krypt. semper pi.

whoami

ruby-core

ruby openssl

freelancer

whoami

germany

ドイツ人

crypto is hard

暗号は難しい

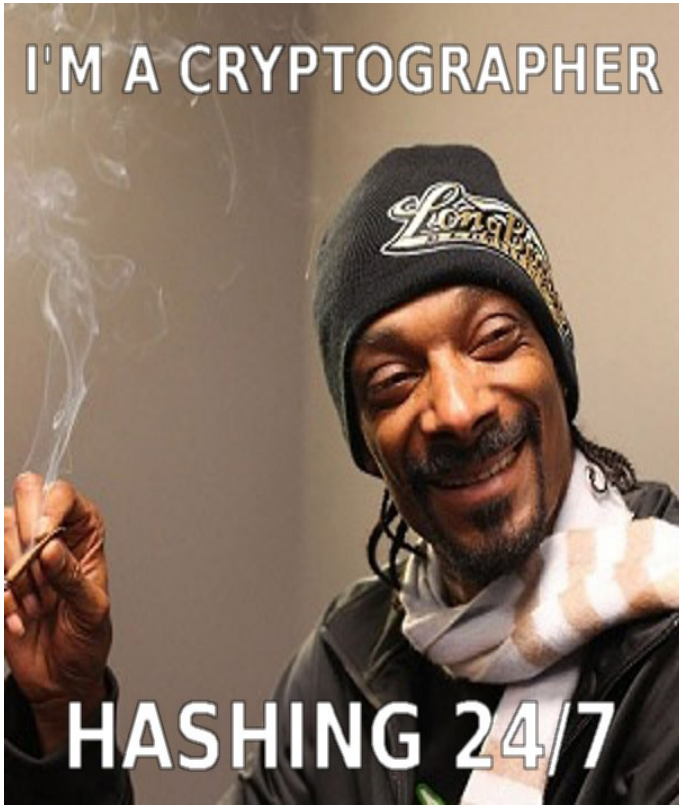
u can't touch this

ムリ

touched crypto

暗号に触れたヒト

gets crypto



モノにしたヒト：
なんでもハッシュ！

so, if crypto is hard

暗号が難しいとして

do crypto apis have to be, too?

暗号 API も難しい？

two opposing forces

相反するチカラ

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

「ログイン部分をいじる時はセキュリティのことを考えてた。誰がアルゴリズムを気にする？私は自分の問題を解決したいんだ！」

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

「私は最近 SSL 筆記試験を受けた。全てのパラメータの細かな制御が必要。 あと古いアルゴリズムもな」

everyday development:

日々の開発：

security sucks

セキュリティなんてくそくらえ

„oh no, not security again!“

「二度とセキュリティなんて!」

security is the jar jar binks
of software development

セキュリティはソフトウェア開発の嫌われ者

it's also a huge pain

巨大な痛み

that unfortunately cannot be overlooked

残念ながら見逃してもらえない

and must be dealt with

対処が必要

security can be a real bitch

セキュリティはビッチになりうる

THIS BITCH

good design
==
making complex things easy

よいデザイン == 難しいことを簡単に

databases are hard -> active record

データベースは難しい → **active record**

threads are hard -> stm, actors

Thread は難しい → STM 、アクター

memory is hard -> gc

メモリ管理は難しい → GC

don't bother me with details

細かいことで煩わされたくない

but experts™ need full control

専門家は「全ての制御」をしたい

conflict

矛盾

abstraction vs. oversimplification

抽象化 vs 過度の単純化

there is no golden middle

中庸はない

if full control is your thing:

「全ての制御」が必要なら：

openssl
java security api
&
friends

if all you care about is security by default:

「デフォルトでの安全性」が必要なら：

keyczar (no friends)

（※ Google Security Team がそういう目的で作ったやつ）

(maybe (rb)nacl / libsodium)

あとは

why not just use keyczar and be done with it?

keyczar でやればいいじゃない？

well, there's the experts™

まあ､｢エキスパート｣とかいうヤツがいる

and legacy apps

あとレガシーアプリ

we need both

両方必要

full control if needed

必要なら「全ての制御」

security by default otherwise

そうでなきゃ「デフォルトで安全」

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

哀しいかな両方は無理。だよね？

revolutionary idea

革新的なアイデア

両方作ればいいし？

combine both aspects in one library

両側面を一つのライブラリに

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
?

「今あるもの」を置き換える？

we come in peace

平和的に

krypt first of all is a framework

krypt はまず第一にフレームワーク

using existing libraries

to implement

core cryptography primitives

「今あるもの」を使って
暗号の基本要素を作る

peaceful coexistence

(at least at first)

平和的共存（少なくとも当面は）

expert™

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

「全ての制御」でエキスパー満足する（※手抜き訳）

digest

cipher

signature

ダイジェスト・暗号化・署名

krypt adds asn.1 dsl to the mix

krypt は ASN.1 に DSL を持ち込む

(asn.1 being crypto's lingua franca)

（ASN.1 は暗号（※フォーマット）の共通言語）

„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?“

アプリを１から作るとして、
セキュアにするにはどうする？

implement higher-level protocols
using this basis

この暗号要素を使って
高レベルプロトコルを実装

what's the big deal?

大事なことはなんでしょう？

crypto code today

今ある暗号コードを見てみよう

#1 encrypting data

#1 データの暗号化

what it should look like

あるべきコード

```
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

実際に見られるコード

```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

#2 パスワードハッシュ

what it should look like

あるべきコード

```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

実際に見られるコード

```ruby
require 'openssl'

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

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

```ruby
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

とかいって本当の実際は

```
require 'openssl'

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

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

（※ストレッチしないただのハッシュｗ）

FEYENOORD
ROTTERDAM

#3 certificate validation

#3 証明書検証

what it should look like

あるべきコード

```
require 'openssl'

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

（わかるよね、時間ないし。。。）

almost impossible to do it correct™

「正しく」やるのはほぼ無理

online revocation checks

オンラインでの失効確認

openssl refuses to have dependencies

openssl は依存ライブラリを許さない

relic of bygone times before

dependency management tools

依存管理ツールがなかった頃の遺物

we're left with half-assed validation

不完全な証明書検証

krypto code tomorrow

これからの krypto コード

what it should actually look like

あるべきコード

#1 encrypting data

#1 データ暗号化

krypt.semper pi.

```
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

#2 パスワードハッシュ

```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

#3 証明書検証

```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

細かいことに煩わされたくない

security by default

デフォルトで安全

ONE DOES NOT SIMPLY
REPLACE OPENSSL LIKE THAT

yes we can

use protocols

moar advantages:

更なる利点：

moar tests

より多くのテスト

rspec

FuzzBert

moar docs (non-expert™)

より多くのドキュメント
（非エキスパート向け）

moar ruby

より多くの ruby

easier to understand & maintain

理解・管理し易い

minimal portion of native code

ネイティブコードは最小限

using whatever library is available
in the background

基礎として使えるライブラリなら何でも使う

diversity™

「多様性」

the rest is plain ruby

後はただの ruby

„why should anyone care?“

「誰が気にする？」

write once, run anywhere™^{2}

「一度書けばどこでも動く」

run on all rubies

全ての ruby 実装で動作

https

if https doesn't work,

ruby doesn't work

https 使えなきゃ ruby 使えない

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

openssl isn't available everywhere

openssl はどこでも使えるわけじゃない

c (or java) extensions are not the answer

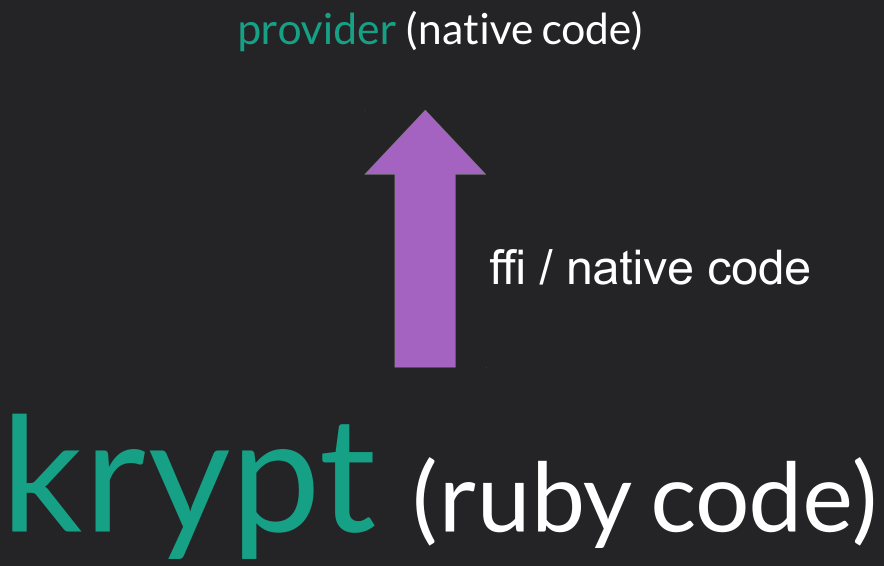
C や Java 拡張は解ではない

only ruby runs everywhere

ruby だけがどこでも動く

this is part of java's success

Java の成功の一つ



ネイティブコードによるプロバイダを使う

„what's yours is mine!“

「お前のものは俺のもの」

use openssl on jruby

JRuby で openssl を使う

w/o changing any of the code

コード書き換えなしに

le demo

same code runs on all platforms

同じコードが全てのプラットフォームで動く

using different parts to get there

異なるパーツを使って

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

future: all-ruby provider

将来：ruby で書いたプロバイダ

„if it runs ruby, it runs krypt“

「 ruby が動くなら krypt が動く」

think:

webrick

vs.

thin, unicorn, puma, torquebox, ...

off-topic: other programming langs

他のプログラミング言語では？

c-based: tightly coupled to openssl

C ベースの言語：openssl 依存

~~write once, run anywhere™^{2}~~

aka: „write many times, run nowhere“

java-based:

tightly coupled to java security api
or bouncy castle

Java ベースの言語：Java の API、
もしくは Bouncy Castle ライブラリ

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

but:

no c libraries
expert™ api only

しかし：C ライブラリは使えない
「エキスパート」向け

why not give those guys a break, too?

彼らにも休息を?

make krypt a full-blown c & java lib

krypt を C および Java ライブラリとしても提供

blueprint for jvm and c-based languages

JVM および C ベース言語なら
使えるように

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

krypt all the things

どこでも krypt

once we take over the world

もし成功したら

we make sure to be gentle

to the people in this room

この部屋にいる人たちに
寛大であります(?)

if nobody picks up the idea?

誰も見向きもしなかったら？

„at least we'll have
flawless design“
「少なくとも我々は
完璧なデザインを手に入れるでしょう」

is it any good?

役に立つ？

participating in jruby gsoc '12 & '13

JRuby の Google Summer of Code に参加

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

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

「 krypt を導入して PKCS5 で使う」

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

„thank you!!!“

still a long road ahead of us

まだまだ長い道のり

plan: krypt as default crypto library

計画：krypt をデフォルト（ ruby 標準）の
暗号ライブラリに

sneaky plan:

use openssl shim for the interim period

秘密の計画：
当面は openssl をラップして使う

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

thank you, @nahi

thank you

https://github.com/krypt

http://martinbosslet.de

martin.bosslet@gmail.com

@_emboss_