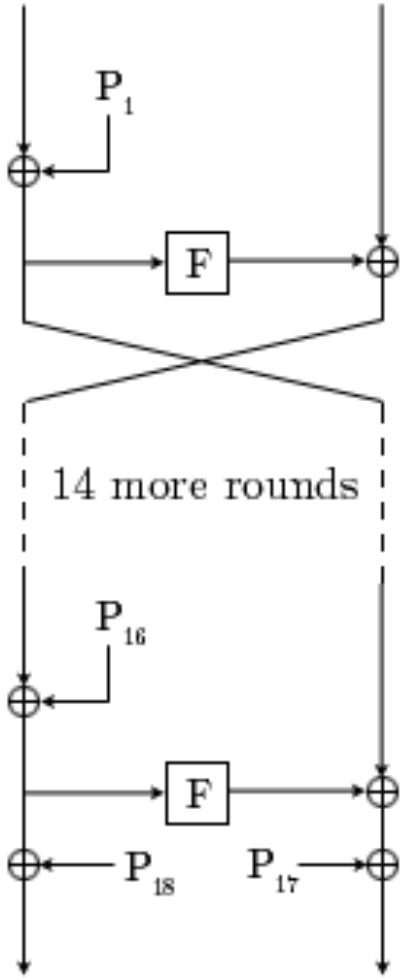
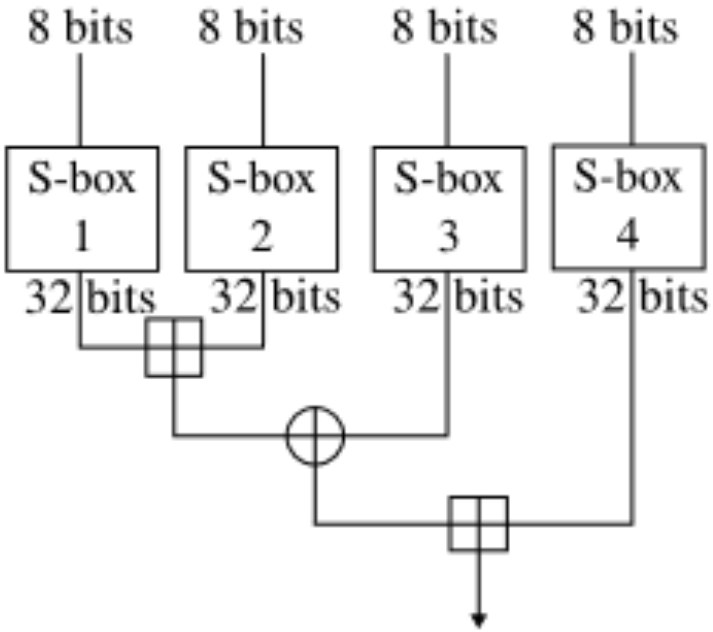
Blowfish

Wesley Wigham, Stephen Yingling, Chad Zawistowski

Blowfish


P1
F
14 more rounds
P16
F
P18
P17


8 bits
8 bits
8 bits
8 bits
S-box 1
S-box 2
S-box 3
S-box 4
32 bits
32 bits
32 bits
32 bits

Original Implementation

- Used a for-loop to iterate through the rounds
- The feistel function was not inlined, and was called extremely often
- Generating S-boxes was slow, but was only performed once
- Python’s pack and unpack methods were used for converting between integers and byte arrays

Timetrial: Original

Encrypting the all-0 bytestring 2.5 million times (3 trials)

- Average Running Time: 197.279 seconds
- Total times spent in a selection of functions:
  - feistel = 82.666 secs
  - encrypt_block = 58.479 secs
  - pack = 31.230 secs
  - encrypt = 6.572 secs
  - unpack = 3.892 secs
  - generate S box = 0.002 secs

Analysis

- Spends a long time in encrypt_block
  - Mostly in ‘feistel’
    - The ‘feistel’ function incurs quite a bit of overhead, but is where most processing takes place
  - Secondarily in ‘pack’
    - python library call, converts bytearray to integer
    - ‘feistel’ also calls ‘pack’

Optimized Implementation

- Did in-place calculations of 2 rounds at once to avoid left-right swapping overhead
- Unrolled the rounds' for-loop
- Inlined the feistel function to remove calling overhead
- Within 'encrypt_block', replaced calls to 'pack' with inline bit operations

Timetrial: Optimized

Encrypting the all-0 bytestring 2.5 million times (3 trials)

- Average Running Time: 68.125 seconds
- Total times spent in a selection of functions:
  - encrypt_block = 46.119 secs
  - encrypt = 6.561 secs
  - pack = 2.249 secs
  - generate S box = 0.012 secs
  - unpack = 0.001 secs

Results of Changes

- 2.94x faster overall
  - Reduced time in pack and unpack by 29 seconds and 3 seconds by switching to manual bitshift operations
  - Eliminated significant overhead by inlining the feistel function
  - Unrolled the for loop in encrypt_block and saved approximately 12 seconds

Optimization Lessons

- Python’s overhead for function calls becomes significant over thousands of calls
- Python’s struct library is fairly inefficient for combining bytearrays of known size into integers

Future Work

- Replace the remaining pack and unpack functions with our own conversions
- Unroll loops within the S box generation function