Lightweight Cryptography

Solomon Himelbloom
CS 665 (Spring 2023)

Index

1. Project Inspiration
2. Managing Complexity
3. Project Demo
4. Basic Analysis
5. Next Steps

Project Inspiration

Ascon

1. Initialization
2. Associated Data
3. Plaintext
4. Finalization

Research Questions

1. *When does it make sense to utilize LWC?*

2. *How can we benchmark & visualize respective results?*

Key Terms

- Combinations vs. permutations
- Cryptographic nonce
- Ephemeral secrets
- Round count

Managing Complexity

	Language Choice	Hardware Decisions	System Constraints
1	JavaScript Framework	≥ Web Server	Networking
2	Ruby & Python	≈ Raspberry Pi	Device Storage
3	C / C++	≤ Arduino	Memory + Message Size

Hash-based Message Authentication Codes

- Hash function
- Block vs. stream cipher
- Known or shared key
  - Message integrity
  - Authentication

- [Securing Stream Ciphers (HMAC) - Computerphile]
  - Dr Mike Pound

⚠️

- Checksum similarity
  - [message | hashed {m}]
- Length extension attack
  - message | hashed {key/m}

```ruby
# Solomon Himelbloom
# CS 665 (Final Project)
# 2023-04-07
#
# Simple hashing function with demo string.

require 'digest'
require 'openssl'

class LWC
  def initialize
    @hash = Digest::SHA256.new
  end

  def hash_string(plaintext)
    @hash.hexdigest(plaintext)
  end

  def generation
    key = 'key'
    data = 'The quick brown fox jumped over the lazy dog.'
    hmac = OpenSSL::HMAC.hexdigest(OpenSSL::Digest.new('sha256'), key, data)
    puts hmac
  end
end

input = "Lightweight Cryptography"

lwc = LWC.new
puts lwc.hash_string(input)
lwc.generation
```

(e.g. Ruby)

Demo

Static Site Generation

lightweight-cryptography.pages.dev

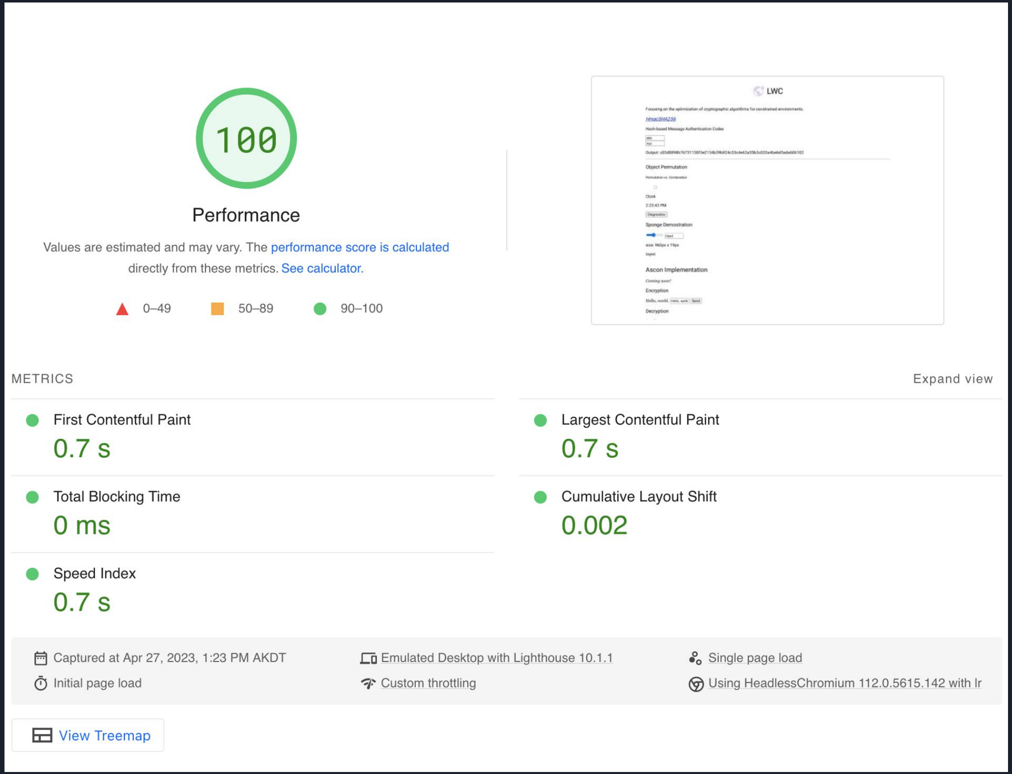
- SvelteKit & Cloudflare Pages
  - GitHub Repository
- Security considerations
  - User input (sanitized?)
  - Status codes + redirects
  - External packages
- Benefits
  - Routing for pages

Basic Analysis

Initial System Benchmarks

- **Arduino Uno Rev3** (SHA-256 Baseline / Rhys Weatherley's LWC Primitives)
  - Hashing: 167.01µs per byte, 5987.68 bytes per second
  - Finalizing: 10721.19µs per op, 93.27 ops per second
  - HMAC Reset: 10722.48µs per op, 93.26 ops per second
  - HMAC Finalize: 32196.03µs per op, 31.06 ops per second
- **Ruby** (Initial Hash + HMAC)
  - real 0m0.142s | user 0m0.101s | sys 0m0.040s
- **Web Version**
  - Svelte + Cloudflare Pages (w/ variable page load times)
  - Jupyter Notebook (pyascon → pictured right)

```
[ ] %time
    demo_aead('Ascon-128')

    CPU times: user 3 µs, sys: 0 ns, total: 3 µs
    Wall time: 7.15 µs
    === demo encryption using Ascon-128 ===
    key:        0xd27a1d2c71997eb418aa2d42d1108ddb (16 bytes)
    nonce:      0xa3accc404eae1c05248c1923abf2c006 (16 bytes)
    plaintext:  0x6173636f6e (5 bytes)
    ass.data:   0x4153434f4e (5 bytes)
    ciphertext: 0x9ac7c12a76 (5 bytes)
    tag:        0x4dc4d2861ca509f743c156b7df3abc4d (16 bytes)
    received:   0x6173636f6e (5 bytes)

[ ] %time
    demo_hash("Ascon-Hash")

    CPU times: user 4 µs, sys: 0 ns, total: 4 µs
    Wall time: 7.15 µs
    === demo hash using Ascon-Hash ===
    message: 0x6173636f6e (5 bytes)
    tag:     0x02c895cb92d79f195ed9e3e2af89ae307059104aaa819b9a987a76cf7cf51e6e (32 bytes)

[ ] %time
    demo_mac("Ascon-Mac")

    CPU times: user 3 µs, sys: 0 ns, total: 3 µs
    Wall time: 6.91 µs
    === demo MAC using Ascon-Mac ===
    key:     0xae9dd032adc49898cb50178529ee482c (16 bytes)
    message: 0x6173636f6e (5 bytes)
    tag:     0x4a264d9c4f3f8b5b5909f35c56e9142c (16 bytes)
```




```
real	0m0.002s
user	0m0.002s
sys	0m0.000s
[ec2-user@ip-172-31-15-70 netrun]$ time ./a.out
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528

real	0m0.002s
user	0m0.002s
sys	0m0.000s
[ec2-user@ip-172-31-15-70 netrun]$ time ./a.out
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528

real	0m0.002s
user	0m0.002s
sys	0m0.000s
[ec2-user@ip-172-31-15-70 netrun]$ time ./a.out
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528

real	0m0.002s
user	0m0.002s
sys	0m0.000s
[ec2-user@ip-172-31-15-70 netrun]$ time ./a.out
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528

real	0m0.002s
user	0m0.002s
sys	0m0.000s
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.394s
user	0m0.362s
sys	0m0.032s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.389s
user	0m0.320s
sys	0m0.068s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.396s
user	0m0.336s
sys	0m0.059s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.399s
user	0m0.342s
sys	0m0.056s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.390s
user	0m0.365s
sys	0m0.025s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.392s
user	0m0.326s
sys	0m0.066s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
[ec2-user@ip-172-31-15-70 netrun]$ time g++ SHA256.cpp; ./a.out

real	0m0.391s
user	0m0.337s
sys	0m0.054s
SHA256('Lightweight Cryptography'): f7efb0761ca6f93f7c88c53a063eafbe021b5c2f4e86f4075159957455806528
```

📊 Data Collection Methodology

Next Steps
+ Lessons Learned

Retrospective

- Continue improving ease-of-use for common web tools
  - Systematic approach for individual tests
- Improved visualizations stemming from benchmarking tasks
  - CPU, memory utilization, & temperature
- Expanding towards smaller devices
  - Industrial control systems → IoT → sensor nodes → RFID
- Standing on the shoulders of…
  - Layers of abstraction
  - Historical examples
  - Giants!
- Start small, finish big

Works Cited

1. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. 2021. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. J. Cryptol. 34, 3 (Jul 2021). https://ascon.iaik.tugraz.at/index.html
2. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. 2011. Cryptographic sponge functions. https://keccak.team/files/CSF-0.1.pdf
3. M. Rahman, S. Karnik, and S. Sarangerel. Lightweight Cryptography. Spring 2022 MIT 6.857: Computer and Network Security, Submitted Final Paper.
4. Lightweight Cryptographic Algorithms. Communications Security (ComSec) Lab, 2019-11-25. https://uwaterloo.ca/communications-security-lab/lwc.
5. R. Mishra, S. Dutta, M. Okade, and K. Mahapatra, "Substitution Permutation Network based Lightweight Ciphers with Improved Substitution Layers for Secure IoT Applications," 2021 2nd International Conference on Range Technology (ICORT), Chandipur, Balasore, India, 2021, pp. 1-6, doi: 10.1109/ICORT52730.2021.9581374.

📆 Upcoming:
⇒ Industry Events
⇒ NIST LWC News

Semester Security Paper Findings

2023.04.28 – *The Ascon Family for Lightweight Cryptography* (talk)

Summer '23 – Sixth Lightweight Cryptography Workshop (virtual)

Questions?