CIRCL
A library for Post-Quantum and Elliptic Curve Cryptography

Armando Faz Hernandez
Cloudflare Research
ask-research@cloudflare.com

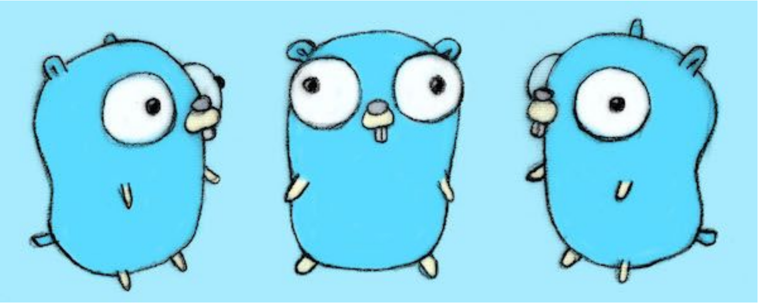
Open Source Cryptography Workshop
March 28th, 2024
Toronto, Canada

Agenda

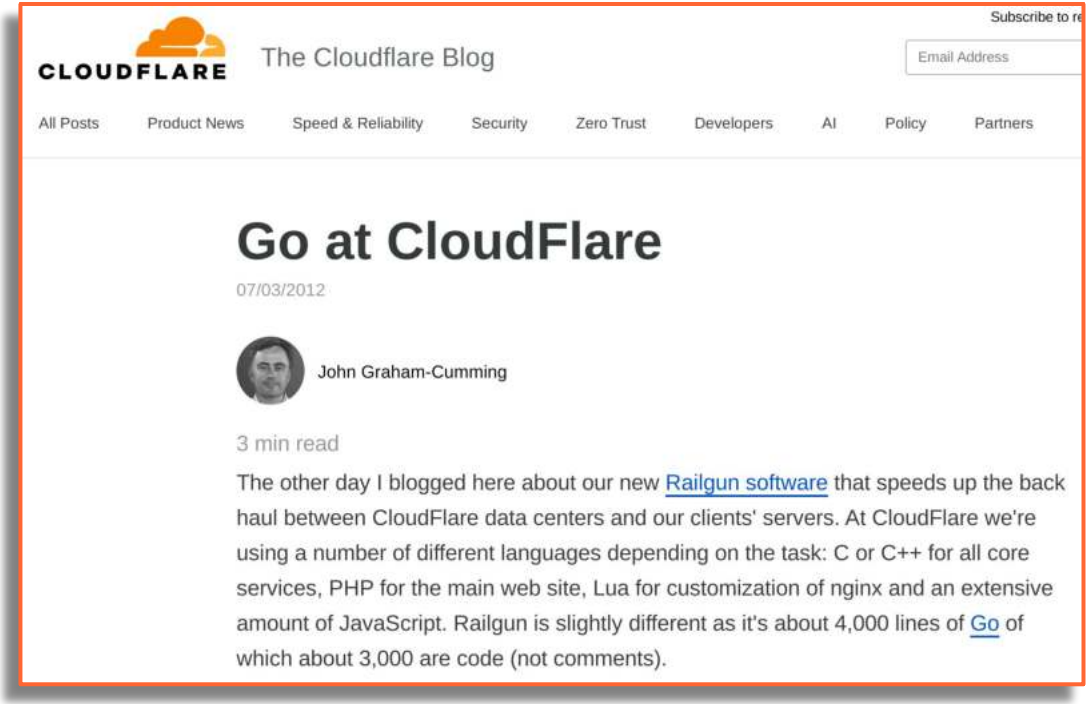
- Go in Cloudflare
- CIRCL
- Lessons
- Takeaways

Go Programming Language

- Compiled & strongly typed
- Garbage collection
- Standard library
- Algorithms for cryptography
  - Standard and experimental
- Pure Go & architecture-specific
- Integration with assembler (Plan 9)
- Easy to learn



Go at Cloudflare



Subscribe to re

CLOUDFLARE The Cloudflare Blog

Email Address

All Posts Product News Speed & Reliability Security Zero Trust Developers AI Policy Partners

Go at CloudFlare

07/03/2012

John Graham-Cumming

3 min read

The other day I blogged here about our new Railgun software that speeds up the back haul between CloudFlare data centers and our clients' servers. At CloudFlare we're using a number of different languages depending on the task: C or C++ for all core services, PHP for the main web site, Lua for customization of nginx and an extensive amount of JavaScript. Railgun is slightly different as it's about 4,000 lines of Go of which about 3,000 are code (not comments).

https://blog.cloudflare.com/go-at-cloudflare

Go concurrency model

Use of channels for communication

Easy to use for developers

Off-the-shelf libraries:

- HTTP, TLS, strings, compression, cryptography, etc

Go at Cloudflare

Cloudflared: Secure tunnel for origins

GoKeyless: TLS termination

cfssl & certmgr: Certificate management

GeoKey Manager: Encryption and distribution of keys

Roughtime: Authenticated time service

...

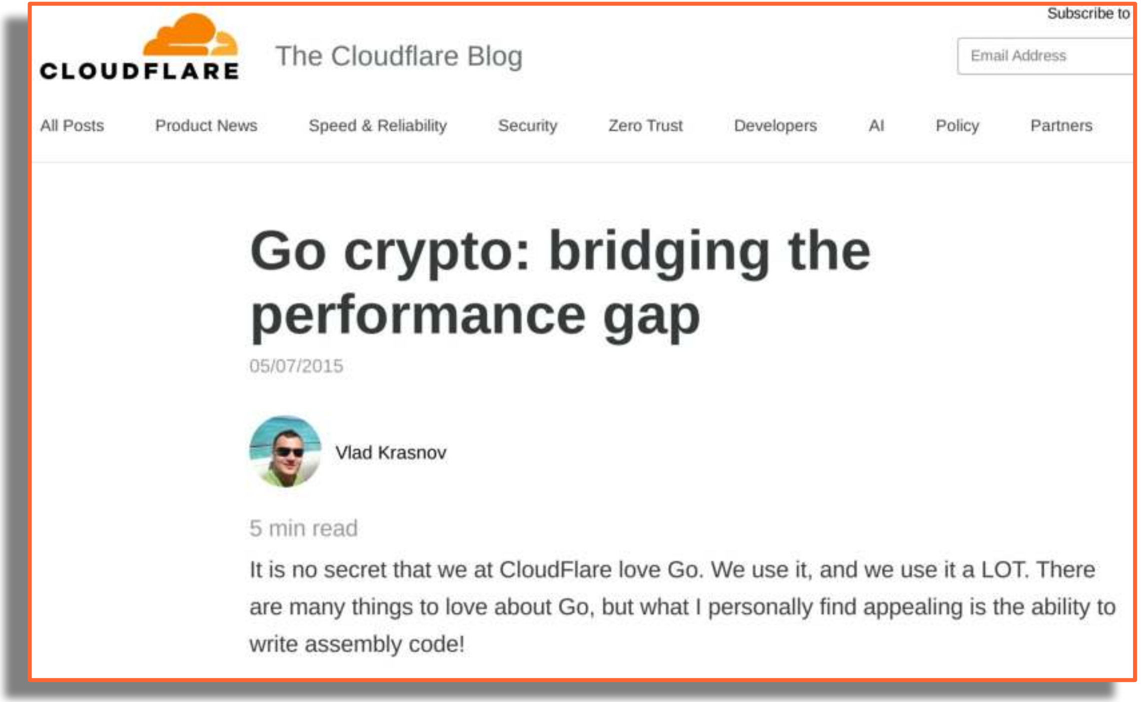
List of open source projects:

https://github.com/cloudflare?q=&language=go

Blog posts:

https://blog.cloudflare.com/tag/go/

Faster Crypto



The Cloudflare Blog

Go crypto: bridging the performance gap

05/07/2015

Vlad Krasnov

5 min read

It is no secret that we at CloudFlare love Go. We use it, and we use it a LOT. There are many things to love about Go, but what I personally find appealing is the ability to write assembly code!

https://blog.cloudflare.com/go-crypto-bridging-the-performance-gap/

Performance Improvements
Go v1.4

- RSA
- P256 Curve
- AES-GCM
  - Assembler for AES-NI

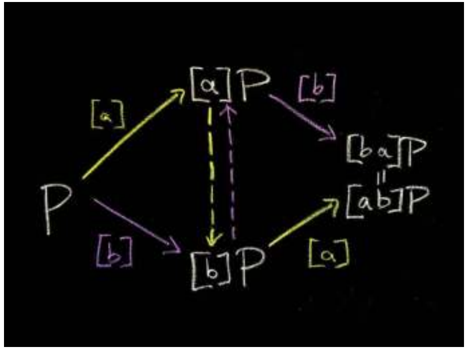
Fork of Go

https://github.com/cloudflare/go

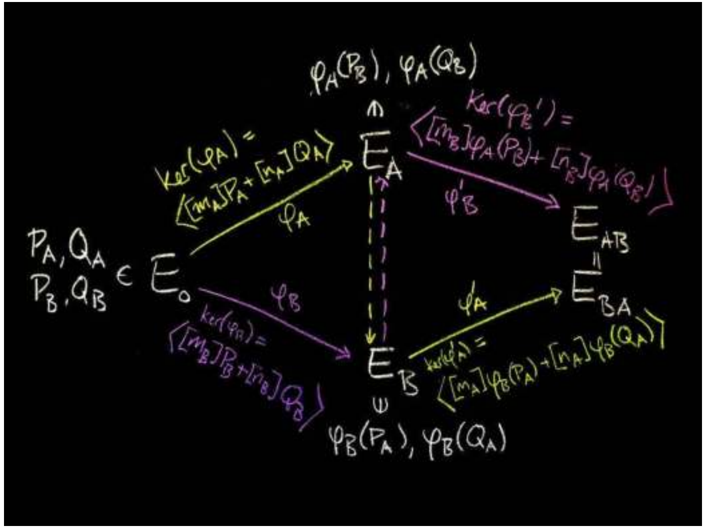
Post-Quantum Crypto

In 2017, Cloudflare started experimentation with PQ algorithms

Isogeny-based Cryptography (Jao-De Feo-Plût)
H. de Valence wrote SIDH in Go+Assembler
https://github.com/cloudflare/p751sidh



Elliptic Curve
Diffie-Hellman



Supersingular Isogeny
Diffie-Hellman

Variety of Applications

Customer Certificates
- Support for P-384 curve

New curves: Curve25519 & Goldilocks
- EdDSA & X Diffie Hellman

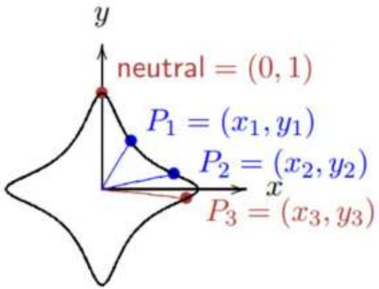
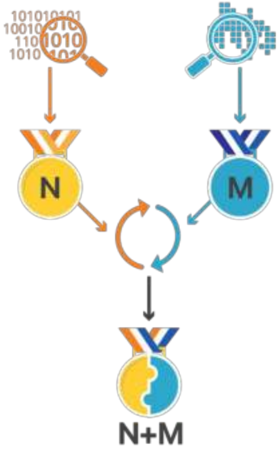


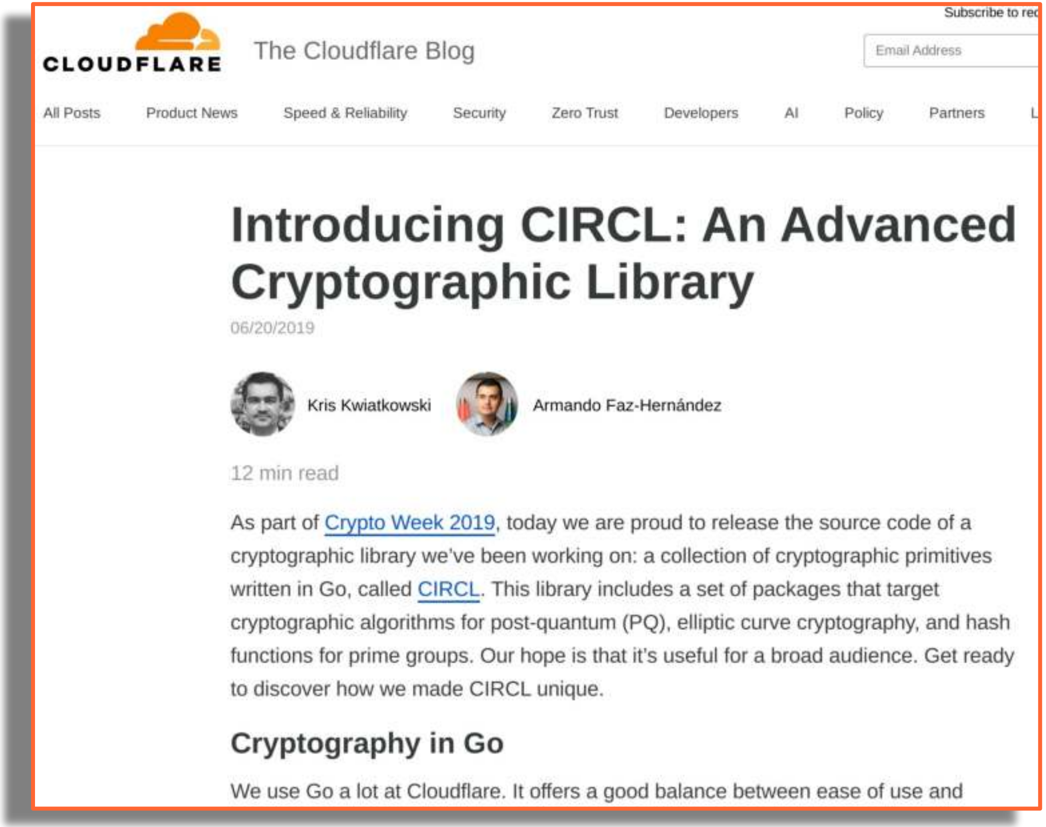
Figure 2.2: Addition law on a unit edwards curve

From El Housni Thesis



Distributed Randomness Beacon
- bn256 pairing-friendly curve

CIRCL



CLOUDFLARE The Cloudflare Blog

Subscribe to rec

Email Address

All Posts Product News Speed & Reliability Security Zero Trust Developers AI Policy Partners L

Introducing CIRCL: An Advanced Cryptographic Library

06/20/2019

Kris Kwiatkowski Armando Faz-Hernández

12 min read

As part of Crypto Week 2019, today we are proud to release the source code of a cryptographic library we've been working on: a collection of cryptographic primitives written in Go, called CIRCL. This library includes a set of packages that target cryptographic algorithms for post-quantum (PQ), elliptic curve cryptography, and hash functions for prime groups. Our hope is that it's useful for a broad audience. Get ready to discover how we made CIRCL unique.

Cryptography in Go

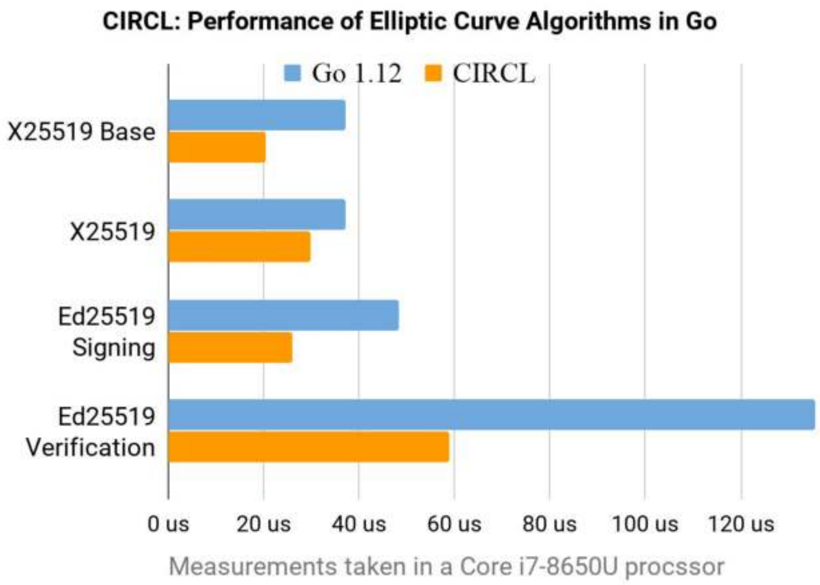
We use Go a lot at Cloudflare. It offers a good balance between ease of use and

Cloudflare
Interoperable,
Reusable
Cryptographic
Library

CIRCL

Goal: Experimentation with
Post-Quantum and
Elliptic Curve Cryptography

- SIDH → SIKE
- X25519/X448
- Ed25519
- P-384 optimizations (by B. McMillion)
- Hash to Curve



Elliptic Curve Groups

A **group** $\langle G, * \rangle$ is a set G, closed under binary operation $*$, such that the following axioms are satisfied:

1. For all $a, b, c \in G$, we have

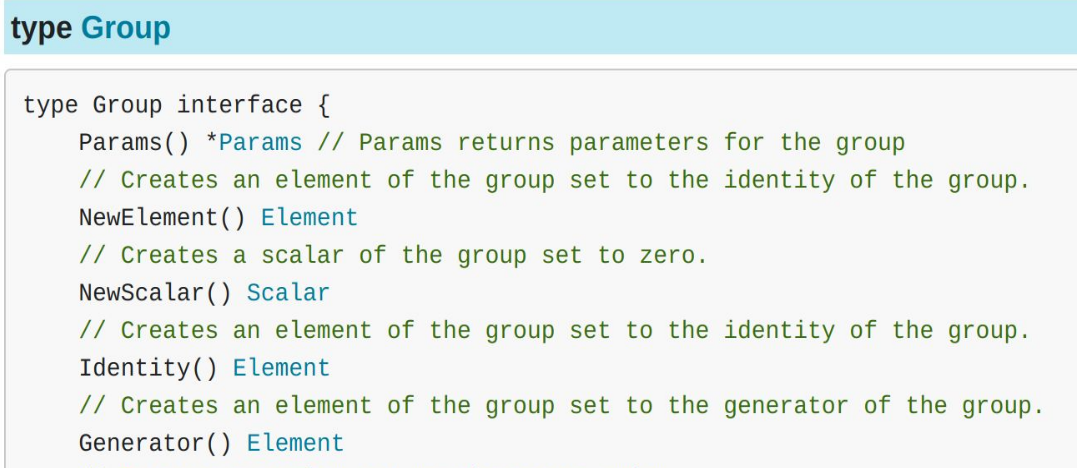
$$(a * b) * c = a * (b * c). \qquad \textbf{associativity of } *$$

2. There is an element e in G such that for all $x \in G$,

$$e * x = x * e = x. \qquad \textbf{identity element } e \textbf{ for } x$$

3. Corresponding to each $a \in G$, there is an element a' in G such that

$$a * a' = a' * a = e. \qquad \textbf{inverse } a' \textbf{ of } a$$



type Group

```
type Group interface {
    Params() *Params // Params returns parameters for the group
    // Creates an element of the group set to the identity of the group.
    NewElement() Element
    // Creates a scalar of the group set to zero.
    NewScalar() Scalar
    // Creates an element of the group set to the identity of the group.
    Identity() Element
    // Creates an element of the group set to the generator of the group.
    Generator() Element
```

Elliptic Curve Groups

A **group** $\langle G, * \rangle$ is a set G, closed under binary operation $*$, such that the following axioms are satisfied:

1. For all $a, b, c \in G$, we have

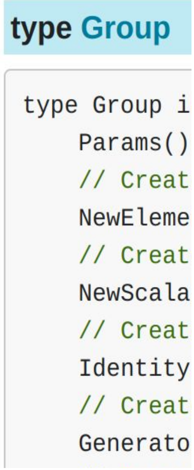
$$(a * b) * c = a * (b * c). \qquad \textbf{associativity of } *$$

2. There is an element e in G such that for all $x \in G$,

$$e * x = x * e = x. \qquad \textbf{identity element } e \textbf{ for } x$$

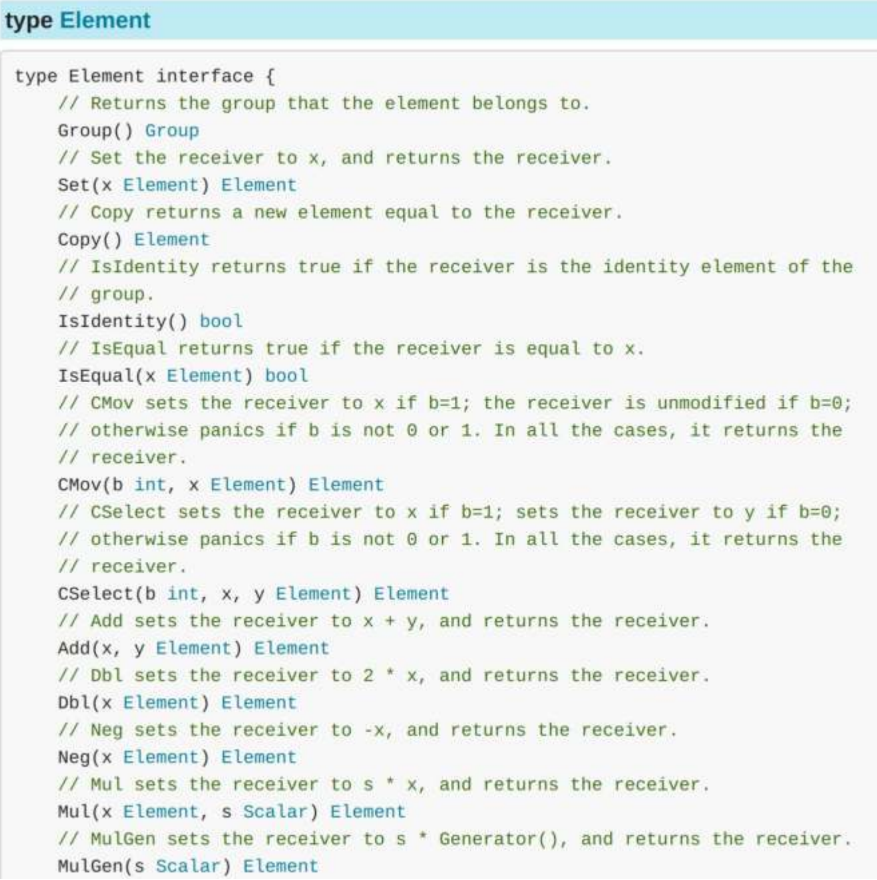
3. Corresponding to each $a \in G$, there is an element a' in G such that

$$a * a' = a' * a = e. \qquad \textbf{inverse } a' \textbf{ of } a$$



type Group

```
type Group i
    Params()
    // Creat
    NewEleme
    // Creat
    NewScala
    // Creat
    Identity
    // Creat
    Generato
```



type Element

```
type Element interface {
    // Returns the group that the element belongs to.
    Group() Group
    // Set the receiver to x, and returns the receiver.
    Set(x Element) Element
    // Copy returns a new element equal to the receiver.
    Copy() Element
    // IsIdentity returns true if the receiver is the identity element of the
    // group.
    IsIdentity() bool
    // IsEqual returns true if the receiver is equal to x.
    IsEqual(x Element) bool
    // CMov sets the receiver to x if b=1; the receiver is unmodified if b=0;
    // otherwise panics if b is not 0 or 1. In all the cases, it returns the
    // receiver.
    CMov(b int, x Element) Element
    // CSelect sets the receiver to x if b=1; sets the receiver to y if b=0;
    // otherwise panics if b is not 0 or 1. In all the cases, it returns the
    // receiver.
    CSelect(b int, x, y Element) Element
    // Add sets the receiver to x + y, and returns the receiver.
    Add(x, y Element) Element
    // Dbl sets the receiver to 2 * x, and returns the receiver.
    Dbl(x Element) Element
    // Neg sets the receiver to -x, and returns the receiver.
    Neg(x Element) Element
    // Mul sets the receiver to s * x, and returns the receiver.
    Mul(x Element, s Scalar) Element
    // MulGen sets the receiver to s * Generator(), and returns the receiver.
    MulGen(s Scalar) Element
```

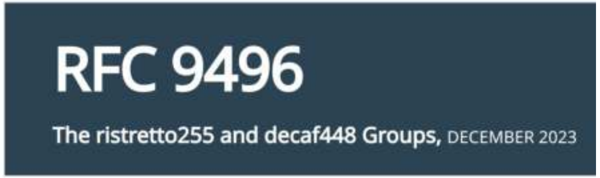
Elliptic Curve Groups

Prime Order Groups

- P256, P384, P521
- Ristretto, Decaf

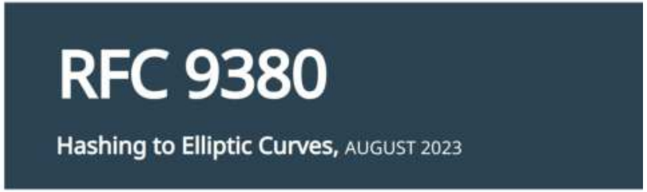
Hash to Curve

- Straight-line methods
- Encoding & Hash (Random Oracle Model)
- Hash to Field



RFC 9496

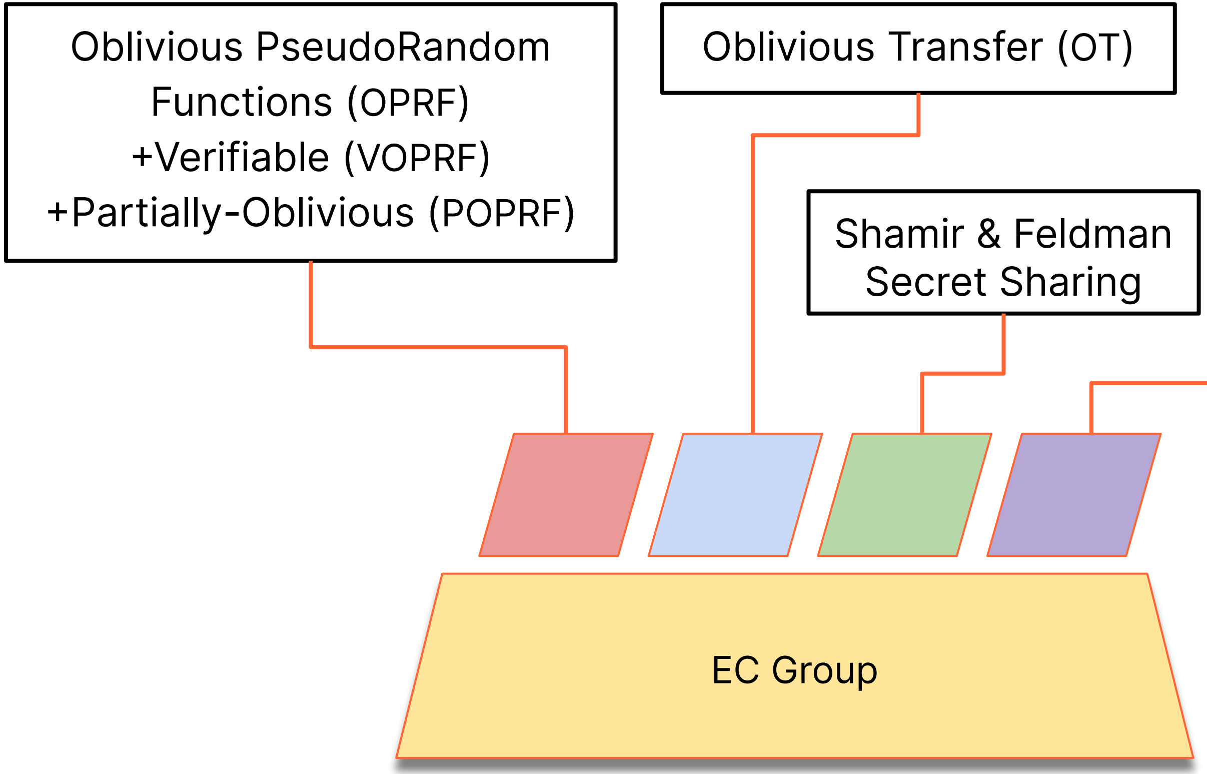
The ristretto255 and decaf448 Groups, DECEMBER 2023



RFC 9380

Hashing to Elliptic Curves, AUGUST 2023

Protocols based on Groups



Oblivious PseudoRandom Functions (OPRF)
+Verifiable (VOPRF)
+Partially-Oblivious (POPRF)

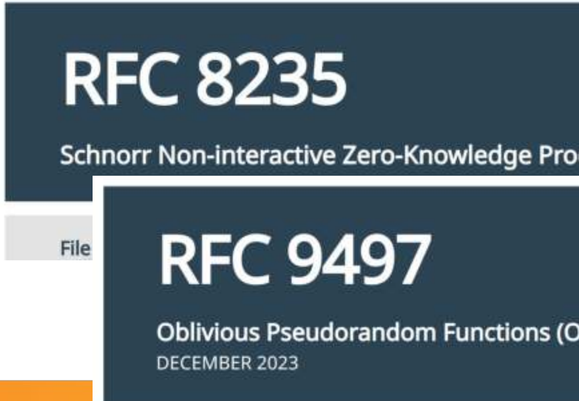
Oblivious Transfer (OT)

Shamir & Feldman Secret Sharing

ZK Discrete-Log (Schnorr)

ZK Discrete-Log Equivalence (DLEQ)

EC Group



RFC 8235

Schnorr Non-interactive Zero-Knowledge Pro

File

RFC 9497

Oblivious Pseudorandom Functions (O

DECEMBER 2023

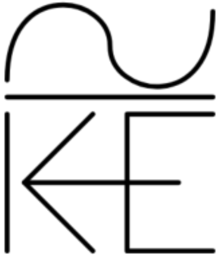
Post-Quantum Algorithms

SIDH (H. de Valence)

SIKE/CSIDH (K. Kwiatkowski)

Frodo (G. Tamvada)

Dilithium, Kyber (B. Westerbaan)



CSIDH

Post-Quantum Algorithms

SIDH (H. de Valence)

SIKE/CSIDH (K. Kwiatkowski)

Frodo (G. Tamvada)

Dilithium, Kyber (B. Westerbaan)

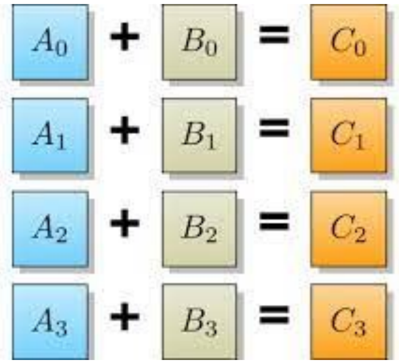
CSIDH

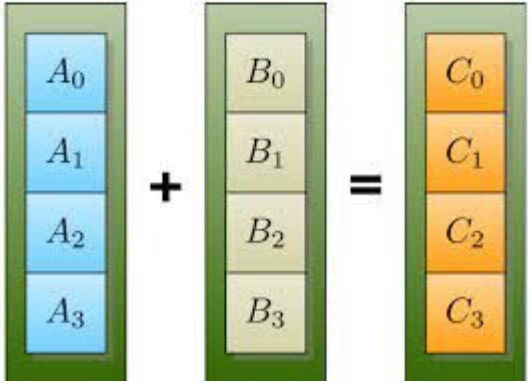
SIMD Execution

Parallel Keccak F-1600 Permutation

Either 2x or 4x parallelization

- NEON for ARM64
- AVX2 for x64



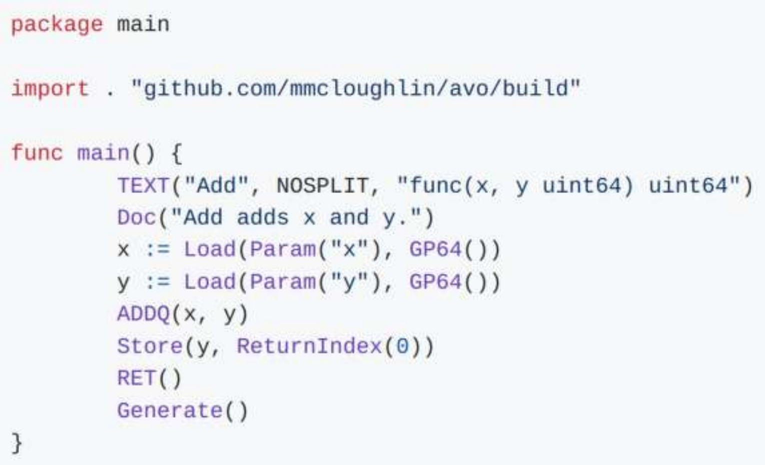


Avo: Assembly Generation in Go

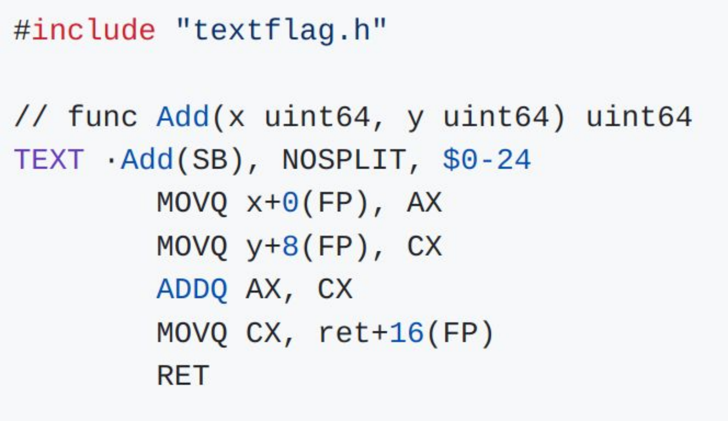
Written by M. McLoughlin.

Use Go code for writing x86 assembler

Easy access to AVX2 instructions



```go
package main

import . "github.com/mmcloughlin/avo/build"

func main() {
        TEXT("Add", NOSPLIT, "func(x, y uint64) uint64")
        Doc("Add adds x and y.")
        x := Load(Param("x"), GP64())
        y := Load(Param("y"), GP64())
        ADDQ(x, y)
        Store(y, ReturnIndex(0))
        RET()
        Generate()
}
```



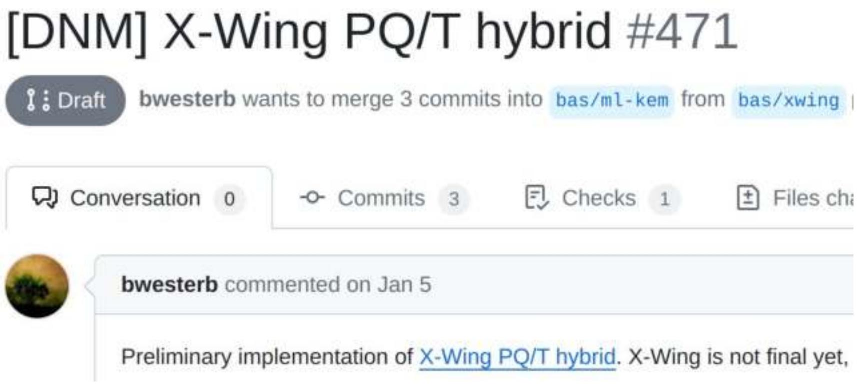
```
#include "textflag.h"

// func Add(x uint64, y uint64) uint64
TEXT ·Add(SB), NOSPLIT, $0-24
        MOVQ x+0(FP), AX
        MOVQ y+8(FP), CX
        ADDQ AX, CX
        MOVQ CX, ret+16(FP)
        RET
```

Hybrid: Pre- and Post-Quantum

Several proposals exist

Example:

Use X25519 and Kyber768



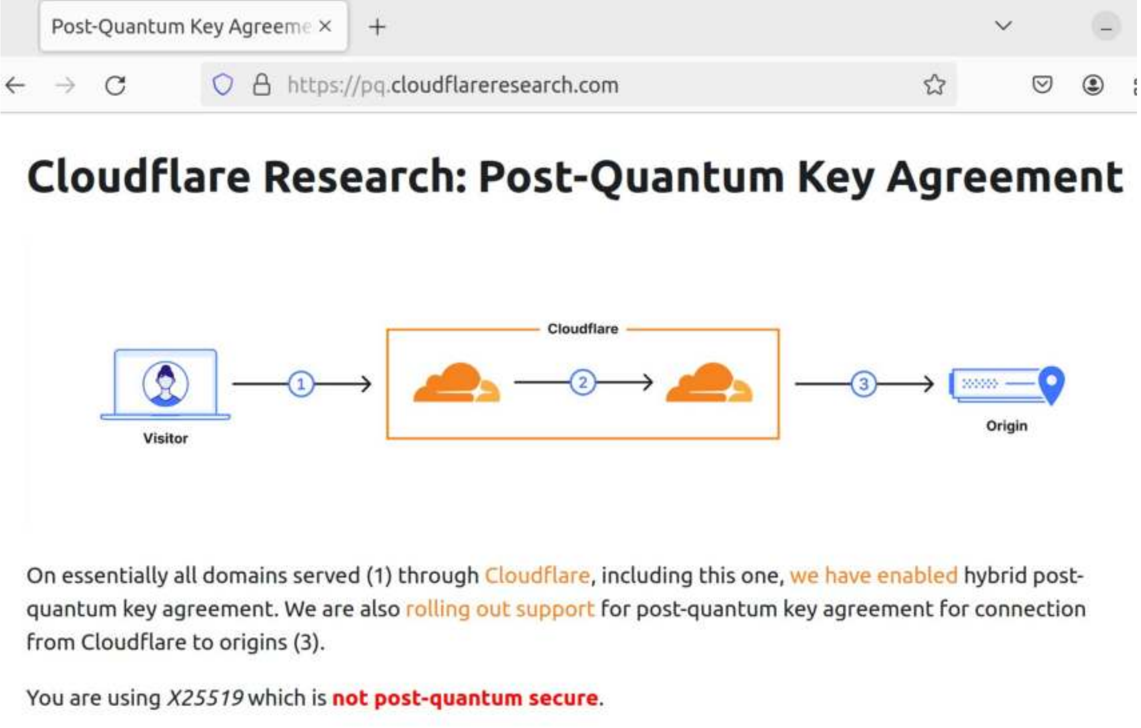
[DNM] X-Wing PQ/T hybrid #471

bwesterb wants to merge 3 commits into `bas/ml-kem` from `bas/xwing`

bwesterb commented on Jan 5

Preliminary implementation of X-Wing PQ/T hybrid. X-Wing is not final yet,

Kyber Supported at Cloudflare



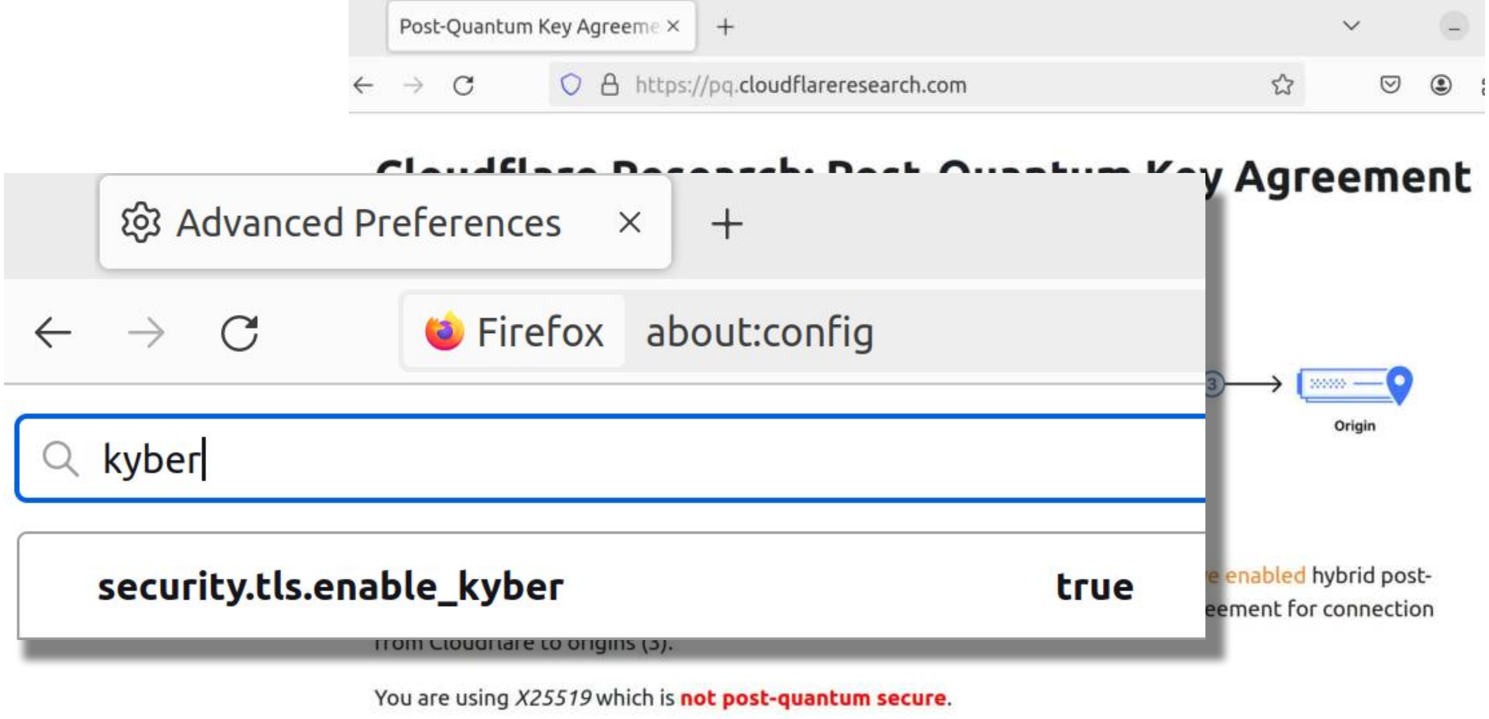
Cloudflare Research: Post-Quantum Key Agreement

On essentially all domains served (1) through Cloudflare, including this one, we have enabled hybrid post-quantum key agreement. We are also rolling out support for post-quantum key agreement for connection from Cloudflare to origins (3).

You are using *X25519* which is **not post-quantum secure**.

https://pq.cloudflareresearch.com/

Kyber Supported at Cloudflare



Post-Quantum Key Agreeme ×
https://pq.cloudflareresearch.com
Cloudflare Research: Post-Quantum Key Agreement
Origin
Advanced Preferences
Firefox
about:config

kyber

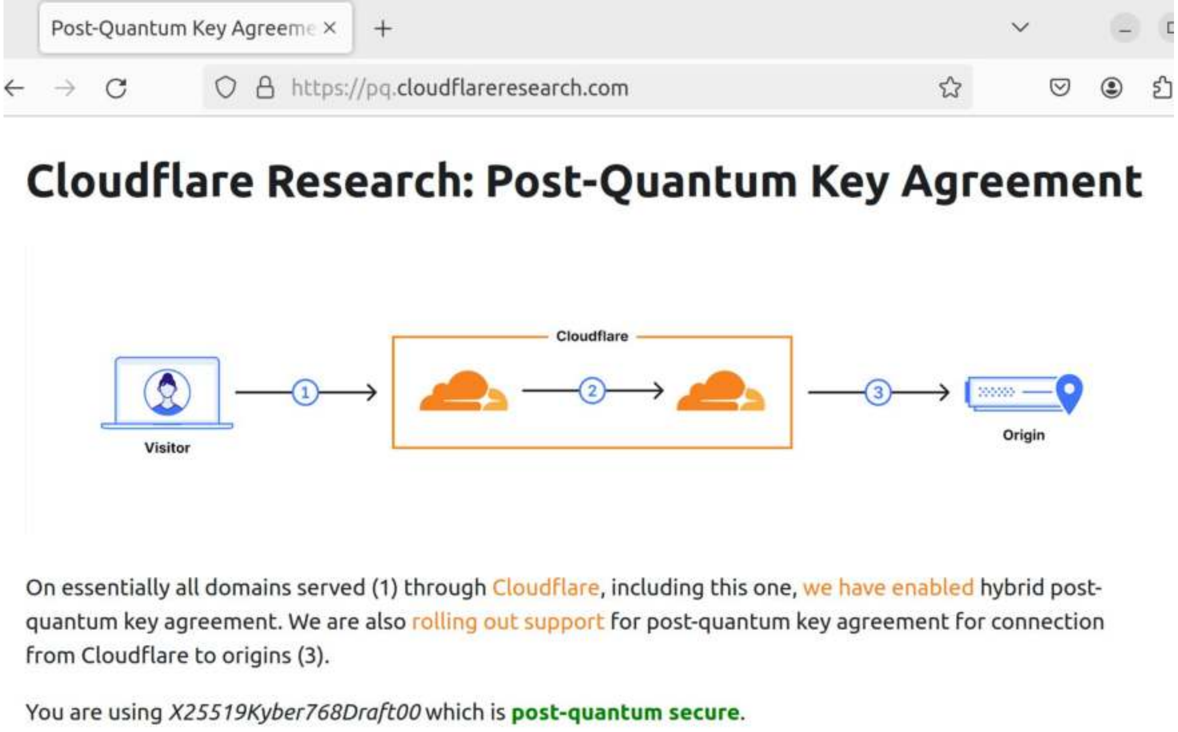
security.tls.enable_kyber true

...e enabled hybrid post- ...eement for connection from Cloudflare to origins (3).

You are using *X25519* which is **not post-quantum secure**.

https://pq.cloudflareresearch.com/

Kyber Supported at Cloudflare



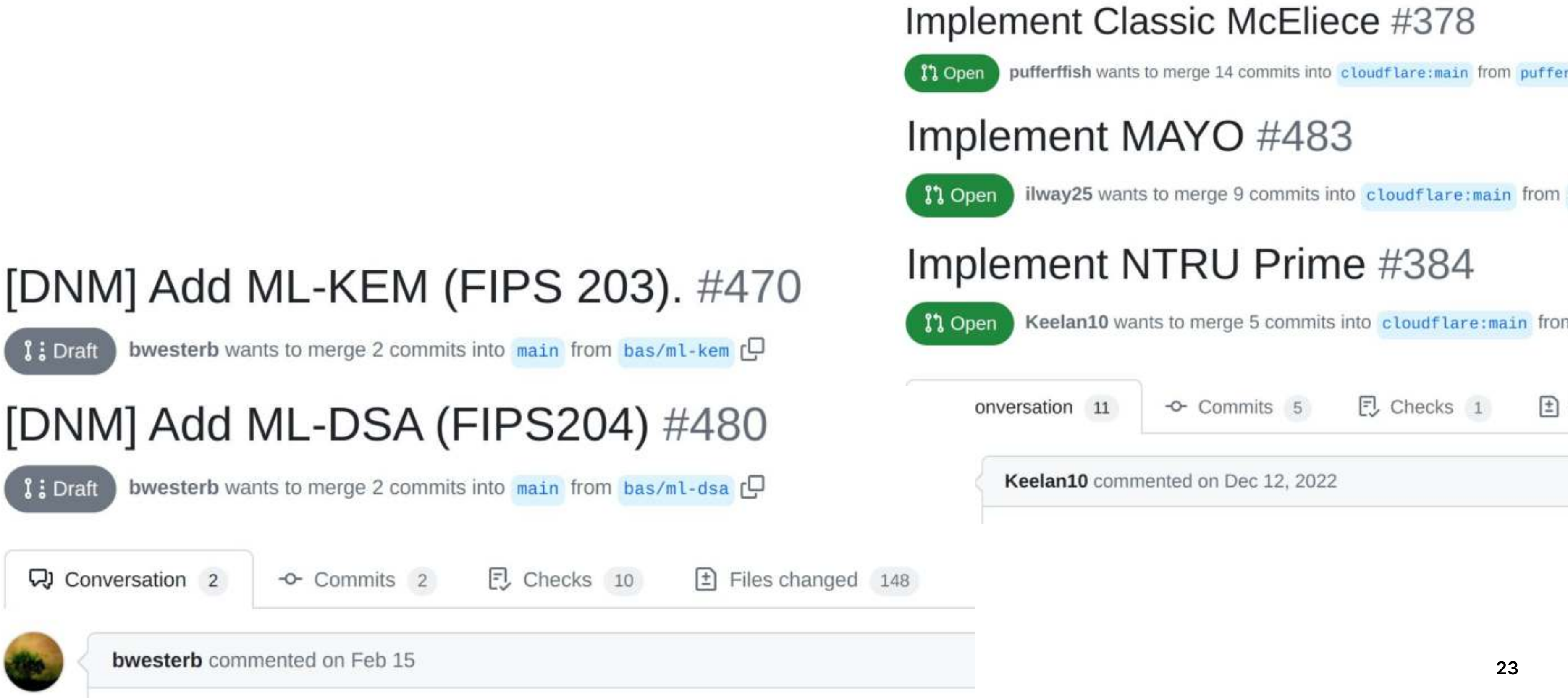
Cloudflare Research: Post-Quantum Key Agreement

On essentially all domains served (1) through Cloudflare, including this one, we have enabled hybrid post-quantum key agreement. We are also rolling out support for post-quantum key agreement for connection from Cloudflare to origins (3).

You are using *X25519Kyber768Draft00* which is **post-quantum secure**.

https://pq.cloudflareresearch.com/

Upcoming PQ Algorithms



Implement Classic McEliece #378

Open pufferfish wants to merge 14 commits into `cloudflare:main` from `puffer`

Implement MAYO #483

Open ilway25 wants to merge 9 commits into `cloudflare:main` from

Implement NTRU Prime #384

Open Keelan10 wants to merge 5 commits into `cloudflare:main` fron

Keelan10 commented on Dec 12, 2022

[DNM] Add ML-KEM (FIPS 203). #470

Draft bwesterb wants to merge 2 commits into `main` from `bas/ml-kem`

[DNM] Add ML-DSA (FIPS204) #480

Draft bwesterb wants to merge 2 commits into `main` from `bas/ml-dsa`

bwesterb commented on Feb 15

Protocols based on RSA

- Blind RSA Signatures
- Partially-Blind RSA Signatures
- Threshold 2-party RSA Signatures
- Safe prime generation

RFC 9474

RSA Blind Signatures, OCTOBER 2023

Symmetric-key Primitives

SHA3: Parallel F-1600 Permutation

XOF Interface:

- SHAKE
- Blake2X
- KangarooTwelve

Cipher:

- ASCON

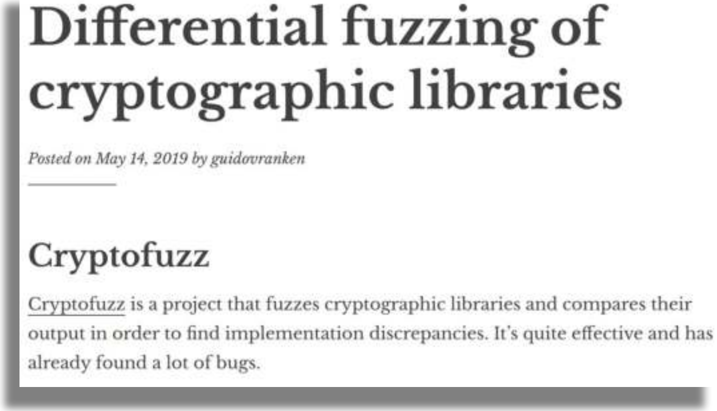
Bugs

- Complete formulas for elliptic curves
- Endianness (s390x is big-endian)
- Modular reduction issues
- Assembler: R15 register
- Hertzbleed Attack for SIDH/SIKE
  - Random blinding
- SIDH got broken :’(
- GoFetch: data memory prefetchers

Fuzz Testing

CryptoFuzz by G. Vranken

- Side-to-side comparisons
- Detected a modular reduction issue in P-384
- BLS12-381 pairing operations



Differential fuzzing of cryptographic libraries

Posted on May 14, 2019 by guidovranken

Cryptofuzz

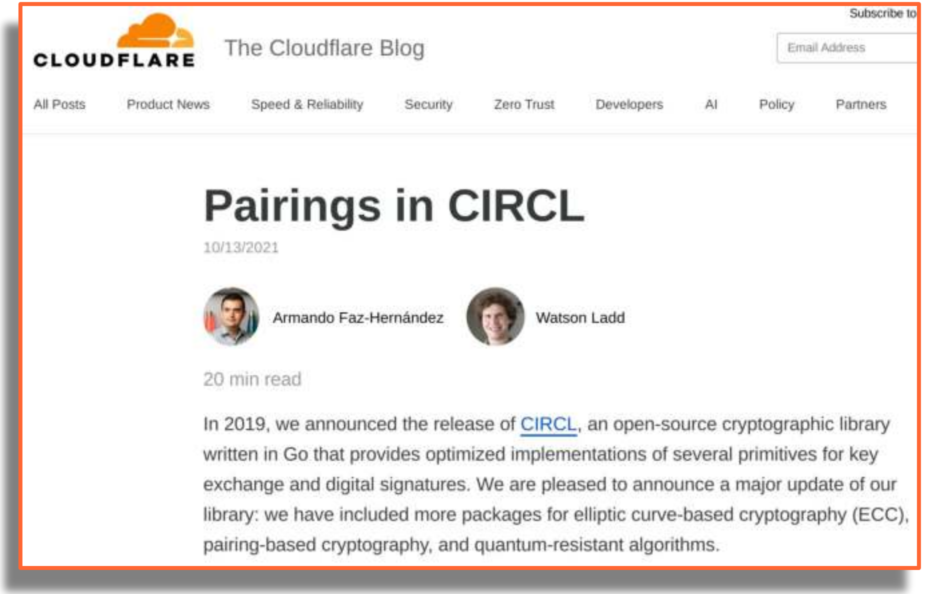
Cryptofuzz is a project that fuzzes cryptographic libraries and compares their output in order to find implementation discrepancies. It's quite effective and has already found a lot of bugs.

https://github.com/guidovranken/cryptofuzz/tree/master/modules/circl

Formal Methods

BLS12-381 pairing-friendly curve

- Uses **fiat-crypto** for prime field arithmetic
- Tower of fields optimized
- Efficient multi-pairing evaluation (W. Ladd)
- Subgroup check
- Hash to curve



Pairings in CIRCL

10/13/2021

Armando Faz-Hernández Watson Ladd

20 min read

In 2019, we announced the release of CIRCL, an open-source cryptographic library written in Go that provides optimized implementations of several primitives for key exchange and digital signatures. We are pleased to announce a major update of our library: we have included more packages for elliptic curve-based cryptography (ECC), pairing-based cryptography, and quantum-resistant algorithms.

Attribute-based Encryption (ABE)

Policy: country: US or region: EU

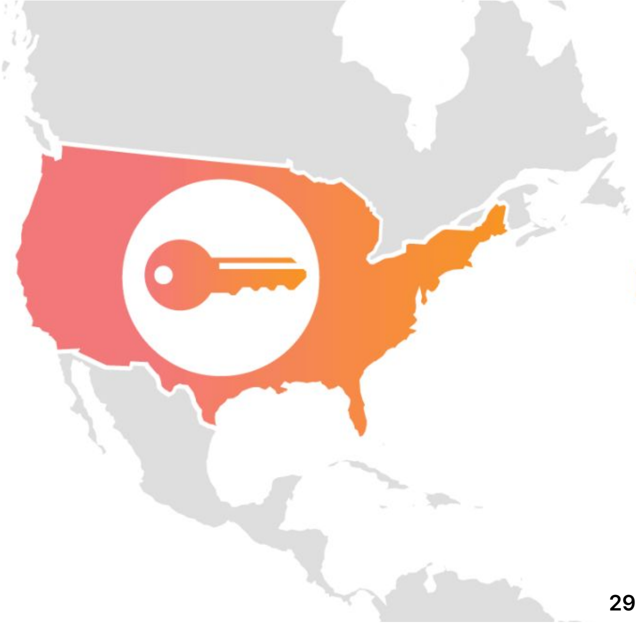
Ciphertext can be only decrypted in the United States or in the European Union

Encryption based takes a policy as input.

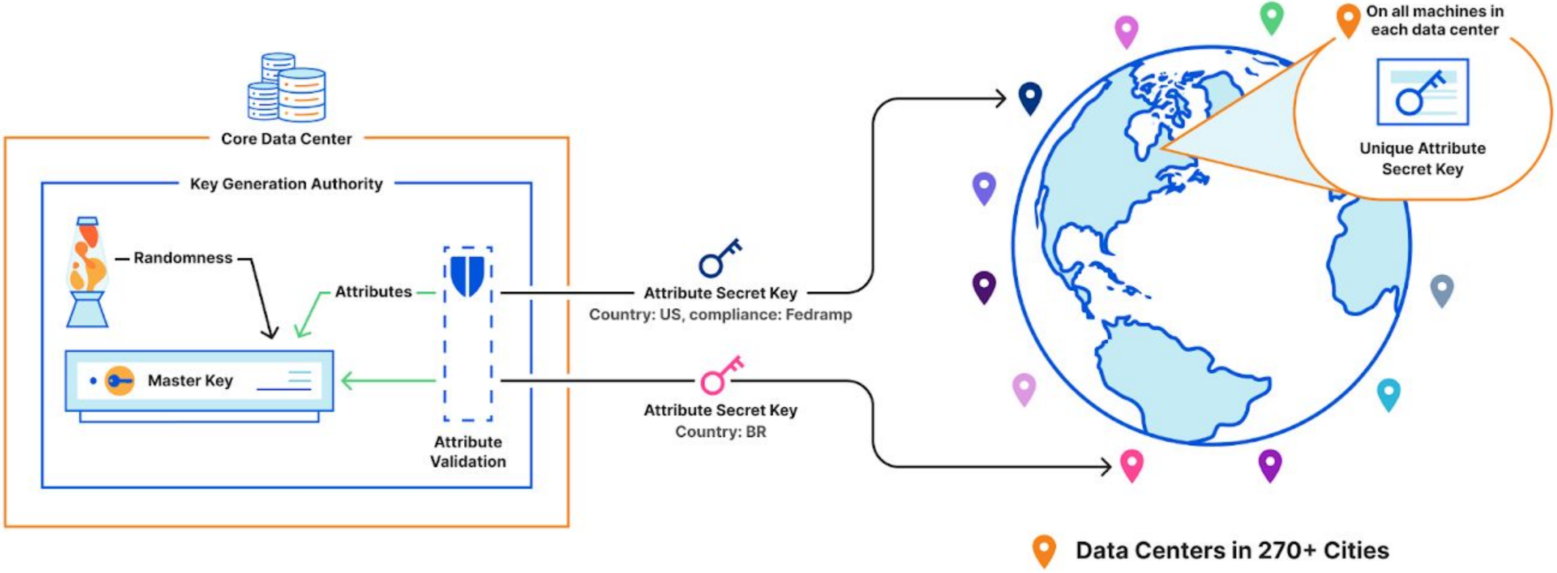
Decryptor has attributes, and can only decrypt if complies with the policy.

Scheme by Tomida, Kawahara and Nishimaki (TKN20)

- Ciphertext-Policy ABE
- Supports Negation of Attributes
- CCA-secure (BK transform)
- Domain-specific language for policies
- Code written by T. Verma, W. Ladd.



Portunus: Encryption of TLS Keys



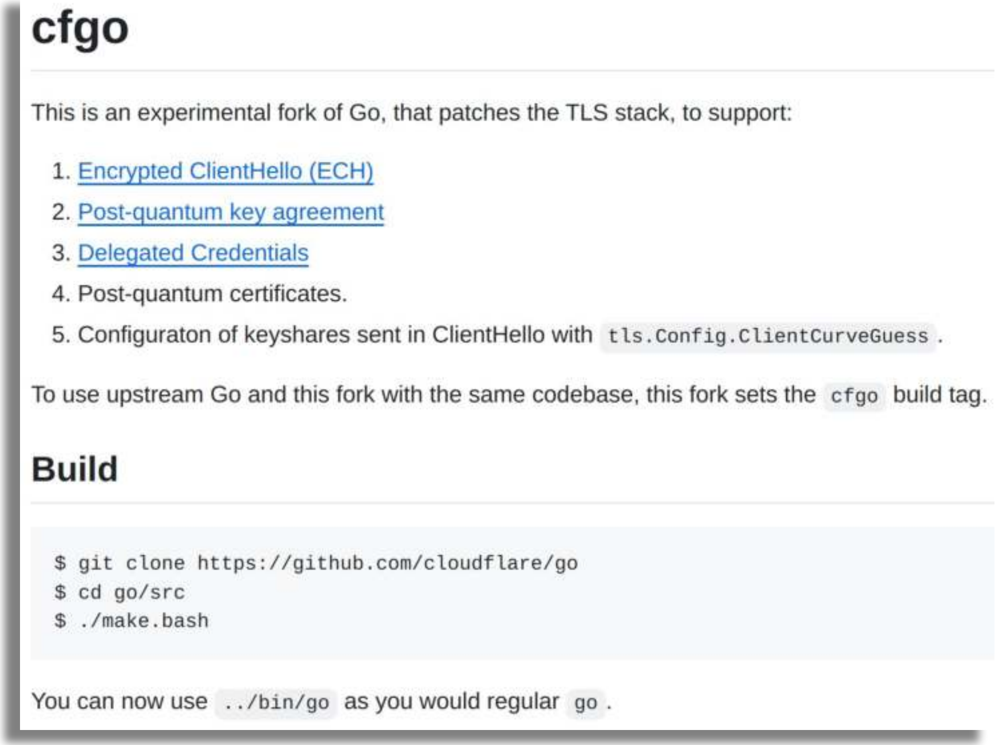
https://www.usenix.org/conference/atc23/presentation/ladd

https://blog.cloudflare.com/inside-geo-key-manager-v2/

Use CIRCL Natively

Experiment with Post-Quantum

https://github.com/cloudflare/go



cfgo

This is an experimental fork of Go, that patches the TLS stack, to support:

1. Encrypted ClientHello (ECH)
2. Post-quantum key agreement
3. Delegated Credentials
4. Post-quantum certificates.
5. Configuraton of keyshares sent in ClientHello with `tls.Config.ClientCurveGuess` .

To use upstream Go and this fork with the same codebase, this fork sets the `cfgo` build tag.

Build

```
$ git clone https://github.com/cloudflare/go
$ cd go/src
$ ./make.bash
```

You can now use `../bin/go` as you would regular `go` .

Takeaways

- Experimentation
  - Identify patterns
  - Few choices for ciphersuites
  - Iterate fast
- As code base increases, reviewing got more challenging
- Testing is a must
  - Enable static analysis, linters, code coverage, etc…
  - Test vectors - edge cases
- Formal methods & Verified implementations for Go
- Verified Assembly
  - Jasmin, Vale

Thanks

Contact us

ask-research@cloudflare.com

https://research.cloudflare.com/