Experimental Analysis of Cryptography Overhead for Secure CubeSat Telemetry

Tyler Field, Computer Systems Engineering
Mentor: Michael Goryll, Associate Professor
School of Electrical, Computer, Energy Engineering

About

This research explores which encryption algorithms are best for cheap microcontrollers to help maximize the use of limited resources.

Abstract

Encryption consumes critical and limited resources, such as computing power, energy, and memory. Results from this research has helped SquidSat, a student-led 3u CubeSat launching sometime in 2027-2028, by providing data showing which microcontrollers are most effective for encryption algorithms.

Algorithm Sizes (bytes-in/out)

- TinyAES-16/16
- TEA-16/8
- RC5-16/8
- Twofish-16/16
- SIMECK-16/8
- HIGHT-16/8
- SEED-16/16
- LEA-16/16

Pico/2

- SIMON-16/16
- SPECK-16/16

UNO/STM32

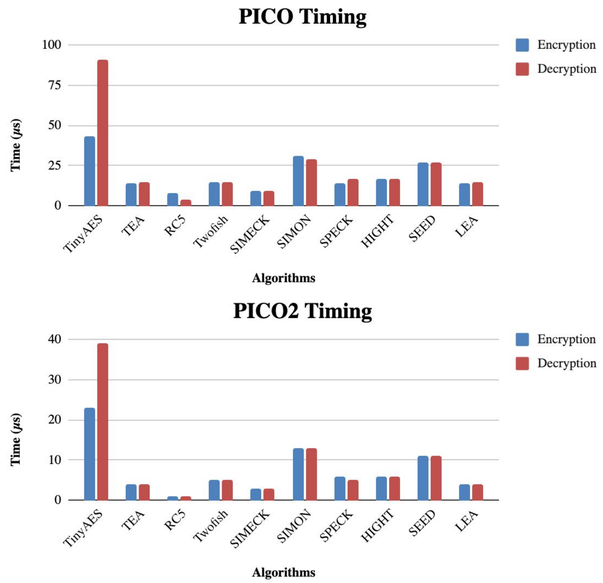
- SIMON-16/8
- SPECK-16/8

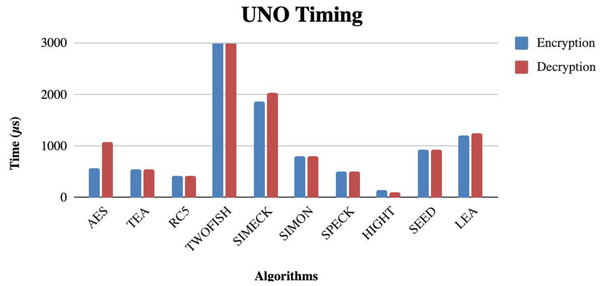
Future Work

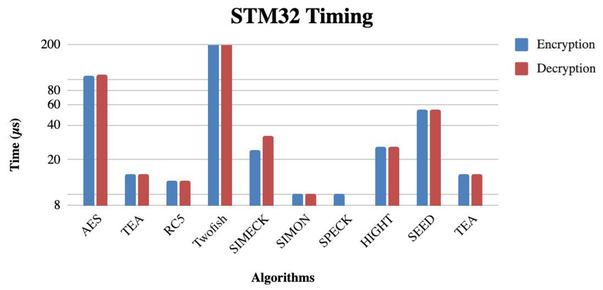
Since only the runtime per operation is measured, the pico/2 is expected to have the most operations per watt, then the STM32, and lastly the Uno. This hypothesis can be validated by measuring the operateration per watt.

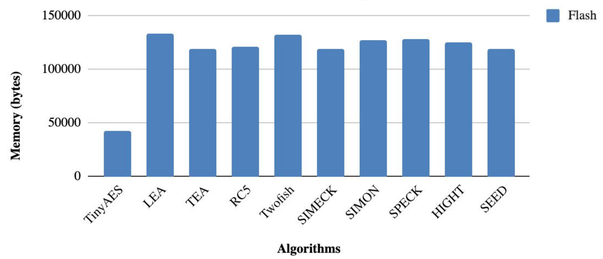
References

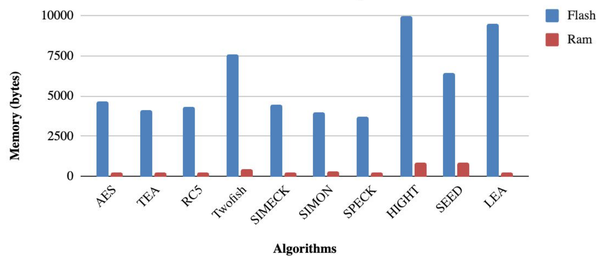
1. El-hajj, M., Mousawi, H., & Fadlallah, A. (2023). Analysis of lightweight cryptographic algorithms on IOT hardware platform. Future Internet, 15(2), 54. https://doi.org/10.3390/fi15020054

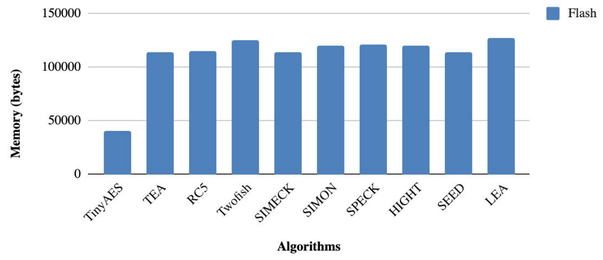


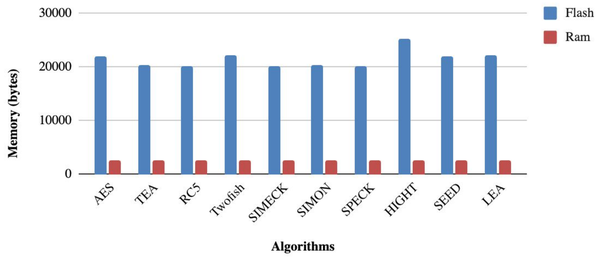












Timing Methodology (refer to image 1)

1. Each algorithm ran 100 trials. Within each trial, 100 operations of the algorithm were performed.
2. Before each trial, a time stamp is marked for when it began and when it ended. The difference between the two time points are added to a cumulative variable.
3. The cumulative variable is divided by the number of trials * the number of operations. This results in the average time taken for each operation.

```
//----------------------------ENCRYPTION BELOW-------------------------------
uint8_t enc_buffer[16];
for (size_t i = 0; i < block_bytes; ++i) {
    enc_buffer[i] = in_block[i];
}
for (int i = 0; i < num_of_trials; i++) {
    start = get_absolute_time();
    for (int j = 0; j < ops_per_trial; ++j) {
        AES_ECB_encrypt(&ctx, enc_buffer);
    }
    trial = absolute_time_diff_us(start, get_absolute_time());
    enc_time_trials += trial;
}
int64_t enc_avg_time = enc_time_trials / (num_of_trials * ops_per_trial);
double enc_avg_time_ns = (enc_time_trials * 1000.0) / (num_of_trials * ops_per_trial);
```

Image 1

Memory Methodology

1. The experiment files were built without the algorithms to measure how much storage is taken (x = without algo).
2. After which, a algorithm at a time is added in the build and then the total memory used is recorded (y = with algo).
3. The result is calculated by subtracting x from y (y-x).

FURI

Ira A. Fulton Schools of Engineering
Arizona State University