krypt & jruby

cryptography's new best friends?

„so will you tell us how krypt can protect our privacy? I heard crypto can do that!“

krypt & jruby.

problems of crypto apis today

problem #1

crypto is hard

only a select few

can really master

the arcane art that is cryptography

krypt & jruby.

```
describe Cryptography do

  it 'shouldn't be touched by mere mortals' do
    expect {
      write_crypto(:joe_programmer)
    }.to raise_error(ArgumentError)
  end

  it 'should only be written by experts in the field' do
    expect { hire_expert }.to be_the_default
  end

end
```

krypt & jruby.

```
describe Cryptography do
  context 'reality' do

    let(:expert) { :joe_programmer }

    it 'shouldn't be touched by mere mortals' do
      hope { write_crypto(expert) }.not_to raise_error(ToldYaSo)
    end

    it 'should only be written by experts in the field' do
      expect { hire_expert }.to raise_error(BudgetError)
    end

  end
end
```

so here's your catch-22:

every app needs a security guy

but not
every security guy needs your app

can't expect everyone to become

an overnight expert

somebody needs to get the job done

problem #2

arcane tools for arcane people

millions of parameters

trillions of combinations

about seven of them are secure

example:

password hashing with pbkdf2

what it should look like

krypt & jruby.

```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

krypt & jruby.

```
require 'openssl'

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

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

krypt & jruby.

```
require 'openssl'

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

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

krypt & jruby.

```
require 'openssl'

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

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

krypt & jruby.

```ruby
require 'openssl'

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

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

what it actually actually looks like

krypt & jruby.

```ruby
require 'openssl'

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

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

krypt & jruby.

this is typical

many different options

that do roughly the same thing

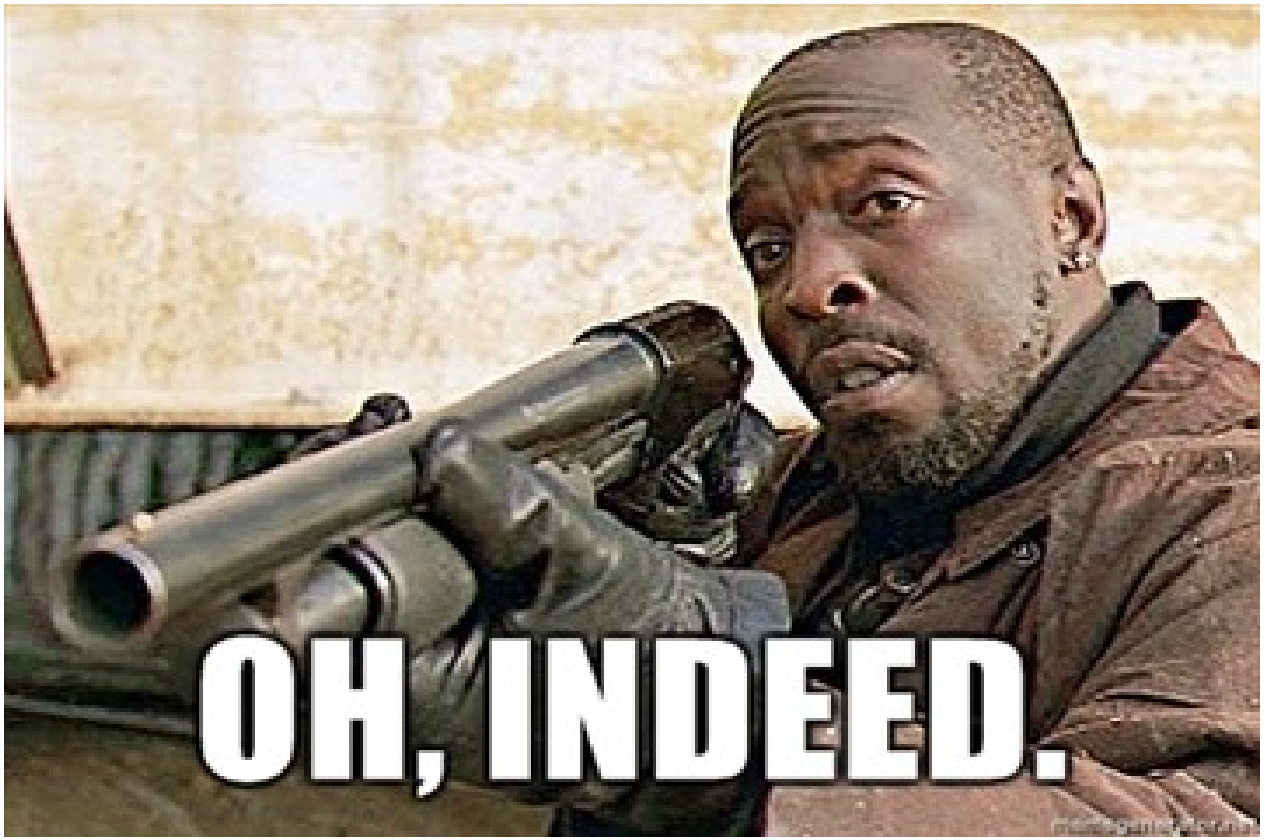
but only one is secure

none of them are documented

„ok, let's see, here's what i got:
aes with 128, 192 and 256 bit keys.
you want modes?
here's ecb, cbc, ofb, cfb,
ctr, and, as of recently gcm.
don't forget the iv!

ps: they're all insecure to some extent,
gcm is probably the only one you might
wanna use. but please, try the others,
it'll be good for your learning curve! “

krypt & jruby.



what we need

simple-to-use API

secure defaults

cover broader variety

of common scenarios

crypto api for mere mortals

„wait a minute, dude -
what about
keyczar or nacl / libsodium?
i've heard they do just that.

so why
bother with yet another library?“

problem #3

„it's either expert™ or high-level“

camp ivory tower

openssl
nss
ms capi
ms cng
gnutls
polarssl
...

camp mere mortals

nacl / libsodium
keyczar

camp ivory tower is good for

full control over all parameters

optimizing performance

shooting yourself in the foot

regularly

camp mere mortals is good for

getting things done

even securely

why not use them all the time?

legacy apps

OH:

„i need to pull encrypted data from your app, you know the one last updated in the 90s. is it cool if i use keyczar for that?“

„sure it is, honey. if your geezer or what's-his-name can do des with 56 bit keys.“

„but... but... your encryption scheme is totally insecure. are you sure you wanna leave it like that?“

„(laughs) oh, dear, security (laughs harder). who cares? nobody here knows cobol anymore anyway. just get it done. (threatening) we're on a schedule here!“

ideally:

crypto library gives us both

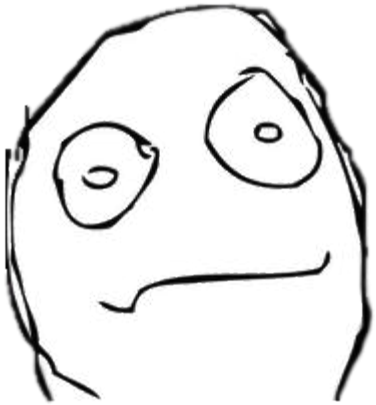
full control when needed

security by default otherwise

revolutionary idea

offer both in one single library

krypt. semper pi.



„yo bro, wasn't this supposed to be jrubyconf?
what's this guy doing? hasn't lost a single word about jruby yet.
let's punch him in the face!“

„sure, if i can keep his money.
krypt my ass, dude.“

problem #4

c rules everything around me

c-based languages

openssl is ubiquitous

ruby c extensions

blessing or curse?

pros:

gets the job done

good performance

cons:

interoperability

even for c-based rubies

think:
*nix, os x, windows
32/64 bit
endianess
gc
native threading
violations of duck typing
...

real problem in jruby land

true, there is ffi

but still:

platform interop issues remain

jit doesn't help much

potentially instabilizes the jvm

without ffi you're screwed

typical problem

many ruby / rails apps use

c extensions

hard to make the switch

jruby-only java extensions

even worse:

no ffi

duplicate the work

drop-in replacement?

summary:

native extensions are

more curse

than blessing

if you don't believe me,
see for yourself on stackoverflow
what a 'smooth experience'
ruby openssl
is right now

ruby openssl

is a pain to emulate

in jruby

maintaining its c version

isn't particularly funny either

-> only ruby code is truly interoperable

and jruby is fast enough to run it

eat your own dog food

(a part of java's success)

what does this mean for krypt?

primary goal:

run on all rubies, equally well

option to avoid c / java where possible

embrace & use ruby

while we do use ffi

we don't stop there

jdk has a perfectly fine crypto library

let's use it

future: all-ruby implementation

diversity

choose what fits best in your situation

rails development:

webrick („it just works“)

vs.

unicorn / thin / puma etc.
(„optimal performance“)

we want the same for krypt:

all-ruby implementation („it just works“)

openssl/ms cng/common crypto etc.

via ffi

(„optimal performance /
os availability“)

jdk security api/bouncy castle etc.

via java extensions

(„optimal performance /
availability in java“)

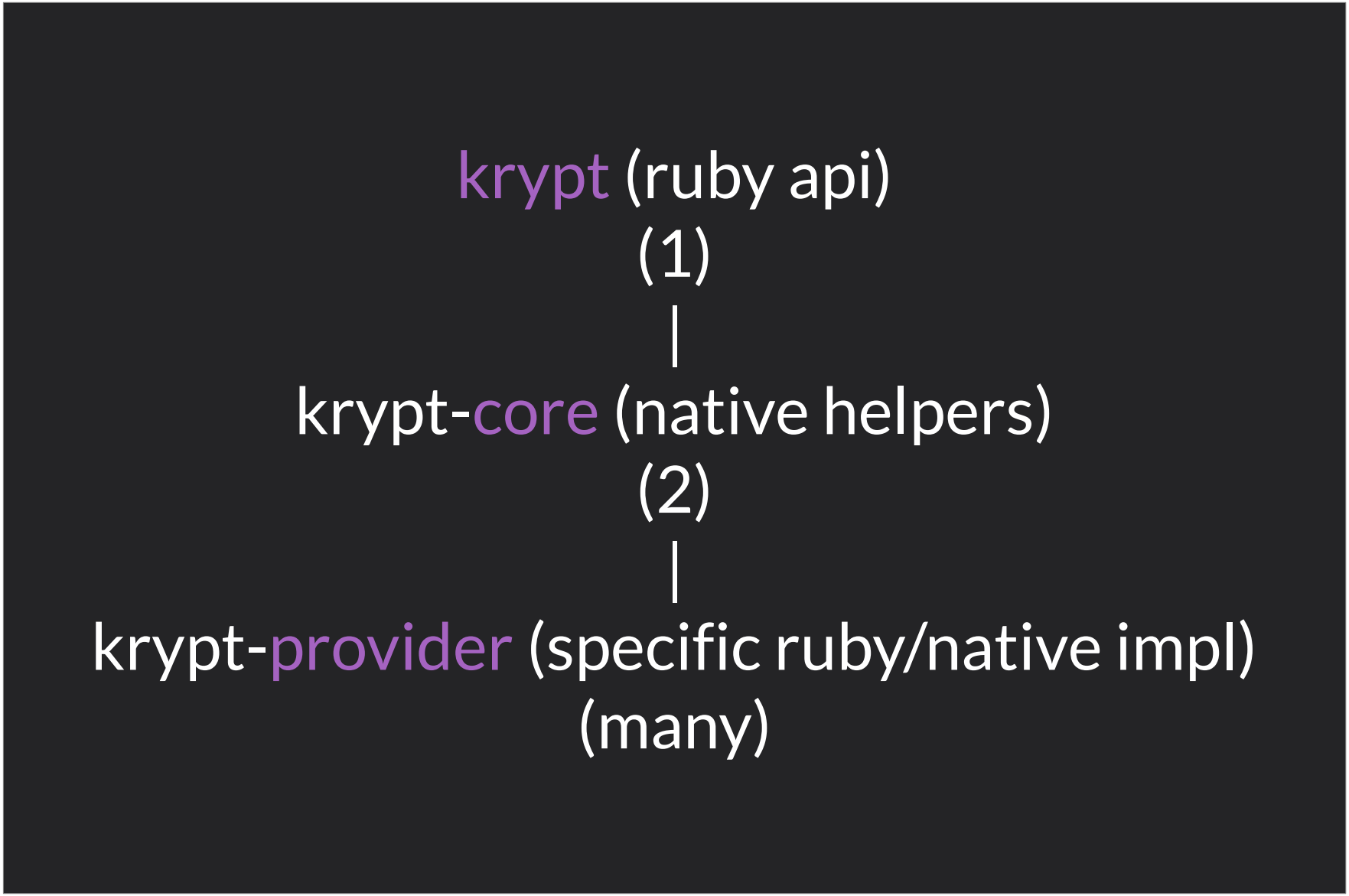
how does it work technically?

provider api defined on the ruby level

different implementations as

gem dependencies

krypt & jruby.


krypt (ruby api)
(1)
|
krypt-core (native helpers)
(2)
|
krypt-provider (specific ruby/native impl)
(many)

easy to add custom providers

helpers for native code

taking care of the boilerplate

krypt is much more than

„yet another crypto library“

crypto framework

with pluggable components

le demo

where krypt is used today

part of jruby

krypt & jruby.

https://github.com/jruby/jruby/commit/cc9acbaf2

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

plan: krypt as default crypto library

(not just in jruby)

openssl shim/polyfill for the interim period

replace openssl components

piece after piece

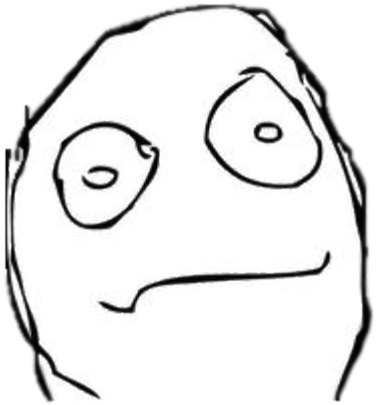
jruby gsoc '13

paves the way for encryption api

still a long road ahead of us

final thought:

why should you care at all?



„linda does all the security code
in our company. i'm not ever gonna touch
that stuff.“

„me neither. we got joe for that.
what do i care about
this guy's stupid crypto library?“

https

if https doesn't work,

ruby doesn't work

https://rubygems.org

krypt. semper pi.

thank you

https://github.com/krypt

http://martinbosslet.de

martin.bosslet@gmail.com

@_emboss_