**Lý do chọn thuật toán Ascon**

Trong hệ thống IoT, vấn đề bảo mật dữ liệu truyền dẫn giữa các thiết bị là yêu cầu thiết yếu, đặc biệt trong bối cảnh các thiết bị IoT thường có tài nguyên phần cứng hạn chế như vi xử lý công suất thấp, bộ nhớ nhỏ và thời lượng pin có hạn. Do đó, việc lựa chọn thuật toán mã hóa phù hợp đóng vai trò rất quan trọng nhằm cân bằng giữa hiệu năng xử lý, mức độ bảo mật và khả năng tiêu thụ tài nguyên phần cứng.

ASCON-128a là một thuật toán mã hóa thuộc nhóm lightweight cryptography, đã được Viện Tiêu chuẩn và Công nghệ Hoa Kỳ (NIST) chính thức lựa chọn năm 2023 làm chuẩn bảo mật nhẹ (Lightweight Cryptography Standard) cho các thiết bị IoT. Thuật toán này được thiết kế nhằm đáp ứng các yêu cầu khắt khe sau:

Bảo mật mạnh mẽ: ASCON-128a hỗ trợ cơ chế mã hóa xác thực (Authenticated Encryption with Associated Data - AEAD), đảm bảo đồng thời tính bảo mật, tính toàn vẹn và xác thực dữ liệu. Đây là yếu tố rất quan trọng trong môi trường IoT, nơi dữ liệu có thể bị nghe lén hoặc giả mạo trong quá trình truyền tải.

Hiệu quả trên thiết bị hạn chế tài nguyên: Thuật toán được tối ưu hóa để hoạt động tốt trên các bộ vi xử lý nhỏ như ESP32, ARM Cortex-M, giúp tiết kiệm tài nguyên bộ nhớ RAM, ROM và giảm tải xử lý, từ đó tiết kiệm điện năng và kéo dài tuổi thọ pin cho thiết bị.

Chống tấn công vật lý: ASCON-128a được thiết kế có khả năng kháng lại các cuộc tấn công kênh kề, điều này rất quan trọng khi triển khai trong các thiết bị thực tế, dễ bị lộ thông tin vật lý như điện áp, dòng điện hoặc thời gian xử lý.

Dễ dàng triển khai: Với cấu trúc đơn giản, kích thước mã nguồn nhỏ, ASCON-128a dễ dàng được tích hợp vào các hệ thống IoT nhúng mà không yêu cầu phần cứng mã hóa chuyên dụng.

So với các thuật toán mã hóa phổ biến như AES, ASCON-128a mang lại hiệu năng cao hơn trên nền tảng vi điều khiển nhỏ, trong khi vẫn đảm bảo mức độ bảo mật tương đương hoặc vượt trội. Điều này giúp hệ thống IoT vừa bảo vệ được dữ liệu, vừa đáp ứng được yêu cầu tiết kiệm năng lượng, chi phí phần cứng và tối ưu hoạt động lâu dài trong thực tế.

Chính nhờ những ưu điểm trên, ASCON-128a được lựa chọn làm thuật toán bảo mật trong hệ thống IoT của đề tài này, nhằm đảm bảo an toàn dữ liệu truyền dẫn giữa các thiết bị cảm biến và gateway một cách hiệu quả và bền vững.

**Lý do lựa chọn Raspberry Pi làm Gateway, ESP32 làm Node:**

Trong hệ thống IoT, các thiết bị thường được phân thành 2 loại chính:

* IoT Node (Cảm biến / Thu thập dữ liệu): Nơi gắn cảm biến, thu thập dữ liệu thực địa.
* IoT Gateway (Cổng trung tâm): Nơi thu nhận dữ liệu từ nhiều node, xử lý, mã hóa/phân tích sâu hơn và truyền dữ liệu lên server.

Việc lựa chọn ESP32 tại Node và Raspberry Pi tại Gateway dựa trên đặc điểm kỹ thuật của từng vi điều khiển

ESP32 đặt tại Node: Node thường bố trí phân tán ngoài hiện trường, có nguồn điện hạn chế (chạy pin hoặc năng lượng mặt trời), nhiệm vụ chính là đọc cảm biến, mã hóa dữ liệu cơ bản rồi gửi về Gateway qua mạng LoRa. ESP32 rất phù hợp vì kích thước nhỏ, tiêu thụ điện năng thấp, hỗ trợ nhiều giao thức cảm biến, dễ lập trình và chi phí thấp.

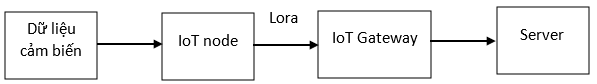
Raspberry Pi đặt tại Gateway: Gateway đóng vai trò trung tâm, có nguồn điện ổn định, cần khả năng xử lý mạnh để:

* Giải mã dữ liệu từ các node gửi về.
* Kiểm tra tính toàn vẹn, xác thực dữ liệu.
* Thực hiện các bước mã hóa tiếp theo với khóa khác.
* Kết nối Internet, giao tiếp với Web Server qua mạng IP (WiFi/Ethernet).
* Raspberry Pi phù hợp với các tác vụ xử lý nặng, mã hóa nhiều lớp (ASCON nhiều lượt, có thể cả TLS/HTTPS khi gửi lên server).

**Thiết kế mô hình phần cứng thiết bị IoT**

Mục tiêu của hệ thống là xây dựng một mô hình truyền nhận dữ liệu cảm biến an toàn sử dụng thuật toán mã hóa nhẹ ASCON. Thiết bị ESP32 thực hiện việc thu thập dữ liệu cảm biến, mã hóa và truyền dữ liệu đến Raspberry Pi thông qua giao tiếp LoRa. Raspberry Pi sau đó tiến hành giải mã, mã hóa lại dữ liệu và gửi lên server web phục vụ mục đích giám sát từ xa.

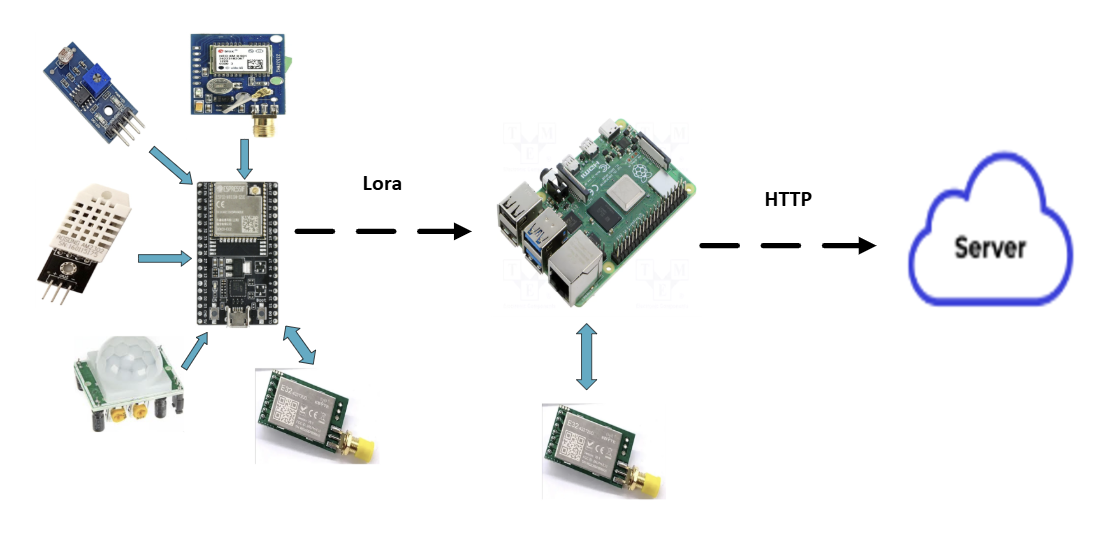
Mô hình phần cứng thiết bị IoT đươc đề xuất đề xuất bao gồm các phần chính là: KIT “IoT Gateway”, KIT “IoT Node”. Các thành phần của hệ thống có thể kết nối với máy tính, điện thoại di động, thực hiện dịch vụ điện toán đám mây hoặc điều khiển thiết bị bên ngoài. Mô hình được đề xuất như dưới đây.



Khối dữ liệu cảm biến: thu thập dữ liệu môi trường thông qua các cảm biến để đưa tới khối IoT node

Khối IoT node: có chức năng nhận thông tin từ khối cảm biến, thực hiện mã hóa dữ liệu bằng thuật toán Ascon-128a, sử dụng truyền thông Lora để gửi dữ về trung tâm.

Khối IoT Gateway: là node trung tâm có chức năng phân biệt dữ liệu từ các node cảm biến, phân tích xử lý dữ liệu đồng thời thực hiện giải mã dữ liệu nhận được từ IoT node bằng thuật toán Ascon-128a. Khối này sử dụng giao tiếp truyền thông Lora để kết nối với khối IoT node. Sau khi thực hiện giải mã dữ liệu khối IoT Gateway thực hiện đẩy dữ liệu lên server, dữ liệu trước khi đẩy lên server được mã hóa tiếp bằng thuật toán Ascon-128a để tăng khả năng bảo mật.

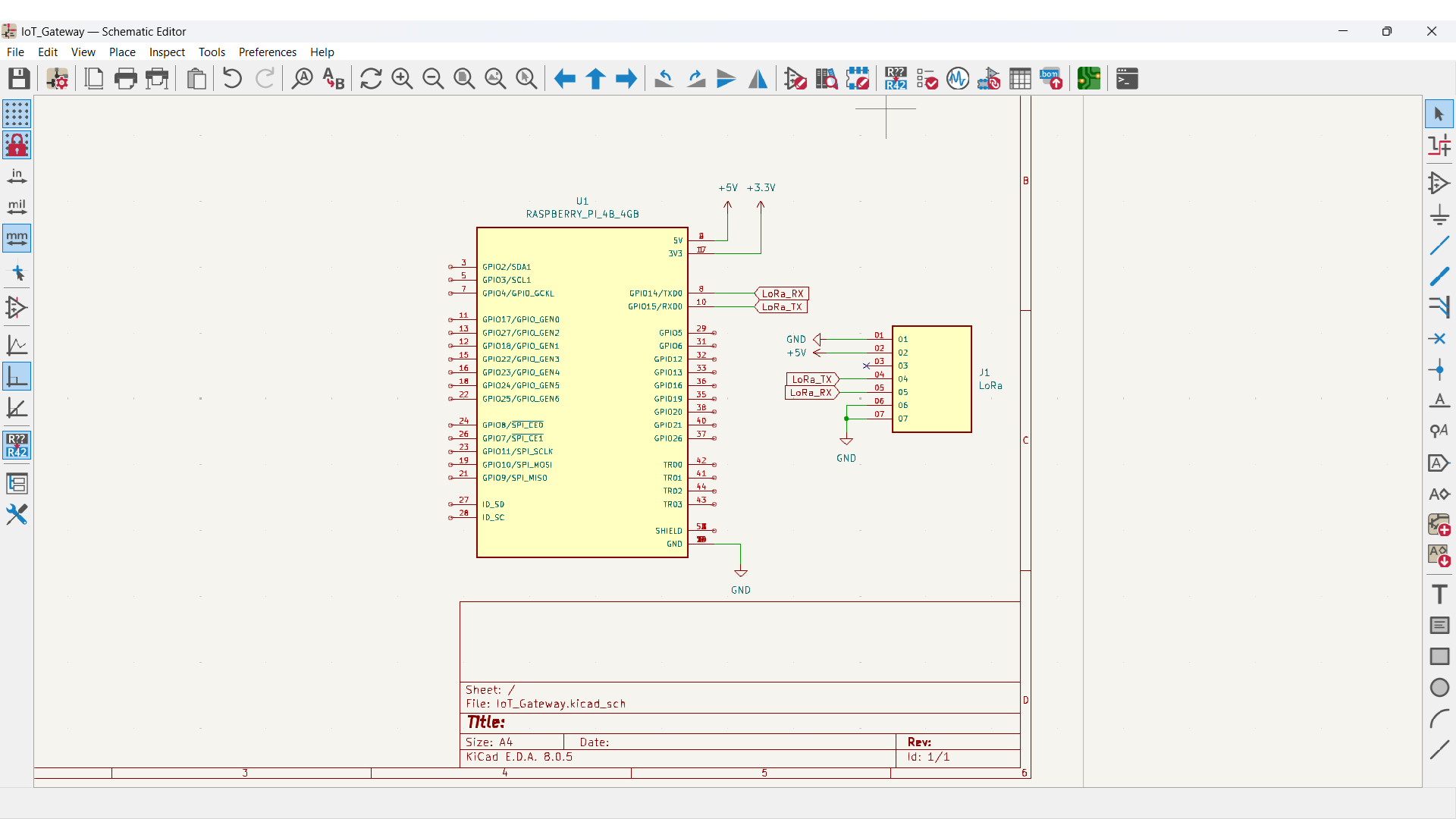


Hình 1. 1: Mô hình thử nghiệm ASCON cho bảo mật truyền thông IoT

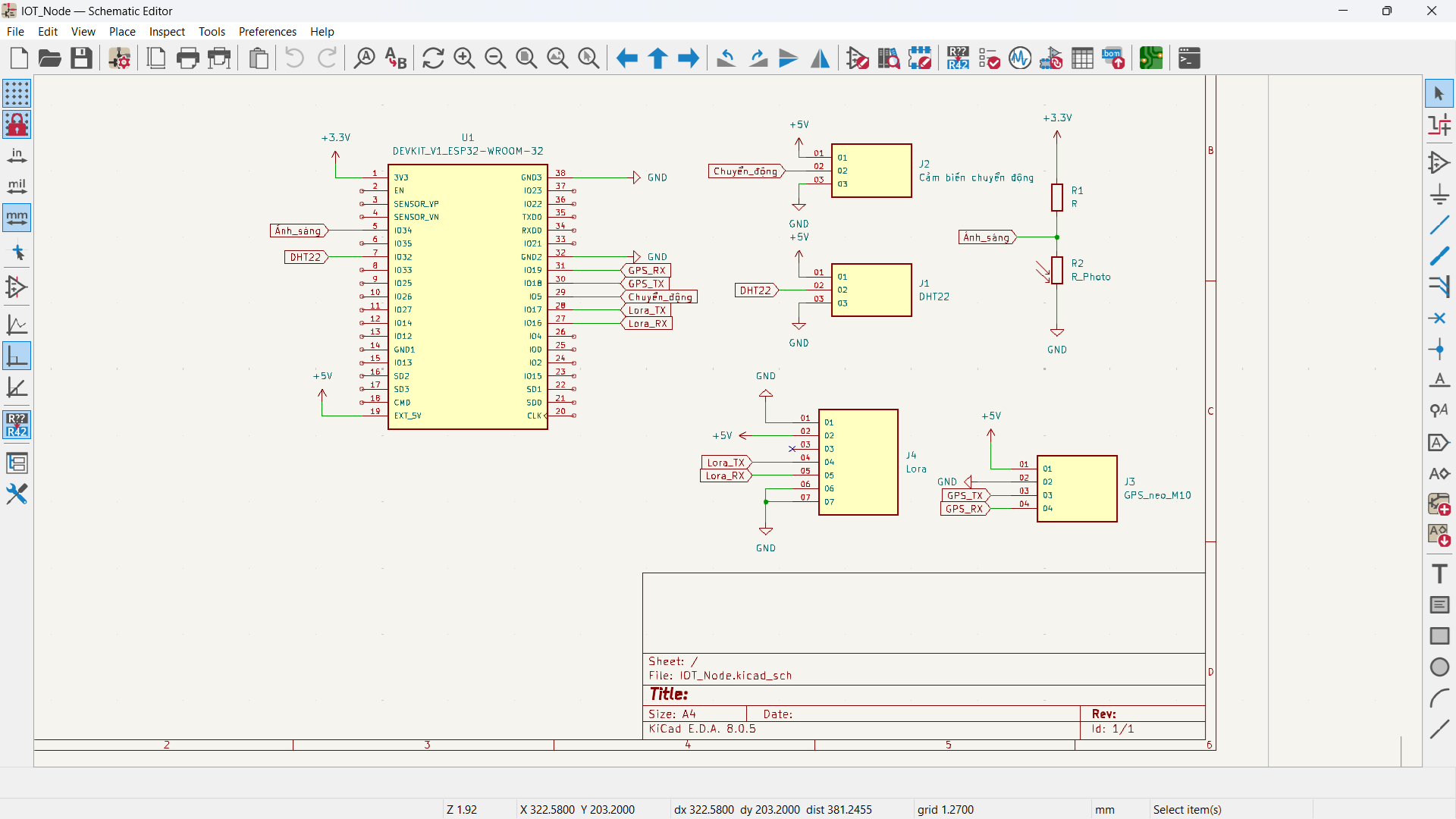
Thành phần phần cứng sử dụng trong mô hình:

|  |  |  |
| --- | --- | --- |
| **Thiết bị** | **Vai trò** | **Mô tả** |
| ESP32 | Vi điều khiển nhúng | Đọc dữ liệu cảm biến, mã hóa, truyền Lora |
| Raspberry pi | Thiết bị trung tâm | Nhận dữ liệu, giải mã, mã hóa, gửi dữ liệu |
| Lora E32 | Giao tiếp không dây | Truyền dữ liệu tầm xa giữa ESP và Rpi |
| DHT22 | Đo nhiệt độ, độ ẩm | Giao tiếp qua one-wire |
| Cảm biến chuyển động | Phát hiện chuyển động | Giao tiếp qua one-wire |
| Cảm biến ánh sáng | Đo cường độ ánh sáng | Giao tiếp qua one-wire |
| Module GPS | Xác định vị trí địa lý: kinh độ, vĩ độ | Giao tiếp với vi điều khiển thông qua UART |
| Server Web | Lưu trữ và hiển thị | Nhận dữ liệu qua API |

Bảng 1. 1: Phần cứng sử dụng trong mô hình

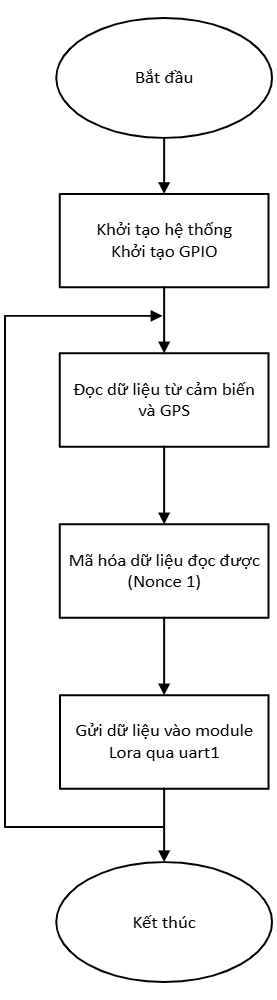


Hình 1. 2: Sơ đồ nguyên lý của khối IoT\_gateway



Hình 1. 3: Sơ đồ nguyên lý của khối IoT\_node

**Thành phần phần mềm:**



Hình 1. 4: Lưu đồ thuật toán của khối IoT node

**Trên khối IoT node:**

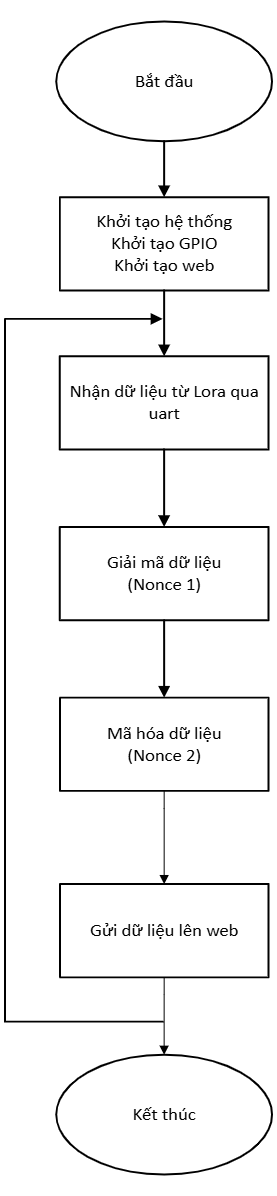
ESP32 thu thập dữ liệu của môi trường từ cảm biến và GPS. Sau đó được chuẩn hóa sang định dạng chuỗi.

Mỗi thông số sẽ được biểu diễn bằng 1 byte (chuyển động, độ ẩm, nhiệt độ, ánh sáng, kinh độ: độ, phút, giây, vĩ độ: độ, phút, giây, tọa độ E và N).

Dữ liệu sẽ được mã hóa bằng ASCON-128a (AEAD), sử dụng khóa đối xứng 128 bit (thống nhất trên 2 khối) và nonce 128 bit được sinh ngẫu nhiên.

Mã hóa dữ liệu theo chuẩn Ascon-128a, đóng gói và truyền qua lora bằng module Lora E32.

**Trên khối IoT gateway:**



Hình 1. 5: Lưu đồ thuật toán của khối IoT gateway

Dữ liệu mã hóa sẽ được nhận tại cổng uart thông qua giao tiếp với Lora

Raspberry Pi sử dụng cùng thuật toán để tiến hành giải mã dữ liệu. Khóa đã được thống nhất từ trước, nonce chính là 128 bit cuối cùng của chuỗi nhận được.

Sau khi giải mã thành công, sẽ tiến hành mã hóa lại với khóa và nonce khác để tăng lớp bảo mật trước khi gửi ra ngoài hệ thống nội bộ.

Dữ liệu được gửi lên web server để giám từ xa hoạt động của hệ thống.

**Kết quả:**

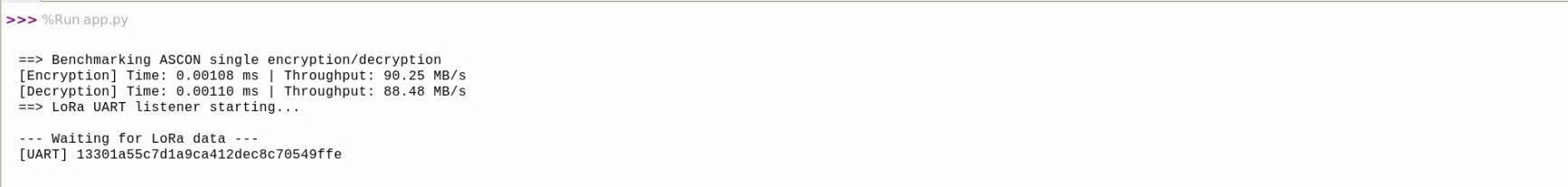
Thông lượng được tính theo công thức:

S: kích thước dữ liệu (MB)

T: thời gian xử lý (s)

****

Hình 1. 6: Kết quả đánh giá trên ESP



Hình 1. 7: Thông lượng mã hóa và giải mã trên raspberry pi

**Đánh giá kết quả thử nghiệm:**

**Kết quả đo tốc độ mã hóa trên ESP32 cho thấy:**

Thuật toán mã hóa nhẹ ASCON128 được triển khai và thử nghiệm trên nền tảng vi điều khiển ESP32 với dữ liệu plaintext có kích thước 12 byte. Các thông số được đo bao gồm: bộ nhớ sử dụng (RAM), thời gian mã hóa, tốc độ mã hóa (throughput) và độ trễ mã hóa tính trên mỗi byte dữ liệu.

Để đánh giá hiệu quả triển khai, kết quả thực nghiệm được so sánh với dữ liệu khảo sát từ bài báo "A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers" của jesus alejandro soto-cruz cùng các cộng sự. Các biểu đồ được sử dụng để đối chiếu gồm:

Hình 7: RAM sử dụng.

Hình 8: Độ trễ mã hóa/giải mã

Hình 11: Tốc độ mã hóa/giải mã

1. **Sử dụng bộ nhớ**

Kết quả RAM sử dụng thực nghiệm thu được khoảng…. trong khi dữ liệu đã được khảo sát trong bài báo khoảng 480B.

Sự chênh lệch đáng kể này chủ yếu đến từ phương pháp đo. Nguyên nhân có thể đến từ việc đo tổng bộ nhớ cấp phát tại thời điểm thực thi, bao gồm cả các vùng đệm phụ trợ như nonce, key, plaintext, ciphertext và tag. Ngoài ra, một phần bộ nhớ stack và vùng nhớ của thư viện tiêu chuẩn cũng góp phần vào con số này. Nếu chỉ xét riêng bộ đệm phục vụ thuật toán, footprint thực tế có thể thấp hơn. Do đó, kết quả đo vẫn nằm trong ngưỡng hợp lý và có thể chấp nhận được trong bối cảnh thử nghiệm trên ESP32.

1. **Độ trễ mã hóa**

Độ trễ mã hóa thực nghiệm của đề tài có giá trị 0.0047ms/B = 4.7µs/B so với kết quả 7.25 µs/B đã được các tác giả thực hiện trong bài báo trên cho thấy độ trễ thấp hơn, phản ánh hiệu quả xử lý khi làm việc với chuỗi trong thuật toán ASCON128. Ngoài ra, sai khác này cũng có thể đến từ sự khác biệt trong môi trường thử nghiệm (xung nhịp CPU, phiên bản thư viện, cách tối ưu bộ đệm)

**Kết quả đo tốc độ mã hóa và giải mã trên Raspberry Pi:**

* Tốc độ mã hóa:
* Tốc độ giải mã:
* Kích thước dữ liệu thử nghiệm: 12 byte

Đánh giá:

Trên Raspberry Pi, với CPU ARM tốc độ cao và hệ điều hành hỗ trợ đa tác vụ, thuật toán ASCON-128a đạt tốc độ xử lý rất nhanh cho cả hai quá trình mã hóa và giải mã.

Thời gian xử lý trung bình chỉ khoảng … micro giây với mỗi gói dữ liệu nhỏ.

Sự cân bằng giữa tốc độ mã hóa và giải mã (chênh lệch không đáng kể) cho thấy thuật toán có tính đối xứng và ổn định hiệu năng.

Tốc độ này giúp Gateway dễ dàng xử lý đồng thời dữ liệu từ nhiều node khác nhau mà không tạo ra độ trễ, đảm bảo hệ thống hoạt động gần như theo thời gian thực.

Tài liệu tham khảo

Lightweight Cryptography for IoT: A Review, M. R. Alenezi et al., IEEE Internet of Things Journal, 2020.