Rust 기반 암호 구현 기술 동향

Trends in Cryptography Implementation Based on Rust

김상원*, 엄시우* , 송민호* , 서화정 **
* 한성대학교 (대학원생), ** 한성대학교(교수)

요약

Rust 프로그래밍 언어의 핵심적인 특성과 다양한 산업 분야에서의 적용 사례를 탐구하며, Rust의 특성이 현대 소프트웨어 개발에서 얼마나 중요한지 강조한다. Rust의 실제 적용 사례를 통해 이론적 이점이 실제 환경에서 어떻게 구현될 수 있는지를 보여 준다. 이러한 사례로, Dropbox, Mozilla, Microsoft 그리고 Amazon 사용 사례를 기술한다. 또한, 암호 구현 분야에서의 Rust 사용 사례를 다양한 논문을 통해 살펴보려 한다.

서론

최근 C, C++ 등의 저수준 언어에서 발생하는 메모리 버그로 인한 보안 결함에 대한 경각심이 높아지고 있다. 이에 따라 메모리 안전한 프로그래밍 언어의 중요성이 강조되면서, Rust 언어의 입지가 점차 확대되고 있다. 미국 사이버국장실(Office of the National Cyber Director)에서 발표된 보고서에는 메모리 안전한 프로그래밍 언어 사용을 촉진하여 소프트웨어 개발 과정에서 발생할 수 있는 보안 취약점을 줄이기 위한 내용이 포함되어 있다. 이러한 배경 아래에서, Rust의 실제 적용 사례들은 그 장점을 더욱 명확히 드러내고 있다.

예를 들어, Dropbox는 대규모 파일 메타데이터 동기화 시스템을 C++에서 Rust로 전환함으로써 메모리 안전성을 크게 향상시켰으며, 이로 인해 시스템의 안정성이 크게 향상되었다. 또한, Mozilla는 Firefox 브라우저의 엔진인 Servo를 Rust로 개발하여 병렬 처리 성능과 메모리 안전성을 크게 개선했으며, 이는 사용자 경험의 질을 높이는 결과를 가져왔다. Microsoft도 Windows의 구성 요소 중 일부를 Rust로 재작성함으로써 보안성을 강화하고 있다. Microsoft의 엔터프라이즈 및 OS 보안 담당 부사장인 David Weston에 따르면, Microsoft는 메모리 안전성 문제를 줄이고 소프트웨어와 하드웨어의 안전성을 높이기 위해, 시스템 수준에서 안전한 Rust 언어의 도입을 추진하고 있다고 했다. Amazon AWS 역시 Rust를 도입하여 서버리스 컴퓨팅 플랫폼인 AWS Lambda의 성능과 안전성을 개선했다. 이로 인해 클라우드 서비스의 처리 능력 및 메모리 관리가 보다 효율적으로 이루어지고 있다. Google 엔지니어링 이사는 최근 진행된 Rust Nation UK 컨퍼런스에서 고(Go)나 C++로 작성된 프로젝트를 러스트로 전환한 경험을 발표하며 Rust는 안정성 뿐만 아니라 생산성 면에서도 우수하다고 발표한 바 있다.

이들 사례는 Rust가 메모리 안전성, 시스템의 안정성, 그리고 높은 성능을 요구하는 다양한 분야에서 어떻게 활용될 수 있는지를 보여 준다.

Rust의 특징과 관련 연구

Rust는 2010년 Mozilla Research의 그레이든 호어에 의해 처음 개발되었다. 이 언어는 메모리 안전성을 보장하면서도 가비지 컬렉션을 사용하지 않고 성능 저하 없이 안정적인 시스템을 구축할 수 있게 하는 시스템 프로그래밍 언어이다. Rust의 컴파일러는 컴파일 시에 고유한 소유권 및 빌림 시스템으로 메모리 안전을 검증하여 널 포인터 역참조, 버퍼 오버플로, 메모리 누수와 같은 일반적인 버그들을 사전에 차단한다.

Rust는 최근 몇 년 간 C++와 시스템 수준 프로그래밍 분야에서 두드러진 차이를 보이며, 진정한 경쟁자로 자리매김하였다. 소켓 서버나 암호화 알고리즘 등의 응용 프로그램 개발에 있어서, 성능의 미세한 향상이 전체 시스템의 성능에 큰 영향을 미칠 수 있다. 이와 관련된 연구에서는 C++와 Rust로 작성된 코드의 속도와 효율성을 비교 분석하였다. 그 결과, Rust는 컴파일 시간이 짧고 메모리 안전성이 뛰어난 것으로 나타났으며, 많은 상황에서 동등하거나 더 나은 성능을 제공하는 것으로 확인되었다.

암호구현 동향

암호화 기술은 데이터 보호와 정보 보안의 핵심 요소로, 그 중요성은 계속해서 증가하고 있다. 최근 다양한 애플리케이션에서 요구되는 보안 수준이 높아짐에 따라, 더욱 효율적이고 강력한 암호화 구현이 필수가 되고 있다. 이러한 배경 속에서, 프로그래밍 언어의 선택은 암호 알고리즘의 성능과 안정성에 직접적인 영향을 미친다.

특히, Rust는 암호화 알고리즘을 구현할 때 발생할 수 있는 여러 보안 취약점을 효과적으로 줄여줄 수 있는 강력한 도구이다. 컴파일 시간에 메모리 안전성 검증을 제공함으로써, 실행 시간 오류로 인한 보안 위협을 사전에 차단할 수 있는 능력은 암호화 애플리케이션에서 특히 중요하다. 이와 관련된 사례들을 살펴보겠다.

한 연구에서는 TinyJAMBU라는 경량 암호화 알고리즘을 중심으로 여러 프로그래밍 언어의 성능을 비교 분석하였다. TinyJAMBU는 리소스가 제한된 IoT 환경에서 사용하기 위해 설계된 알고리즘으로, 낮은 메모리와 처리 능력 요구 사항에도 불구하고 높은 보안성을 제공한다.

이 연구에서 Rust, C, 그리고 Go 언어로 구현된 TinyJAMBU의 성능을 비교한 결과, C와 비교했을 때, Rust는 전력 소비 면에서는 C와 거의 비슷하고, 성능 면에서는 약간 뒤처지지만, 이를 다르게 보자면 Rust가 C와 비슷한 자원을 가지고 비슷한 속도를 내며 메모리 안전성까지 가질 수 있다는 것을 의미한다.

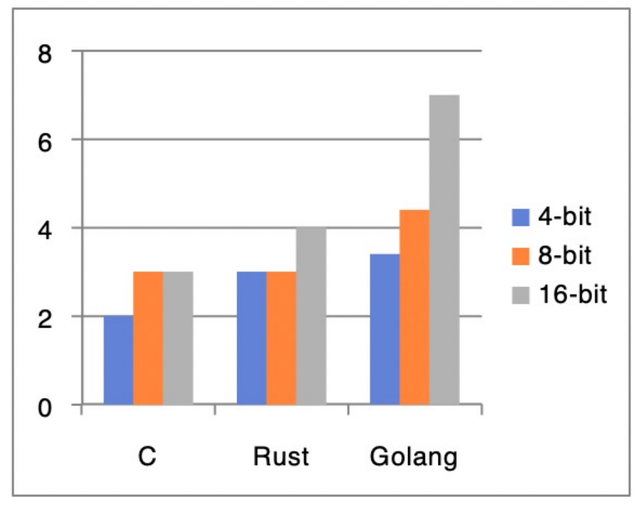

그림 1 Power Consumption Between Different Message Length (Average)

암호학 연구의 중요성이 증가함에 따라, 블록체인 애플리케이션에 필요한 Rust 암호화 라이브러리인 'fastcrypto'는 Rust의 고급 기능을 활용하여 지속적인 성능 향상과 보안 강화를 목표로 한다. 이 라이브러리는 특히 블록체인 애플리케이션에서 요구하는 엄격한 성능 기준을 충족시키기 위해 설계되었다. Rust의 메모리 안전성 보장, 데이터 레이스 방지, 그리고 효율적인 멀티 스레딩 처리 능력은 'fastcrypto'를 이용한 암호화 작업의 신뢰성과 효율성을 높여준다. 또한, 'fastcrypto'는 자동화된 벤치마킹 도구를 통해 라이브러리의 각 구성 요소가 최적의 성능을 발휘하도록 지속적으로 검토하고 개선한다. 이러한 접근 방식은 암호학 연산의 속도를 높이는 동시에 잠재적인 보안 취약점을 식별하고 해결하는 데 큰 도움이 된다. 예를 들어, 디지털 서명, 해시 함수, 암호화 프로토콜 등의 구현이 이루어질 때, 각 기능의 성능이 지속적으로 모니터링되어 효율적인 암호학 솔루션을 제공하기도 한다.

사이버 물리 시스템(CPS)와 사물인터넷(IoT)의 보안과 관련된 RISC-V 병렬 저전력 아키텍처(PULP)를 활용한 연구도 진행되었다. 특히, 이 연구에서는 비디오 스트림의 보안을 강화하기 위해 스트림 암호화 기법을 적용하고, Rust 코드 안전성을 활용하여 ChaCha20 및 AES-CTR 스트림 암호화 알고리즘을 효율적으로 구현하였다. 또한, Rust의 외부 함수 인터페이스(FFI)를 사용하여 기존의 C 언어로 작성된 PULP SDK와의 연동을 가능하게 하며, PULP 아키텍처의 병렬 처리 능력을 최대한 활용하여 암호화 알고리즘의 성능을 향상시켰다. 이로 인해, PULP 아키텍처에서 스트림 암호화를 구현할 때 발생할 수 있는 보안 취약점을 최소화할 수 있게 됐다. 결과적으로, 이 연구는 저전력이면서 고성능을 요구하는 임베디드 시스템에서의 스트림 암호화 구현에 있어 Rust의 유용성을 입증한다.

양자내성암호 분야에서도 Rust의 사용을 엿볼 수 있다. 이 연구에서는 Jasmin 프레임워크와 Rust 프로그래밍 언어의 연결을 탐구하여, 양자내성 암호를 구현하는 방법을 제시한다. Rust는 시스템 프로그래밍에 널리 사용되고 있는데, 그 이유는 그 효율성과 안전한 코딩 개념 때문이다. 이 연구에서는 Jasmin으로 구현된 기존의 Kyber 알고리즘을 Rust로 확장하여, Rust 프로그램에서 안전하게 호출할 수 있는 고효율의 Kyber 라이브러리(RjKyber)를 제공한다. 이 과정에서 Jasmin 구현이 Rust에서 호출되면서 두 언어의 보안 제약을 유지하는 방법을 조사한다. 이를 통해 얻은 통합의 효율성, 안전성 및 사용 용이성은 단독 언어로는 달성할 수 없는 수준이다. 이는 특히 공개키 암호화 기법이 양자 컴퓨터로 인해 취약해질 수 있는 현 상황에서 매우 중요하다.

Function	Pqc kyber(us)	Rjkyber(us)
Keygen	168.51	73.882
Encap	174.06	88.832
Decap	174.21	101.90

결론

본 논문에서 검토한 Rust의 적용 사례와 연구 결과들은 Rust가 단순히 메모리 안전성만을 제공하는 것이 아니라, 복잡한 멀티스레드 환경에서의 데이터 무결성 보장, IoT 및 클라우드 컴퓨팅 환경에서의 안전성 및 성능 향상, 그리고 고급 암호화 알고리즘의 구현 등 다양한 프로그래밍 요구를 충족시키는 유연성을 갖추고 있음을 확인할 수 있다.