krypt. semper pi.

whoami

ruby-core

ruby openssl

freelancer

whoami

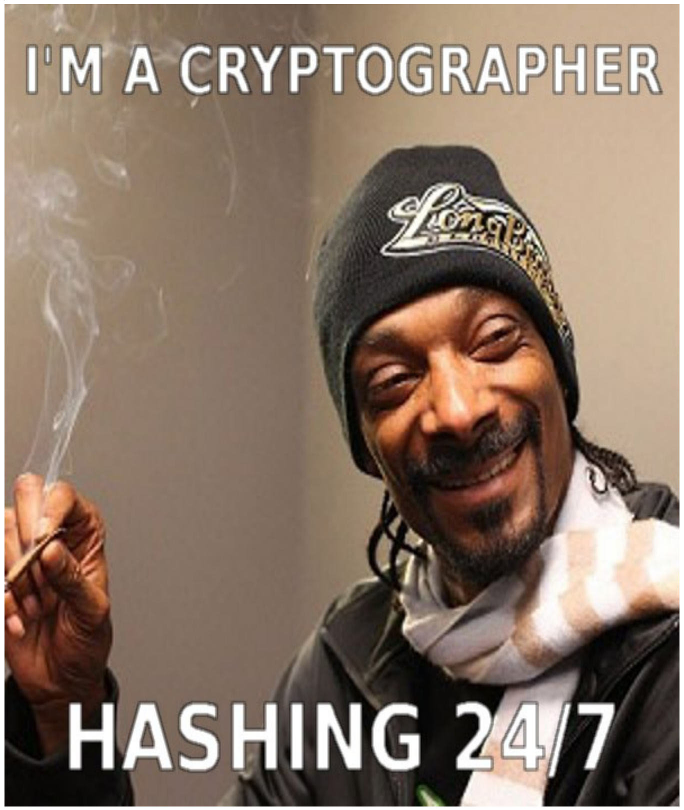
germany

crypto is hard

u can't touch this

touched crypto

gets crypto



so, if crypto is hard

do crypto apis have to be, too?

two opposing forces

„when i refactored our login,
i mostly cared about security.
what do i care about algorithms?
i need my time to work out!"

„i did a lot of ssl pentesting lately.
i need full control of all parameters.
and i sure do need my legacy algorithms, dude.“

everyday development:

security sucks

„oh no, not security again!“

„today was awesome. i got to write crypto code again. boy, did i have fun integrating with our customer's century-old security interface.“

- **noone**, **never**

security is the jar jar binks
of software development

it's a huge pain

that must be dealt with

security can be a real bitch

THIS BITCH

good design

==

making complex things easy

many examples where this happened

databases are hard -> active record

threads are hard -> stm, actors

memory is hard -> gc

don't bother me with details

don't make me think

crypto is hard -> ???

but experts™ need full control

conflict

abstraction vs. oversimplification

there is no golden middle

if full control is your thing:

openssl
java security api
&
friends

if all you care about is security by default:

keyczar (no friends)

(maybe (rb)nacl / libsodium)

why not just use keyczar

and be done with it?

well, there's the experts™

and legacy apps

we need both

full control if needed

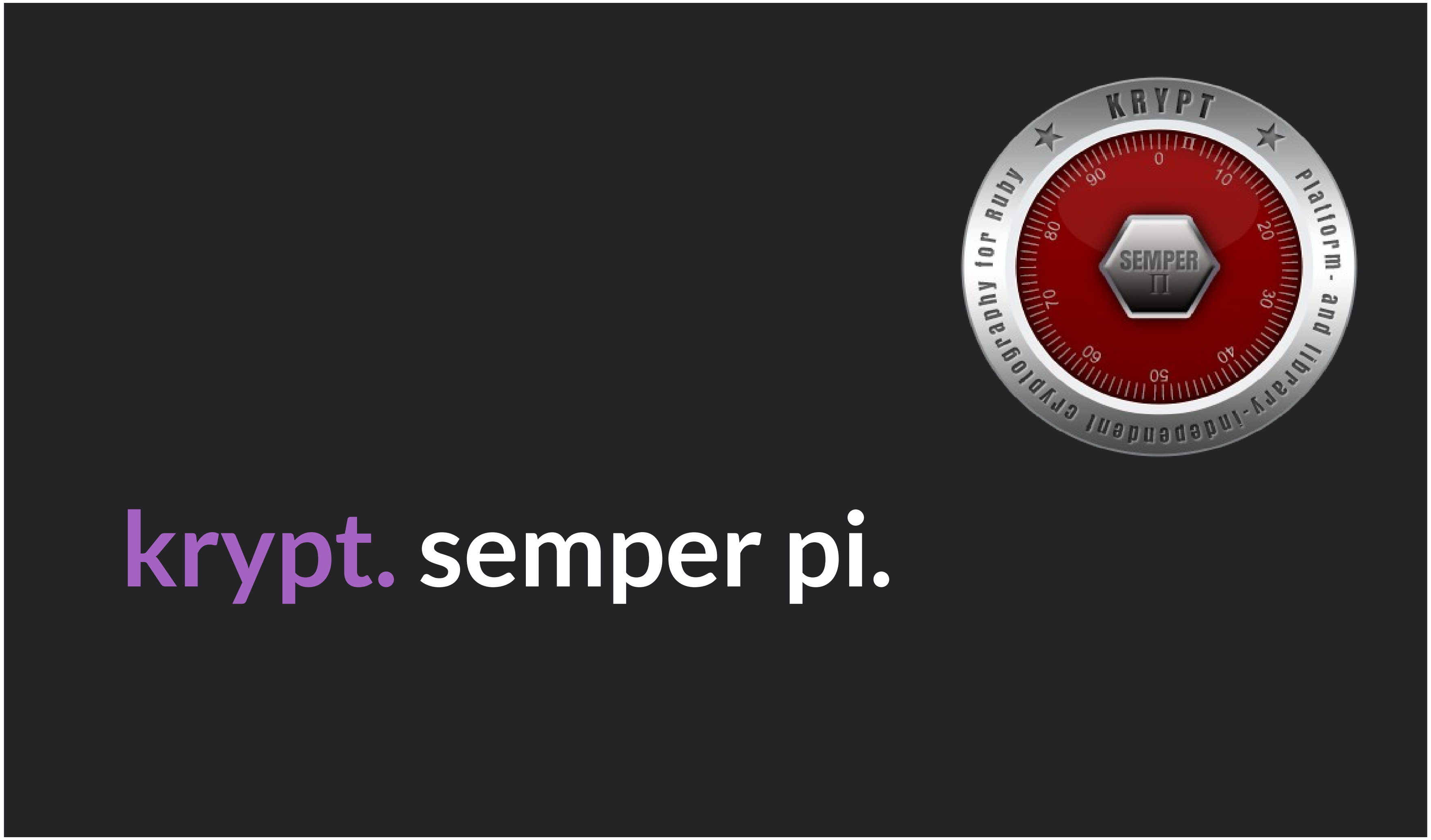
security by default otherwise

„sadly we can't have both. or can we?“

revolutionary idea

Y U NO
MAKE BOTH

combine both aspects in one library



krypt. semper pi.

ok, cool, but...

WTF
IS THIS SHIT?

who we are

@nahi, @vipulnsward, @abstractj, @qmx, @_emboss_

will it replace

openssl
java security api
my favorite crypto library
?

we come in peace

krypt first of all is a framework

using existing libraries

to implement

core cryptography primitives

peaceful coexistence

(at least at first)

„i see – this is the low-level api that gives me the full control that i so badly need“

digest

cipher

signature

krypt adds asn.1 dsl to the mix

(asn.1 being crypto's lingua franca)

„let's say i write
an app from scratch
and all i really care
about is the thing
being secure – what
do i do then?“

krypt offers high-level api

implemented on this basis

what's the big deal?

crypto code today

#1 encrypting data

what it should look like

```
require 'openssl'

data = 'le secret'

cipher = OpenSSL::Cipher.new('aes-128-cbc')
cipher.encrypt
key = cipher.random_key
iv = cipher.random_iv

enc = cipher.update(data) + cipher.final


decipher = OpenSSL::Cipher::AES.new('aes-128-cbc')
decipher.decrypt
decipher.key = key
decipher.iv = iv

plain = decipher.update(enc) + decipher.final
```

what it actually looks like

```ruby
require 'openssl'

data = 'le secret'
key = 'lepasswordlepassword'

cipher = OpenSSL::Cipher.new('AES-128-ECB')
cipher.encrypt
cipher.key = key

enc = cipher.update(data) + cipher.final

decipher = OpenSSL::Cipher::AES.new('AES-128-ECB')
decipher.decrypt
decipher.key = key

plain = decipher.update(enc) + cipher.final
```

```
require 'openssl'

data = 'le secret'
key = 'lepasswordlepassword' # fail

cipher = OpenSSL::Cipher.new('AES-128-ECB')
cipher.encrypt
cipher.key = key

enc = cipher.update(data) + cipher.final

decipher = OpenSSL::Cipher::AES.new('AES-128-ECB')
decipher.decrypt
decipher.key = key

plain = decipher.update(enc) + cipher.final
```

```
require 'openssl'

data = 'le secret'
key = 'lepasswordlepassword'

cipher = OpenSSL::Cipher.new('AES-128-ECB') # fail
cipher.encrypt
cipher.key = key

enc = cipher.update(data) + cipher.final

decipher = OpenSSL::Cipher::AES.new('AES-128-ECB') # fail
decipher.decrypt
decipher.key = key

plain = decipher.update(enc) + cipher.final
```

```
require 'openssl'

data = 'le secret'
key = 'lepasswordlepassword'

cipher = OpenSSL::Cipher.new('AES-128-ECB')
cipher.encrypt
cipher.key = key
# no iv -> fail

enc = cipher.update(data) + cipher.final

decipher = OpenSSL::Cipher::AES.new('AES-128-ECB')
decipher.decrypt
decipher.key = key
# no iv -> fail

plain = decipher.update(enc) + cipher.final
```

```
require 'openssl'

data = 'le secret'
key = 'lepasswordlepassword'

cipher = OpenSSL::Cipher.new('AES-128-ECB')
cipher.encrypt # design fail
cipher.key = key

enc = cipher.update(data) + cipher.final

decipher = OpenSSL::Cipher::AES.new('AES-128-ECB')
decipher.decrypt # design fail
decipher.key = key

plain = decipher.update(enc) + cipher.final
```

```
require 'openssl'

data = 'le secret'
key = 'lepasswordlepassword'

cipher = OpenSSL::Cipher.new('AES-128-ECB')
cipher.encrypt
cipher.key = key

enc = cipher.update(data) + cipher.final # design fail

decipher = OpenSSL::Cipher::AES.new('AES-128-ECB')
decipher.decrypt
decipher.key = key

plain = decipher.update(enc) + cipher.final # design fail
```

#2 pbkdf2 password hash

what it should look like

```ruby
require 'openssl'

pass = 'le secret'
salt = OpenSSL::Random.random_bytes(16)
iter = 20000
len = OpenSSL::Digest::SHA1.new.digest_len # 20

hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

what it actually looks like

```
require 'openssl'

pass = 'le secret'
salt = pass
iter = 10
len = password.size

hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

```
require 'openssl'

pass = 'le secret'
salt = pass # fail
iter = 10
len = password.size

hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

```
require 'openssl'

pass = 'le secret'
salt = pass
iter = 10 # fail
len = password.size

hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

```
require 'openssl'

pass = 'le secret'
salt = pass
iter = 10
len = password.size # fail

hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

```
require 'openssl'

pass = 'le secret'
salt = pass
iter = 10
len = password.size
                                             # design fail
hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

```
require 'openssl'

pass = 'le secret'
salt = pass
iter = 10
len = password.size
                              o_O
hash = OpenSSL::PKCS5.pbkdf2_hmac_sha1(pass, salt, iter, len)
```

what it actually **actually** looks like

```
require 'openssl'

pass = 'le secret'
salt = pass
digest = OpenSSL::Digest::MD5.new

hash = digest.digest(salt + pass)
```

FEYENOORD
ROTTERDAM

#3 certificate validation

what it should look like

```
require 'openssl'

store = OpenSSL::X509::Store.new
store.set_default_paths
oh, you know what, i don't
have the time for this...
```

almost impossible to do it right™

online revocation checks

openssl refuses to have dependencies

relic of bygone times before

dependency management tools

we're left with half-assed validation

krypto code tomorrow

what it should actually look like

#1 encrypting data

krypt.semper pi.

```ruby
require 'krypt'

data = 'le secret'

encrypter = Krypt::Encrypter.new
key = encrypter.generate_key

enc = encrypter.encrypt(data)


decrypter = Krypt::Decrypter.new
decrypter.key = key

plain = decrypter.decrypt(data)
```

#2 password hash

```ruby
require 'krypt'

pass = 'le secret'

hash = Krypt::PasswordHash.hash(pass)

begin
  Krypt::PasswordHash.verify(hash, pass)
rescue Krypt::PasswordHash::InvalidPassword
  # react
end

#swell
```

#3 certificate validation

```ruby
require 'krypt'

certificate = Krypt::X509::Certificate.new(bytes)

begin
  certificate.verify
rescue Krypt::X509::VerificationError
  # react
end

#swell
```

don't bother me with details

don't make me think

security by default

ONE DOES NOT SIMPLY
REPLACE OPENSSL LIKE THAT

yes we can

use protocols

moar advantages:

moar tests

rspec

FuzzBert

moar docs (non-expert™)

moar ruby

easier to understand & maintain

minimal portion of native code

krypt.semper pi.

using whatever library is available

in the background

diversity

the rest is plain ruby

„why should anyone care?“

write once, run anywhere$^{\text{TM}^2}$

run on all rubies

„what do i care

if your stupid crypto library works?“

https

if https doesn't work,

ruby doesn't work

(https://rubygems.org, anyone?)

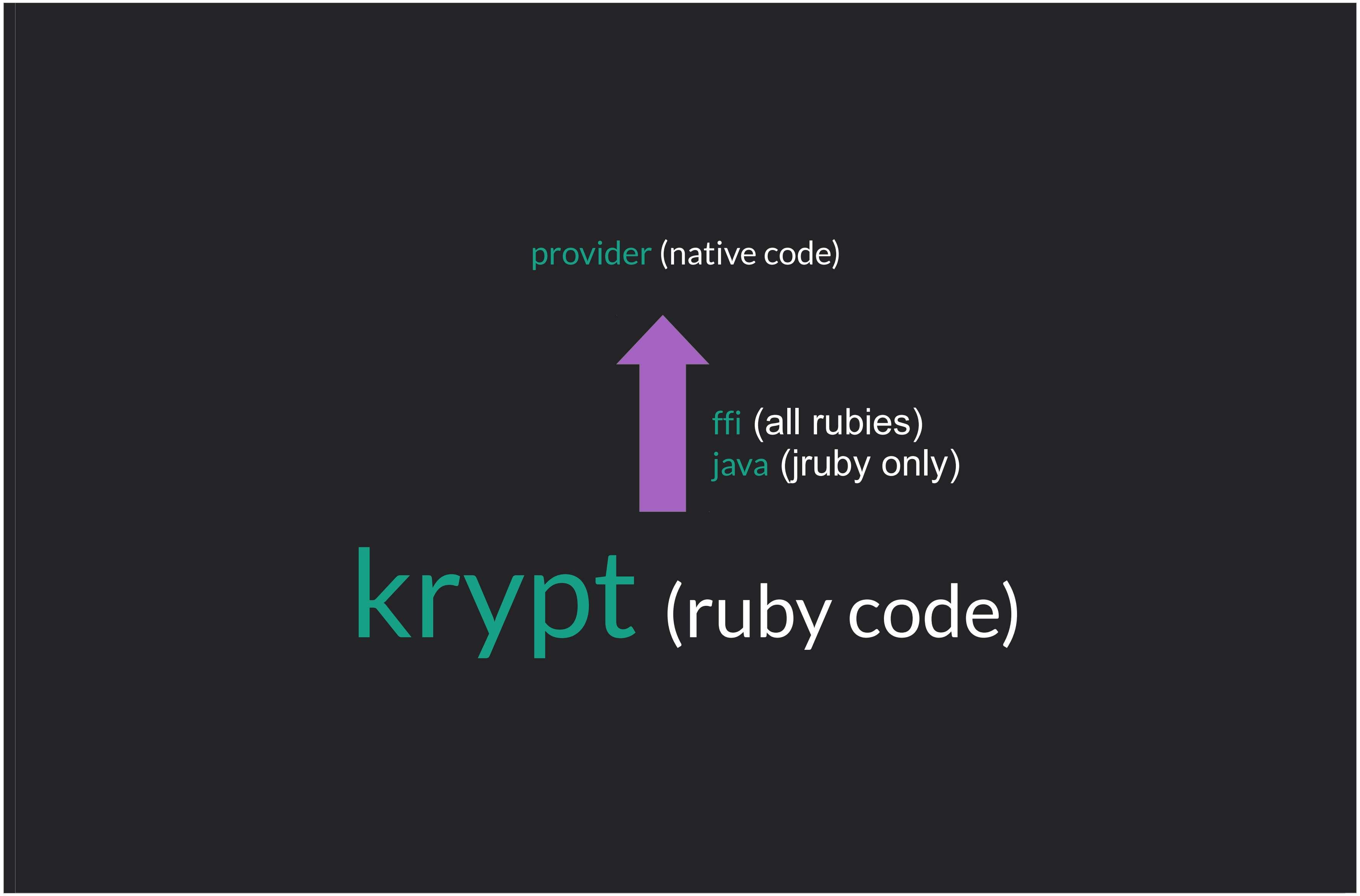
openssl isn't available everywhere

yet another

c (or java) extension is not the answer

only ruby runs on every ruby

this is part of java's success


provider (native code)
ffi (all rubies)
java (jruby only)
krypt (ruby code)

„what's yours is mine!“

use openssl on jruby

benefit from fips validation

w/o changing any of the code

le demo

same code runs on all platforms

using different parts to get there

write once, run anywhere$^{\text{TM}^2}$

future: all-ruby provider

„if it runs ruby, it runs krypt“

think:

webrick

vs.

thin, unicorn, puma, torquebox, ...

off-topic: other programming langs

c-based: tightly coupled to openssl

~~write once, run anywhere™^{2}~~

aka: „write many times, run nowhere“

java-based:

tightly coupled to java security api
or bouncy castle

write once, run anywhere$^{\text{TM}^2}$

but:

no c libraries
expert™ api only

why not give those guys a break, too?

make krypt a full-blown c & java lib

blueprint for jvm- and c-based languages

write once, run anywhere$^{\text{TM}^2}$

krypt all the things

once we take over the world

we make sure to be gentle

to the people in this room

if nobody picks up the idea?

„at least we'll have
flawless design“

is it any good?

participating in jruby gsoc '12 & '13

two projects featuring devwrat & matthew https://github.com/jruby/jruby/commit/cc9acbaf2

„Incorporate Krypt and wire it up for OpenSSL::PKCS5.“

ONE SMALL STEP FOR MAN,
ONE GIANT LEAP FOR MANKIND...

„thank you!!!“

still a long road ahead of us

plan: krypt as default crypto library

sneaky plan:

use openssl shim for the interim period

https://github.com/krypt/krypt-ossl

thank you

https://github.com/krypt

http://martinbosslet.de

martin.bosslet@gmail.com

@_emboss_