Cryptographic primitives in Roc

Writing code as learning material

Matthieu Pizenberg – 22/06/2024

Last update: 22/06/2024

Agenda

Context: learning cryptography

Online course by Dan Boneh - Cryptography I

- Stream ciphers → ChaCha, Salsa20 🔗, ...
- Block ciphers → AES-128 🔗, Twofish, ...
- Message Authentication Codes (MAC) → HMAC, Poly1305 🔗, ...
- Hash functions → MD5, SHA-256, BLAKE-2 🔗, ...
- ... WIP

⚠ These Roc implementation are strictly for learning purposes, or for when there is no security risk.

What I liked

All in one file

```
app [main] { pf: platform "https://github.com/roc-lang/basic-cli/releases/down...ucY.tar.br" }

import pf.Stdout
import pf.Task

main = Stdout.line "Call instead: roc test 04-salsa20.roc"

# TESTS

# A word is an element of {0, 1, ..., 2³²-1}
expect 0xc0a8787e == 3232266366

# IMPLEMENTATION

## Wrapping bitwise left rotation for U32 words
wrapShiftLeftWordBy : U32, U8 -> U32
wrapShiftLeftWordBy = \word, n ->
    Num.bitwiseOr (Num.shiftLeftBy word n) (Num.shiftRightZfBy word (32 - n))
```

Roc syntax

```
addChecked : U256, U256 -> Result U256 [Overflow]
addChecked = \{ high: high1, low: low1 }, { high: high2, low: low2 } ->
    (newLow, carryLow) =
        when Num.addChecked low1 low2 is
            Ok newLow12 -> (newLow12, 0)
            Err Overflow -> (Num.addWrap low1 low2, 1)

    high12 <- Num.addChecked high1 high2 |> Result.try
    newHigh <- Num.addChecked high12 carryLow |> Result.map
    { high: newHigh, low: newLow }
```

What I struggled with

No builtin comparison in tests

```
## Salsa20 initialization with just k
expect
    k = List.range { start: At 1, end: At 16 }
    n = List.range { start: At 101, end: At 116 }
    output = [
        [39, 173, 46, 248, 30, 200, 82, 17, 48, 67, 254, 239, 37, 18, 13, 247],
        [241, 200, 61, 144, 10, 55, 50, 185, 6, 47, 246, 253, 143, 86, 187, 225],
        [134, 85, 110, 246, 161, 163, 43, 235, 231, 94, 171, 51, 145, 214, 112, 29],
        # [14, 232, 5, 16, 151, 140, 183, 141, 171, 9, 122, 181, 104, 182, 177, 193],
        [15, 232, 5, 16, 151, 140, 183, 141, 171, 9, 122, 181, 104, 182, 177, 193],
    ]
    salsa20k k n == output
```

```
❯ roc test 04-salsa20.roc
── EXPECT FAILED in 04-salsa20.roc ─────────────────────────────────────────────

This expectation failed:

196|>  ## Salsa20 initialization with just k
197|>  expect
198|>      k = List.range { start: At 1, end: At 16 }
199|>      n = List.range { start: At 101, end: At 116 }
200|>      output = [
201|>          [39, 173, 46, 248, 30, 200, 82, 17, 48, 67, 254, 239, 37, 18, 13, 247],
202|>          [241, 200, 61, 144, 10, 55, 50, 185, 6, 47, 246, 253, 143, 86, 187, 225],
203|>          [134, 85, 110, 246, 161, 163, 43, 235, 231, 94, 171, 51, 145, 214, 112, 29],
204|>          # [14, 232, 5, 16, 151, 140, 183, 141, 171, 9, 122, 181, 104, 182, 177, 193],
205|>          [15, 232, 5, 16, 151, 140, 183, 141, 171, 9, 122, 181, 104, 182, 177, 193],
206|>      ]
207|>      salsa20k k n == output

When it failed, these variables had these values:

k : List (Int Unsigned8)
k = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

n : List (Int Unsigned8)
n = [101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116]

output : List (List (Int Unsigned8))
output = [[39, 173, 46, ...], ..., [15, 232, 5, 16, 151, 140, 183, 141, 171, 9, 122, 181, 104, 182, 177, 193]]


1 failed and 27 passed in 715 ms.
```

No std U256

see GitHub file: U256.roc 🔗

Some UB

see https://github.com/roc-lang/roc/issues/6789

Some unimplemented operators

```
❯ roc repl

  The rockin' roc repl
————————————————————————

Enter an expression, or :help, or :q to quit.

» n = 0_u128

0 : U128
» Num.shiftLeftBy n 128
thread 'main' panicked at crates/compiler/gen_dev/src/generic64/mod.rs:4711:48:
not yet implemented
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

ChatGPT helpful?

see https://chatgpt.com/share/9f9176e7-6340-4c65-aec0-765feb9ca7f1

What's next?

- continue learning cryptography & writing Roc impls
- maybe make a package?