Go cryptography without bugs

(Ok, with fewer bugs)

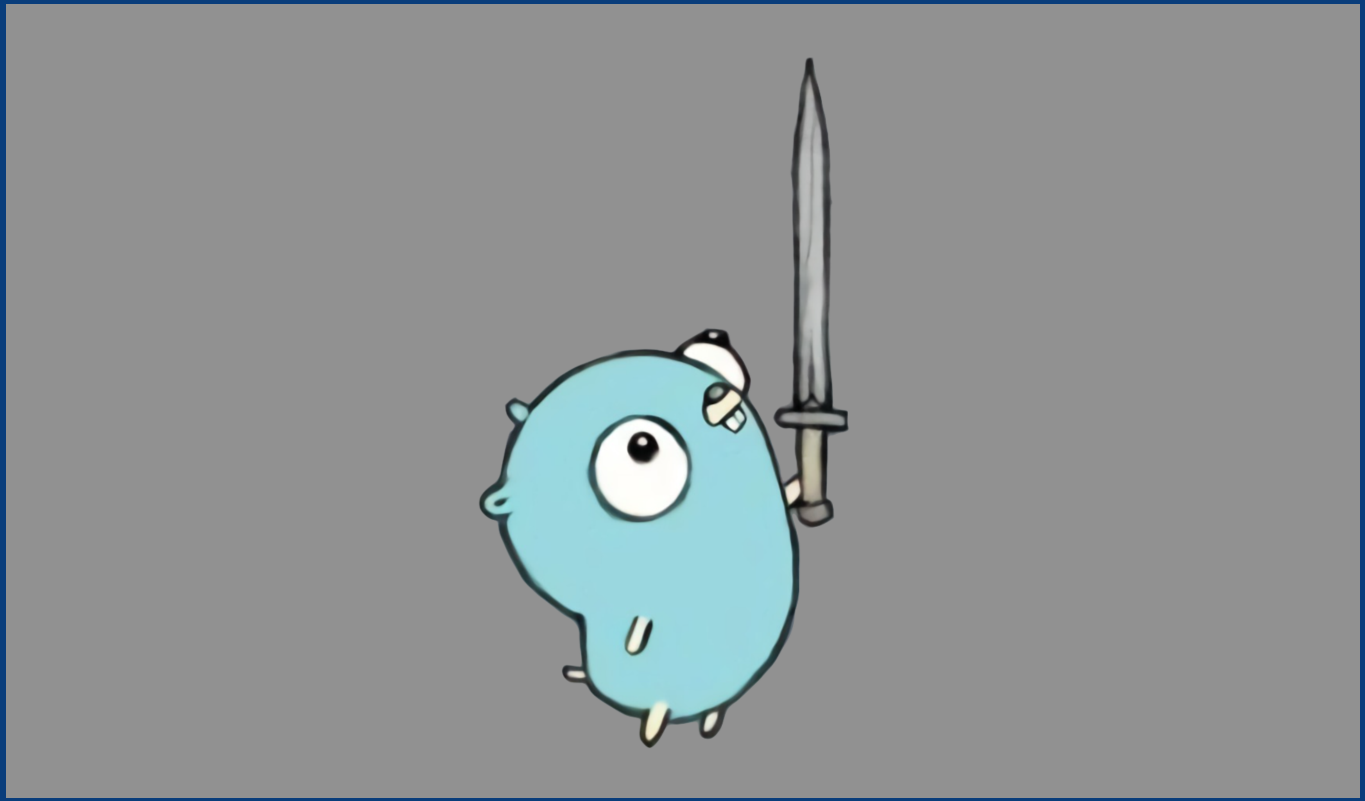
Filippo Valsorda

Filippo Valsorda

Go cryptography maintainer since 2018

age and mkcert

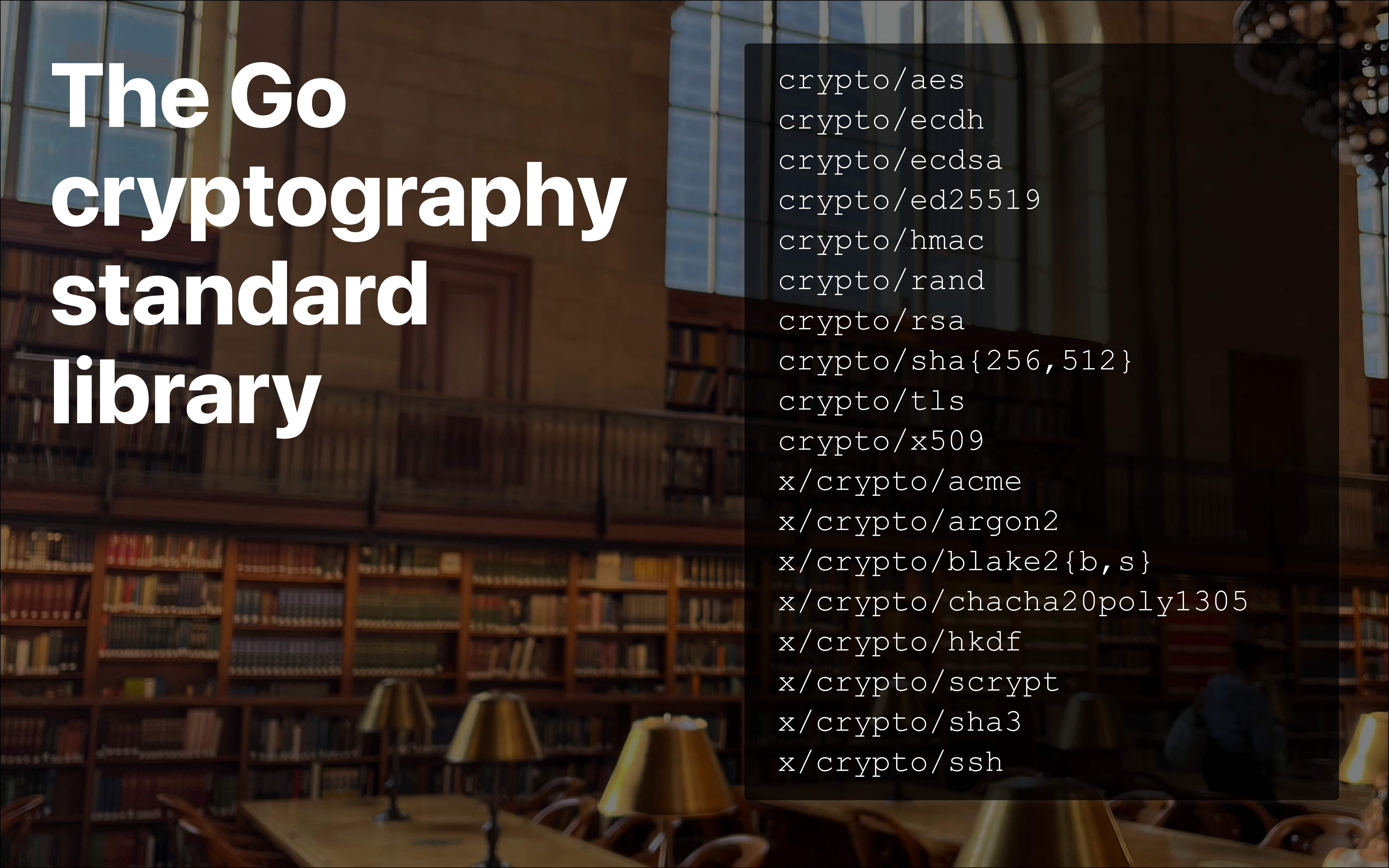
Go Checksum Database, TLS 1.3, Privacy Pass



age

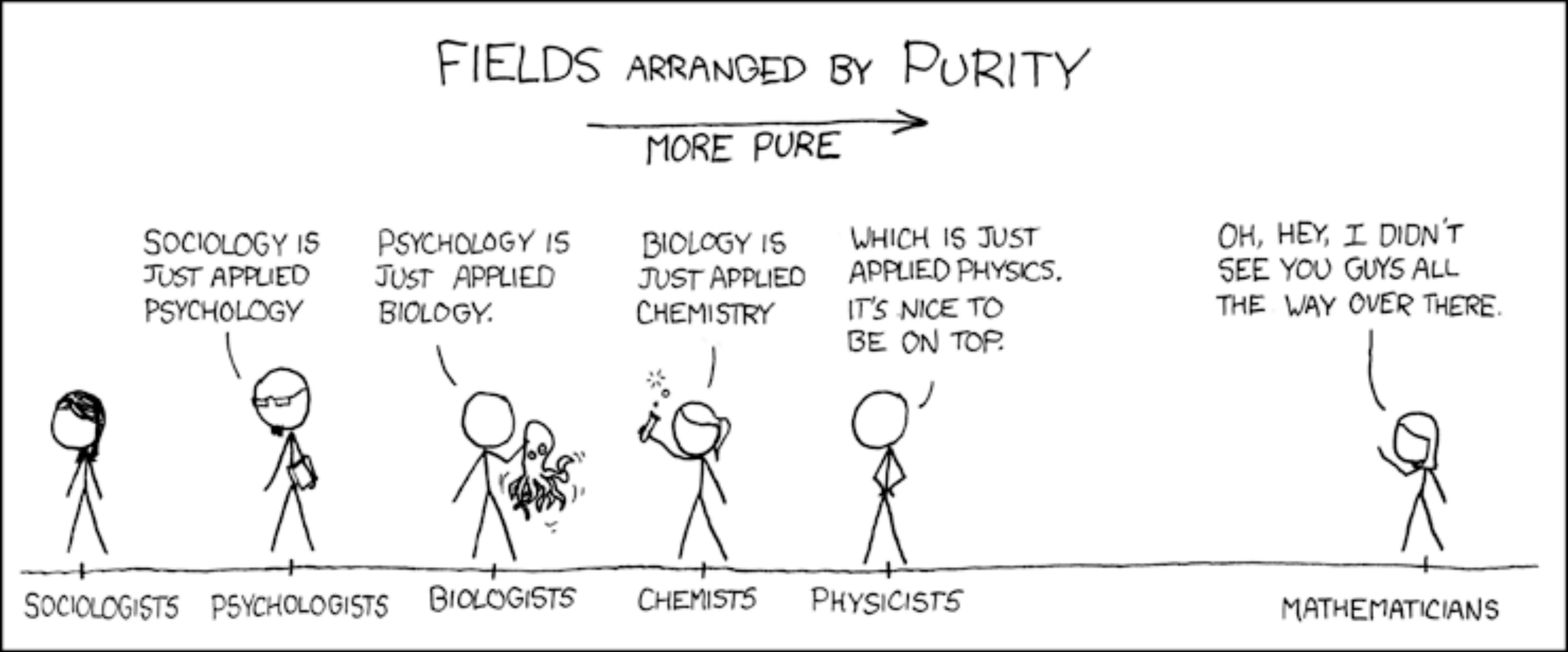
FILE ENCRYPTION

mkcert https://localhost



The Go cryptography standard library

```
crypto/aes
crypto/ecdh
crypto/ecdsa
crypto/ed25519
crypto/hmac
crypto/rand
crypto/rsa
crypto/sha{256,512}
crypto/tls
crypto/x509
x/crypto/acme
x/crypto/argon2
x/crypto/blake2{b,s}
x/crypto/chacha20poly1305
x/crypto/hkdf
x/crypto/scrypt
x/crypto/sha3
x/crypto/ssh
```


FIELDS ARRANGED BY PURITY
MORE PURE
SOCIOLOGY IS JUST APPLIED PSYCHOLOGY
PSYCHOLOGY IS JUST APPLIED BIOLOGY.
BIOLOGY IS JUST APPLIED CHEMISTRY
WHICH IS JUST APPLIED PHYSICS. IT'S NICE TO BE ON TOP.
OH, HEY, I DIDN'T SEE YOU GUYS ALL THE WAY OVER THERE.
SOCIOLOGISTS
PSYCHOLOGISTS
BIOLOGISTS
CHEMISTS
PHYSICISTS
MATHEMATICIANS

Memory safety,

tests, fuzzing,

safe APIs, code generation,

low complexity, readability

Memory safety

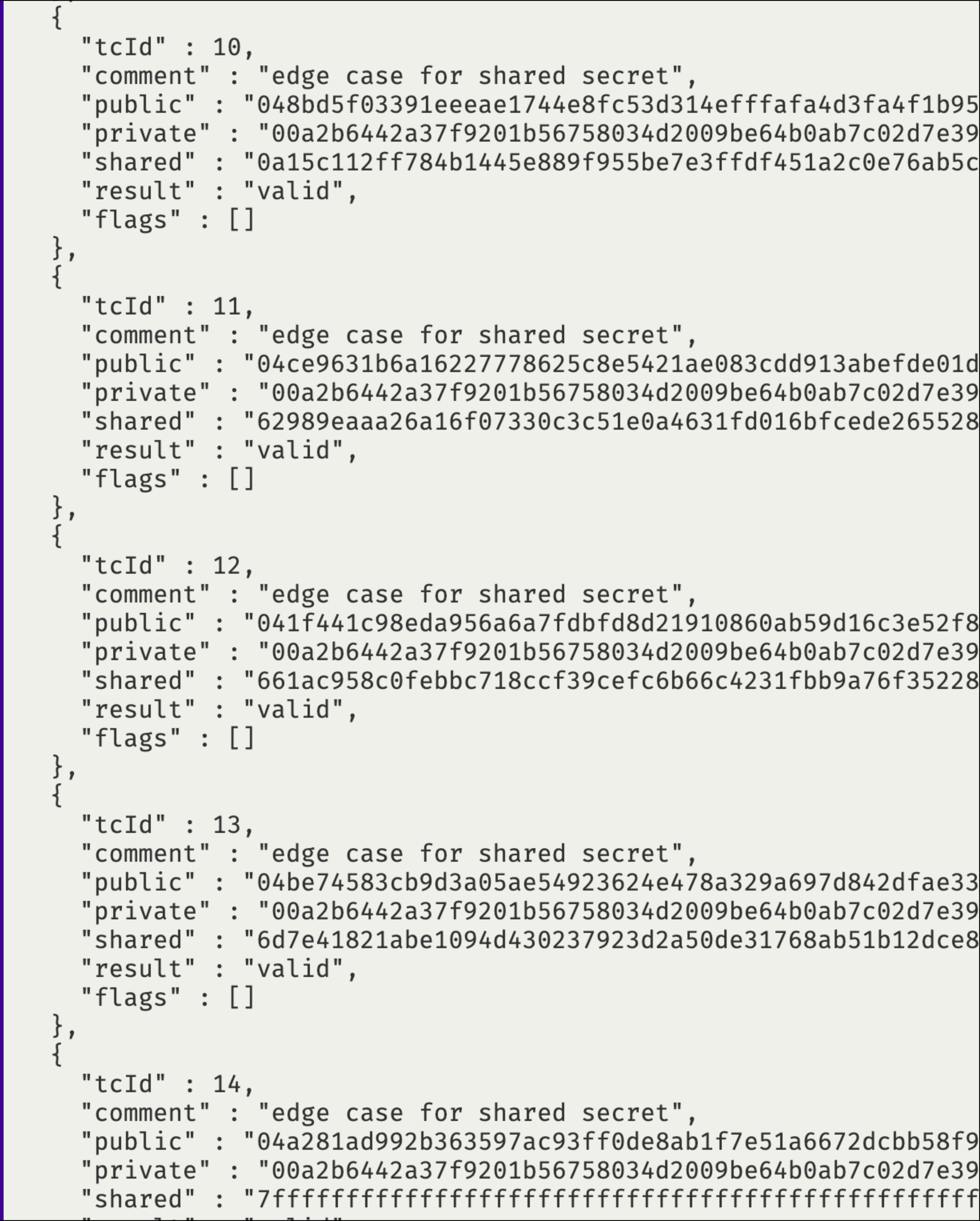
Bounds checks and garbage collector

(or, you know, 🦀👀)

Take the performance hit

Test vectors

Run tests!



```
  {
    "tcId" : 10,
    "comment" : "edge case for shared secret",
    "public" : "048bd5f03391eeeae1744e8fc53d314efffafa4d3fa4f1b95
    "private" : "00a2b6442a37f9201b56758034d2009be64b0ab7c02d7e39
    "shared" : "0a15c112ff784b1445e889f955be7e3ffdf451a2c0e76ab5c
    "result" : "valid",
    "flags" : []
  },
  {
    "tcId" : 11,
    "comment" : "edge case for shared secret",
    "public" : "04ce9631b6a16227778625c8e5421ae083cdd913abefde01d
    "private" : "00a2b6442a37f9201b56758034d2009be64b0ab7c02d7e39
    "shared" : "62989eaaa26a16f07330c3c51e0a4631fd016bfcede265528
    "result" : "valid",
    "flags" : []
  },
  {
    "tcId" : 12,
    "comment" : "edge case for shared secret",
    "public" : "041f441c98eda956a6a7fdbfd8d21910860ab59d16c3e52f8
    "private" : "00a2b6442a37f9201b56758034d2009be64b0ab7c02d7e39
    "shared" : "661ac958c0febbc718ccf39cefc6b66c4231fbb9a76f35228
    "result" : "valid",
    "flags" : []
  },
  {
    "tcId" : 13,
    "comment" : "edge case for shared secret",
    "public" : "04be74583cb9d3a05ae54923624e478a329a697d842dfae33
    "private" : "00a2b6442a37f9201b56758034d2009be64b0ab7c02d7e39
    "shared" : "6d7e41821abe1094d430237923d2a50de31768ab51b12dce8
    "result" : "valid",
    "flags" : []
  },
  {
    "tcId" : 14,
    "comment" : "edge case for shared secret",
    "public" : "04a281ad992b363597ac93ff0de8ab1f7e51a6672dcbb58f9
    "private" : "00a2b6442a37f9201b56758034d2009be64b0ab7c02d7e39
    "shared" : "7ffffffffffffffffffffffffffffffffffffffffffffffff
```

Test frameworks

The easier you make writing tests, the more you will have.

Examples:

- BoringSSL’s BoGo
- acmetest
- testscript
- age’s testkit

age testkit

```go
package main

import "filippo.io/age/internal/testkit"

func main() {
    f := testkit.NewTestFile()
    f.VersionLine("v1")
    f.X25519(testkit.TestX25519Recipient)
    f.HMAC()
    f.Payload("age")
    f.Generate()
}
```

age testkit

Serialized format

```
expect: success
payload:
013f54400c82da08037759ada907a8b864e97de81c088a182062c4b5622fd2ab
file key: 59454c4c4f57205355424d4152494e45
identity: AGE-SECRET-KEY-
1XMWWC06LY3EE5RYTXM9MFLAZ2U56JJJ36S0MYPDRWSVLUL66MV4QX3S7F6

age-encryption.org/v1
-> X25519 TEiF0ypqr+bpvcqXNyCVJpL7OuwPdVwPL7KQEbFDOCc
EmECAEcKN+n/Vs9SbWiV+Hu0r+E8R77DdWYyd83nw7U
--- Vn+54jqiiUCE+WZcEVY3f1sqHjlu/z1LCQ/T7Xm7qI0
[binary gibberish omitted]
```

Fuzzing

The easiest way to write test vectors is to let the computer come up with them

P-224 field: $2^{224} - 2^{96} + 1$

Limb size: 28 bits

Lots of shifts by 28 and by 96 % 28 = 12

Candidate values:

2^{28}, $2^{28} - 1$, 2^{12}, $2^{12} - 1$, 1, 0

plus their sums and differences

```
0xfffffff, 0xfffffff, 0xfffffff,
0xfffffff, 0xffff???, 0x???????,
0x???????, 0x0000000
```

Random chance: 2^{-154}

Weighted chance: 2^{-18}

Safe interfaces

Internal and external

Even better than finding bugs is being unable to write them

Before

```
package crypto/elliptic

type Curve interface {
    IsOnCurve(x, y *big.Int) bool

    ScalarBaseMult(k []byte) (x, y *big.Int)

    ScalarMult(x1, y1 *big.Int, k []byte)
        (x, y *big.Int)
}
```

After: public interface

```
package crypto/ecdh

type Curve interface {
    NewPrivateKey(key []byte) (*PrivateKey, error)

    NewPublicKey(key []byte) (*PublicKey, error)
}

func (k *PrivateKey) ECDH(r *PublicKey) ([]byte, error)
```

After: private interface

```
package crypto/internal/nistec

func (p *P256Point) Bytes() []byte

func (p *P256Point) SetBytes(b []byte) (*P256Point, error)

func (p *P256Point) ScalarMult(
    q *P256Point,
    scalar []byte,
) (*P256Point, error)
```

> *Software engineering is what happens to programming when you add time and other programmers.*
> — Russ Cox

Code generation

Sometimes it's safest to let the computer write the code

- fiat-crypto
- avo assembly

fiat-crypto generated code

```
package fiat

type p256MontgomeryDomainFieldElement [4]uint64

func p256Mul(
    out1 *p256MontgomeryDomainFieldElement,
    arg1 *p256MontgomeryDomainFieldElement,
    arg2 *p256MontgomeryDomainFieldElement)

func p256FromBytes(
    out1 *[4]uint64, arg1 *[32]uint8)
```

```go
import . "github.com/mmcloughlin/avo/[...]"

type uint128 struct {
    hi, lo GPVirtual
}

// addMul64 sets r to r + i * aX * bX.
func addMul64(r uint128, i uint64, aX, bX Component) {
    switch i {
    case 1:  Load(aX, RAX)
    default: IMUL3Q(Imm(i), Load(aX, GP64()), RAX)
    }
    MULQ(mustAddr(bX)) // RDX, RAX = RAX * bX
    ADDQ(RAX, r.lo)
    ADCQ(RDX, r.hi)
}
```

Complexity reduction

The safest code is the one you didn’t write

- Assembly Policy
- Cryptography Principles
- Deprecations
- Limited scope, flexibility
- 95% of use cases with 5% of code

Deprecated

(But not removed.)

~~x/crypto/openpgp~~

~~x/crypto/poly1305~~

~~crypto/dsa~~

~~crypto/elliptic~~

~~x/crypto/blowfish, bn256, cast5, md4, ripemd160, tea, xtea, twofish~~

> *No is temporary,*
> *yes is forever.*
> *— Solomon Hykes*

The Go Cryptography Principles

In this order.

Secure.

Safe.

Practical.

Modern.

Readability

If the code is complex it should be easier to read, not harder

Again, take the performance hit

We're not done!

We'll never be done!

Look at Go for your next paper.

filippo@golang.org

https://filippo.io/rwc2023/talk