Tilo Müller, Hans Spath, Richard Mäckl and Felix Freiling

STARK

TAMPERPROOF AUTHENTICATION TO RESIST KEYLOGGING

FRIEDRICH-ALEXANDER UNIVERSITY (FAU) OF ERLANGEN-NUREMBERG, DERPARTMENT OF COMPUTER SCIENCE (INFORMATIK 1)

FINCANCIAL CRYPTOGRAPHY AND DATA SECURITY (FC 2013), SEVENTEENTH INTERNATIONAL CONFERENCE, APRIL 1-5

April 4th '13

www1.cs.fau.de/stark

Chapter 1

Threat Scenario

Physical Access Threats

- 2008: “Airport Insecurity: The case of Missing Laptops” (Ponemon)
  - 12,255 laptops are lost at U.S. airports per week
  - 53% of traveling salesmen state to carry along sensitive data
- 2011: “The Billion Euro Lost Laptop Problem” (Ponemon)
  - 275 organizations in Europe
  - 8% of all company laptops are lost during their lifetime
    (in the 12-month study, 72,789 laptops were lost)
- 2012: “2011 Annual Study: U.S. Cost of a Data Breach” (Ponemon)
  - cost per data record: 194 USD
  - average cost per stolen laptop: ~45,000 USD

Solution: Disk Encryption

- protects data against *physical* loss and theft
- disk remains unreadable until a user enters the correct passphrase

- examples: BitLocker, FileVault, TrueCrypt, ...

  (most configurations based on AES)

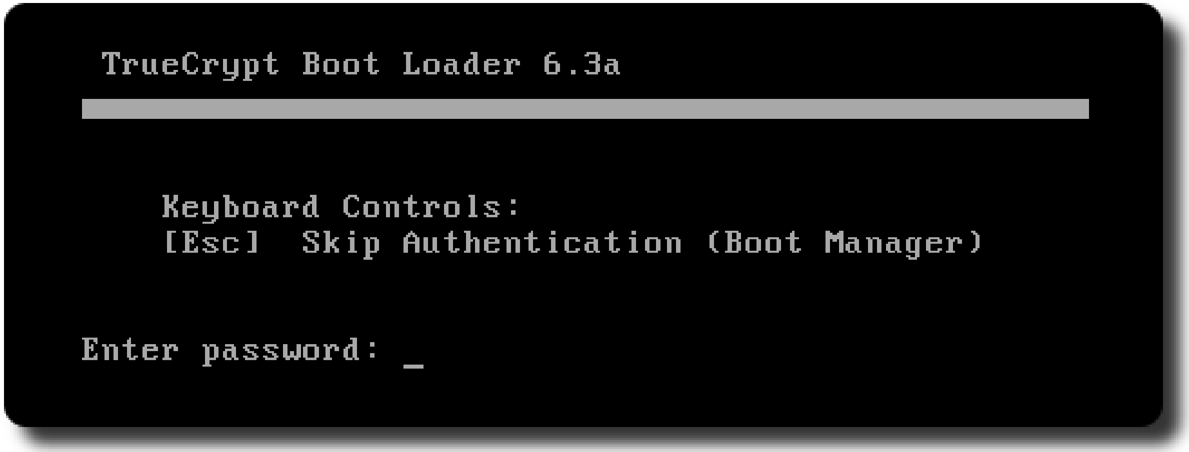
Full Disk Encryption

- FDE = *full* disk encryption
- supported mode by TrueCrypt, BitLocker, ...
- encrypts a whole disk including the OS

- the whole disk?
- no!
  for *bootstrapping* reasons, at least a small part of the disk must be present unencrypted

Preboot Environments

- placed inside the master boot record (MBR)
- *uniform* (often text-based) password prompts



- problem: these prompts can easily be forged!

Evil Maid Attacks

- coined by Rutkowska, 2009: *Evil Maid goes after TrueCrypt*
- basically a keylogger placed inside the MBR
- a.k.a. *bootkit*

- scenario:
  1. salesman leaves hotel room
  2. an *evil maid* manipulates the MBR
  3. salesman enters PW and leaves again
  4. the evil maid reads out the logged PW

Consequences

- today's FDE does not protect against *system subversion*, but only against loss and theft
- TrueCrypt says that bootkits

  “require the attacker to have [...] physical access to the computer, and the attacker needs you to use the computer after such an access. However, if any of these conditions is met, it is actually impossible to secure the computer”

- so it's a matter of opinion if FDE should or should not protect against evil maid attacks
- in our opinion, it should!

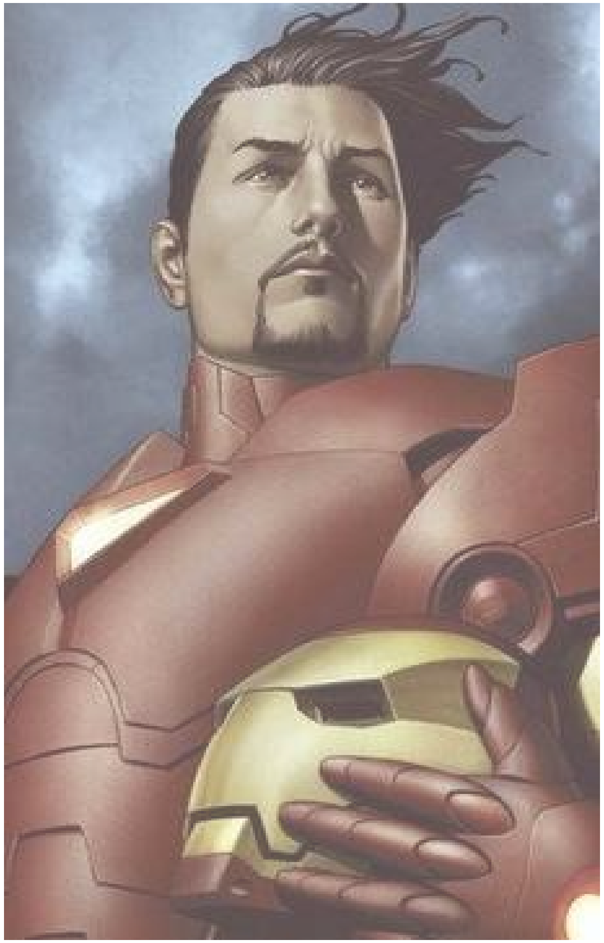
BitLocker and the TPM

- BitLocker supports the Trusted Platform Module (TPM)
- TPM is used to “unseal” the data encryption key
- if MBR is manipulated, the correct key cannot be unsealed and the data cannot be decrypted

Tamper-and-Revert Attacks

- Türpe et al., 2009: “Attacking the BitLocker Boot Process”
- Tamper-and-Revert:
  1. *tamper* with the bootloader to introduce keylogging
  2. victim enters PW into forged prompt
  3. *revert* to the original bootloader
  4. reboot

- Most likely, the user enters PW again and proceeds working as usual



Chapter 2

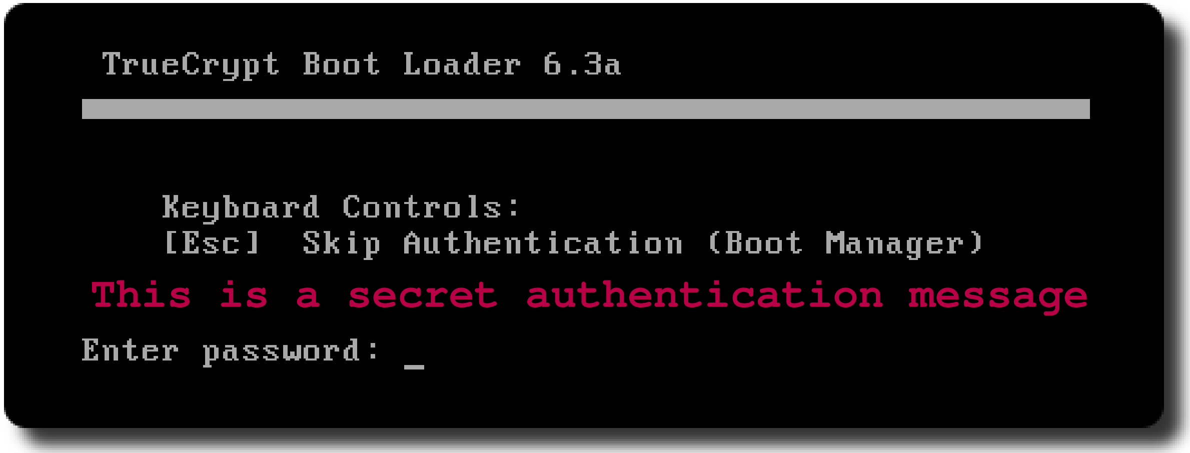
STARK Authentication

Mutually Authentication

- STARK T*amperproof* A*uthentication to* R*esist* K*eylogging*
- idea: mutually authenticate users and PCs
  1. *PC proves to user that it is not manipulated*
  2. only then, user enters password
- how can the PC prove its *integrity*?

Personalized prompts

- only if the PC is not manipulated, a user-defined message can be unsealed (**TPM**)
- this message must be shown to the user before he enters the password



Problems

- an evil maid can boot the machine, write down the unsealed message, and build a forged MBR

- Solutions?

  PIN for the TPM (i.e., “pwd > msg > pwd”)?

  No (two keylogging attacks)

  Auth. message on USB drive?

  No (MBR can clone the USB drive)

STARK

- mutual authentication in FDE by
  - *trust bootstrapping* (from USB drive)
  - *one-time prompts* (by socalled *monces*)
- “monce” = message used once

  (because humans cannot remember nonces)
- basic authentication scheme:
  1. PC auth. towards user by unsealing a monce (TPM)
  2. user auth. towards PC by a traditional password
  3. user updates the old monce with a new one

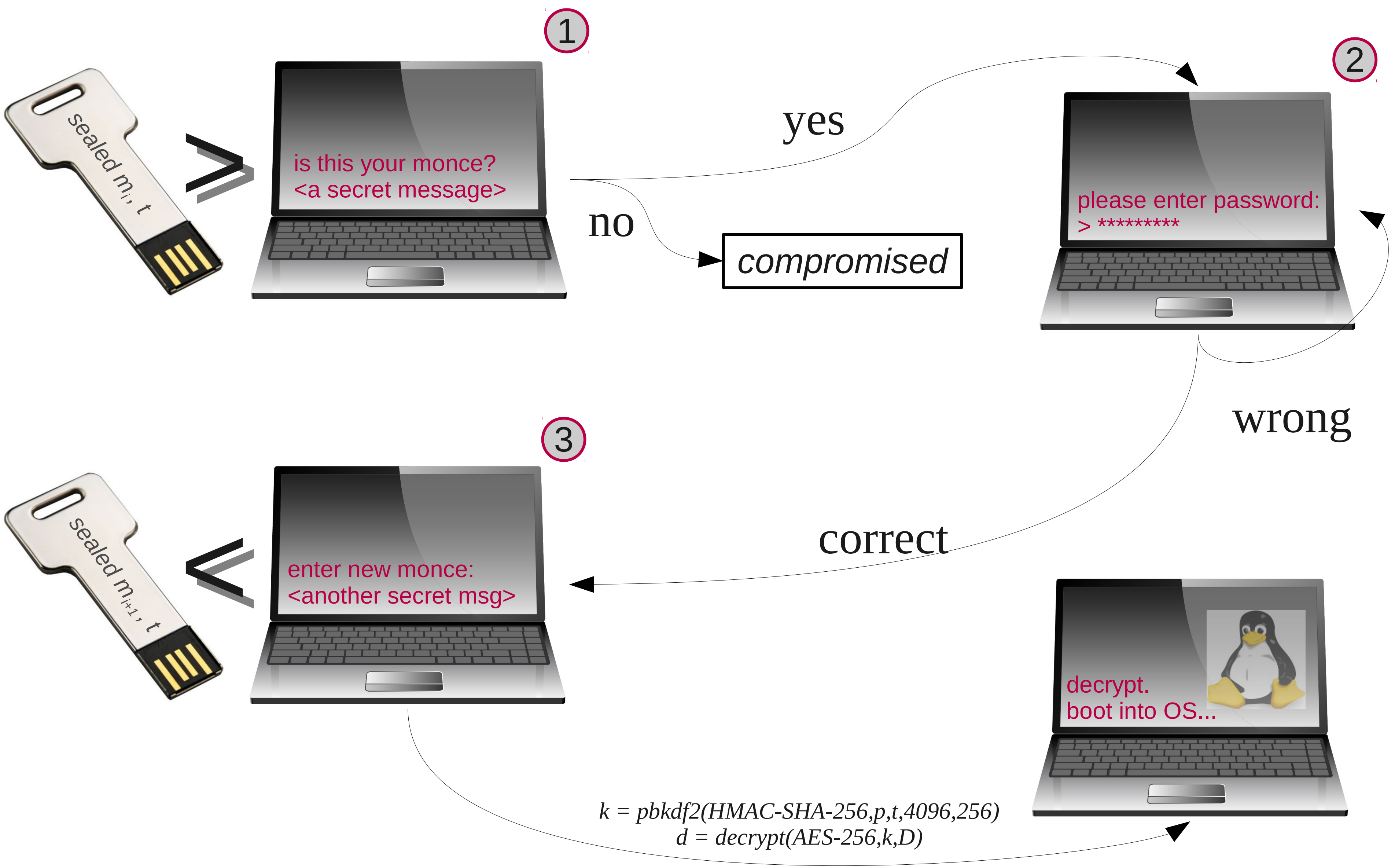
STARK: Bootstrapping Phase

USER		COMPUTER
construct p and m_0		
	p, m_0 $\longrightarrow$	
		construct t *save hash(p,t)* *seal (m_0,t)*
	$\longleftarrow$ *$sealed_C(m_0,t)$*	
save $sealed_C(m_0,t)$ to USB		

STARK: Authentication Sessions

	USER		COMPUTER	
	plug in the USB drive	$\xrightarrow{sealed_C(m_{i-1},t)}$		
			unseal (m_{i-1},t)	
1		$\xleftarrow{m_{i-1}}$		
	authenticate the computer and only then enter p			
		$\xrightarrow{p}$		
			calc hash(p,t) to authenticate the user	2
		$\xleftarrow{ACK}$		
	construct m_i			
		$\xrightarrow{m_i}$		
3			*seal* (m_i,t)	
	save & unplug USB drive	$\xleftarrow{sealed_C(m_i,t)}$		

STARK Overview



STARK Characteristics

- *one-time boot prompts*
  - each auth. message is valid for only one boot
- *trust bootstrapping*
  - USB drive must be handled like a physical key
- additionally: *two-factor authentication*
  - USB drive is bounded to decryption process

► ►evil maids catch outdated monces only!

Chapter 3

POTTS Implementation

POTTS

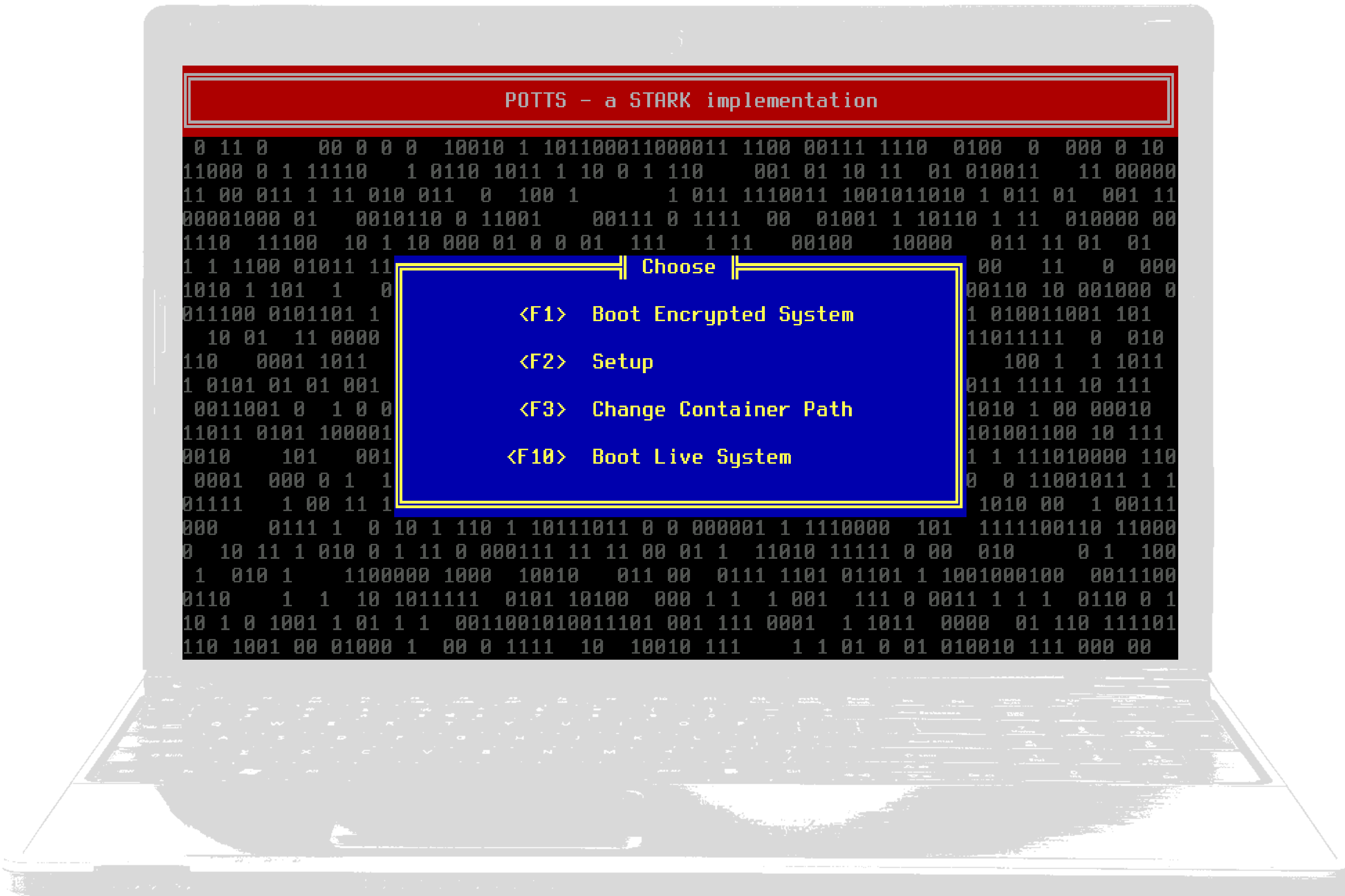
- Linux-based Implementation of STARK
- POTTS: P*revents* O*pportunistic and* T*argeted* T*hreat Scenarios*
  - *targeted*: system manipulation (by STARK)
  - *opportunistic*: physical loss and theft (by TRESOR)

TRESOR

- Academic disk encryption solution
- T. Müller, F. Freiling, A. Dewald (USENIX 2010)
  TRESOR Runs Encryption Securely Outside RAM
- Prevents *cold boot attacks*

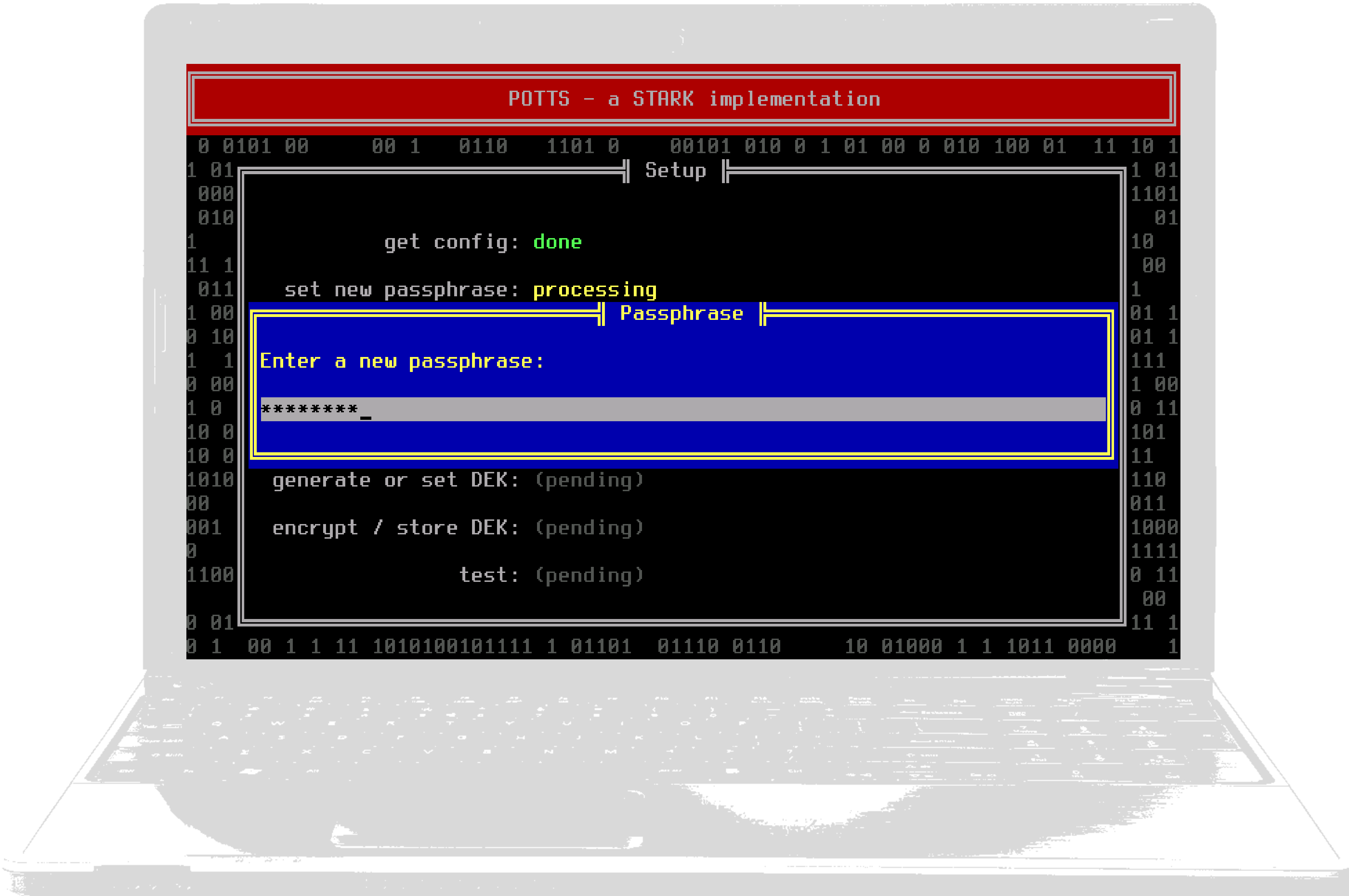
(BitLocker, TrueCrypt, etc. are all vulnerable to cold boot attacks)

Setup Phase



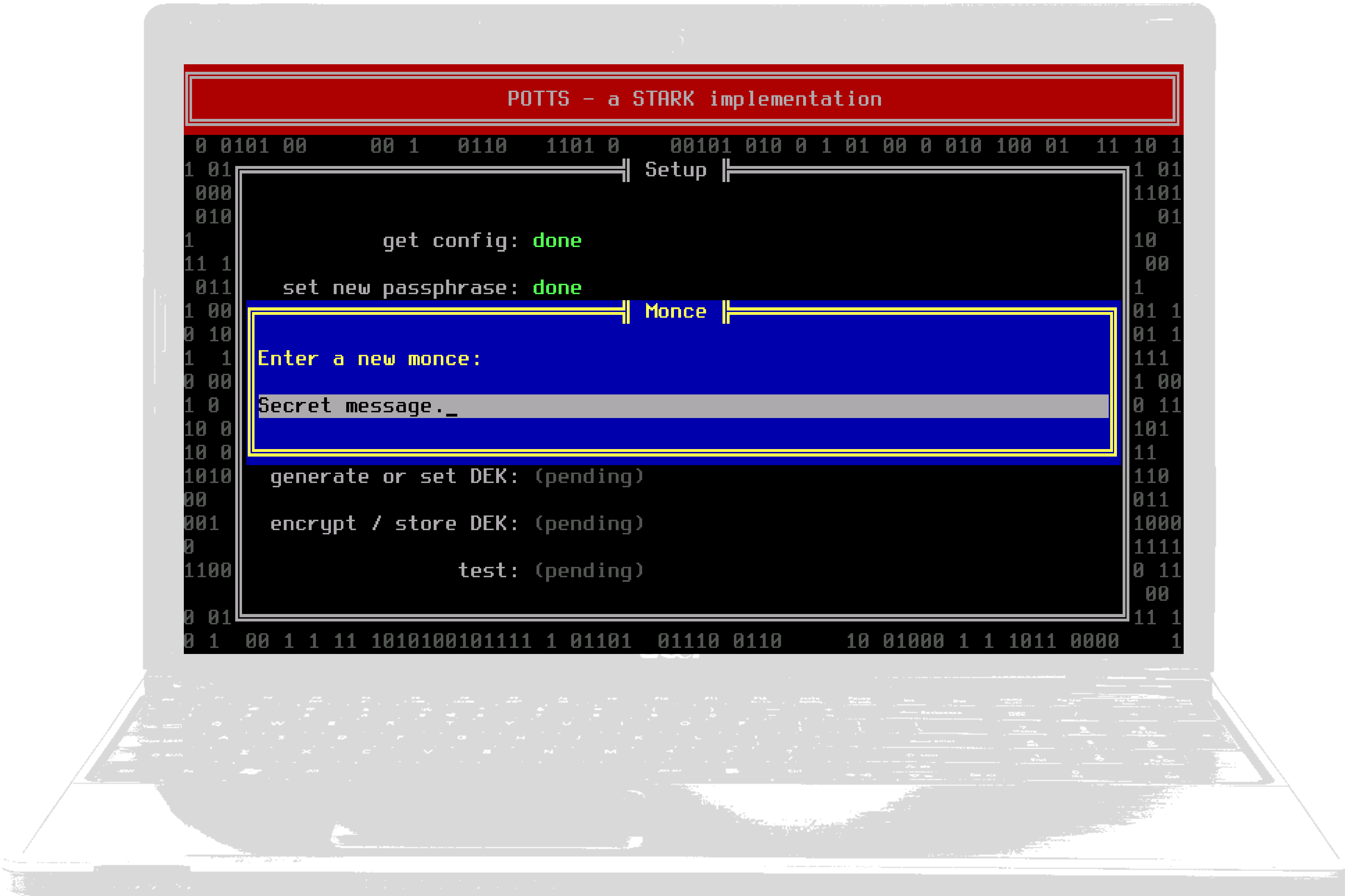
POTTS – a STARK implementation
Choose
<F1> Boot Encrypted System
<F2> Setup
<F3> Change Container Path
<F10> Boot Live System

Setup Phase

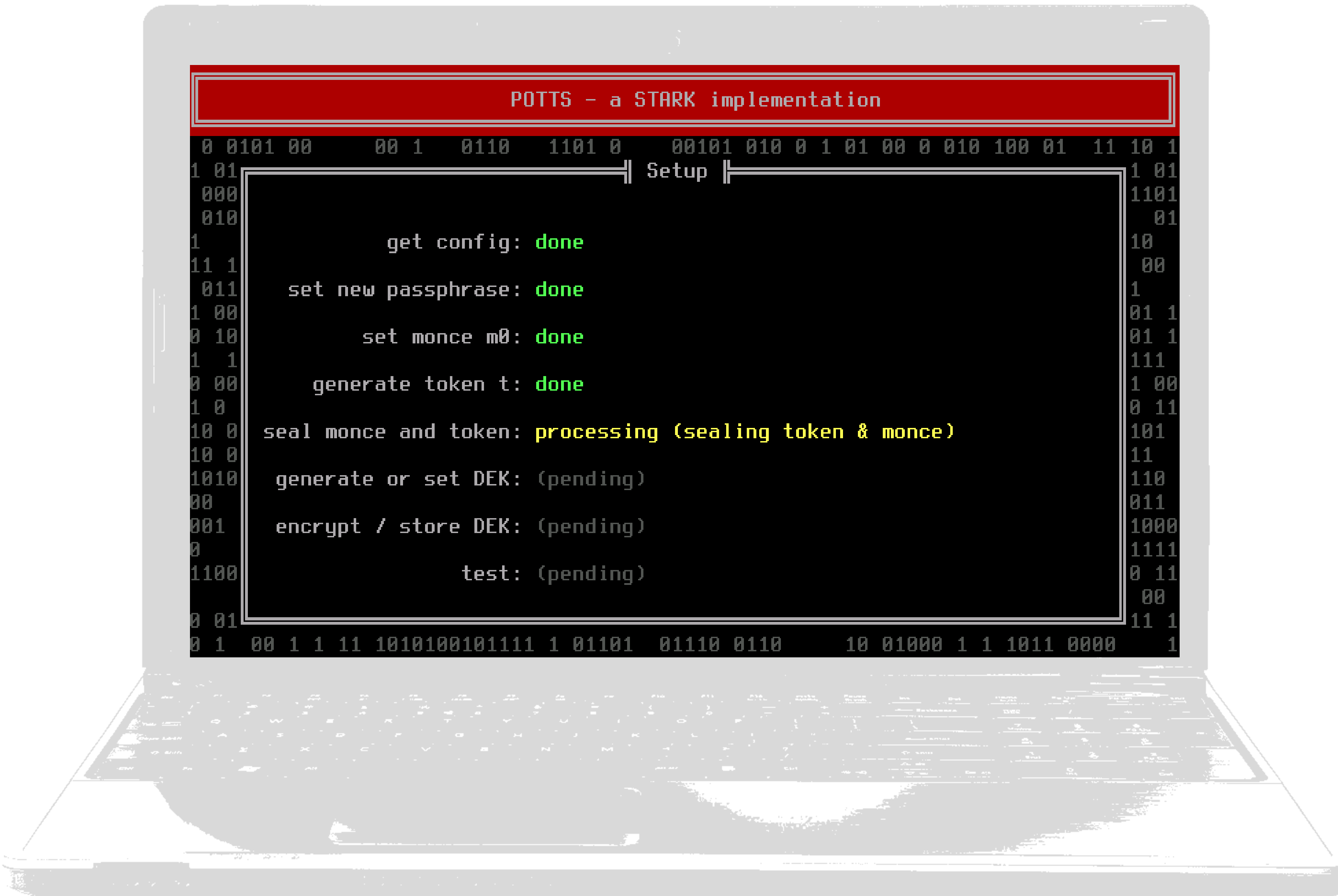


POTTS - a STARK implementation
Setup
get config: done
set new passphrase: processing
Passphrase
Enter a new passphrase:
********_
generate or set DEK: (pending)
encrypt / store DEK: (pending)
test: (pending)

Setup Phase

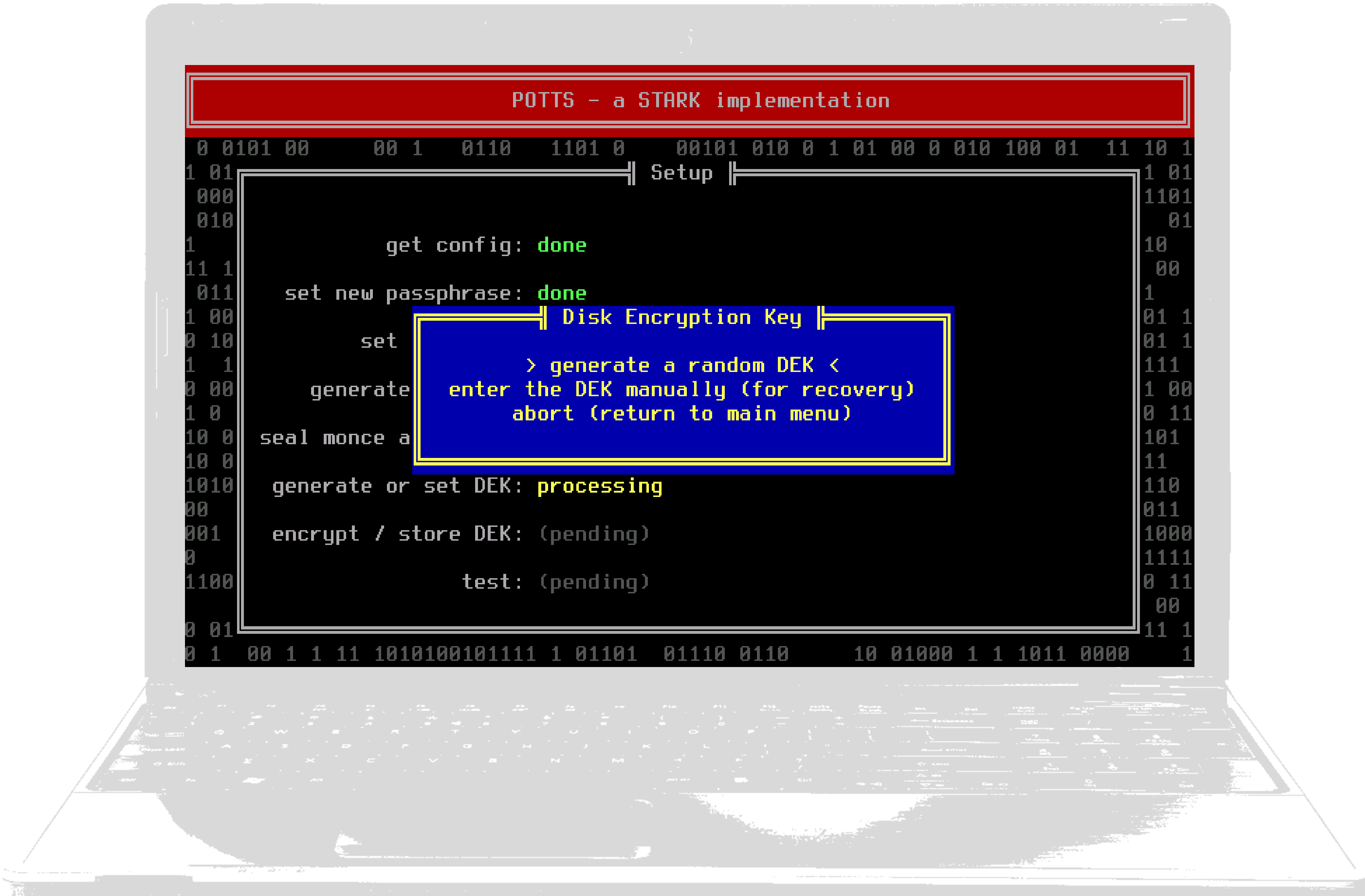

POTTS - a STARK implementation
Setup
get config: done
set new passphrase: done
Monce
Enter a new monce:
Secret message._
generate or set DEK: (pending)
encrypt / store DEK: (pending)
test: (pending)

Setup Phase

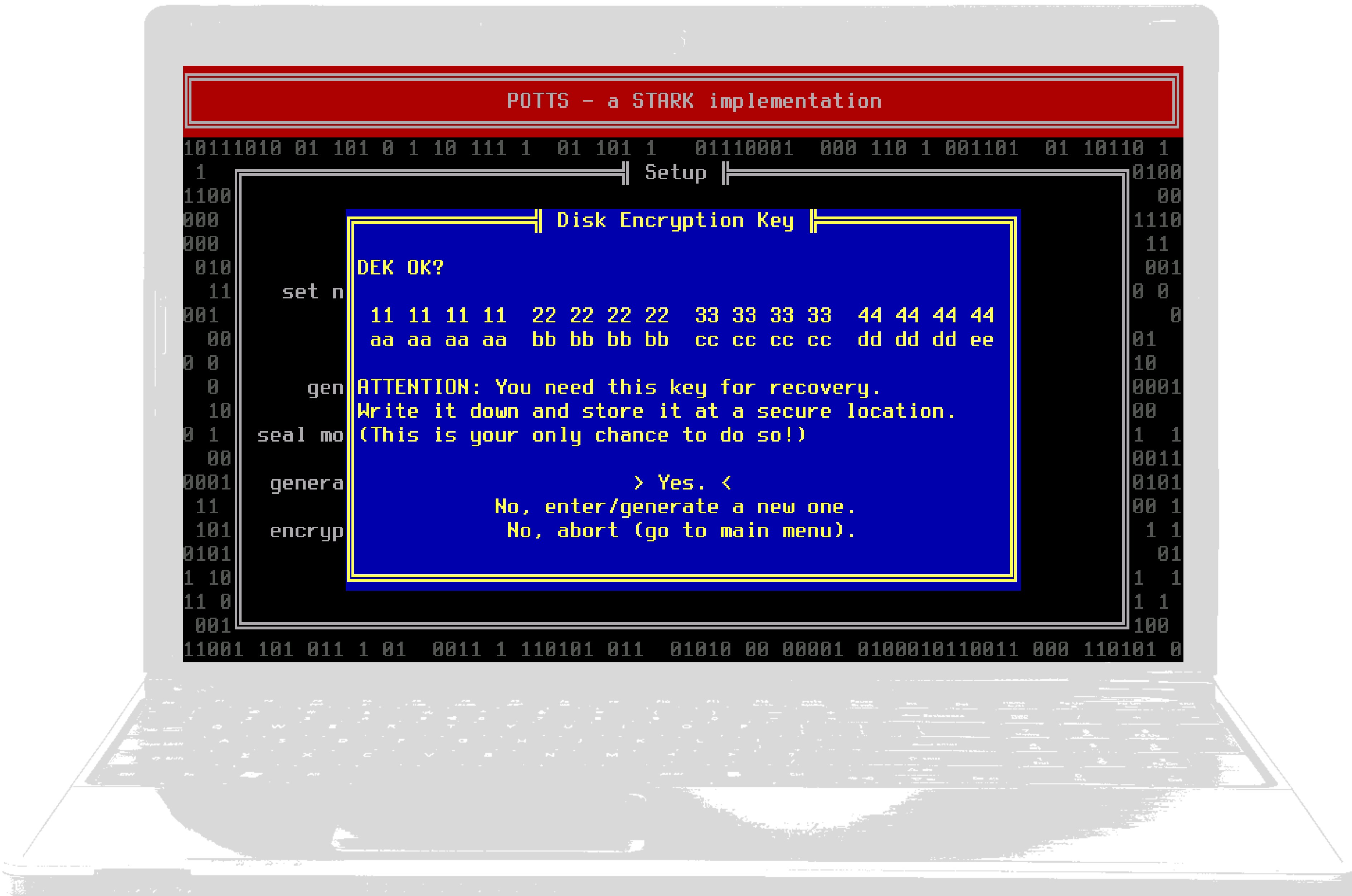


POTTS - a STARK implementation
Setup
get config: done
set new passphrase: done
set monce m0: done
generate token t: done
seal monce and token: processing (sealing token & monce)
generate or set DEK: (pending)
encrypt / store DEK: (pending)
test: (pending)

Setup Phase

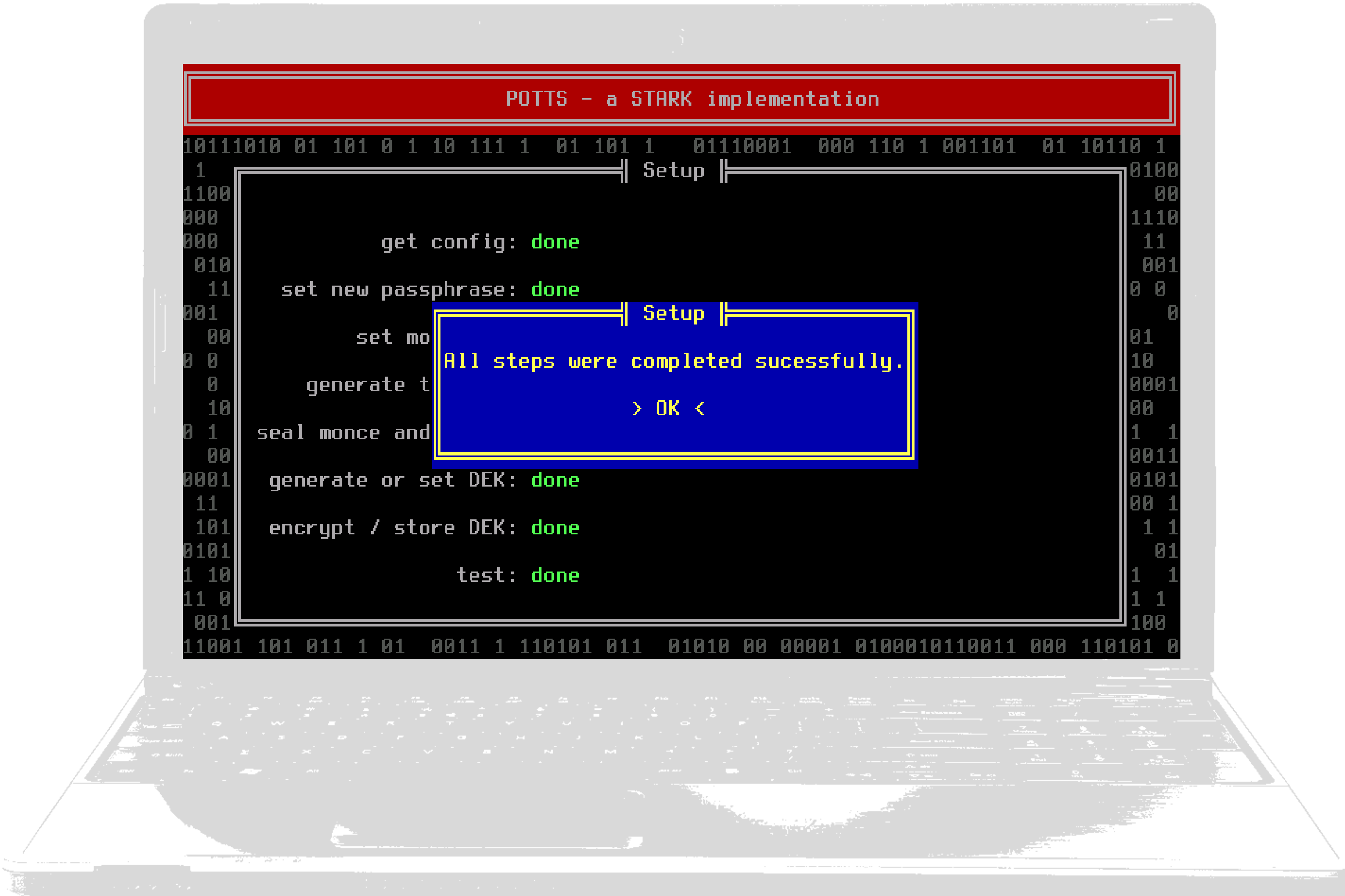

POTTS – a STARK implementation
Setup
get config: done
set new passphrase: done
set
generate
seal monce a
generate or set DEK: processing
encrypt / store DEK: (pending)
test: (pending)
Disk Encryption Key
> generate a random DEK <
enter the DEK manually (for recovery)
abort (return to main menu)

Setup Phase



POTTS – a STARK implementation
Setup
Disk Encryption Key
DEK OK?
11 11 11 11 22 22 22 22 33 33 33 33 44 44 44 44
aa aa aa aa bb bb bb bb cc cc cc cc dd dd dd ee
ATTENTION: You need this key for recovery.
Write it down and store it at a secure location.
(This is your only chance to do so!)
> Yes. <
No, enter/generate a new one.
No, abort (go to main menu).

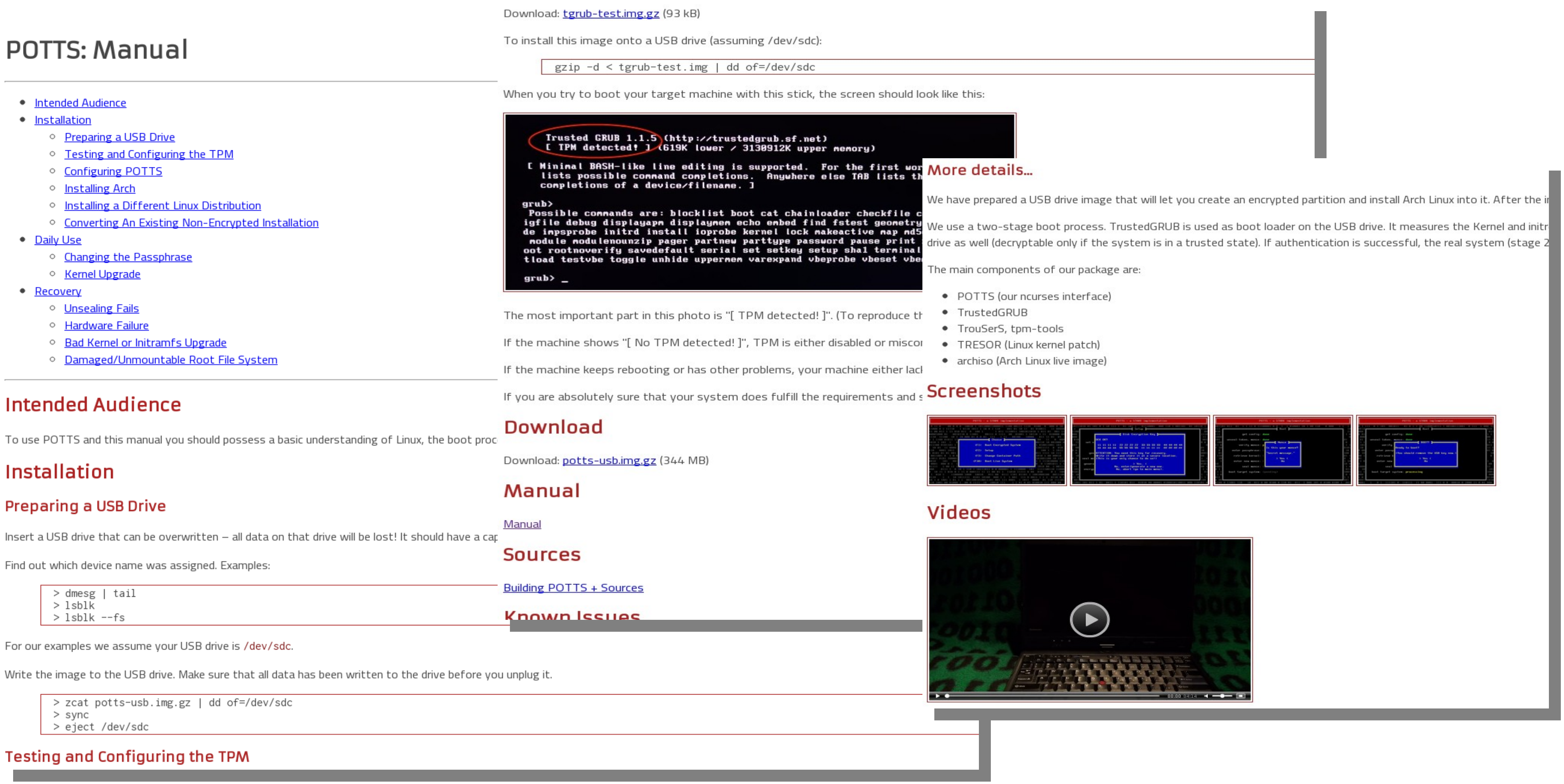
Setup Phase


POTTS - a STARK implementation
Setup
get config: done
set new passphrase: done
set mo
generate t
seal monce and
generate or set DEK: done
encrypt / store DEK: done
test: done
Setup
All steps were completed sucessfully.
> OK <

Authentication Session (Video)

Availability

- STARK: http://www1.cs.fau.de/stark/
- POTTS: http://13.tc/potts/



POTTS: Manual

- Intended Audience
- Installation
  - Preparing a USB Drive
  - Testing and Configuring the TPM
  - Configuring POTTS
  - Installing Arch
  - Installing a Different Linux Distribution
  - Converting An Existing Non-Encrypted Installation
- Daily Use
  - Changing the Passphrase
  - Kernel Upgrade
- Recovery
  - Unsealing Fails
  - Hardware Failure
  - Bad Kernel or Initramfs Upgrade
  - Damaged/Unmountable Root File System

Intended Audience

To use POTTS and this manual you should possess a basic understanding of Linux, the boot proc

Installation

Preparing a USB Drive

Insert a USB drive that can be overwritten – all data on that drive will be lost! It should have a cap

Find out which device name was assigned. Examples:

```
> dmesg | tail
> lsblk
> lsblk --fs
```

For our examples we assume your USB drive is `/dev/sdc`.

Write the image to the USB drive. Make sure that all data has been written to the drive before you unplug it.

```
> zcat potts-usb.img.gz | dd of=/dev/sdc
> sync
> eject /dev/sdc
```

Testing and Configuring the TPM

Download: tgrub-test.img.gz (93 kB)

To install this image onto a USB drive (assuming /dev/sdc):

```
gzip -d < tgrub-test.img | dd of=/dev/sdc
```

When you try to boot your target machine with this stick, the screen should look like this:

```
Trusted GRUB 1.1.5 (http://trustedgrub.sf.net)
[ TPM detected! ] (619K lower / 3130912K upper memory)

[ Minimal BASH-like line editing is supported.  For the first wor
  lists possible command completions.  Anywhere else TAB lists th
  completions of a device/filename. ]

grub>
 Possible commands are: blocklist boot cat chainloader checkfile c
igfile debug displayapm displaymem echo embed find fstest geometry
de impsprobe initrd install ioprobe kernel lock makeactive map md5
 module modulenounzip pager partnew parttype password pause print
oot rootnoverify savedefault serial set setkey setup shal terminal
tload testvbe toggle unhide uppermem varexpand vbeprobe vbeset vbe

grub> _
```

The most important part in this photo is "[TPM detected!]". (To reproduce th

If the machine shows "[No TPM detected!]", TPM is either disabled or miscor

If the machine keeps rebooting or has other problems, your machine either lacl

If you are absolutely sure that your system does fulfill the requirements and s

Download

Download: potts-usb.img.gz (344 MB)

Manual

Manual

Sources

Building POTTS + Sources

Known Issues

More details...

We have prepared a USB drive image that will let you create an encrypted partition and install Arch Linux into it. After the i

We use a two-stage boot process. TrustedGRUB is used as boot loader on the USB drive. It measures the Kernel and initr drive as well (decryptable only if the system is in a trusted state). If authentication is successful, the real system (stage 2

The main components of our package are:

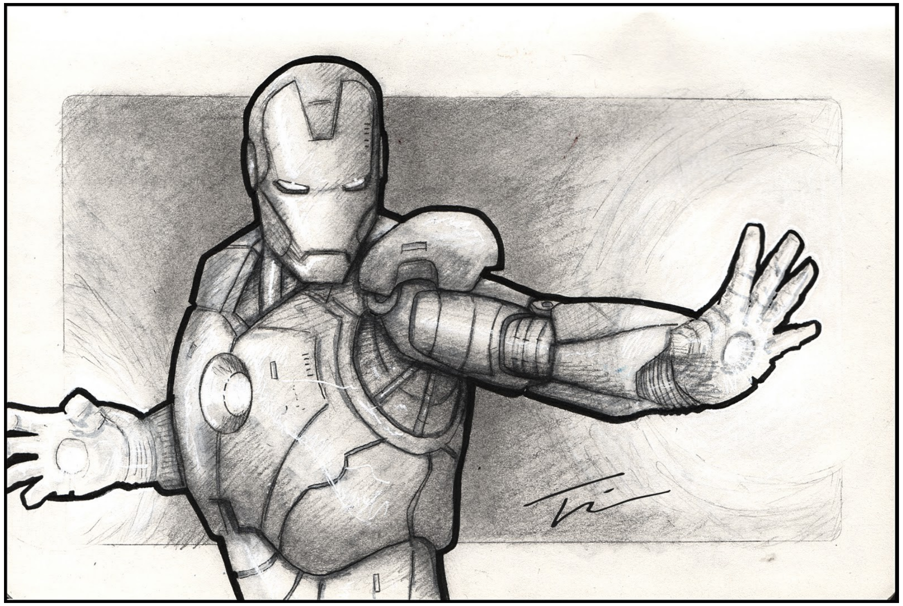
- POTTS (our ncurses interface)
- TrustedGRUB
- TrouSerS, tpm-tools
- TRESOR (Linux kernel patch)
- archiso (Arch Linux live image)

Screenshots

Videos

Chapter 4

Limitations and Future Work



Limitations

- POTTS enables users to *identify* a system compromise, but does not regulate which actions to take afterwards
- STARK defeats only traditional evil maid attacks

  [x] software-based boot manipulations

  [ ] hardware-based attacks

Future Work

- monces are hard to generate and remember
- problem: one communication end point is human (nonces are even harder to generate and remember).

- future: use active USB drives and real nonces

Questions?