --raw -Sdisable -Sqrcode.enable <image.jpg> | sort -n | cut -d ' ' -f 2 |
base64 -d > mykey.bin
```

Output:

```
scanned 43 barcode symbols from 1 images in 1.5 seconds
```

Make sure that all bar codes are detected. The one-liner doesn't check for number of detected QR codes or correctness of the md5 checksum.

For a more sophisticated recovery, do

Recreate private key using Python, zbar and OpenCV

```
sudo apt-get install zbar-tools git
git clone https://github.com/x42x64/key2paper.git
cd key2paper
# optional: create virtual environment: python3 -m venv .venv && source .venv/bin/activate
pip3 install -r requirements.txt
python3 pick_and_transform.py -i <image.jpg>
```

If the python script detected all QR codes, it won't bother you. If there are codes missing, it will show you a window, where you have to select the 4 corners (including some margin) of the undetected QR code by clicking on the corners. Detected corners are framed green. You may select more than one QR code at once.

The script will save 2 files next to the image you selected:

- <image.jpg>.decoded.b64: The base64 encoded text file of the payload
- <image.jpg>.decoded.bin: The binary file stored on the paper

Restore the key from the recovered .bin-file

```
gpg --import $FILE
```

List your private keys:

```
gpg -K
```

Change/add password of your private key

```
gpg --textmode --change-passphrase <KeyID>
```

Decrypt a file

```
gpg --decrypt test/secret.org.gpg
```