

SECURING AUTOMATED DECRYPTION

Thorsten Scherf
Red Hat

FFG 2019, Karlsruhe,

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

Booting...

Disk Password:

YESTERDAY

Standards (AES, PCI-DSS, etc.)

TODAY

Automation

TOMORROW

Policy

YESTERDAY

TODAY

TOMORROW

Standards (AES, PCI-DSS, etc.)

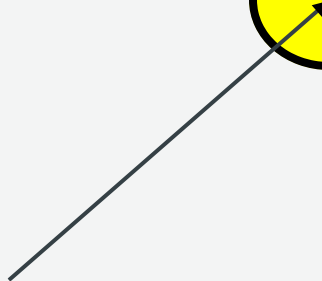
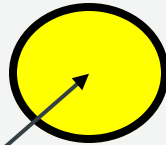
Automation

Policy

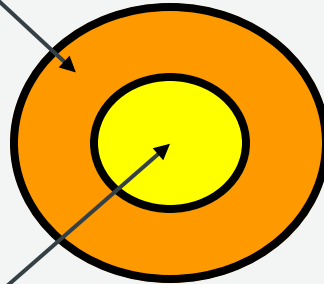


HOW DO WE AUTOMATE?

Shh... I'm Secret!



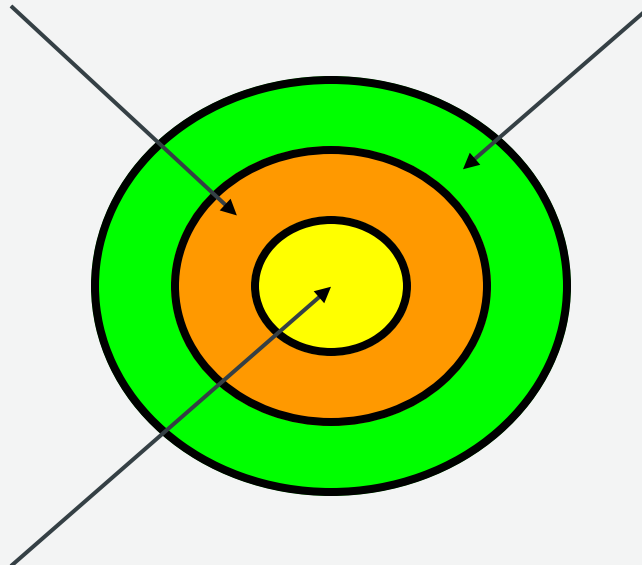
Encryption Key



Shh... I'm Secret!

Encryption Key

Key Encryption Key



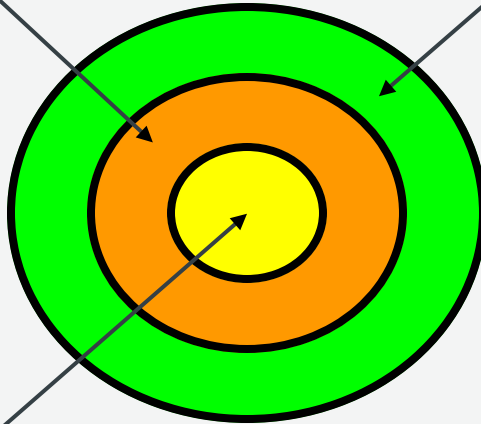
Shh... I'm Secret!

Encryption Key

Key Encryption Key

"correct battery horse staple"

Shh... I'm Secret!



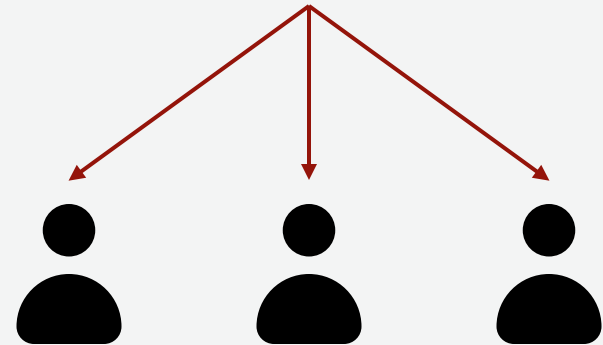
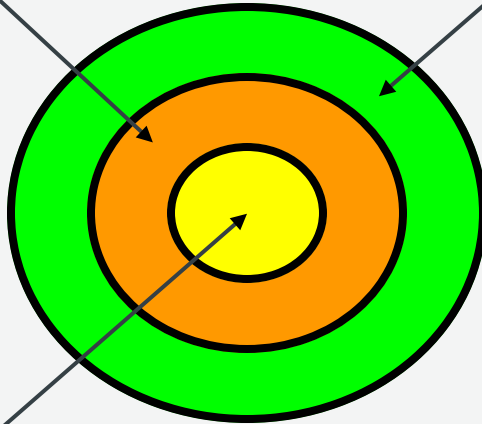
STANDARD PASSWORD MODEL

Encryption Key

Key Encryption Key

"correct horse battery staple"

Shh... I'm Secret!

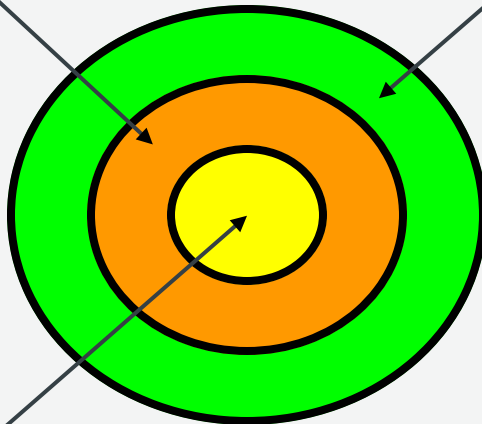


Encryption Key

Key Encryption Key

"d41d8cd9...ecf8427e"

Shh... I'm Secret!



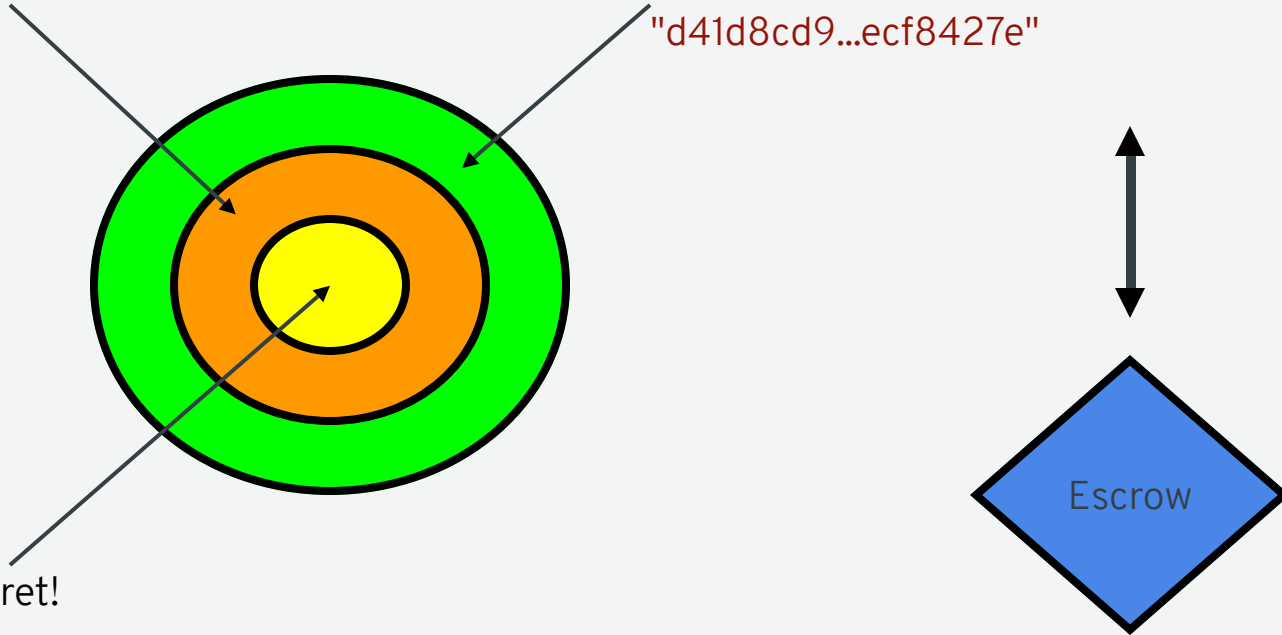
STANDARD ESCROW MODEL?

Encryption Key

Key Encryption Key

"d41d8cd9...ecf8427e"

Shh... I'm Secret!



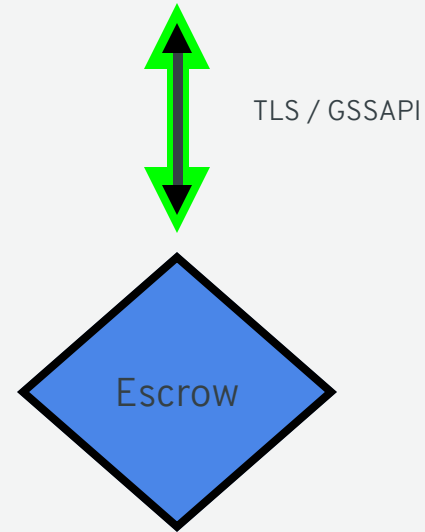
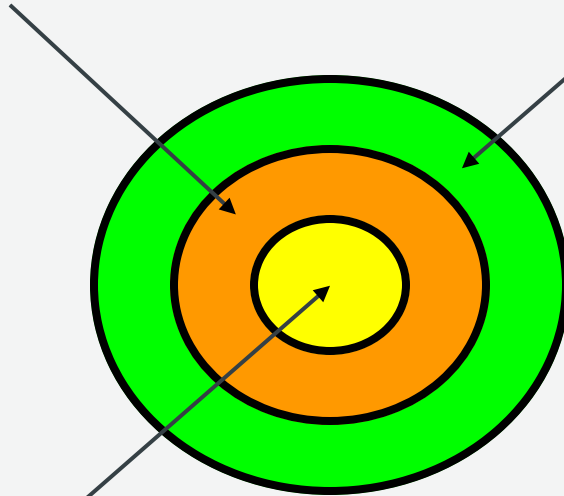
STANDARD ESCROW MODEL?

Encryption Key

Key Encryption Key

"d41d8cd9...ecf8427e"

Shh... I'm Secret!



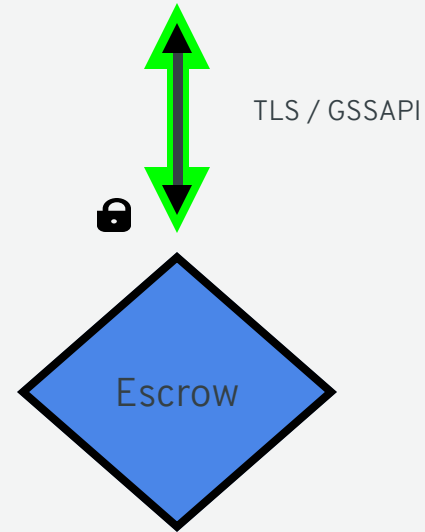
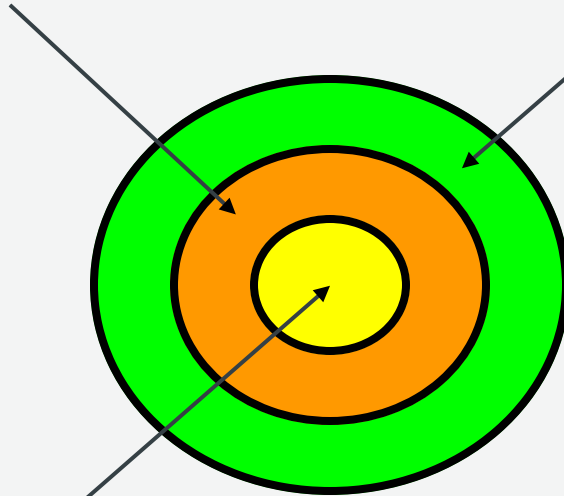
STANDARD ESCROW MODEL?

Encryption Key

Key Encryption Key

"d41d8cd9...ecf8427e"

Shh... I'm Secret!



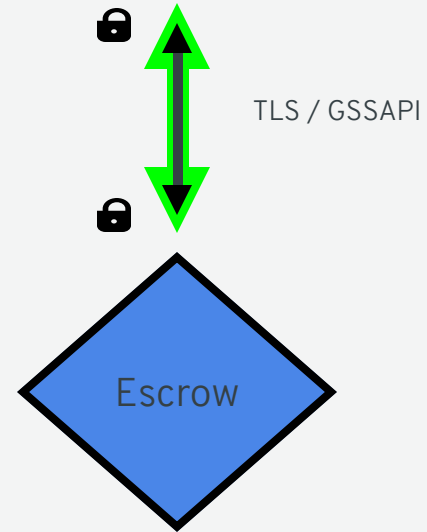
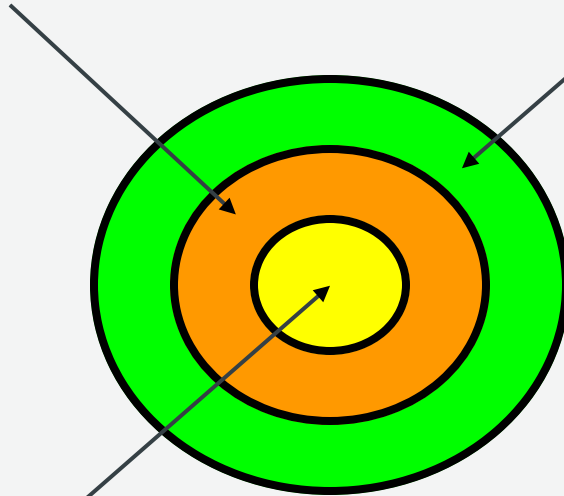
STANDARD ESCROW MODEL?

Encryption Key

Key Encryption Key

"d41d8cd9...ecf8427e"

Shh... I'm Secret!



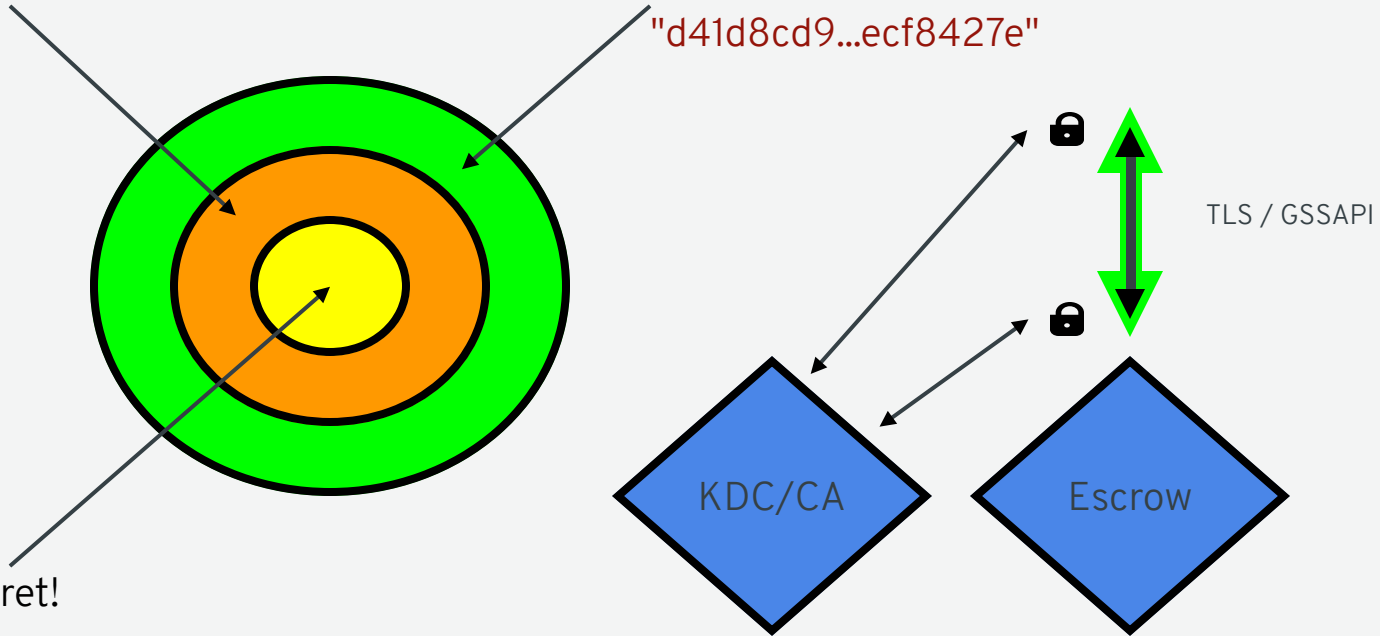
STANDARD ESCROW MODEL?

Encryption Key

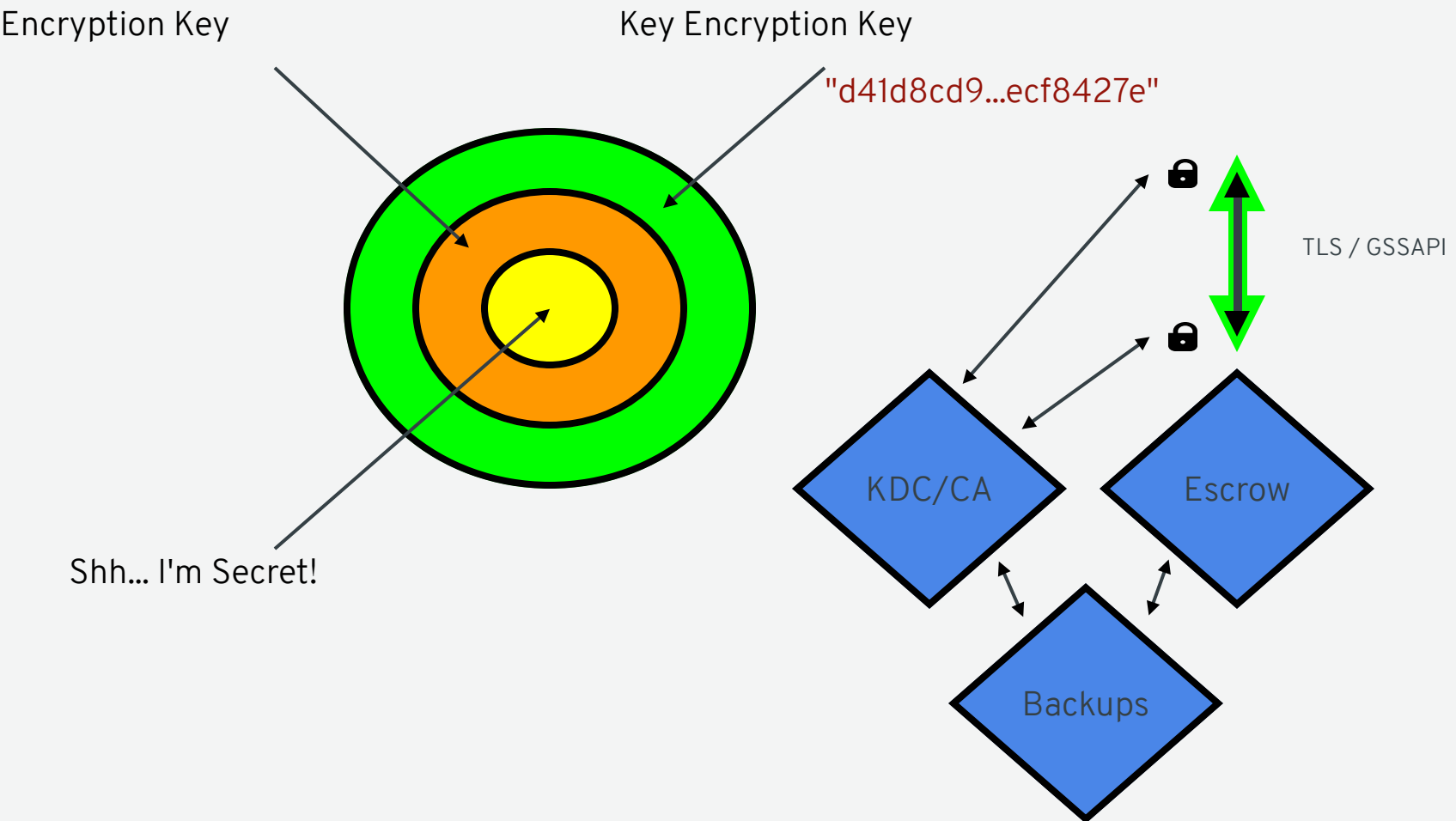
Key Encryption Key

"d41d8cd9...ecf8427e"

Shh... I'm Secret!



STANDARD ESCROW MODEL



STANDARD ESCROW MODEL

Encryption Key

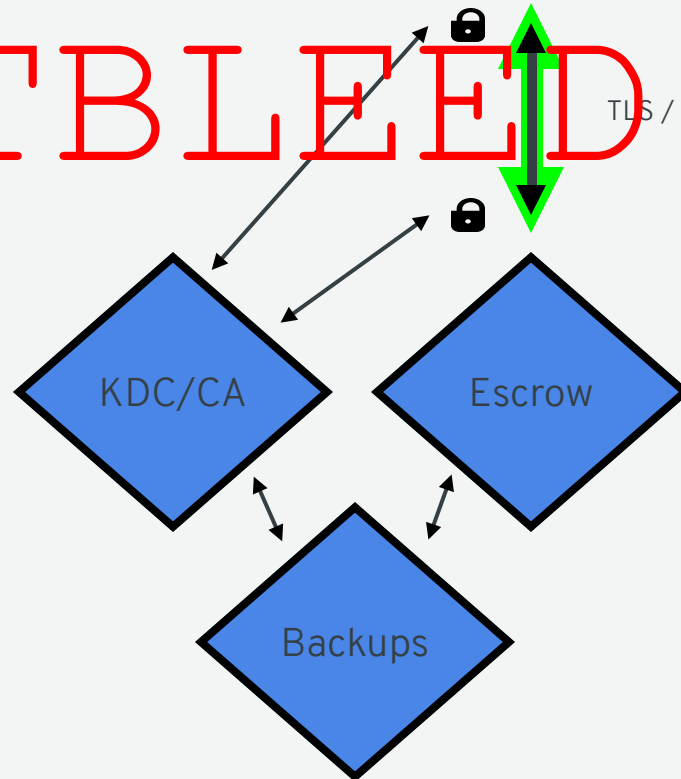
Key Encryption Key

"d41d8cd9...ecf8427e"

HEARTBLEED

TLS / GSSAPI

Shh... I'm Secret!



LESSONS LEARNED

- Presuming TLS will protect key transfer is dangerous
- Complexity increases attack surface
- Escrows are difficult to deploy
- X.509 is hard to get right

ASYMMETRIC CRYPTO?

BRACE YOURSELVES,

MATH IS COMING

imgflip.com



(EC) DIFFIE-HELLMAN KEY EXCHANGE

$$C \in_R [1, p-1]$$

$$c = gC$$

$c \longrightarrow$

$$K = gSC = sC$$

$$S \in_R [1, p-1]$$

$$s = gS$$

$\longleftarrow s$

$$K = gCS = cS$$

MCCALLUM-RELYEA KEY EXCHANGE

PROVISIONING

$C \in_R [1, p-1]$
 $c = gC$
 $K = gSC = sC$
Discard : K, C
Retain : s, c

$S \in_R [1, p-1]$
 $s = gS$
 $\leftarrow s$

RECOVERY

$E \in_R [1, p-1]$
 $e = gE$
 $x = c + e$
 $x \longrightarrow$

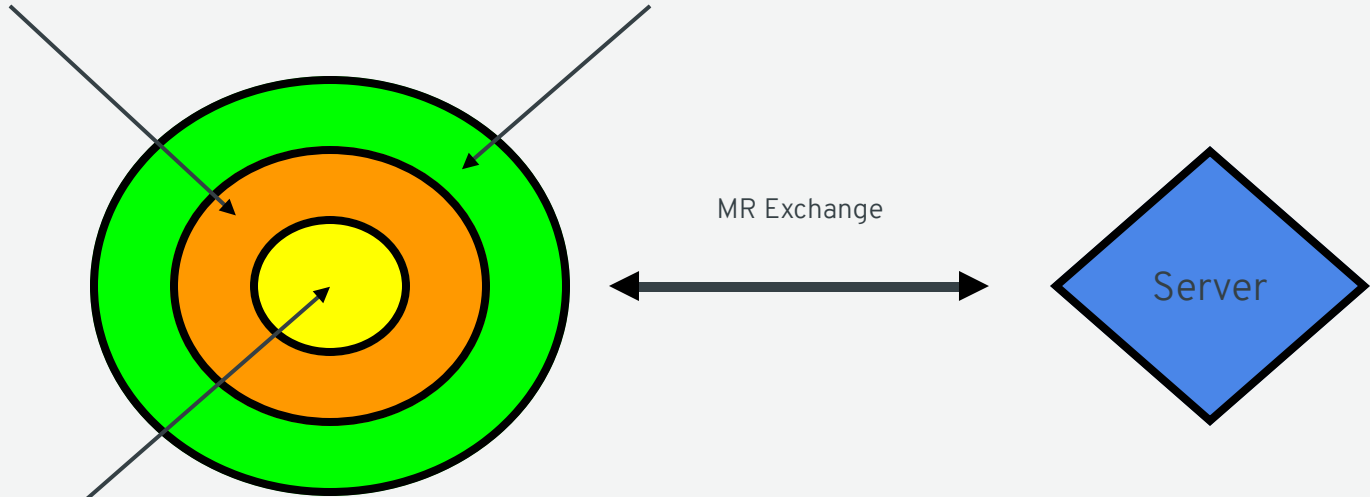
$y = xS$
 $\leftarrow y$
 $K = y - sE$
Because : $K = gCS + gES - gSE$

To keep c private, e & E **MUST** be private.

Encryption Key

Key Encryption Key

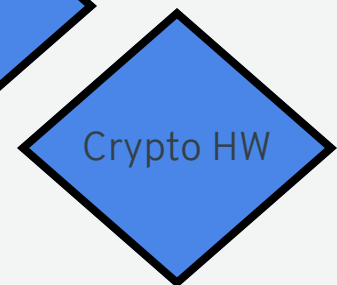
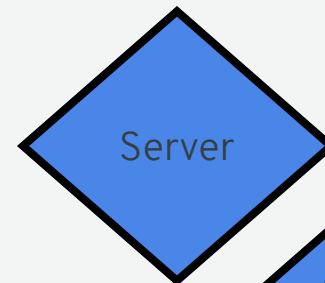
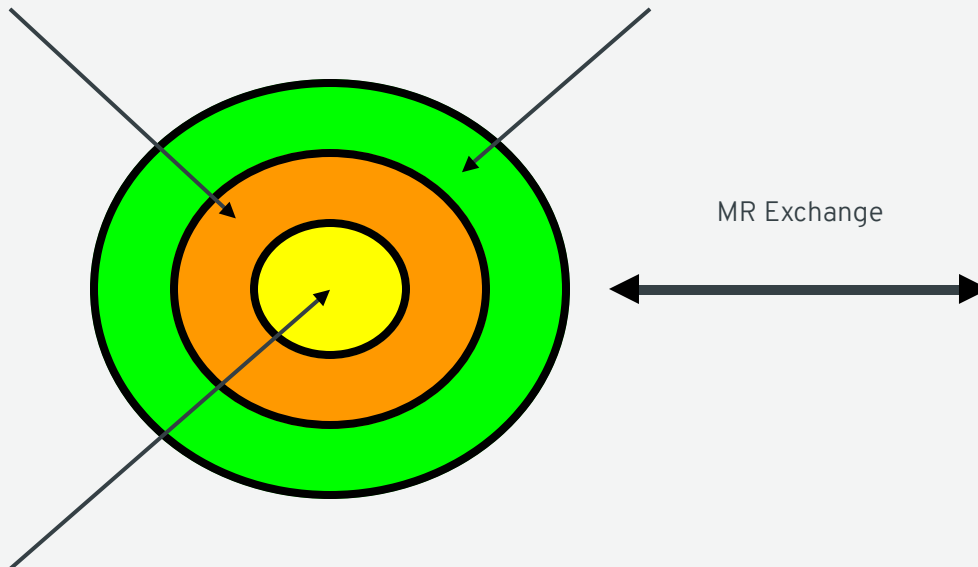
Shh... I'm Secret!



Encryption Key

Key Encryption Key

Shh... I'm Secret!



Property	Escrow	MR Exchange
Server presence during provisioning	Required	Optional
Server presence during recovery	Required	Required
Server knowledge of keys	Required	None
Key transfer	Required	None
Client authentication	Required	Optional
Transport encryption	Required	Optional
End-to-end Encryption	Difficult	Unneeded

TANG

- <https://github.com/latchset/tang>
- Server-side daemon
- Simple: HTTP + JOSE
- Fast (>2k req/sec)
- Extremely small
- Minimal dependencies
- Fedora 26+, RHEL 7.4, Debian (soon)

INSTALLING A TANG SERVER

```
$ sudo dnf install tang
```

```
$ sudo systemctl enable --now tangd.socket
```

ON THE CLIENT...

CLEVIS

- <https://github.com/latchset/clevis/>
- Decryption automation and policy framework
- Minimal dependencies
- Early boot integration
- GNOME integration
- Fedora 26+, RHEL 7.4, Debian (soon)

BASIC ENCRYPTION WITH TANG

```
$ dnf install clevis
```

```
$ echo PT | clevis encrypt tang '{"url":"http://localhost"}' > mydata.jwe
```

The advertisement is signed with the following keys:

```
haD7Y-8VkAyJo6-vdZMrGQXCSfI
```

Do you wish to trust the advertisement? [yN] y

```
$ cat mydata.jwe
```

```
{"ciphertext":"-059czAqybvxDme2t3I5A", ...}
```

```
$ clevis decrypt < mydata.jwe
```

```
PT
```

```
$ sudo systemctl stop tangd.socket
```

```
$ clevis decrypt < mydata.jwe
```

```
$ echo $?
```

```
1
```

BASIC ENCRYPTION WITH AN ESCROW

```
$ dnf install clevis
```

```
$ echo PT | clevis encrypt http '{"url":"http://localhost/key"}' > mydata.jwe
```

```
$ cat mydata.jwe
```

```
{"ciphertext":"-O59czAqybvxHdme2t3I5A", ...}
```

```
$ clevis decrypt < mydata.jwe
```

```
PT
```

DISK BINDING WITH TANG

```
$ sudo clevis bind luks -d /dev/sda1 tang '{"url":"http://tang.srv"}'
```

The advertisement is signed with the following keys:

```
haD7Y-8VKAyJo6-vdZMrGQXCSfI
```

Do you wish to trust the advertisement? [yN] y

Enter passphrase **for** /dev/sda1:

```
$ sudo luksmeta show -d /dev/sda1
```

```
0 active empty
```

```
1 active cb6e8904-81ff-40da-a84a-07ab9ab5715e
```

```
2 inactive empty
```

```
3 inactive empty
```

```
...
```

```
# For root volume unlocking at boot:
```

```
$ sudo dnf install clevis-dracut
```

```
$ sudo dracut -f
```

```
$ reboot
```

```
# For removable storage GNOME unlocking:
```

```
$ sudo dnf install clevis-udisks2
```

FROM AUTOMATION TO POLICY

YESTERDAY

Standards (AES, PCI-DSS, etc.)

TODAY

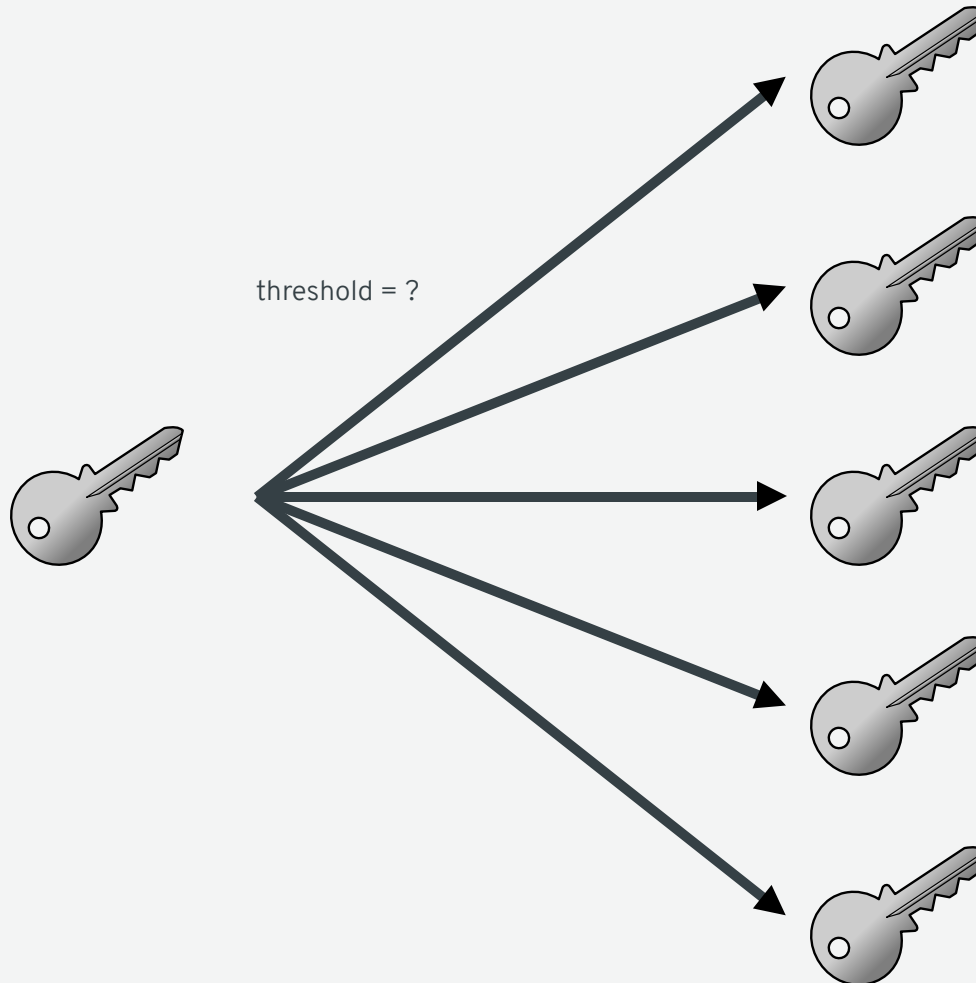
Automation

TOMORROW

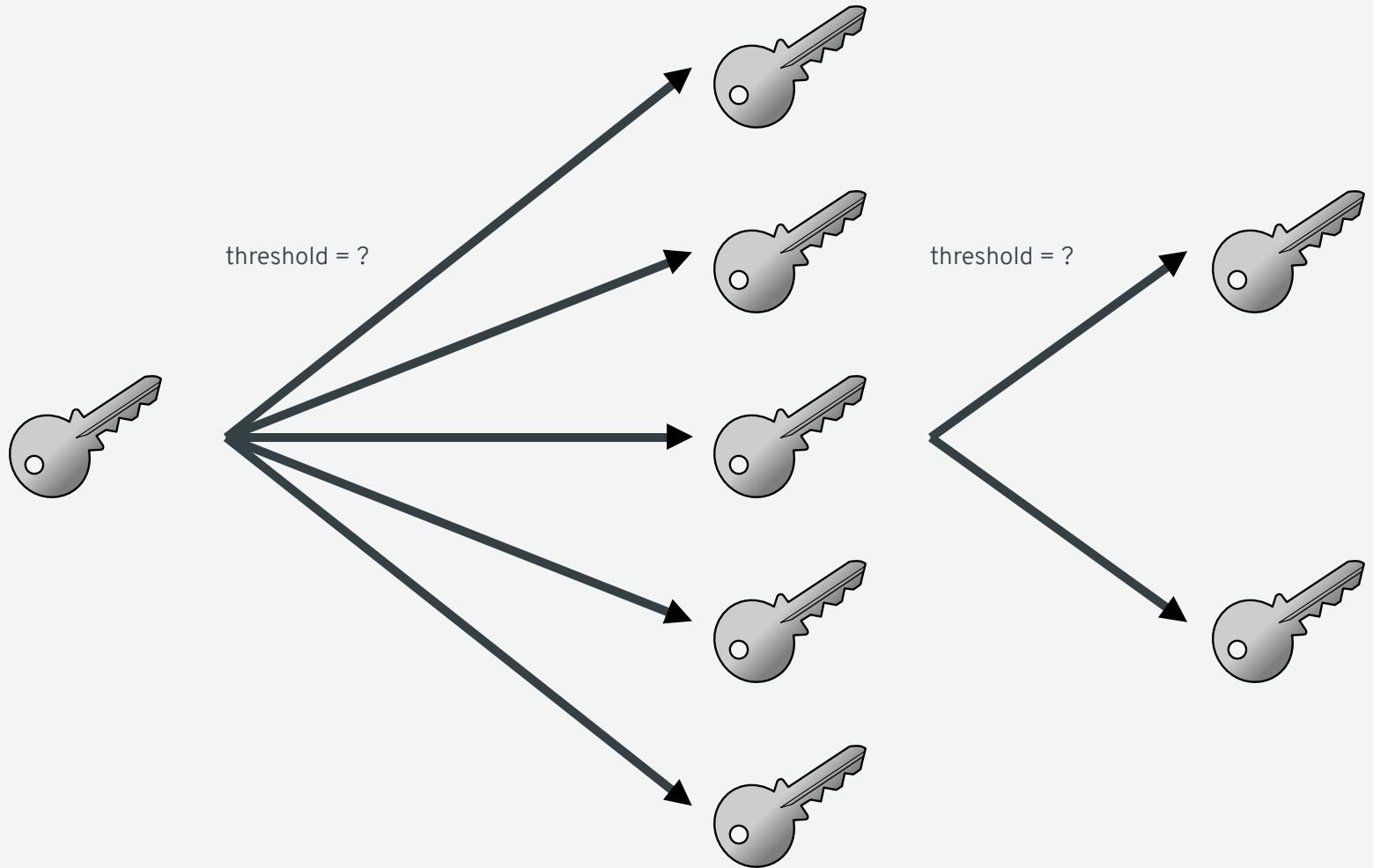
Policy



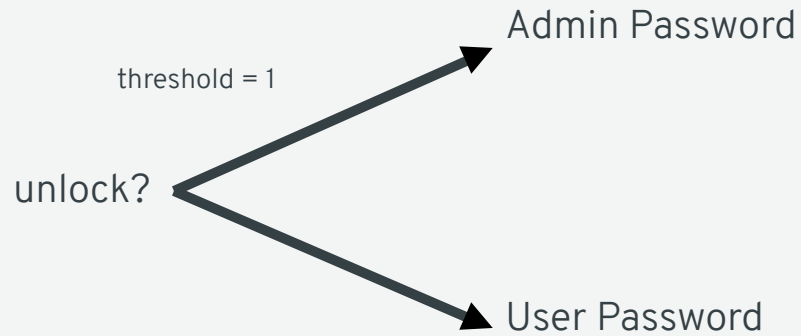
SHAMIR SECRET SHARING



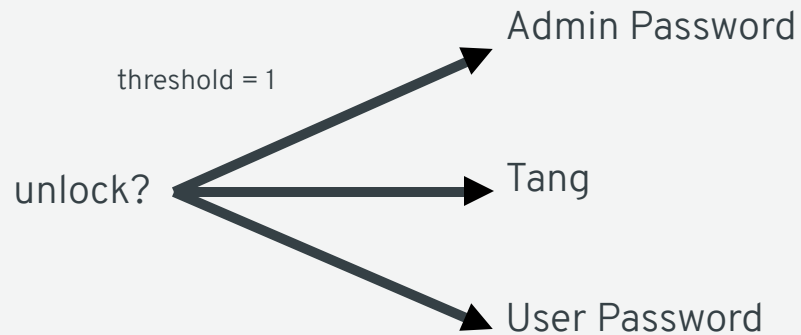
SHAMIR SECRET SHARING



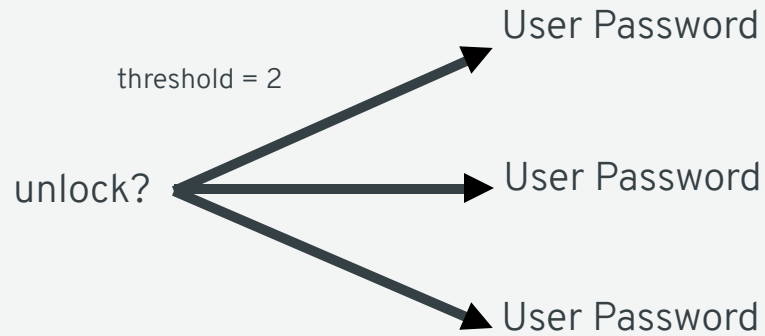
SIMPLE LAPTOP



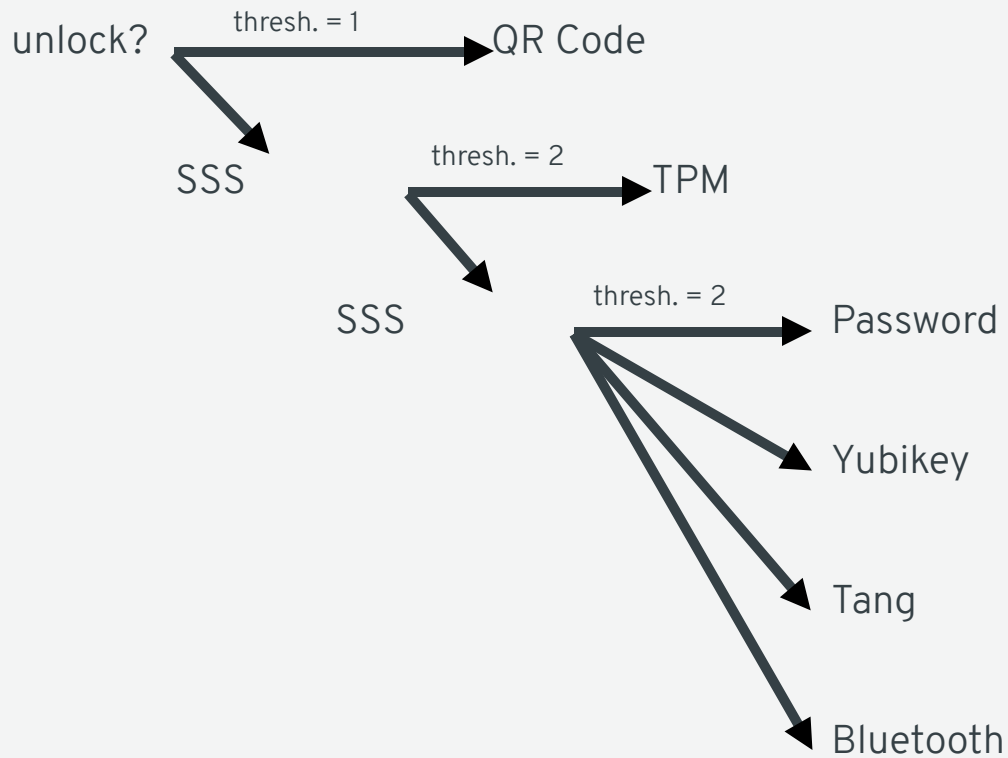
AUTOMATED LAPTOP



HIGH SECURITY SYSTEM



SOPHISTICATED LAPTOP POLICY



BASIC SHAMIR'S WITH TANG

```
$ echo PT | clevis encrypt sss \  
'{"pins": {"tang": [{"url": "http://a.tang.srv"}, {"url": "http://b.tang.srv"}]}, "t": 1}' \  
> out.jwe
```

The advertisement is signed with the following keys:

```
haD7Y-8VkAyJo6-vdZMrGQXCSfI
```

Do you wish to trust the advertisement? [yN] y

The advertisement is signed with the following keys:

```
Edp-ESShUx4_95kGt-DTsCBbPag
```

Do you wish to trust the advertisement? [yN] y

```
$ clevis decrypt < out.jwe
```

```
PT
```

```
# Bring Down Tang Server A
```

```
$ clevis decrypt < out.jwe
```

```
PT
```

```
# Bring Down Tang Server B
```

```
$ clevis decrypt < out.jwe
```

```
$ echo $?
```

```
1
```

EXPLORING THE ECOSYSTEM

DEPENDENCY: JOSÉ

- <https://github.com/latchset/jose>
- JSON Object Signing and Encryption
- C Library & Command Line Utility
- Bottom Line: User-Friendly, Standards Compliant Crypto

```
$ jose jwk gen -i '{"alg": "A128GCM"}' -o oct.jwk
$ jose jwk gen -i '{"alg": "RSA1_5"}' -o rsa.jwk
$ jose jwk gen -i '{"alg": "ES256"}' -o ec.jwk

$ echo hi | jose jwe enc -i- -k rsa.pub.jwk -o msg.jwe
$ jose jwe dec -i msg.jwe -k rsa.jwk
hi
$ jose jwe dec -i msg.jwe -k oct.jwk
Decryption failed!

$ echo hi | jose jws sig -i- -k ec.jwk -o msg.jws
$ jose jws ver -i msg.jws -k ec.pub.jwk
hi
$ jose jws ver -i msg.jws -k oct.jwk
No signatures validated!
```

DEPENDENCY: LUKSMETA

- <https://github.com/latchset/luksmeta>
- Store metadata in LUKSv1 header gap
- C library & Command Line Utility

```
$ echo hi | luksmeta save -d /dev/sdc1 -s 2 -u EC998562-B60D-47F0-A579-DCA8C12F5BF6
```

```
$ luksmeta load -d /dev/sdc1 -s 2 -u EC998562-B60D-47F0-A579-DCA8C12F5BF6  
hi
```

```
$ luksmeta load -d /dev/sdc1 -s 2 -u 12618962-A1E5-48F1-B327-D7C60E20FC02  
Slot contains different UUID
```

THE NEAR FUTURE

JOSÉ

- PKCS#11 Support
- Python Bindings
- Additional crypto backends
- Additional algorithms

CLEVIS

- Password Pin ([Issue #52](#))
- PKCS#11 Pin

TANG

- Binding IDs (Optional; sacrifices anonymity)
- Revocation (requires Binding IDs)

**PATCHES
WELCOME!**

QUESTIONS?

