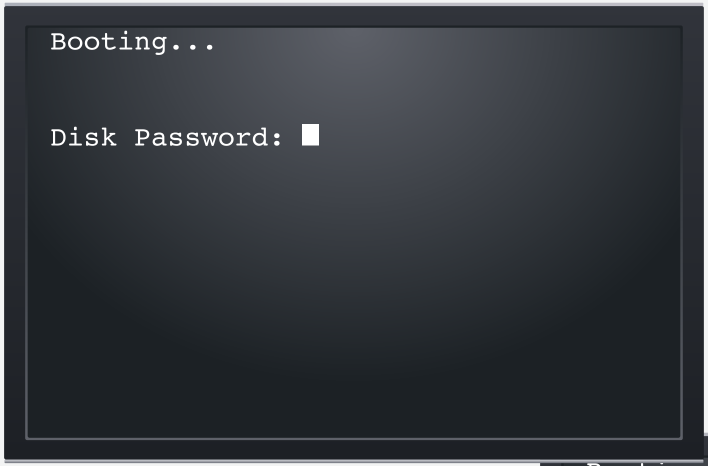
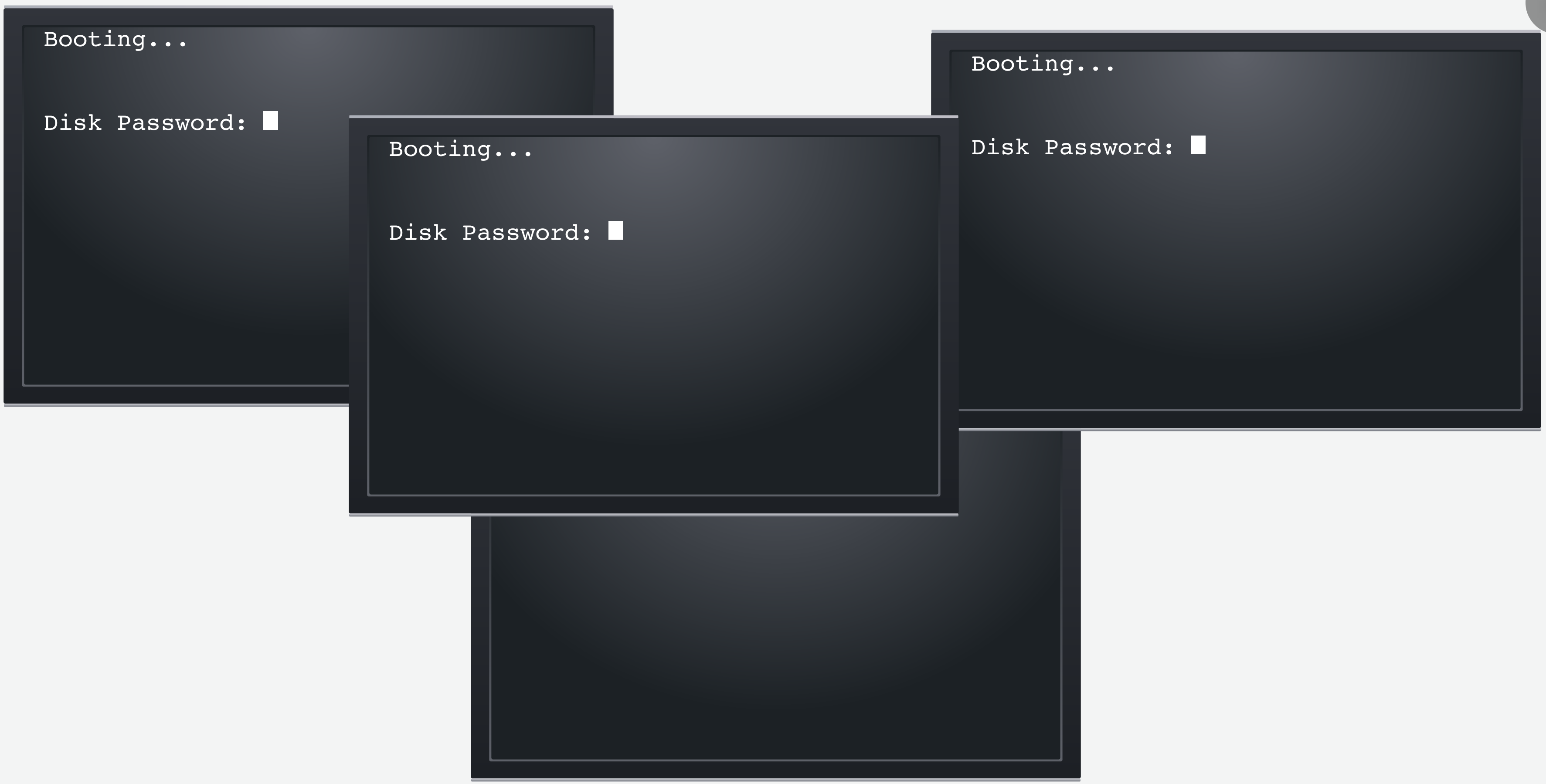
NETWORK-BOUND DISK ENCRYPTION

Nathaniel McCallum
Principal Engineer - Red Hat, Inc.
Alexander Bokovoy
Sr. Principal Engineer - Red Hat, Inc.


Booting...
Disk Password:


Booting...
Disk Password:


Booting...
Disk Password:

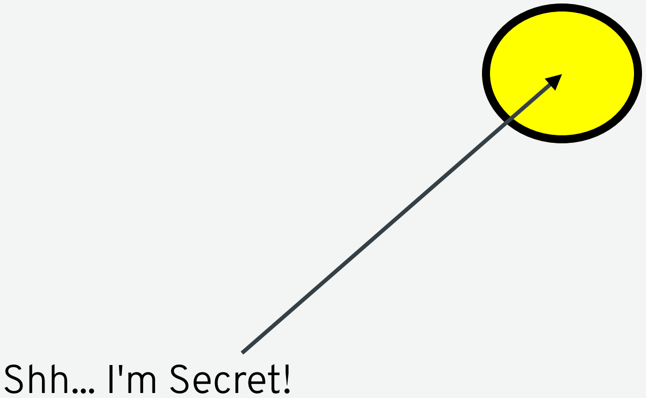

Booting...
Disk Password:
Booting...
Disk Password:
Booting...
Disk Password:

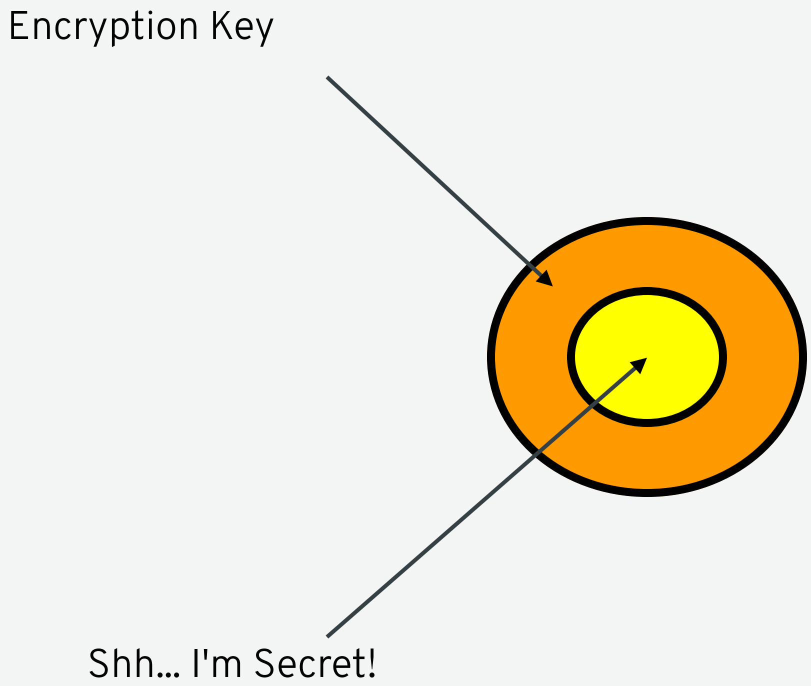

Booting...
Disk Password:
Booting...
Disk Password:
Booting...
Disk Password:
Booting...
Disk Password:
Booting...
Disk Password:
Booting...
Disk Password:

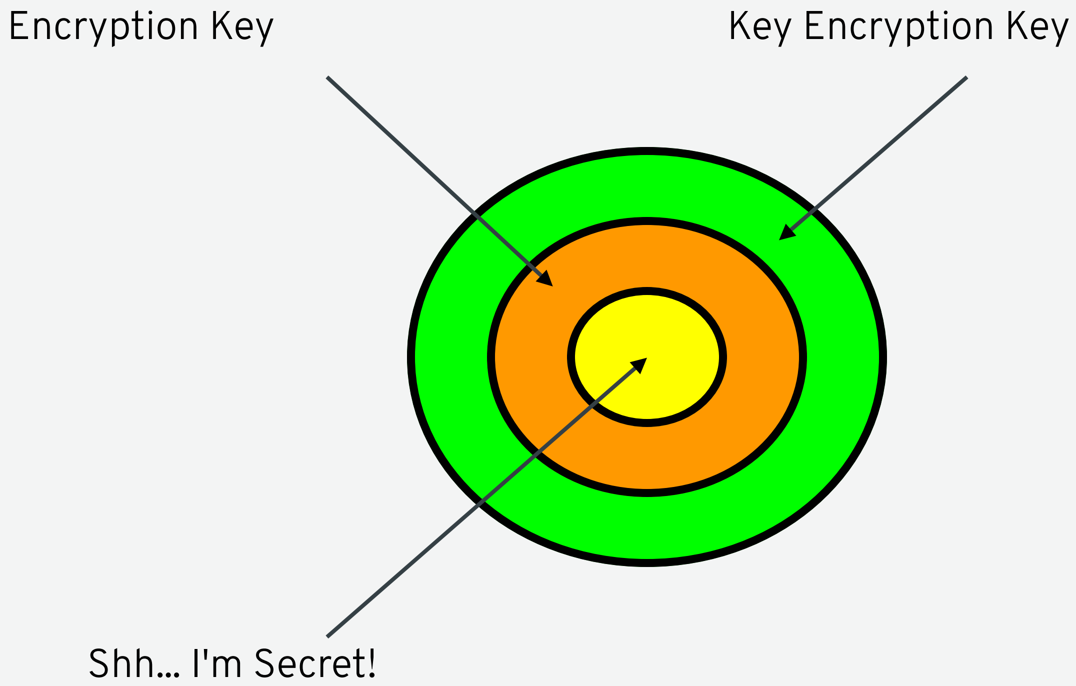
YESTERDAY	TODAY	TOMORROW
Standards (AES, PCI-DSS, etc.)	Automation	Policy

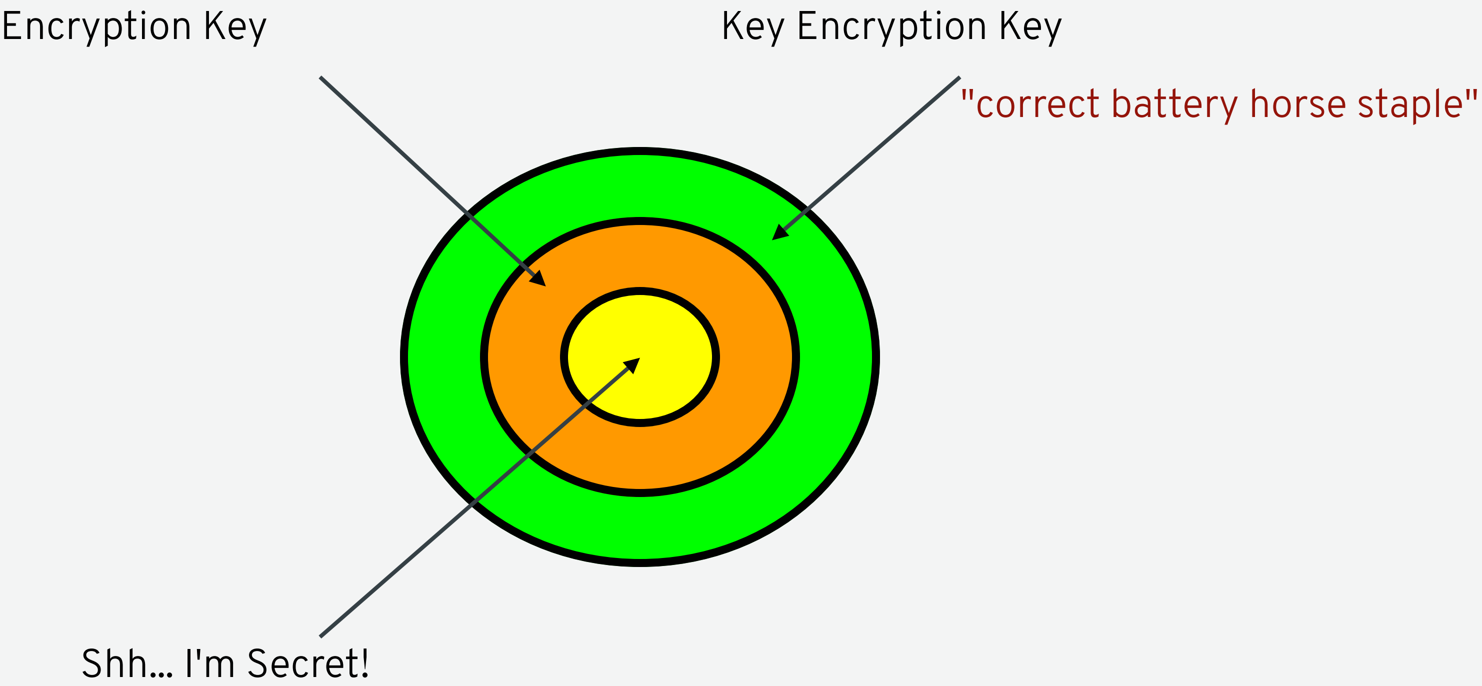
YESTERDAY	TODAY	TOMORROW
Standards (AES, PCI-DSS, etc.)	Automation	Policy

HOW DO WE AUTOMATE?

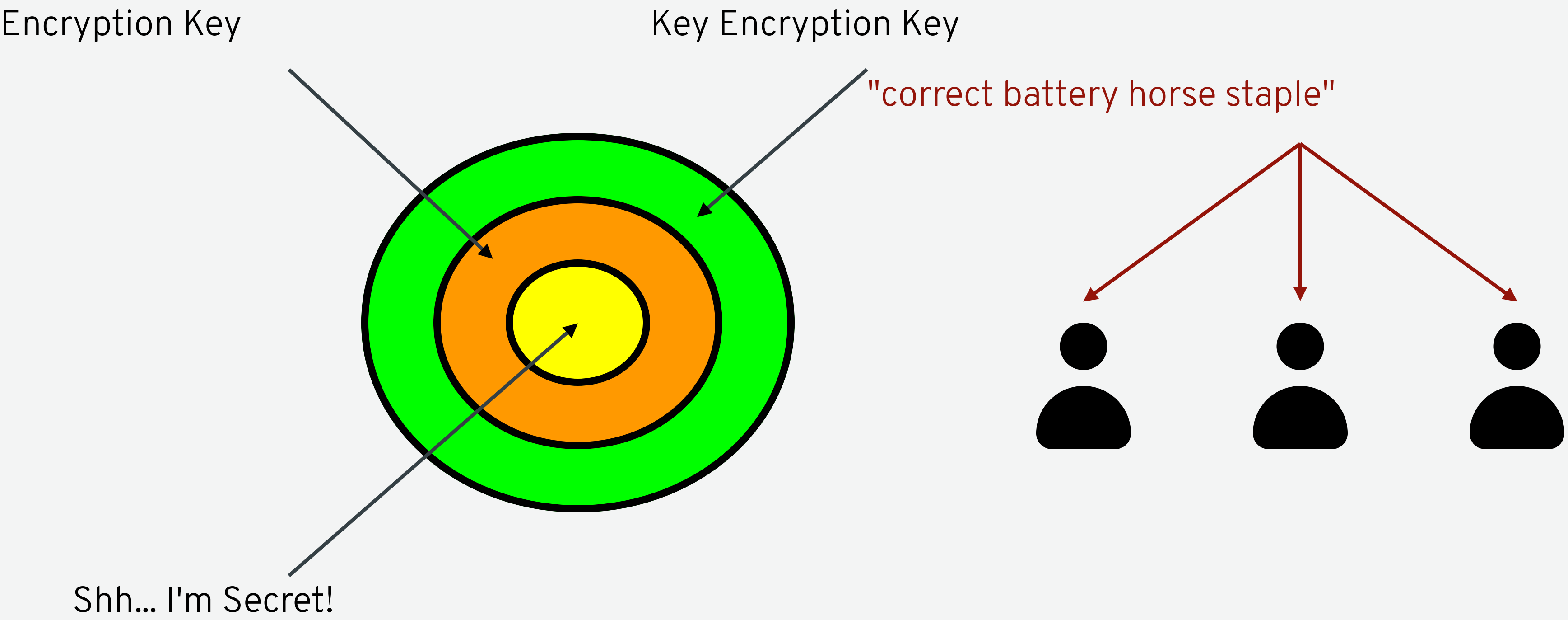

Shh... I'm Secret!

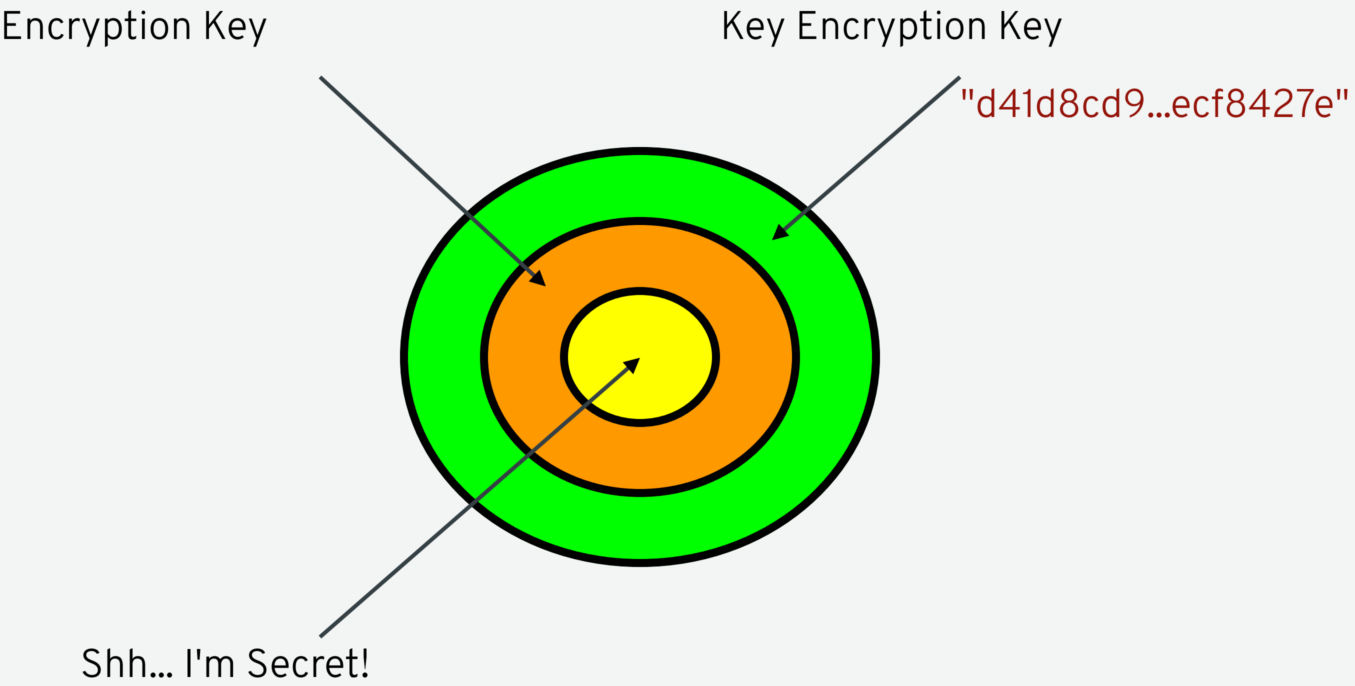

Encryption Key
Shh... I'm Secret!


Encryption Key
Key Encryption Key
Shh... I'm Secret!

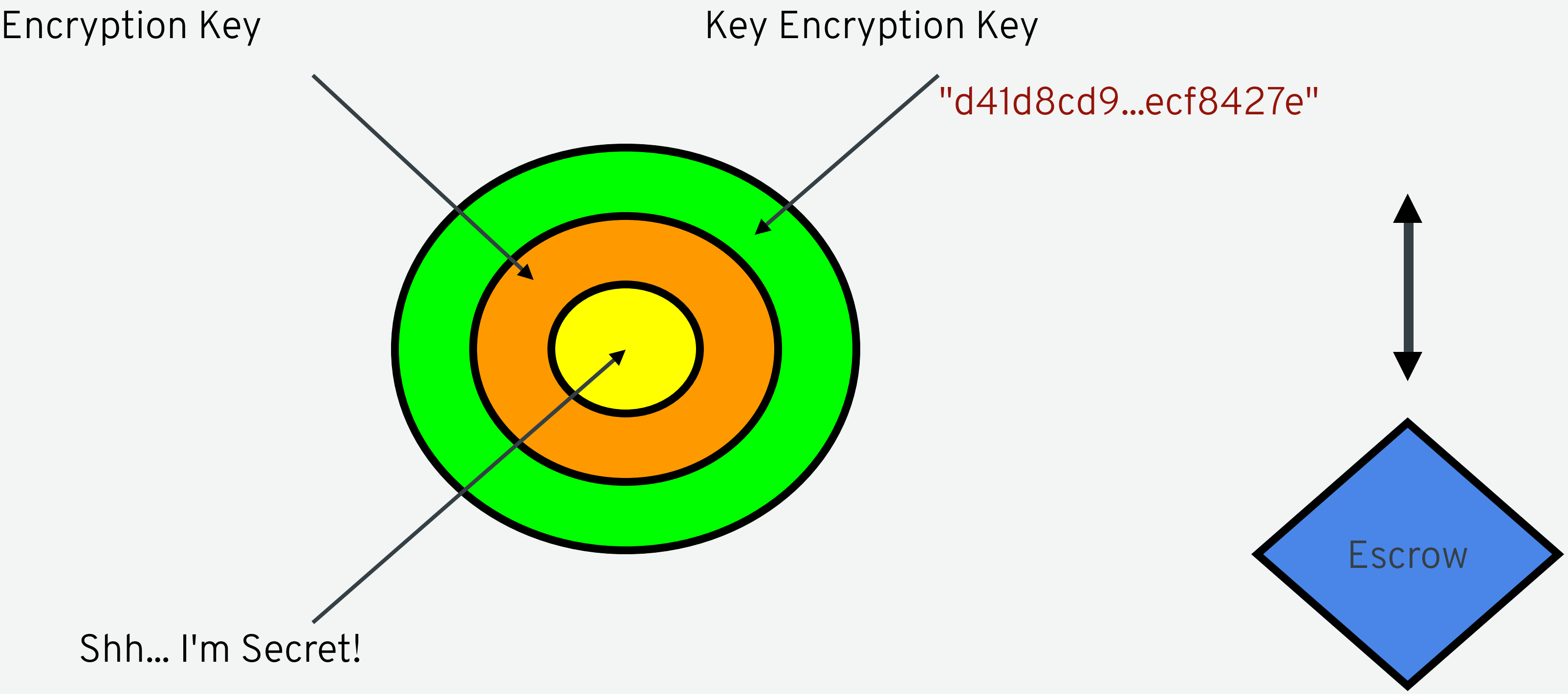

Encryption Key
Key Encryption Key
"correct battery horse staple"
Shh... I'm Secret!

STANDARD PASSWORD MODEL

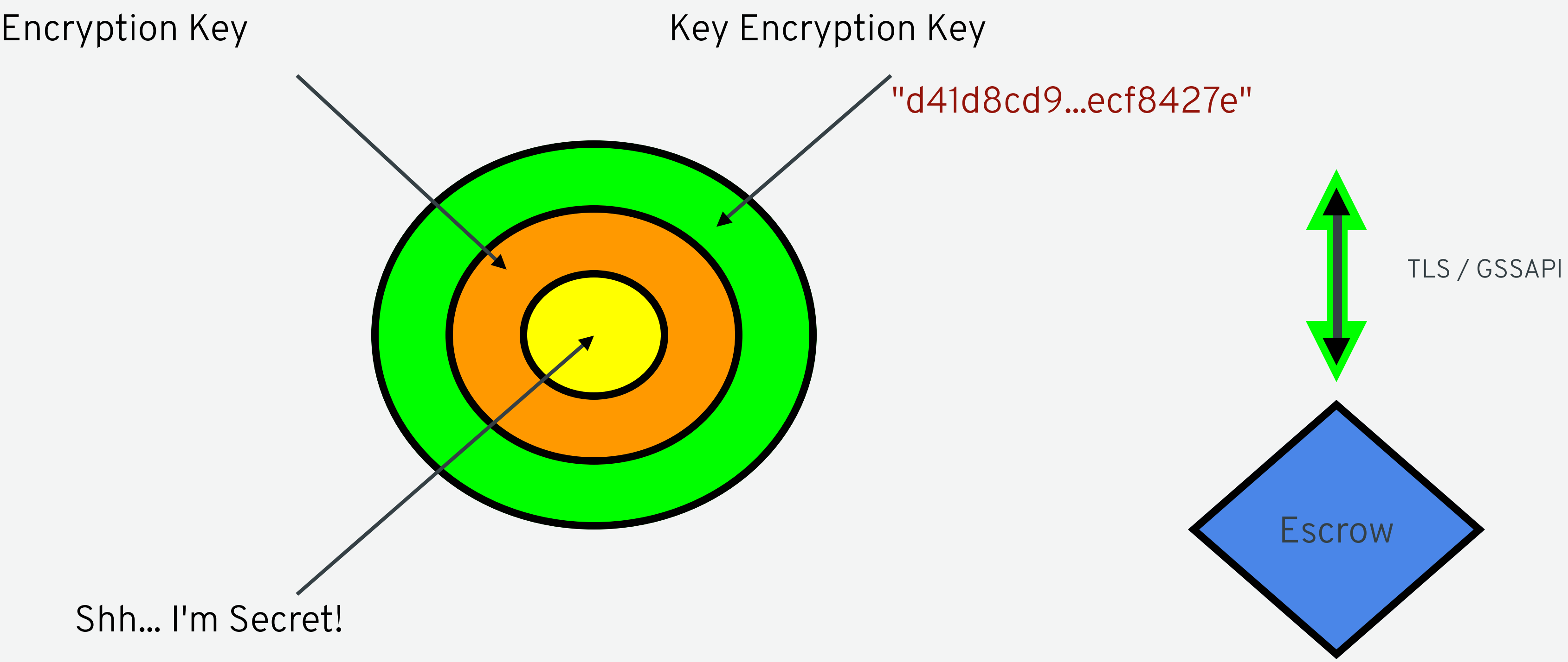



Encryption Key
Key Encryption Key
"d41d8cd9...ecf8427e"
Shh... I'm Secret!

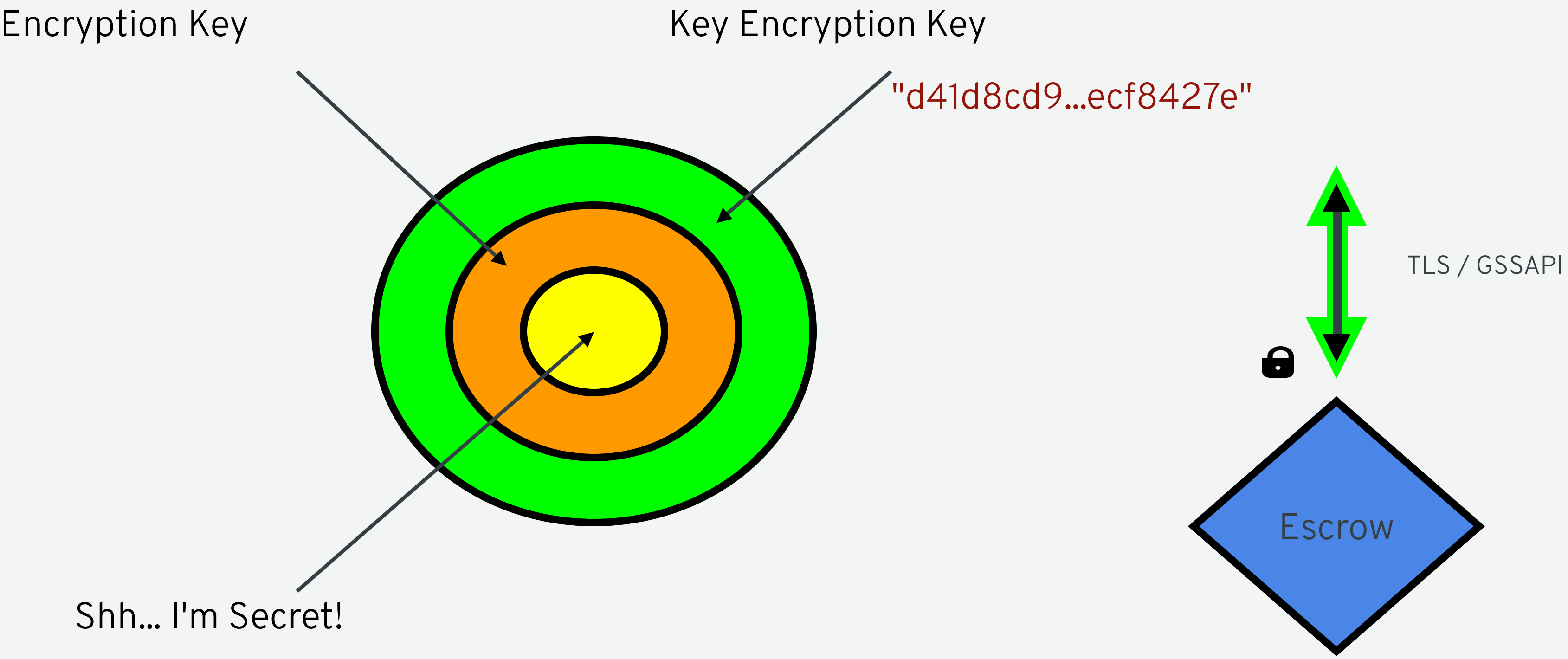
STANDARD ESCROW MODEL?



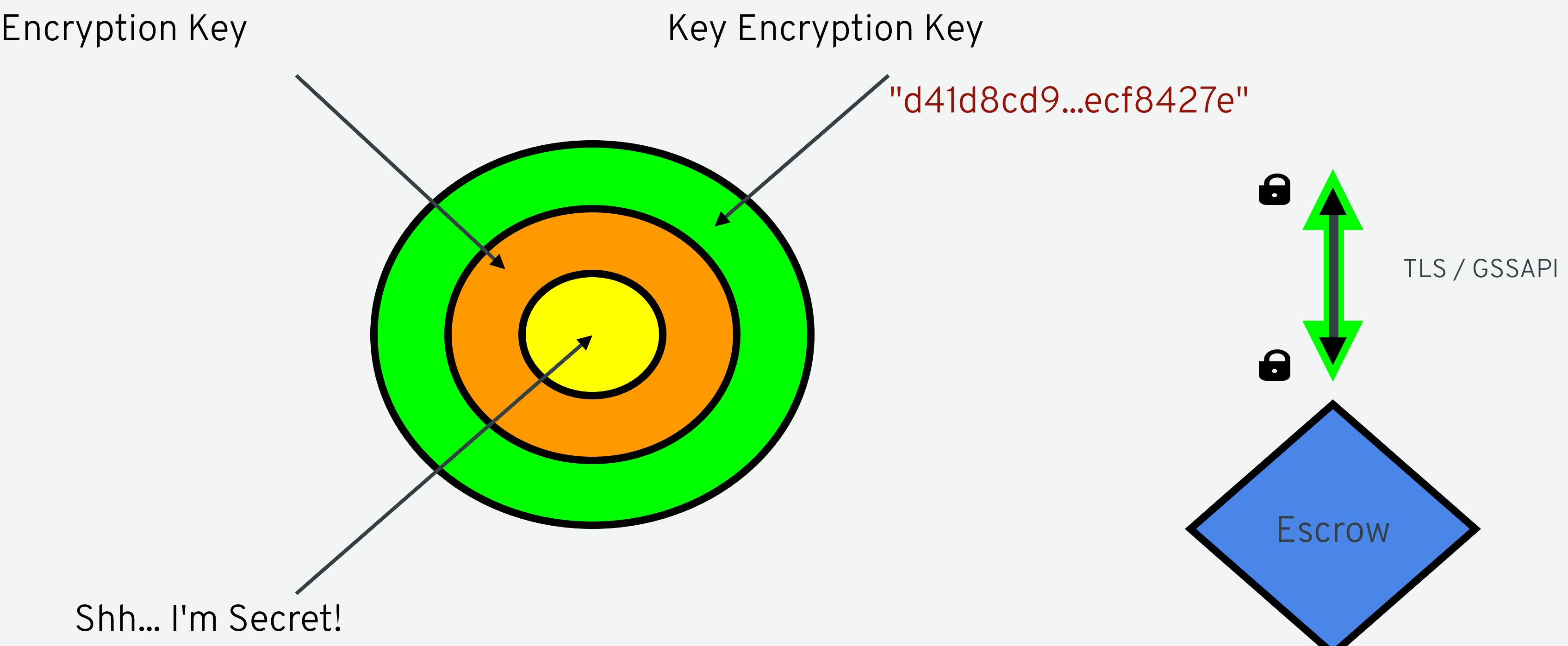
STANDARD ESCROW MODEL?



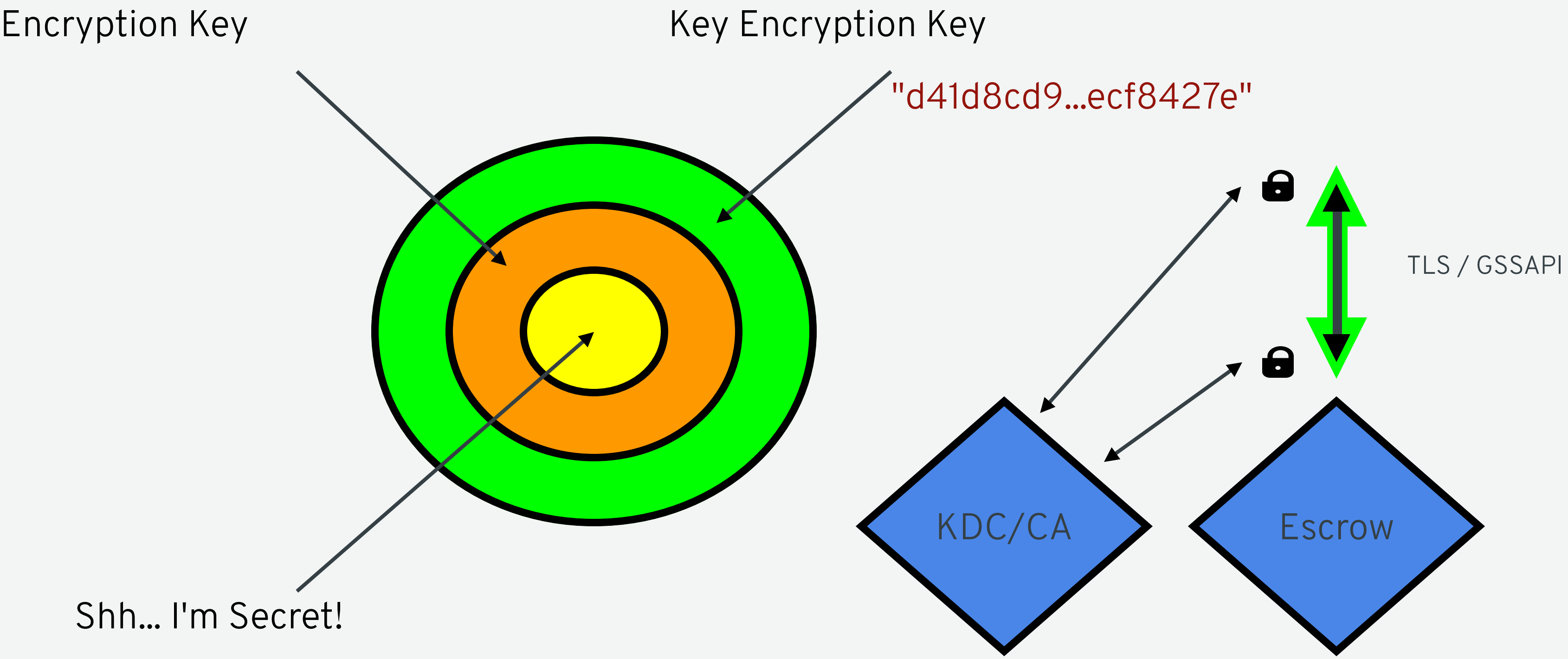
STANDARD ESCROW MODEL?



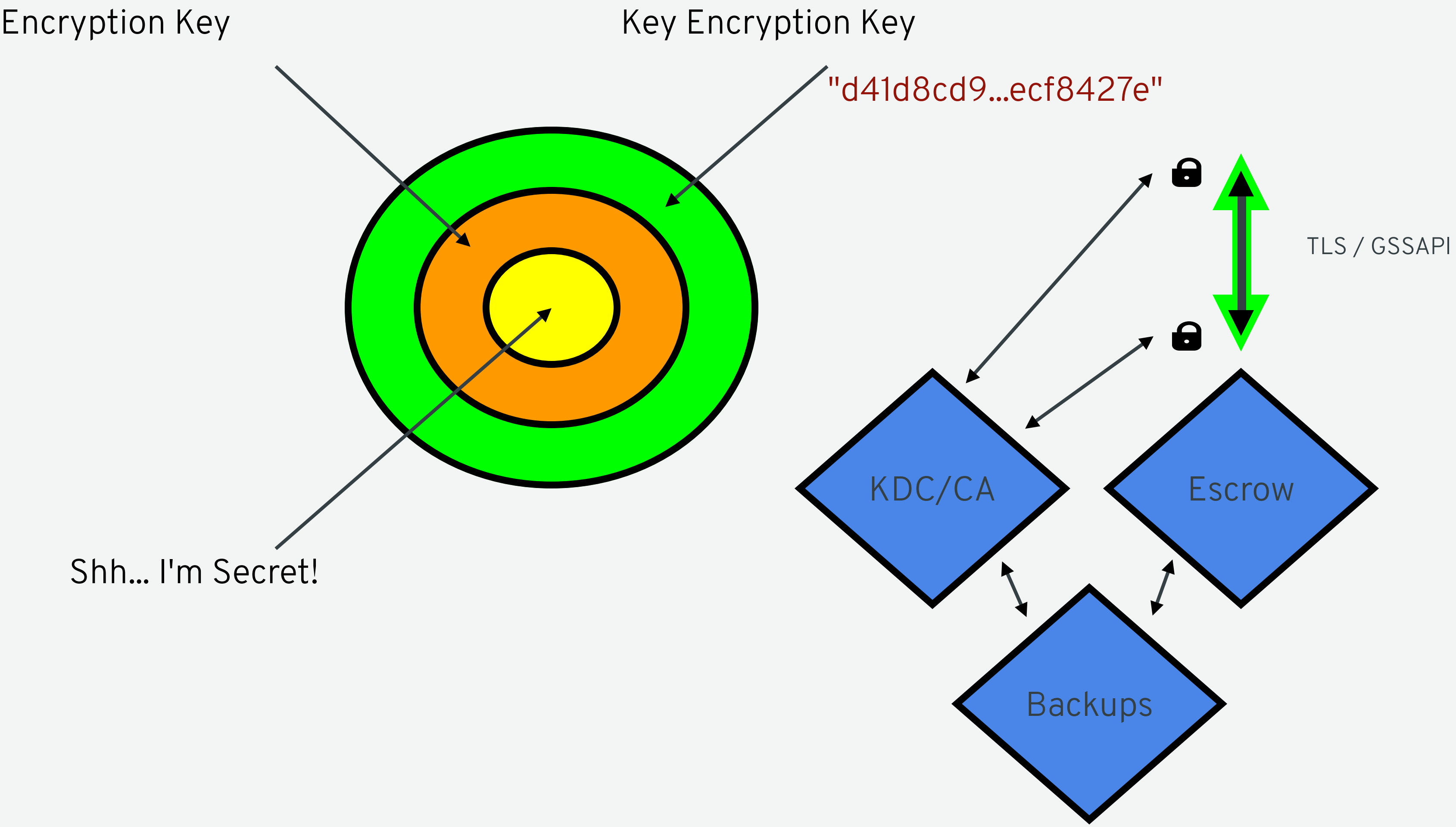
STANDARD ESCROW MODEL?



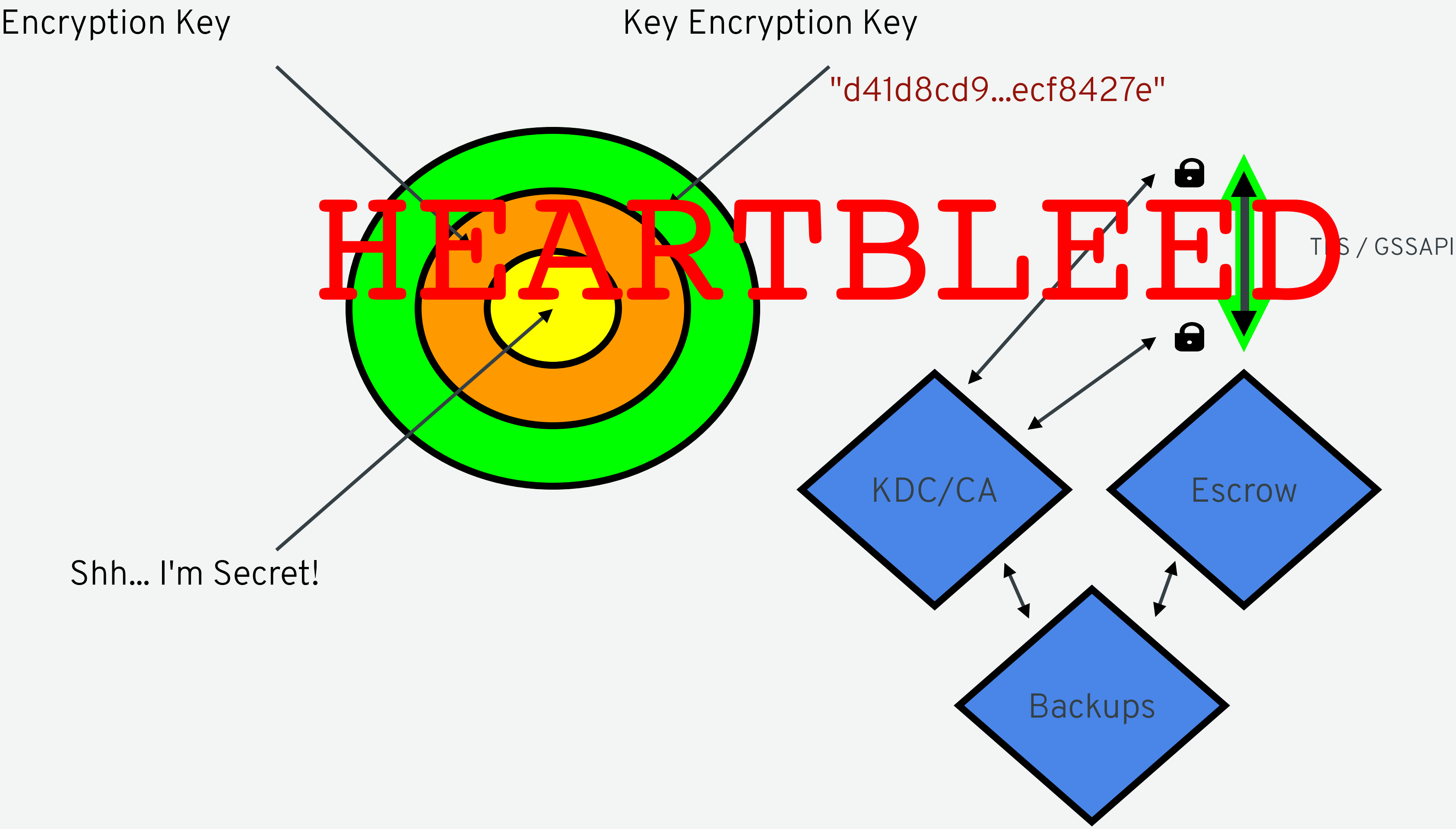
STANDARD ESCROW MODEL?



STANDARD ESCROW MODEL



STANDARD ESCROW MODEL



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?

DIFFIE-HELLMAN IS COMING

Everything simple is false. Everything which is complex is unusable.

Paul Valéry, 1937

(EC) DIFFIE-HELLMAN KEY EXCHANGE

Client side		Server side
$C \in_R [1, p-1]$		$S \in_R [1, p-1]$
$c = gC$		$s = gS$
$c \longrightarrow$		$\longleftarrow s$
$K = gSC = sC$		$K = gCS = cS$

C is a private key

c is a public key

K is a common key

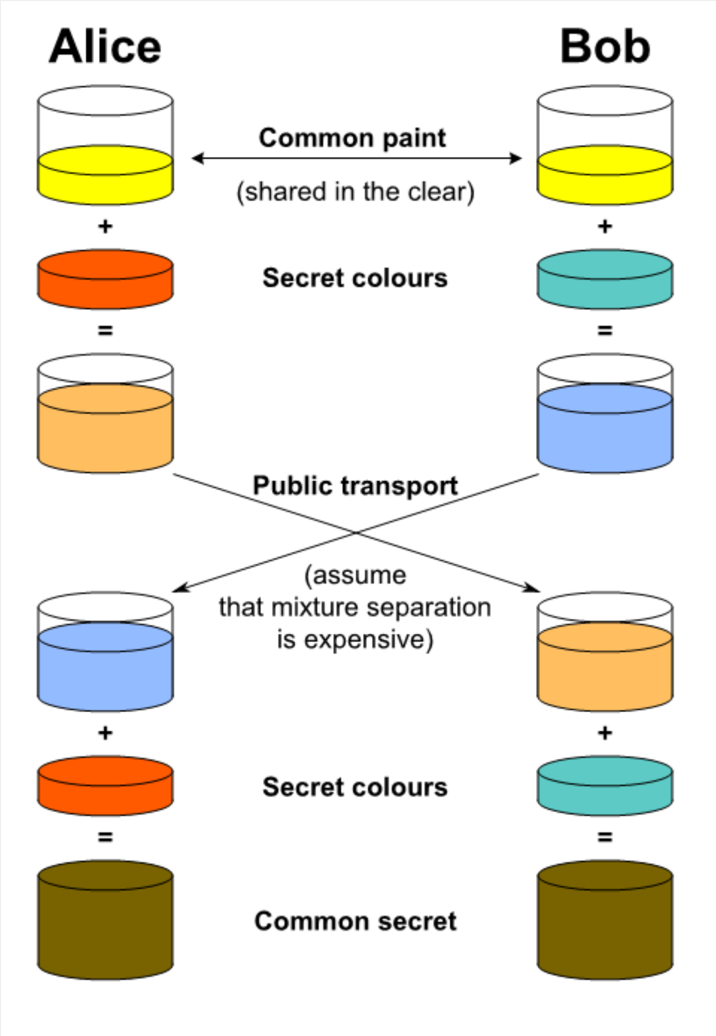
g is a large prime, primitive modulo *p*, *known to both parties*

S is a private key

s is a public key

K is a common key

(EC) DIFFIE-HELLMAN KEY EXCHANGE



BINDING WITH ECDH (INSECURE)

PROVISIONING

Client side	Server side
	$S \in_R [1, p-1]$
	$s = gS$
	$\longleftarrow s$
$C \in_R [1, p-1]$	
$c = gC$	
$K = gSC = sC$	
$Discard: K, C$	
$Retain: s, c$	

RECOVERY

Client side	Server side
$c \longrightarrow$	
	$K = xS$
	$\longleftarrow K$

Weaknesses:

1. K is revealed to a passive attacker.
2. With c, the passive attacker can get K.
3. Server learns c and therefore K.

Resolved: c **MUST** be private

MCCALLUM-RELYEA KEY EXCHANGE

PROVISIONING

Client side | Server side

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

$s = gS$

Text

$\longleftarrow s$

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

$c = gC$

$K = gSC = sC$

$Discard : K, C$

$Retain : s, c$

C is a private key

c is a public key

S is a private key

s is a public key

RECOVERY

Client side | Server side

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

$e = gE$

$x = c + e$

$x \longrightarrow$

$y = xS$

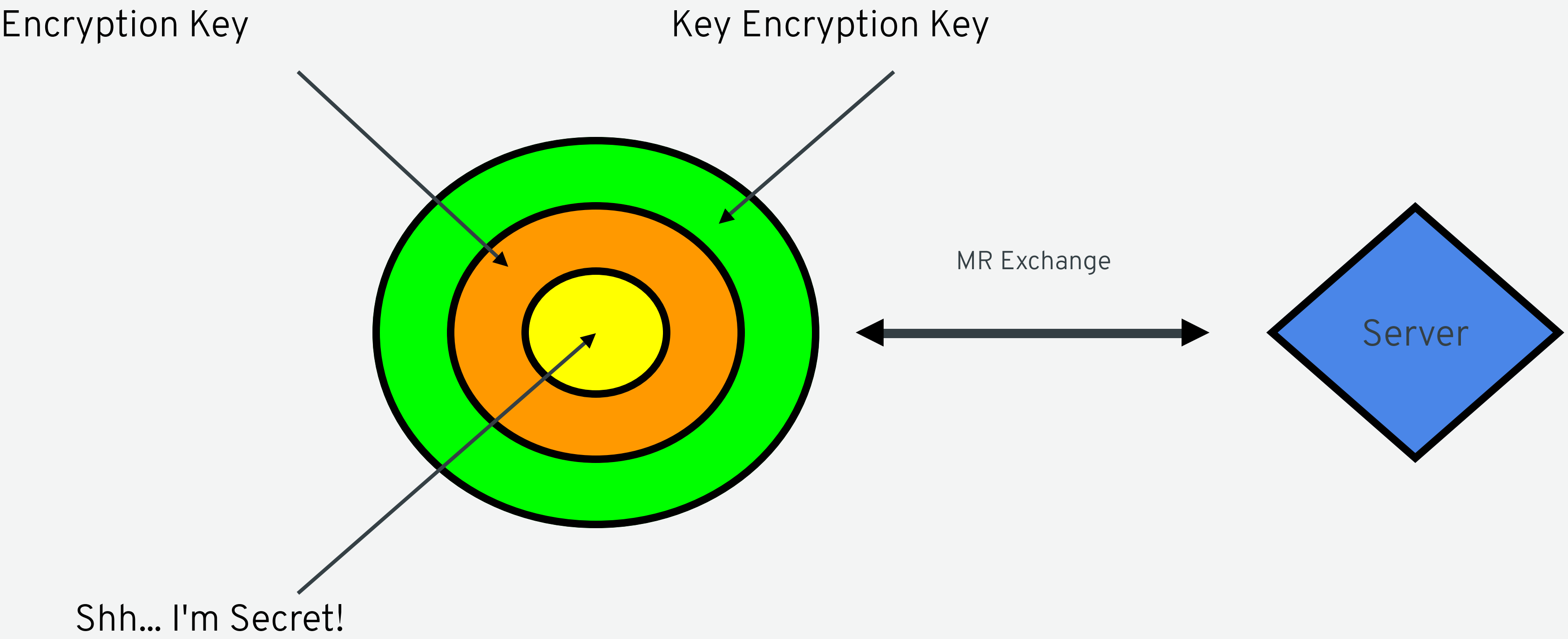
$\longleftarrow y$

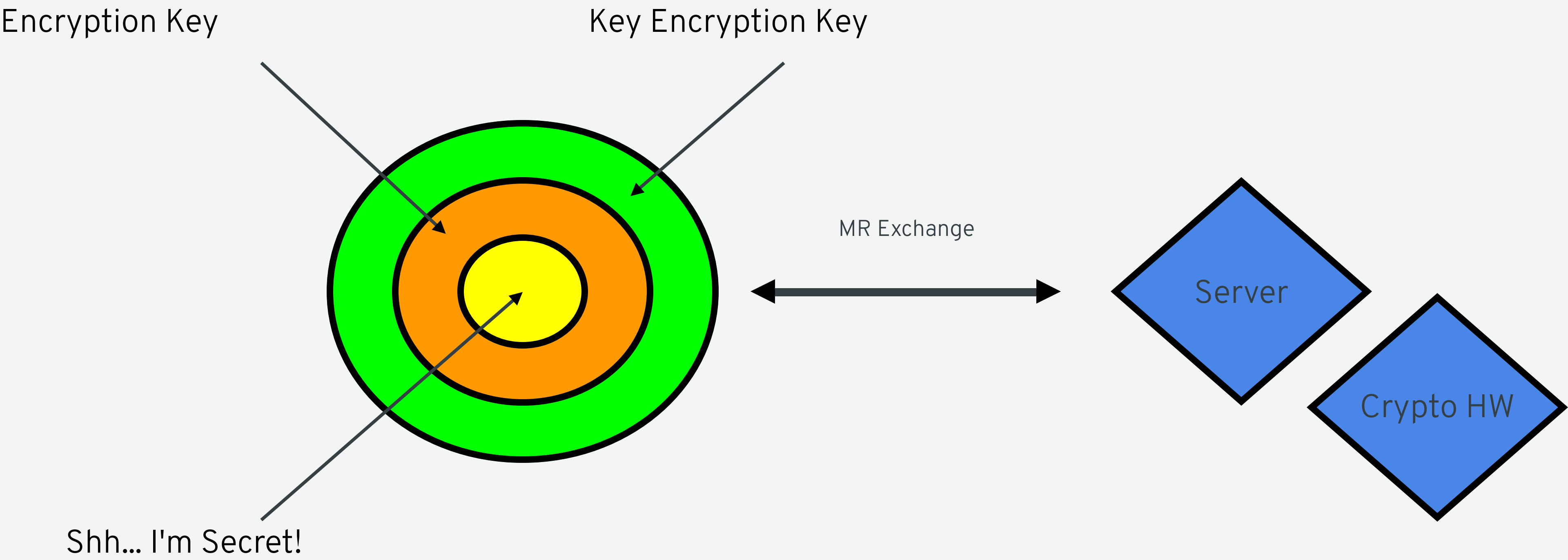
$K = y - sE$

$Because : K = gCS + gES - gSE$

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

e & E are ephemeral keys


Encryption Key
Key Encryption Key
MR Exchange
Server
Shh... I'm Secret!


Encryption Key
Key Encryption Key
MR Exchange
Server
Crypto HW
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, ITP 854409)

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, ITP 854410)

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":"-O59czAqybvxHdme2t3I5A", ...}

$ 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	TODAY	TOMORROW
Standards (AES, PCI-DSS, etc.)	Automation	Policy

SHAMIR'S SECRET SHARING (1979)

Based on the idea of Lagrange polynomial interpolation

Given t distinct points (x_i, y_i) of the form $(x_i, f(x_i))$, where $f(x)$ is a polynomial of degree less that t, then $f(x)$ is determined by

$$f(x) = \sum_{i=1}^{t} y_i \prod_{1 \le j \le t} \frac{x - x_j}{x_i - x_j}$$

Shamir's secret sharing

for a secret $s \in Z/pZ$ with a prime p, set $a_0 = s$, and choose $a_1, \ldots, a_{t-1}$ at random in Z/pZ. The trusted party then computes $f(i)$, where $f(x)$ is

$$f(x) = \sum_{k=0}^{t-1} a_k x^k$$

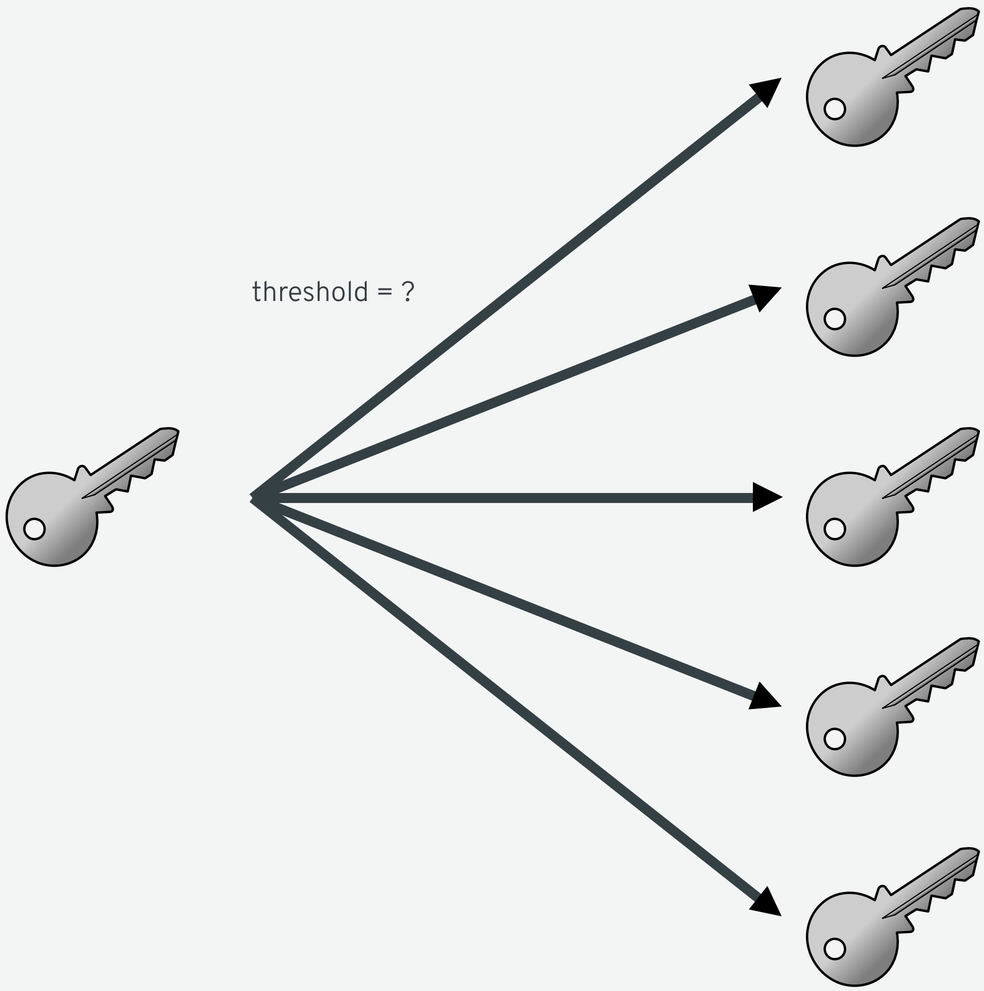
for all $1 \le i \le n$. The shares $(i, f(i))$ are distributed to n distinct parties.

Recovery of a secret s

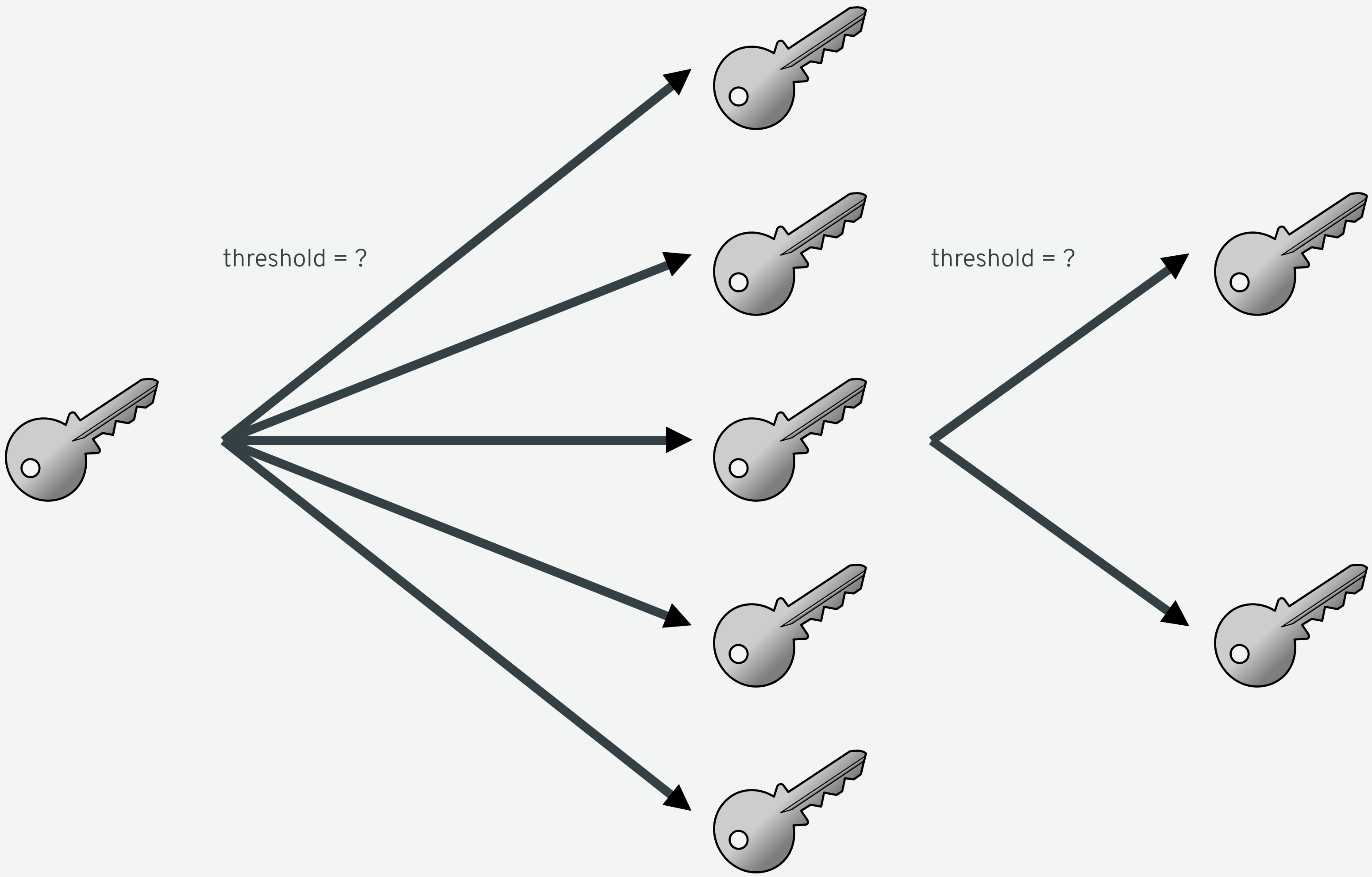
Secret $s = a_0 = f(0)$ is recovered from any t shares $(i, f(i))$, for $I \subset \{1, \ldots, n\}$

$$s = \sum_{i \in I} f(i) \prod_{j \in I, j \neq i} \frac{i}{j - i}$$

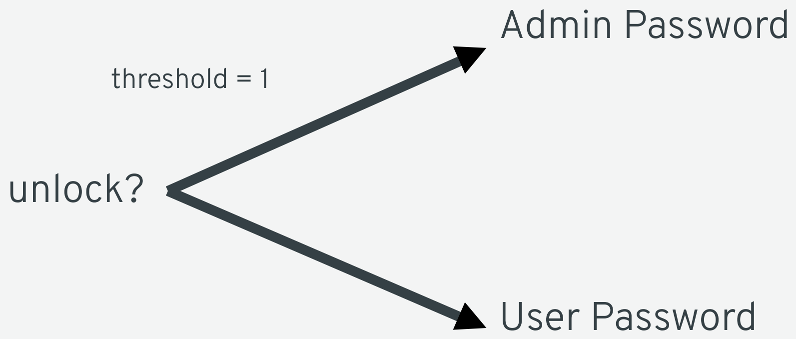
SHAMIR'S SECRET SHARING (1979)



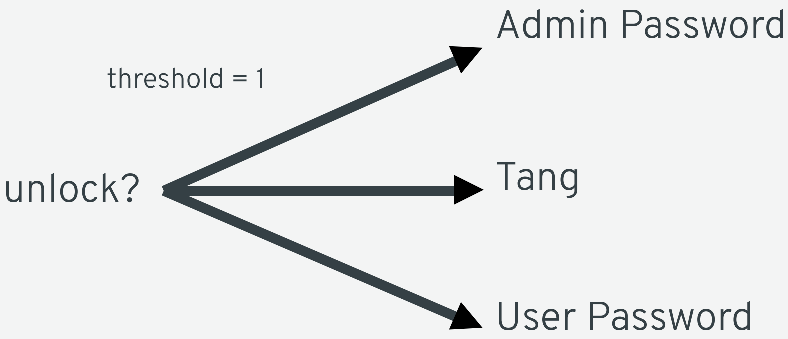
SHAMIR'S SECRET SHARING (1979)



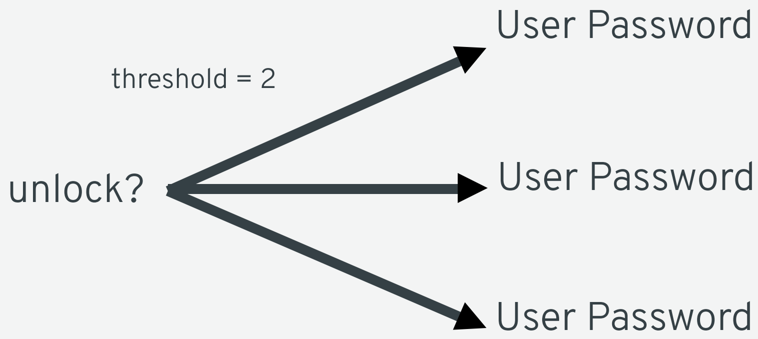
SIMPLE LAPTOP



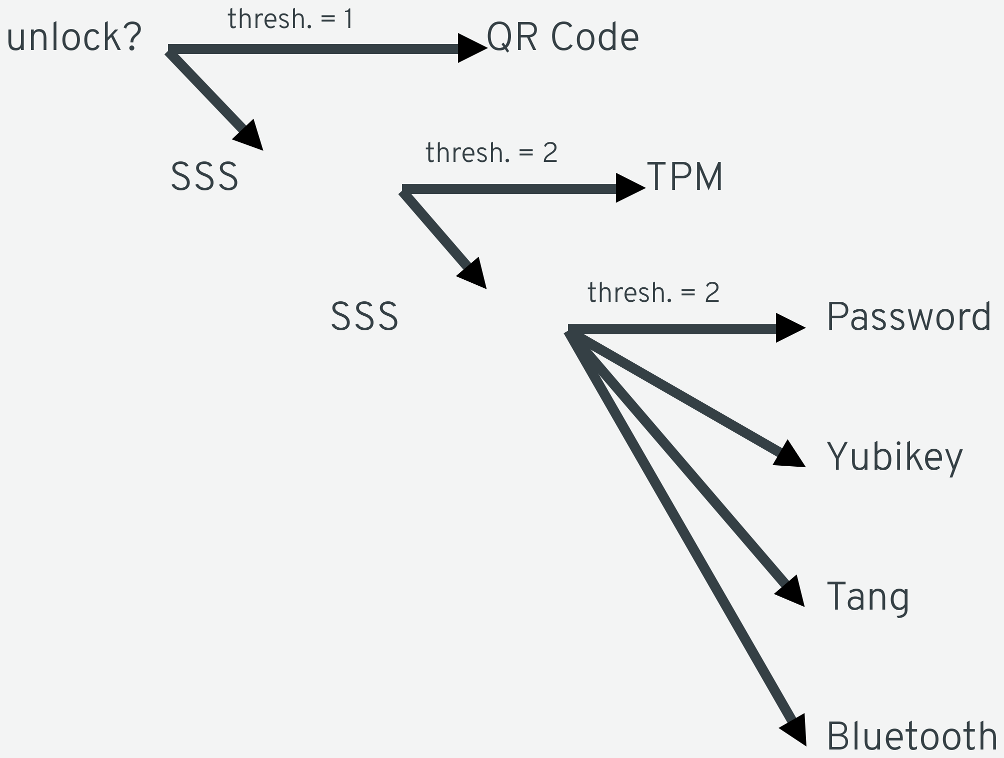
AUTOMATED LAPTOP



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

PATCHES WELCOME!

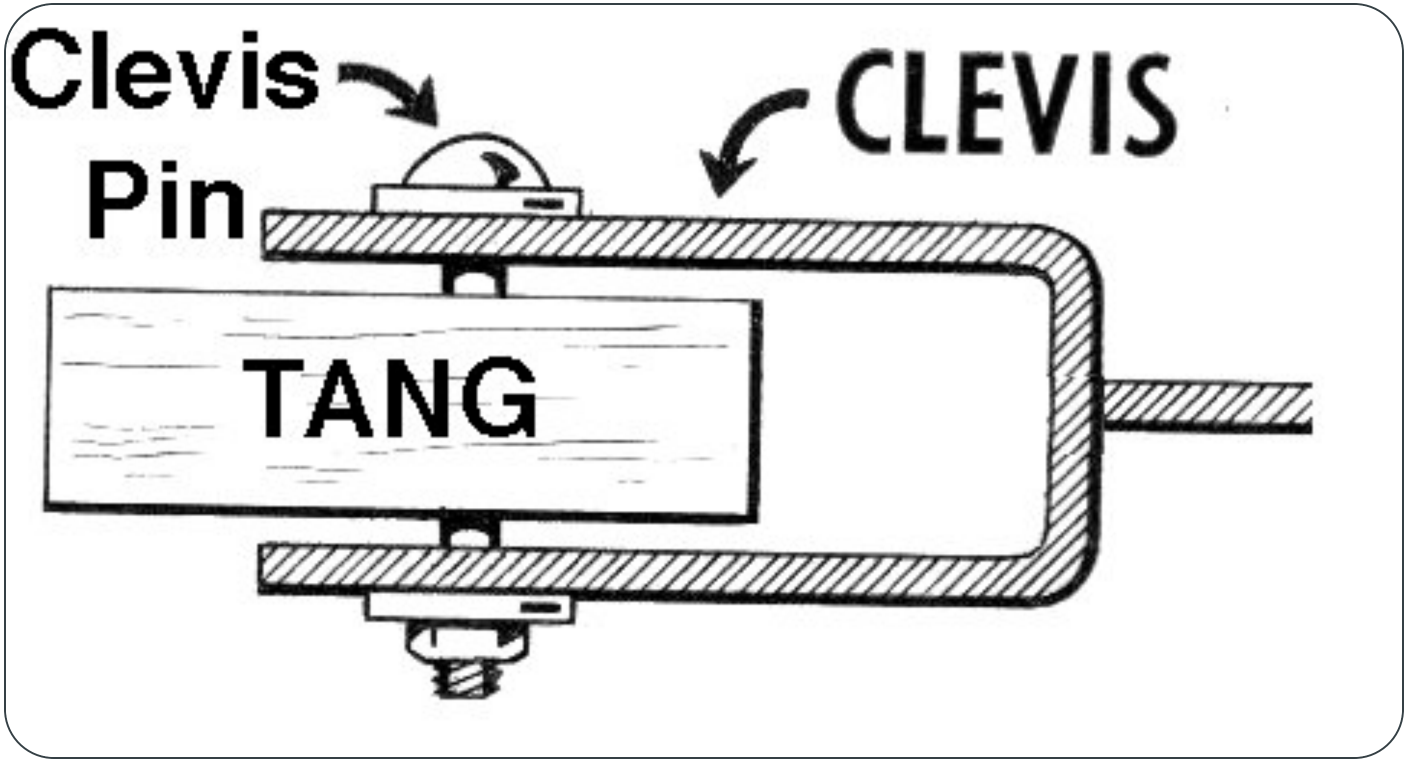
CLEVIS

- Support for non-root, non-removable volumes
- TPM v2.0 Support (PR#17)
- Password Pin
- PKCS#11 Pin
- Ext4 encryption support

TANG

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

QUESTIONS?



All related projects are in the Latchset: https://github.com/latchset

Feel free to ask questions:
Alexander Bokovoy: abokovoy@redhat.com