웹 어셈블리를 활용한 초경량 블록암호 CHAM 최적화 구현

안규황* 권혁동* 김현준* 서화정*†

*한성대학교 정보시스템공학과

tigerk9212@gmail.com, hdgwon@naver.com, khj930704@gmail.com, hwajeong84@gmail.com

Implementation of ultra-light block cipher CHAM optimization using Web Assembly

Kyuhwang An* Hyeokdong Kwon* Hyunjun Kim* Hwajeong Seo*†

*Division of information system, Hansung University.

요 약

웹 페이지를 구현하기 위해선 주로 자바스크립트를 사용한다. 자바스크립트는 high-level 언어로, 사용하기 편하다는 장점이 있지만, 그만큼 상대적으로 low-level 언어인 C/C++보다 구현 성능이 떨어진다. 이러한 자바스크립트의 단점을 보안하고자 웹 어셈블리라는 기술이 탄생하였다. 웹 어셈블리는 C/C++로 작성 된 코드를 웹 어셈블리 파일인 이진 바이너리 파일로 변환한 후 하나의 웹 페이지로 변환하는 기술이다. 단순히 자바스크립트로 구현하는 것보다 low-level 언어를 이용하는 웹 어셈블리로 구현하는 것이 엄청난 성능향상을 보여주며 다양한 분야에 활용될 것을 기대하고 있다. 또한 웹 어셈블리는 기존의 언어들과 다르게 연산을 수행하는 별도의 선형 메모리에서 연산을 수행하며, 사전에 권한을 부여 받은 객체만 선형 메모리에 접근할 수 있다. 따라서 자바스크립트보다 보안성과 성능 면에서 웹 어셈블리가 뛰어나다. 이를 실제로 검증하기 위해 본 논문에서는 자바스크립트로 구현한 CHAM과 웹 어셈블리로 구현한 CHAM간의 성능 비교를 하였으며 실제로 자바스크립트로 구현한 CHAM보다 웹 어셈블리로 구현한 CHAM이 약 6.15배의 성능 향상을 볼 수 있었다.

I. 서론

자바스크립트는 웹 페이지 내에서 동적으로 객체를 활용하고 싶을 때 사용되는 객체 기반의 스크립트 프로그래밍 언어이다. 자바스크립트는 컴파일을 수행하는데 있어 JIT(Just-In-Time) 컴파일 방법을 사용한다. JIT 컴파일이란 프로그램을 실제 실행하는 시점에 인간의 언어로 작성 된 자바스크립트 코드를 기계어로 번역하여 컴파일하는 기법이다.

자바스크립트가 최초로 탄생했을 때는 JIT 컴파일 기법을 사용하지 않아 엄청나게 느린 환경을 가지고 있었지만, 2008년에 들어 많은 웹 브라우저들이 JIT 컴파일러를 수용함에 따라 자바스크립트의 성능이 엄청나게 향상될 수 있었다.

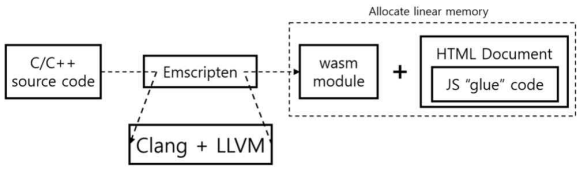
자바스크립트는 과거에 프론트단(front-end)에서 주로 활용이 되었지만 근래에 들어 node.js와 같이 백단(back-end)을 커버해주는 서버 사이드 네트워크 프로그래밍에도 활용되며, 모바일 웹을 개발하는데 사용되고 있다.

모바일 웹의 경우 치명적인 단점을 가지고 있다. 바로 속도이다. 모바일 웹은 네이티브 웹&앱에 비해 속도가 많이 느려 연산이 많이 필요로 하는 작업을 할 경우 완벽히 수행하지 못하는 등 문제를 발생시켰다. 이를 해결하기 위해 마이크로소프트사에서 개발한 ActiveX가 배포되었지만, ActiveX를 이용할 경우 보안에 취약한 등 새로운 문제가 발생하였고 해당 기술을 적용할 수 있는 웹 브라우저에서만 호환이 가능하다는 치명적인 단점이 발생하였다.

웹 어셈블리(Web Assembly)는 앞에서 언급한 속도와 같은 성능에 대한 문제점을 해결해 줄 수 있는 대책으로 기계어로 그 즉시 컴파일 시키기 때문에 JIT 컴파일 기법보다 훨씬 좋은

성능을 자랑한다.

웹 어셈블리는 자바스크립트로 작성하는 것이 아닌 C/C++로 작성한 코드를 웹상에서 컴파일하는 저수준 바이트코드로 바이너리 형식을 통하여 실행되며 독립적인 메모리를 할당하여 사용한다. 웹 어셈블리를 활용한다면 기존에 웹 혹은 사물인터넷 플랫폼 상에서 단순히 자바스크립트로 사용하였던 암호 알고리즘들에 대해 보다 빠른 성능을 보일 것을 기대한다. 동작 구조는 fig. 1과 같다.



Fig, 1. How to run web assembly

본 논문의 구성은 다음과 같다. 2장에서는 웹 어셈블리에 대해 설명한다. 3장에서는 웹 어셈블리를 이용하여 구현한 CHAM에 대하여 성능 평가를 하겠으며, 4장에서 끝을 맺도록 하겠다.

II. 웹 어셈블리를 이용한 최적 구현

본 절에서는 CHAM을 구현하기 위해 참조 논문[1]에서 나타낸 바를 웹 어셈블리로 구현[1] 하였으며 아래 각주에서 확인 가능하다. 구현한 환경은 table. 1과 같다.

웹 어셈블리를 구현함에 있어 가장 중요한 것은 웹 어셈블리를 지원하는 웹 브라우저를 이용해야 하며, 해당 웹 브라우저가 웹 어셈블리 기능을 지원한다고 하더라도 그 중에 지원하지 않는 모듈도 존재한다. 따라서 본인이 원하는 모듈[2]이 무엇인지 확인한 이후에 지원하는 웹 브라우저를 선택해서 구동해야한다. 웹 어셈블리를 지원하는 브라우저별 최소 버전은 table. 2와 같다.

Table 1. The information of experiment environment

OS	macOS 10.12.6
IDE	Brackets 1.10
Web Browser	Chrome 69.0.3497.100

1)https://github.com/kyu-h/WebAssembly_CHAM

Table 2. The list of Web Assembly support web browser

Chrome	57
Edge	16
Firefox	52
Explorer	Not Support
Safari	11

웹 어셈블리는 기존 웹 구동 방식과 다르게 선형 메모리(Linear Memory)를 할당하고 해당 영역에서 동작한다. 기존 메모리 영역의 경우 관리자의 승인 없이도 데이터가 유동적으로 이동할 수 있다. 하지만 선형 메모리의 경우 기존 메모리에 있던 데이터를 관리자의 승인 하에 이동 후 선형 메모리에 저장 된 데이터를 보다 안전하게 사용할 수 있다.

만약 웹 어셈블리를 이용하여 컴퓨터에 저장된 파일을 읽어서 사용하고자 할 때, 관리자가 선형 메모리에 파일을 할당하지 않는다면, 웹 어셈블리로 만든 html과 같은 폴더 내부에 있어도 해당 파일을 읽을 수 없다.

따라서 기존 메모리 방식보다 선형 메모리를 이용한 웹 어셈블리 방식이 데이터를 안전하게 보관할 수 있음을 fig. 2와 같이 보여준다.

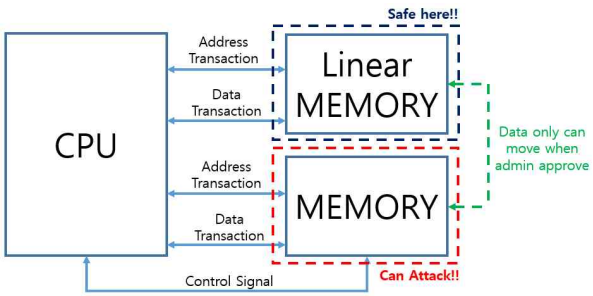


Fig. 2. How to allocate linear memory

III. 성능평가

CHAM에 대하여 3개의 모듈에 대해 1회 암호화 수행하는데 있어, 자바스크립트는 0.008초가 걸렸으며, 웹 어셈블리는 0.0013초가 걸려 약 6.15배 성능 향상을 볼 수 있다. 이는 table. 3과 같다.

앞에서도 설명했듯이 웹 어셈블리는 기존 컴파일 방식과 다르게 독자적인 선형 메모리상에

서 컴파일을 수행한다. 웹의 경우 프론트, 엔드단을 자유롭게 상호작용하며 수행되어야하기 때문에 권한을 할당 받은 파일만 접근할 수 있는 특성을 갖고 있는 웹 어셈블리의 경우 1회 암호화 수행하는 것 보다 성능 저하를 예상하였다.

이를 검증하기 위해 테스트 벡터로 작성 된 메모장을 읽어 3개의 모듈에 대하여 각각 100회 총 300회 암호화 연산을 수행하였을 경우 얼마만큼의 연산 속도가 측정되는지 실험하였다.

각 모듈에 대해 100회 암호 연산을 수행할 때 자바스크립트는 0.51초, 웹 어셈블리는 0.0083초가 걸려 약 6.14배 성능 향상을 볼 수 있었으며, 각 모듈에 대해 10,000회 암호 연산을 수행할 때 자바스크립트는 0.259초, 웹 어셈블리는 0.0873초가 걸려 약 2.96배 성능 향상을 볼 수 있었다.

각 모듈에 대해 10,000회 암호화를 진행할 때는 앞에서 실험한 1, 100회보다 성능을 떨어졌지만, 그래도 단순히 자바스크립트로 구현한 것보다 2.96배의 성능 향상을 하였으며, 2.96배도 엄청난 성과이다.

Table. 3. Performance comparison between javascript and web assembly implementing CHAM

Language	Time(s)	cpb
Time taken to perform 1 encryption operation		
Javascript	0.008s	385,714
Web Assembly	0.0013s	62,678
Time taken to perform 100 encryption operation		
Javascript	0.051s	2,458,928
Web Assembly	0.0083s	400,178
Time taken to perform 10,000 encryption operation		
Javascript	0.259s	12,487,500
Web Assembly	0.0873s	4,209,107

IV. 결 론

본 논문에서는 초경량 블록암호 CHAM에 대하여 웹 어셈블리를 이용하여 최초로 구현하였다. 구현한 결과물과 코드에 대해 깃허브[3]에서 확인할 수 있다.

CHAM을 웹상에서 구현할 때 웹 어셈블리를 이용함으로써 단순히 자바스크립트로 구현한 것 보다 최대 6.15배 빠르게 구현하였다. 또한 기존의 자바스크립트는 단순히 CPU 메모리에 올려 암호화를 수행하기 때문에 보안성이 높지 않으나, 웹 어셈블리 같은 경우 선형 메모리상에 올려 암호화를 수행함으로써 권한을 부여받지 않은 객체는 접근이 불가하다. 따라서 자바스크립트보다 높은 보안성을 제공한다. 추후 연구로는 표준화된 국제 블록암호 혹은 해시함수에 웹 어셈블리를 적용하는 방안에 대해 확인해 볼 계획이다.

[참고문헌]

[1] B. W. Koo, D. Roh, H. Kim, Y. Jung, D. G. Lee, D. Kwon, “CHAM: A Family of Lightweight Block Ciphers for Resource-Constrained Devices.” *International Conference on Information Security and Cryptology*. Springer, Cham, 2017.

[2] MDN web docs, “Browser compatibility, available: https://developer.mozilla.org/ko/docs/WebAssembly#Browser_compatibility

[3] Github, “CHAM with Web Assembly, ” available: https://github.com/kyu-h/WebAssembly_CHAM