

Mavlink에 SIKE 알고리즘을 적용하여 PQC공격에 대비한

보안 강화

2017103951 고성훈

요약

최근 여러 산업에서 드론이 사용되면서 보안 문제가 떠오르고 있다. 이 연구는 드론통신에서 보안을 강화하기 위해, 양자 암호학 기술인 Post-Quantum Cryptography(PQC)을 결합하여 드론 통신의 라이브러리 중 하나인 MAVLINK를 강화하는 방법을 제안한다. 이를 위해, PQC 알고리즘 중 하나인 SIKE 사용하여 효율적이고 안전한 공유키를 생성 후 libsodium을 통해 데이터를 암호화하고 복호화하여 안전한 통신을 구현하고 검증하고자 한다.

1. 서론

1.1 연구배경

드론 산업은 농업, 방재, 산림, 해양 등 다양한 분야에서 사용되고 있으며, 무인 항공기의 사용 분야는 지속적으로 확대되고 있다. 이에 따라 드론 기술은 기존의 방식보다 더욱 높은 수준의 안전성과 보안성이 요구된다.

드론은 자동차와 달리 공중에서 움직이는 차량으로, 사용자가 조종하지 않을 경우 다양한 위험 요소가 존재한다. 그리고 특히 군사 드론인 경우 전송하는 데이터가 타인에게 노출될 경우 불법적인 스파이나 공격 등의 위협을 받을 수 있다. 따라서 드론 통신 보안 문제는 드론의 사용자 및 외부인에게 큰 문제로 작용할 수 있다.

드론 통신 보안 문제를 해결하기 위한 기존의 암호화 기술은 대체로 전통적인 RSA, AES

등의 기술을 사용하였다. 하지만 이러한 기술들은 양자 공격에는 취약다는 점이 있다. 그래서 미래에는 안전성이 보장되지 않는 기술로 인식되고 있다.

이에 반해, 포스트-퀀텀 암호(PQC) 기술은 전통적인 암호화 기술의 한계를 극복하고, 향후 포스트-퀀텀 시대에 적용될 수 있는 안전한 암호화 기술로 주목받고 있다. 이러한 PQC 기술 중 Supersingular Isogeny Key Encapsulation(SIKE) 알고리즘은 안전성과 효율성 모두에서 우수한 성능을 보이고 있다.

SIKE를 이용하여 안전하고 보안성이 높은 공유키를 생성하고 이 공유키를 이용하여 주고받을 데이터를 암호화 및 복호화하여 통신함으로써 안전한 통신을 구현할 수 있다. 따라서 본 연구에서는 Mavlink 프로토콜에 SIKE 알고리즘을 추가하여 드론 통신 보안성을 높이는 방안을 제시하고자 한다.

1.2 연구목표

기존의 드론 통신 프로토콜인 Mavlink는 속도와 안정성 면에서 우수한 성능을 보이지만, 보안성 측면에서는 제한된 보호 기능만을 제공하고 있다.

이에 대한 보완책으로 포스트-퀀텀 암호(PQC) 기술을 이용한 알고리즘인 SIKE(Supersingular Isogeny KeyEncapsulation)를 Mavlink 프로토콜에 적용하여 드론 통신 보안성을 높이는 방안을 연구한다.

본 논문에서는 SIKE 알고리즘을 Mavlink 프로토콜에 적용하는 방법과 그 효과에 대해 제시하고, 기존의 통신 기술과 비교하여 그 우수성을 검증하였다. 이를 통해 드론 통신 보안성의 향상에 대한 필요성을 제시하고, SIKE 알고리즘이 보안성 강화에 대한 해결책으로 활용될 수 있다는 가능성을 제시한다.

2. 관련연구

2.1 양자내성암호

2.1.2 박찬희. "mbedTLS 상에서 격자기반 양자내성암호의 구현 및 성능평가."

국내석사학위논문 부산대학교 대학원, 2020. 부산

구분	특성
격자기반	- 구현의 용이함으로 다양한 응용환경 지원이 가능함
코드기반	- 고속의 암호화, 복호화가 가능함 - 키의 크기가 크다는 단점이 있음
해시기반	- Collision Resistance에 의한 안전성을 보장하고 있음 - 큰 서명 사이즈를 가지고 있음
아이소제니기반	- 키의 크기가 작으며, 구현의 용이성을 가짐 - 연산속도가 느리다는 단점이 있으나, 고속 구현 연구가 활발히 진행되고 있음
다변수기반	- 서명의 크기가 작다는 장점이 있으며, 이로 인해 빠른 계산속도를 가짐 - 키의 크기가 크다는 단점이 있음

[표1] 양자내성암호의 종류에 따른 특성

본 논문에서는 양자컴퓨터 환경에서도 안전한 양자내성암호의 mbedTLS 적용에 대한 구현 결과를 다루고 있다 상에 . TLS 양자내성암호를 적용하고자 하는 연구는 다수 진행되고 있으며 대부분의 , OpenSSL . 구현 결과물은 상에 적용 결과물을 제시하고 있다. mbedTLS Handshake Message Fragmentation의 경우 단계에서 기능을 제공하고 있지 않아 키의 크기가 큰 양자내성암호를 적용하기에는 무리가 있었으나 이 논문에서는 mbedTLS Handshake Message Fragmentation 상에 양자내성암호를 적용하기 위해 기능을 구현한다.

2.1.2 저성능 사물인터넷 상에서의 양자 내성 암호 구현

기존 공개키 암호의 보안성을 저하시킬 수 있는 양자 컴퓨터와 양자 알고리즘의 급속한 발전으로 인해 미국의 NIST에서 는 양자 내성 암호 표준화를 수행하고 있다. 전 세계에서 다양한 양자 내성 암호 표준안을 제시하였으며 현재 2 단계로 넘어가 차세대 양자 내성

암호를 다방면에서 평가하고 있다. 양자 내성 암호를 평가하는 다양한 항목 중에는 암호 보안 강도와 암호 구현 효율성이 있다. 자원적인 제한사항이 많은 저성능 사물인터넷 상에서의 양자 내성 암호 구현에 대해 확인해 보도록 하는 논문이다.

2.2 라이브러리

안전한 통신을 하기위한 라이브러리들에 대해 서술한다. Mavlink 라이브러리를 이용해 기본적인 데이터를 교환하며 SIKE 라이브러리를 통해 보안이 강력한 공유키를 생성한다. 마지막으로 libsodium 라이브러리를 통해 생성한 공유키를 이용하여 데이터를 암호화 및 복호화하여 안전한 통신을 할 수 있도록 한다.

2.2.1 MavLink

Mavlink는 무인 항공기, 로봇 및 차량과 같은 자율 주행 시스템의 통신 프로토콜 중 하나이다. 이는 경량화된 시리얼 통신 프로토콜로서, 오픈 소스 프로젝트로 관리되며, 무인항공기를 비롯한 다양한 자동화 시스템에서 사용된다.

Mavlink는 무인 항공기와 지상 장비간의 통신에 사용됩니다. 무인 항공기가 수집한 데이터(예: GPS 위치, 배터리 상태 등)를 지상 장비로 전송하고, 지상 장비에서는 이를 분석하여 조종 신호를 보낸다. Mavlink는 이러한 통신에 필요한 데이터 구조, 메시지 프로토콜 및 통신 방식을 제공한다.

Mavlink는 다양한 프로그래밍 언어와 하드웨어 플랫폼에서 사용될 수 있으며, 일반적으로 무인 항공기 제어를 위한 시스템에서 많이 사용된다. 또한 Mavlink는 오픈소스 프로젝트로 개발되어 있으며, 다양한 커뮤니티에 의해 지속적으로 발전하고 있다.

2.2.2 SIKE

SIKE는 Supersingular Isogeny Key Encapsulation 알고리즘의 약자로, 공개키 암호화 기술

중 하나이다. SIKE는 NIST에서 후보 PQC(Post-Quantum Cryptography) 알고리즘으로 선정되었다. SIKE는 쌍곡선 위에서 정의된 이소젠스를 이용하여 공개키를 생성하고, 이를 기반으로 하는 키패킷을 생성한다. SIKE는 이소젠스를 기반으로 하는 암호화 방식으로서, 현재까지의 컴퓨팅파워로는 불가능한 계산을 요구하는 이소젠스의 속성을 이용해 안전성을 보장한다.

2.2.3 libsodium

Sodium은 암호화, 암호 해독, 서명, 암호 해싱 등을 위한 사용하기 쉬운 현대식 소프트웨어 라이브러리이다.

이식 가능하고 교차 컴파일 가능하며 설치 가능하고 패키지 가능한 NaCl포크이며 호환 가능하지만 확장된 API를 통해 사용성을 더욱 향상시킨다.

목표는 더 높은 수준의 암호화 도구를 구축하는 데 필요한 모든 핵심 작업을 제공한다.

3. 프로젝트 내용

3.1 알고리즘

3.1.1 Supersingular Isogeny Key Encapsulation

SIKE 알고리즘은 공개키 기반의 암호화 기술 중 하나이다. SIKE 알고리즘은 고전적인 암호화 기술과는 다른 계산 복잡도를 가지기 때문에, 머신러닝 등 현재까지 사용되고 있는 컴퓨팅 기술로는 해독이 어렵다. 따라서, SIKE 알고리즘은 현재까지는 해독이 불가능한 안전성을 가진 알고리즘으로 평가받고 있다.

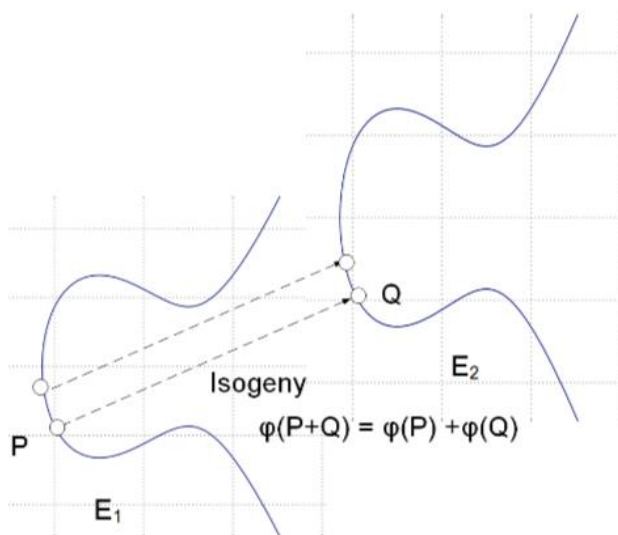
SIKE 알고리즘의 수식을 보면 크게 파라미터 설정, 공개키 설정, 키패킷 생성으로 나눌 수 있다. 파라미터 설정부터 보면 SIKE는 고정된 파라미터를 사용한다. p 는 소수, $2^e \bmod p = 1$, $2^{\lfloor \log_2(p) \rfloor} < N < 2^{\lceil \log_2(p) \rceil}$ 이다. E 는 $x, y \in E(\mathbb{F}_p)$, $\#E(\mathbb{F}_p) = N$, $(2^e \bmod$

$N \cdot E = \infty$ 이다. P와 Q는 E의 유한군 $E(F_p)$ 에서 생성된 무작위 점이다.

공개키 생성에는 S, Ppublic, Qpublic이 있다. S는 무작위 비밀값이다. $P_{public} = S \cdot P$ 이고 $Q_{public} = S \cdot Q$ 이다.

키패킷 생성에는 R, K, Pkey, Shared Secret가 있다. R은 무작위 비밀값이다. $K = \text{hash}(Q_{public} \parallel R)$ 이다. $Pkey = R \cdot P + K \cdot Q_{public}$ 이다. Shared Secret = $\text{hash}((S \cdot Pkey) \cdot P)$ 이다.

두 개의 타원 곡선(E_1 과 E_2)이 있는 경우, E_1 의 점(P)을 E_2 의 점(Q)에 매핑하는 함수를 만들 수 있다. 이 함수를 아이소제니라고 한다. 이 함수를 매핑할 수 있다면 E_1 의 모든 점을 E_2 에 매핑할 수 있다. 그러면 비밀 키는 아이소제니이고 공개 키는 타원 곡선이다. 키 교환을 위해 밥과 앨리스는 자신의 아이소제니와 상대방의 곡선을 섞어 비밀 곡선을 생성한다.



[그림1] 타원 곡선과 아이소제니

따라서 동등 타원 곡선을 사용하면 단일 타원 곡선이 아니라 타원 곡선 계열을 다루게 된다.

그러면 아이소제너레이션은 맵($\phi: E_1 \rightarrow E_2$)이며 소스 커브 E_1 의 점이 타겟 커브 E_2 에

매핑되는 곳이다. 모든 동등성($\phi: E1 \rightarrow E2$)에 대해 이중 동등성($\phi: E2 \rightarrow E1$)이 존재하므로, 두 커브가 동등성으로 연결되어 있으면 두 커브는 동등성이라고 말할 수 있다.

3.1.2 Crypto_secretbox_easy

crypto_secretbox_easy는 libsodium에서 제공하는 secret-key encryption(암호화) 알고리즘 중 하나이다. 이 알고리즘은 XSalsa20 스트림 암호화 알고리즘과 Poly1305 인증 알고리즘을 조합하여, 안전하고 빠른 대칭키 암호화를 제공합니다.

3.1.3 XSalsa20

XSalsa20는 스트림 암호화 알고리즘으로, 무제한 길이의 데이터를 암호화할 수 있다. Salsa20과 마찬가지로 XSalsa20는 대칭키 암호화 알고리즘이기 때문에, 암호화와 복호화에 동일한 비밀키를 사용한다. XSalsa20의 작동 방식은 다음과 같다. nonce와 비밀키를 입력으로 받는다. 그 후 256-bit의 확장된 키 스케줄링 함수를 사용하여 256-bit 키와 64-bit 카운터를 생성한다. 생성된 256-bit 키와 192-bit nonce를 사용하여 Salsa20 알고리즘을 실행한다. 이 과정에서 64-bit 카운터는 0으로 초기화된다. 만약 Salsa20가 생성한 64 바이트 출력이 필요하지 않으면, nonce와 카운터를 증가시켜 새로운 64 바이트 출력을 생성한다. 출력은 키 스트림(keystream)으로 사용되어 메시지와 XOR 연산을 수행한다.

3.1.4 Poly1305

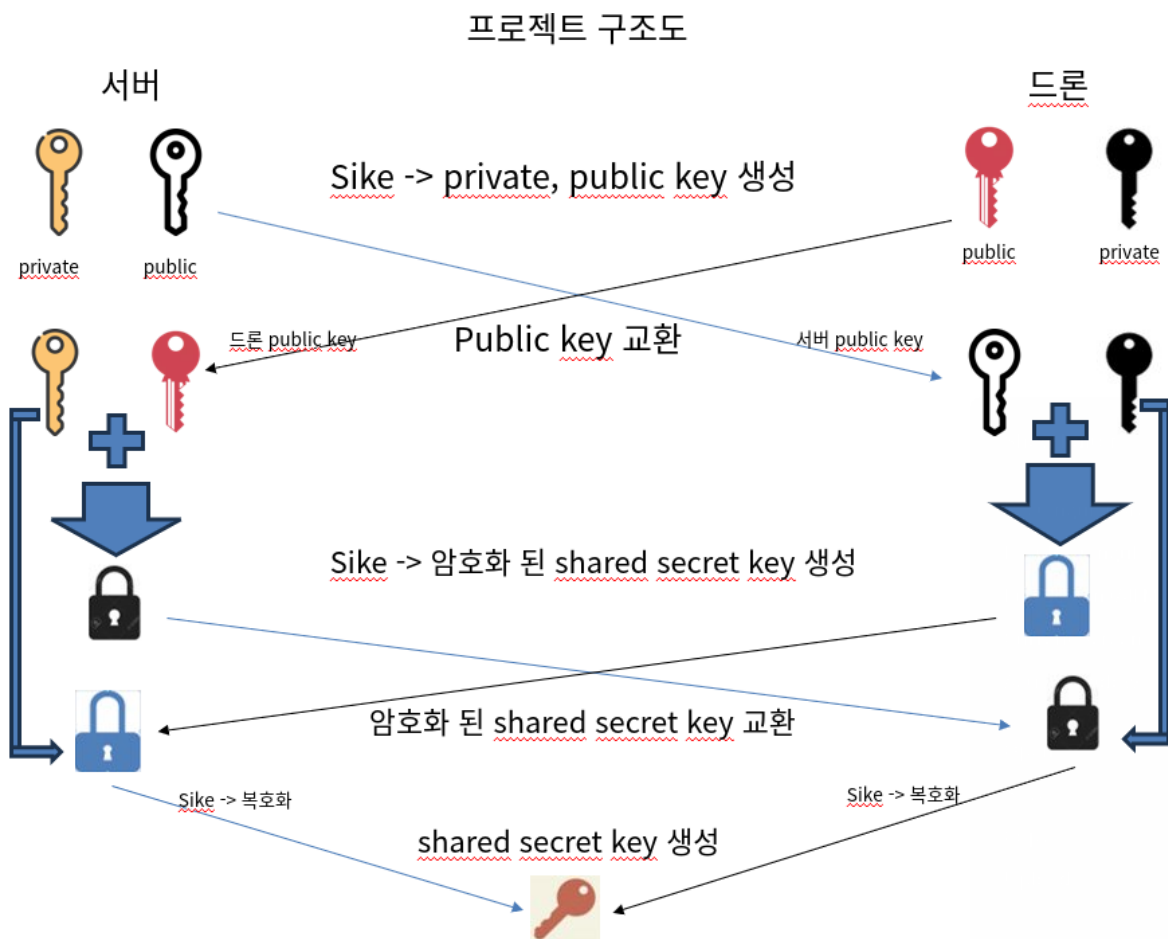
Poly1305는 xsalsa20과 함께 사용되는 인증 알고리즘이다. Poly1305는 고속 하드웨어에서도 매우 빠르고 안전한 HMAC 대안으로 사용된다. Poly1305는 메시지 인증 코드(MAC)를 생성하는 데 사용된다. 이 MAC은 메시지의 무결성을 보호하기 위한 목적으로

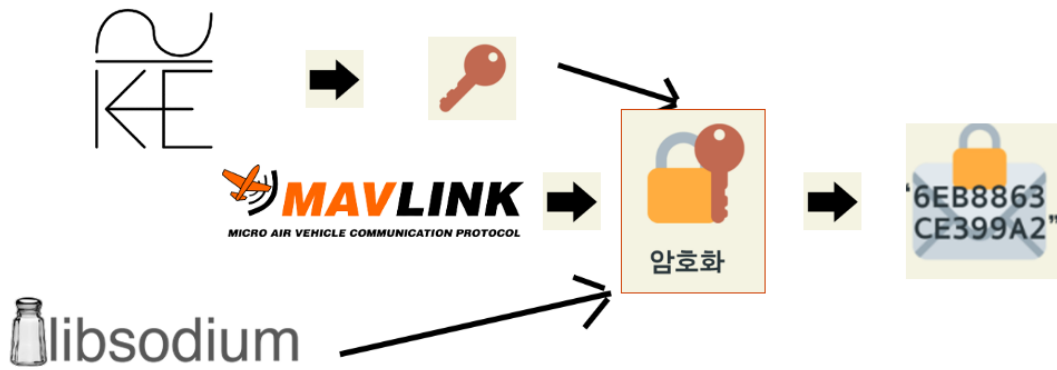
사용된다.

Poly1305의 작동 방식은 다음과 같다. Poly1305는 256-bit 비밀키를 입력으로 받는다. 메시지를 16 바이트 블록으로 나눈다. 마지막 블록은 메시지의 길이를 인코딩하여 블록의 끝에 추가한다. 그 후 각 블록과 이전 블록에서 생성된 Poly1305 MAC을 입력으로 사용하여 다음 MAC을 생성한다.

마지막 블록의 생성이 끝나면, 최종 Poly1305 MAC을 생성한다. XSalsa20와 Poly1305를 함께 사용하는 것은 높은 보안성과 높은 성능을 제공한다. XSalsa20는 안전하고 빠른 스트림 암호화 알고리즘이고, Poly1305는 안전하고 빠른 메시지 인증 알고리즘이다.

3.2 프로젝트 전체 구조





[그림 3] 프로젝트 전체 구조 순서도

서버에서 클라이언트에게 비밀 공유 키를 전송하는 방법은 일반적으로 보안적으로 안전하지 않는다. 비밀 공유 키는 제 3자에게 노출되지 않아야 하기 때문에, 안전한 키 교환 프로토콜을 사용하여 클라이언트와 서버 간에 비밀 공유 키를 안전하게 생성하고 교환해야 한다.

SIKE 프로토콜은 키 교환을 위한 보안 알고리즘이다. 따라서, SIKE를 사용하여 서버와 클라이언트 간에 안전하게 키 교환을 수행할 수 있다. 이를 위해 SIKE의 키 교환 프로토콜을 사용하여 클라이언트와 서버가 비밀 공유 키를 생성하고 교환하게 된다.

키교환 과정은 이렇다. 먼저 클라이언트와 서버는 `crypto_kem_keypair`를 통해 서로 public과 private key를 생성한다. 그 후 서로의 public키를 교환한다. 클라이언트는 서버의 public key와 자신의 private키를 이용하여 `crypto_key_enc()`를 통해 암호화된 비밀 공유키를 생성한 후 서버에게 전송한다. 서버는 클라이언트의 public key와 자신의 private키를 이용하여 암호화된 비밀 공유키를 생성한 후 클라이언트에게 전송해준다. 그럼 각자 받은

비밀 공유키와 자신의 private key를 이용하여 `crypto_kem_dec()`를 통해 비밀 공유키를 복호화하여 사용한다. 이렇게 하면 클라이언트

와 서버 간에 안전하게 공유 비밀 키를 생성하고 교환할 수 있다.

클라이언트측에서는 `mavlink_msg_heartbeat_pack()`을 통해서 원하고자 하는 데이터를 패킹한다. `mavlink_msg_heartbeat_pack()` 함수는 Mavlink 메시지 형식으로 heartbeat 메시지를 패킹하는 함수이다. 이 함수는 주어진 매개변수를 사용하여 heartbeat 메시지를 생성하고 mavlink메세지 버퍼에 패킹한다. 본 논문에서는 Type, Autopilot, Base Mode, Custom Mode, System Status를 정보를 저장했다.

그 후 `mavlink_msg_to_send_buffer()` 함수는 생성된 Mavlink 메시지를 전송 버퍼에 복사한다. 이 함수는 Mavlink 메시지와 메시지 버퍼를 매개변수로 사용한다. 메시지 버퍼는 메시지를 송신하기 전에 전송할 준비가 된 바이트 배열이다.

그 다음 `randombytes_buf()`함수를 사용하여 nonce를 생성해 준다. `randombytes_buf()` 함수는 암호학적으로 안전한 무작위 바이트 시퀀스를 생성하는 함수이다. 이 함수는 주어진 버퍼에 무작위 값을 채워 넣는다. 무작위 바이트를 생성하는 것은 암호학적으로 안전한 난수 생성이 필요한 애플리케이션에서 중요하다. 예를 들어 암호화 키, 초기화 벡터 등을 생성할 때 사용될 수 있다.

앞서 생성한 nonce와 비밀 공유키를 이용하여 `crypto_aead_chacha20poly1305_ietf_encrypt` 함수를 통해 앞서 만든 mavlink 메시지를 암호화한다.

`crypto_aead_chacha20poly1305_ietf_encrypt()`함수는ChaCha20-Poly1305 암호화 스트림과 인증 태그를 사용하여 평문을 암호화하는 함수이다. 이 함수는 주어진 평문 데이터, 평문 길이, 추가 데이터, 비밀 키, 공용 인증 키를 사용하여 암호문을 생성한다.

ChaCha20-Poly1305은 안전한 인증 암호화(AEAD) 알고리즘으로, 대칭키 암호화와 메시지

인증 부호를 결합한 것이다. 이 알고리즘은 데이터의 기밀성과 무결성을 동시에 제공한다. 평문은 암호화되고, 추가 데이터와 함께 인증 태그가 생성됩니다. 이를 통해 평문이 위조되지 않았는지 검증할 수 있다.

마지막으로 암호화된 메시지와 nonce를 memcpy를 통해 packet 변수에 복사한다. 그 후 sendto를 통해 패킷을 전송하게 된다.

서버 측에서는 암호화된 mavlink 메시지를 수신하고, 공유 비밀키를 사용하여 복호화하고 처리하는 과정을 가진다.

암호화된 MAVLink 메시지를 저장할 변수 encrypted_packet을 선언한다.

recvfrom 함수를 통해서 소켓 sockfd에서 클라이언트로부터 암호화된 패킷을 수신한다. 그 후 encrypted_packet에 수신된 데이터가 저장되고, 수신된 데이터의 길이는 n에 저장된다. 복호화된 MAVLink 메시지를 저장할 변수 decrypted_mavlink을 선언한다.

이 변수는 암호화된 패킷의 길이에서 nonce 크기를 뺀 크기로 설정해준다.

crypto_aead_chacha20poly1305_ietf_decrypt() 함수는 encrypted_packet에 저장된 암호화된 MAVLink 메시지를 shared_secret 비밀키를 사용하여 복호화한다. 복호화된 메시지는 decrypted_mavlink에 저장된다.

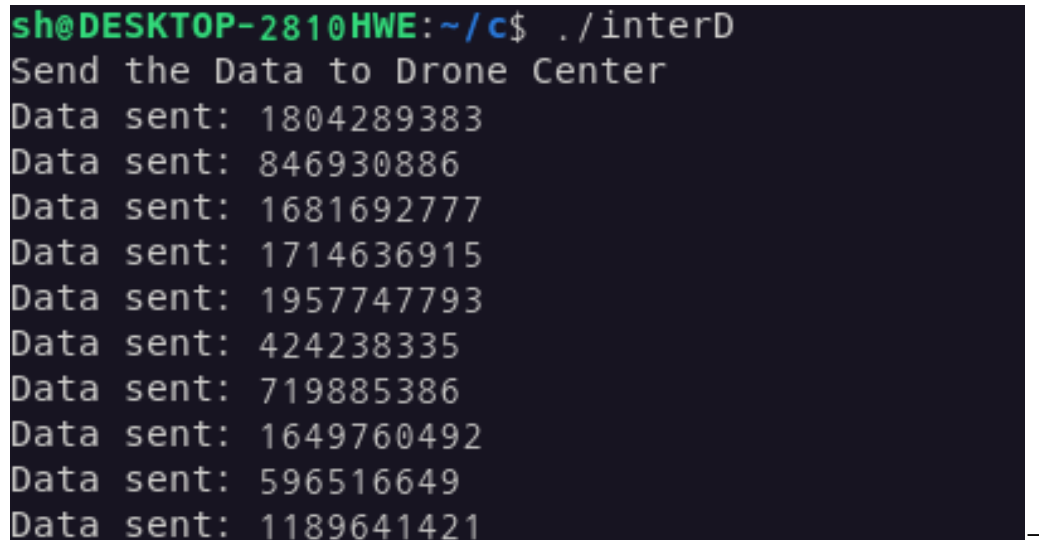
복호화된 MAVLink 메시지를 한 바이트씩 반복하여 처리한다. decrypted_mavlink 배열에는 현재 반복되고 있는 바이트를 byte 변수에 저장한다. mavlink_parse_char() 함수를 사용하여 수신된 바이트를 MAVLink 메시지로 파싱한다. 메시지가 성공적으로 파싱되면, msg에 파싱된 메시지가 저장되고 status에 상태 정보가 저장된다. 이 정보는 메시지의 타입(msgid)을 기준으로 분기하여 처리한다. Heartbeat 메시지를 저장할 변수 heartbeat을 선언하고 msg에 저장된 MAVLink Heartbeat 메시지를 heartbeat 변수에 디코딩하여 저장해준다. 이를 통해 Heartbeat 메시지의 필드 값들을 추출할 수 있다. 이렇게 복호화된 메시지의 필드 값을 출력해준다. 이는 드론의 유형, 자동 조종 장치, 기본 모드, 사용자 정의

모드, 시스템 상태 등을 표시한다. 필요에 따라 다른 메시지 유형을 처리하는 부분도 추가로 구현할 수 있다. 본 논문에서는 이륙, 목표 위치 이동, 착륙 명령을 전송하고 처리하는 것으로 구성했다.

4. 프로젝트 결과

구현 환경은 데비안11이다. 언어는 C/C++을 사용하여서 작성하였다. 사용 라이브러리는 sike.h, mavlink.h, libsodium을 사용하였으며 모두 공식 홈페이지에서 다운 후 빌드하여 사용하였다.

첫 번째는 mavlink만 사용하여 데이터를 주고 받는 모습이고 두 번째는 비밀 공유키를 생성하는 모습이다. 마지막으로는 mavlink 메시지의 오토파일럿의 custom mode 타입, 이륙, 목표 위치 이동, 착륙 명령을 사용자가 선택하여 명령을 암호화하여 전송하고 드론측은 명령을 받아 복호화하여 명령을 처리하였다.

A terminal window with a dark background and green text. The prompt is 'sh@DESKTOP-2810HWE:~/c\$'. The command './interD' has been executed. The output shows a series of 'Data sent:' messages followed by 10-digit hexadecimal values. The values are: 1804289383, 846930886, 1681692777, 1714636915, 1957747793, 424238335, 719885386, 1649760492, 596516649, and 1189641421.

```
sh@DESKTOP-2810HWE:~/c$ ./interD
Send the Data to Drone Center
Data sent: 1804289383
Data sent: 846930886
Data sent: 1681692777
Data sent: 1714636915
Data sent: 1957747793
Data sent: 424238335
Data sent: 719885386
Data sent: 1649760492
Data sent: 596516649
Data sent: 1189641421
```

[그림4] 드론측 데이터 전송

```

sh@DESKTOP-2810HWE:~/c$ ./interS
Receive drone's Data << '
Received data: 1804289383
Received data: 846930886
Received data: 1681692777
Received data: 1714636915
Received data: 1957747793
Received data: 424238335
Received data: 719885386
Received data: 1649760492
Received data: 596516649
Received data: 1189641421

```

[그림5] 서버측 데이터 수신

먼저 mavlink만 사용하여 데이터를 전송하고 전송받는 모습이다.

```

sh@DESKTOP-2810HWE:~/c$ ./_s1
server start
Shared secret: 003038dbc3117f00e03ee7c3117f00e03ee7c3117f00040e7c3117f
sh@DESKTOP-2810HWE:~/c$ ./_d1
Shared secret: 003038dbc3117f00e03ee7c3117f00e03ee7c3117f00040e7c3117f

```

[그림6] 서버에서 생성한 비밀공유키 이미지(위), 드론 이미지(아래)

sike를 통해서 서로의 public key를 전송하고 자신의 private key를 이용하여 서로 같은 비밀 공유키를 생성한 모습이다. 이 비밀 공유키는 데이터를 암호화하고 복호화하는데에 쓰인다.

```

sh@DESKTOP-2810HWE:~/c$ ./_s4
server start
Shared secret: 0098fe3e38ef7f0098fe3e38ef7f002013f38ef7f00741f3f38ef7f

1. 비행 타입 명령
2. 이륙 명령
3. 목표 위치 이동 명령
4. 착륙 명령
5. 종료
명령 입력 : 1
비행 타입 명령 전송 - custom mod 입력: 134
Encrypted Mavlink message sent to drone

1. 비행 타입 명령
2. 이륙 명령
3. 목표 위치 이동 명령
4. 착륙 명령
5. 종료
명령 입력 : 2
이륙 명령 - 목표 고도 입력 : 42
Encrypted Mavlink message sent to drone

1. 비행 타입 명령
2. 이륙 명령
3. 목표 위치 이동 명령
4. 착륙 명령
5. 종료
명령 입력 : 3
목표 위치, 목표 가속도, 회전 반경 전송
목표 위치 입력 m - (x, y, z) : 50 50 600
속도 입력 m/s - (vx, vy, vz) : 1 2 3
목표 가속도 입력 m/s/s - (afx, afy, afz) : 4 5 6
회전 반경 & 비율 degree 입력 - (yw, yrate) : 1 22
Encrypted Mavlink message sent to drone

1. 비행 타입 명령
2. 이륙 명령
3. 목표 위치 이동 명령
4. 착륙 명령
5. 종료
명령 입력 : 4
착륙 위치 입력 (x,y,z) : 1 5 3
착륙 방향 입력 : 10
Encrypted Mavlink message sent to drone

```

[그림7] 서버에서 드론으로 mavlink 명령을 암호화 후 전송

```

sh@DESKTOP-2810HWE:~/c$ ./_d3
Shared secret: 0098fe3e38ef7f0098fe3e38ef7f002013f38ef7f00741f3f38ef7f

Received encrypted Mavlink message from server
비행 타입 정보: Type=2, Autopilot=3, Base Mode=0, Custom Mode=134, System Status=4

Received encrypted Mavlink message from server
이륙 명령 수신 - 목표 고도: 42m

Received encrypted Mavlink message from server
비행 명령 수신 - 위치: (50, 50, 600) m
비행 명령 수신 - 속도: (1, 2, 3) m/s
비행 명령 수신 - 목표 가속도: (4, 5, 6) m/s/s
비행 명령 수신 - Yaw: 1, Yaw Rate: 22 degree

Received encrypted Mavlink message from server
착륙 명령 수신 - 위치: (1, 5, 3) m
착륙 방향 10 degree

```

[그림 8] 드론에서 mavlink 명령을 수신 후 복호화

서버는 mavlink 메시지를 이용해서 오토파일럿 모드를 설정하고 이륙, 비행하는 위치, 착륙 명령까지 암호화하여 전달한다. 이때 데이터는 비밀 공유키로 암호화 된 후 전송이 된다. 드론에서는 서버에서 전송된 mavlink 메시지를 받아 복호화하고 받은 명령을 출력하는 모습이다.

5. 결론

5.1 기대효과

이 프로젝트를 통해 드론 통신 보안 강화를 위해 PQC 의 SIKE 알고리즘을 적용하고 시뮬레이션 및 테스트를 진행한다. 실험 결과, 기존 MAVLINK 통신에 비해 더욱안전하고 보안성이 높은 통신 방식을 제안한다. 특히 PQC를 이용한 SIKE 알고리즘의 적용으로, 기존의 RSA 나 DH 등 대칭키 알고리즘과 달리 공개키 기반의 암호화를 가능하게 하여 중간자 공격 등에 대한 취약성을 보완한다. 보안성이 더욱 강화된 드론 통신 시스템을 구축할 수 있다. 따라서 이 프로젝트를 통해 드론 통신 보안 분야에서의 기술 발전과 안정적인 통신시스템 구축에 기여할 수 있으며, 이를 통해 드론을 보다 안전하게 운용할 수

있는 효과가 기대된다.

5.1 향후 연구 방향

SIKE는 아이소지니 기반 키교환 프로토콜을 KEM 으로 확장한 프로토콜로 ECC의 프리미티브 연산에 기반을 두고 있는 양자 내성 암호이다. SIKE는 매우 작은 키와 암호문의 크기를 가지지만 연산 속도가 느린 단점을 가지고 있다. 따라서 저성능 cpu 상에서 만족할 만한 성능을 도출하는 것이 매우 중요하다. CANS'19에서 발표된 연구 결과에서는 Montgomery multiplication을 최적화하고 연산자들의 파이프라이닝 을 만족하는 형식으로 명령어셋을 조정함으로써 높은 연산 성능을 달성하는 것이 가능함을 보였다. 따라서 향후에 연구를 지속할 수 있다면 높은 연산 성능을 달성할 수 있는 방법을 고안한다면 더욱더 차별화된 PQC알고리즘으로 활용할 수 있을 것이다.

연산 속도와 메모리 사용량의 비교는 [표 1] 그리고 [표 2]에 각각 나타나 있다.

(연산 단위: clock cycles)

Scheme	Function	PQM4 [2]	Seo et al. [12]
SIKEp434	KeyGen	-	74
	Encaps	-	122
	Decaps	-	130
SIKEp503	KeyGen	1,086	104
	Encaps	1,799	172
	Decaps	1,912	183
SIKEp751	KeyGen	3,651	282
	Encaps	5,918	455
	Decaps	6,359	491

[표1]

Scheme	Function	PQM4 [2]	Seo et al. [12]
SIKEp434	KeyGen	-	6,580
	Encaps	-	6,916
	Decaps	-	7,260
SIKEp503	KeyGen	-	6,204
	Encaps	-	6,588
	Decaps	-	6,974
SIKEp751	KeyGen	-	11,116
	Encaps	-	11,260
	Decaps	-	11,852

[표2]

6. 참고문헌

[1] 박찬희. "mbedTLS 상에서 격자기반 양자내성암호의 구현 및 성능평가."

국내석사학위논문 부산대학교 대학원, 2020. 부산

[2] 저성능 사물인터넷 상에서의 양자 내성 암호 구현. 정 보 보 호 학 회 지 제 30 권 제1호, 2020. 2

[3] <https://en.wikipedia.org/wiki/MAVLink>

[4] <https://doc.libsodium.org>

[5] <https://mavlink.io/kr>

[6] <https://github.com/microsoft/PQCrypto-SIDH>

[7] <https://sike.org>

[8] <https://eprint.iacr.org/2016/413.pdf>