차량통신 보안을 위한 암호화 알고리즘의 성능 분석

최원석, 김영진, 정서현, 정준시, 이은향
한국정보통신기술협회 정보통신연구소
wschoi@tta.or.kr, networker@tta.or.kr, shjeong@tta.or.kr, jless@tta.or.kr, ehlee@tta.or.kr

The Performance analysis of cryptography algorithms for vehicle communication

WonSeok Choi, Young-Jin Kim, SeoHyun Jeong, Joonsi Jeong, Eun-Hyang Lee
ICT Testing & Certification Lab., Telecommunication Technology Association

요 약

차량용 통신 시스템은 다양한 센서 정보, 제어 정보를 주고받게 되며 자율주행 등 차량 시스템이 고도화됨에 따라 보안성이 중요해지고 있다. 이에 따라 기존 MCU 구조에서 고성능의 ARM 아키텍처를 사용하는 것이 일반화됨에 따라 본 논문에서는 ARM 기반의 시스템에서 다양한 암호화 알고리즘의 속도와 메모리, CPU 리소스 사용량을 비교 분석한다.

1. 서론

최근 차량용 MCU들은 기존 AVR, MIPS 등 단순한 MCU 구조에서 범용적이고 널리 사용되는 ARM 기반의 아키텍처로 변화되어 가고 있다. 특히 운전 보조, 자율주행 등을 위해 대용량 데이터 처리가 요구되면서 고성능의 CPU 사용이 증가하고 있으며 이에 따라 가용 메모리 용량도 날로 늘어나고 있다. 그러나 차량용 MCU들은 센서의 모니터링, 제어 등 본연의 기능 수행을 상시 수행하고 있어 보안을 위한 암호화 적용에는 한계가 있을 수밖에 없다. 이러한 연유로 HSM(Hardware Security Module) 등을 별도 탑재하기도 하나, 최근에는 MCU 고성능화, 암호화 가속 회로의 내장으로 MCU에서 자체적으로 암호화 연산을 수행하기도 한다. 본 논문에서는 메모리 사용량에 따른 암호화 알고리즘의 성능을 분석하고, 가용 메모리에 따라 적합한 암호화 알고리즘을 제시한다.

2. 암호화 알고리즘의 종류 및 적용 가능성

차량용 MCU의 경우 한정된 CPU 리소스와 메모리를 이용하여 암호화, 복호화를 수행해야 하므로 비대칭 암호화나 복잡한 연산의 암호화는 적용하기 어려운 상황이다. 따라서 일반적으로 널리 사용되는 대칭키 암호화 알고리즘 중 암호화 강도와 보안성이 검증된 알고리즘을 사용하는 것이 유리하다. 현재 가장 널리 사용되는 AES(Advanced Encryption Standard) 암호화 방식은 연산에 높은 CPU 리소스가 요구되어 ARM 기반의 장치에서는 느리다는 단점이 있었다. 이러한 문제점을 해결하기 위해 최근 출시되는 ARM CPU에는 AES 가속을 위한 별도의 Instruction을 탑재하여 출시하기도 하고 있다. 하드웨어적 개선뿐 아니라 암호학 분야에서도 더욱 가볍고 강력한 암호화 기법을 개발하기 위해 노력하였고 이러한 노력으로 LEA(Lightweight Encryption Algorithm), 최근에는 ChaCha20 암호화 알고리즘이 개발되어 여러 분야에서 사용되고 있다. 두 알고리즘은 모두 한정된 CPU 리소스와 메모리에서 적용하기 적합하며 기존 AES 보다 고속으로 암호화 및 복호화가 가능하다는 장점이 있다.[1]

LEA는 2013년 국내에서 개발한 대칭키 블록 암호화 알고리즘으로 ARX(Addition, Rotation, XOR) 연산만으로 구현되어 있어 처리 속도가 AES에 비해 크게 향상되었다. 또한 S-box의 사용을 배제하였기에 코드 크기가 작아진다는 장점이 있다. LEA는 ARM 기반의 디바이스에 다수 사용되고 있으며 암호화 강도 역시 S-box를 사용하지 않아 높다고 평가되고 있다.[2]

ChaCha 암호화 방식은 Salsa20에서 개선된 암호화 알고리즘으로 2008년 처음으로 제안되었다.[3] LEA와 유사하게 ARX 연산을 이용하고 있으며, 암호화 라운드에 따라 ChaCha20, ChaCha12, ChaCha8 등이 제안되어 있다. ChaCha20의 경우 IETF의 RFC 8439로 제안되어 있으며, 현재 BSD계열(FreeBSD, OpenBSD, NetBSD 등) OS의 난수 생성기와 OpenSSH 등 다양한 분야에서 널리 사용되고 있다.

이 밖에도 보안 취약점이 발견된 RC4 암호화와 개선한 RC5, RC6 등이 경량 암호화 방식으로 제안되었으나, 개발한 RSA Security 사의 특허로 상업적으로 사용은 제한적이다.

3. 암호화 알고리즘의 성능 분석 환경 및 지표

차량에 적용하기에 적합한 암호화 기술을 선정하기 위해, 위에서 언급한 경량 암호화 알고리즘인 LEA, ChaCha, AES의 암호화 성능을 비교 분석한다. 분석을 위한 시험 환경은 아래 표와 같다.

<표 1> 암호화 알고리즘 암호화/복호화 속도 시험 환경

CPU	Broadcom BCM2711B0 quad-core A72 (ARMv8-A) 64-bit
Kernel	Linux kernel 5.15.61
Compiler Version	GCC12.2
Memory	8GB LPDDR4-3200
Storage	micro SDXC 16GB UHS-I(U3)

성능 비교 분석을 위해 각 암호화 알고리즘은 Buffer 크기를 1024 bytes부터, 10,240 bytes까지 증가시키면서 성능을 측정하였다. 통신 메시지를 이용하여 측정하기에는 메시지 전송 주기, Jitter 등의 다양한 변수가 존재할 수 있어 본 시험에서는 1MBytes의 파일을 암호화하고, 다시 복호화하는 시간을 측정하고 해당 시간을 이용하여 초당 처리 속도를 계산한다. 이때, Test Vector 값을 이용하여 올바르게 복호화되었는지 확인한다. CPU 리소스 사용량 확인을 위해 CPU의 클럭을 이용하여 아래와 같은 수식으로 CPB(Cycle per byte)를 계산한다.

$$Cycle\,per\,byte = \frac{Cycle\,per\,\mathrm{second}(CPU\,Clock)}{Bytes\,per\,\mathrm{second}}$$

정확한 CPU 리소스 사용량 계산을 위해 모든 시험은 리눅스 System의 CPU 클럭을 1.8GHz로 동작할 수 있도록 코어별 CPU 클럭을 고정한 상태에서 측정하였다.

4. 암호화 알고리즘의 성능 분석

분석 결과, 전반적으로 ChaCha12 암호화 방식이 가장 높은 성능을 보였다. ChaCha12 암호화 방식의 경우 초당 암호화 처리 속도는 Buffer 크기가 9,216 bytes로 설정하였을 때 가장 높은 649,238 KBytes/s로 측정되었다. 복호화 속도도 ChaCha12 암호화 방식이 Buffer 크기가 10,240 bytes일 때, 가장 빠른 640,160 KBytes/s의 속도를 보여주었다.

<표 2> 암호화 알고리즘 암호화/복호화 속도 시험 결과

[단위: KBytes/s]

버퍼 사이즈 (Bytes)	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192	9,216	10,240
ChaCha20 암호화	349,388	381,389	404,072	414,214	399,121	402,913	408,418	411,483	415,467	404,927
ChaCha20 복호화	353,908	380,972	399,150	409,916	395,538	403,572	404,037	411,234	410,767	409,267
ChaCha12 암호화	559,540	592,385	630,162	646,513	624,874	625,135	638,235	638,892	649,238	627,120
ChaCha12 복호화	554,731	592,385	622,768	637,365	616,165	628,295	627,739	638,720	636,827	640,160
AES-256 암호화	48,749	49,108	48,757	49,179	49,004	49,300	49,096	49,316	48,997	49,283
AES-256 복호화	23,587	23,432	23,682	23,745	23,656	23,450	23,665	23,498	23,609	23,643
LEA-256 암호화	113,945	114,491	111,429	113,216	113,297	115,457	113,809	113,385	112,641	115,078
LEA-256 복호화	110,669	113,483	112,274	113,663	90,083	114,296	112,512	76,059	111,202	112,639
LEA-192 암호화	127,522	129,408	127,780	130,399	128,574	130,122	130,174	128,416	128,608	128,452
LEA-192 복호화	128,181	126,702	129,208	130,788	128,719	127,600	130,290	124,927	128,210	130,532

특이한 사항은 일반적으로 Buffer 크기가 증가함에 따라 암호화, 복호화 속도가 증가하는 것이 일반적이나, 시험에 사용된 암호화 알고리즘의 경우 4,096 bytes 이상의 Buffer 크기를 사용하여도 유의미하게 증가하지는 않았다.

시험에 사용된 암호화 알고리즘 중, 가장 느린 속도를 보여준 것은 전통적인 AES-256 암호화 방식이었으며, ChaCha20과 비교해 보았을 때, 약 10%~14% 사이의 성능을 보여주었다.

특이하게, LEA 암호화 알고리즘의 경우 Buffer 사이즈의 영향을 크게 받지 않았으며, LEA 192, 256 알고리즘 모두 일정한 수준의 성능을 보여줌을 알 수 있다. 이는 LEA 암호화 알고리즘 자체가 경량화를 추구하여 개발되었기 때문으로 보인다.

암호화 알고리즘의 CPU 리소스 사용률 확인을 위해 CPB 값을 살펴보면, ChaCha12 암호화 방식이 가장 낮은 CPB 값을 보임을 알 수 있었다. 즉, 해당 암호화 알고리즘이 CPU 리소스를 가장 적게 사용하여 암호화, 복호화를 수행할 수 있음을 보여준다. LEA 암호화 알고리즘의 경우 경량화에 중점을 두어 개발되었으나 CBP 값은 암호화 시 13.7~14.2 사이의 값을 보여 CPU 리소스는 ChaCha 암호화 알고리즘보다 더 많이 사용함을 알 수 있었다. AES의 경우 가장

높은 CPB 값을 보이고 있으며, 가장 많은 CPU 리소스를 사용하며 연산에 많은 시간이 소비됨을 알 수 있다.

<표 3> 암호화 알고리즘별 CPB 산출 결과

[단위: CPB(Cycle Per Byte)]

버퍼 사이즈 (Bytes)	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192	9,216	10,240
ChaCha20 암호화	5.152	4.720	4.455	4.346	4.510	4.467	4.407	4.374	4.332	4.445
ChaCha20 복호화	5.086	4.725	4.510	4.391	4.551	4.460	4.455	4.377	4.382	4.398
ChaCha12 암호화	3.217	3.039	2.856	2.784	2.881	2.879	2.820	2.817	2.772	2.870
ChaCha12 복호화	3.245	3.039	2.890	2.824	2.921	2.865	2.867	2.818	2.827	2.812
AES-256 암호화	36.924	36.654	36.918	36.600	36.731	36.510	36.662	36.499	36.736	36.523
AES-256 복호화	76.310	76.816	76.006	75.803	76.088	76.756	76.060	76.599	76.241	76.131
LEA-256 암호화	15.797	15.722	16.154	15.836	15.887	15.590	15.816	15.875	15.980	15.642
LEA-256 복호화	16.265	15.861	16.032	15.899	19.981	15.748	15.998	23.666	16.187	15.980
LEA-192 암호화	14.115	13.909	14.087	13.804	14.000	13.833	13.828	14.017	13.996	14.013
LEA-192 복호화	14.043	14.207	13.931	13.763	13.984	14.106	13.815	14.408	14.039	13.790

5. 결론

시험 결과, 암호화 알고리즘에 따라, 메모리의 크기, CPU 리소스에 따라 큰 성능 차이를 보였다. 이러한 차이는 차량 통신의 안전성과 성능에 큰 영향을 미칠 수 있으며, 다수의 센서와 고속 통신 기술(Automotive Ethernet 등)이 적용되는 차량 통신 상황에 맞추어 암호화 알고리즘을 적절하게 적용해야 한다. 특히 차량에 탑재되는 CPU와 가용 메모리에 따라 적절한 암호화 알고리즘을 선택하여야 지연시간 등을 최소화 할 수 있다. 본 논문에서 시험하였던 ChaCha 등의 새로운 암호화 알고리즘은 기존 암호화 알고리즘 대비 가장 좋은 성능을 보였으며, 경량으로 구현이 가능하다는 장점이 있다. 그러나 지금까지 오랜 시간 검증이 이루어지지 않았기 때문에 좀 더 많은 연구와 노력이 필요하리라 생각된다. ChaCha 외의 더 좋은 성능의 암호화 방식에 대한 연구도 함께 이루어져야 할 것이다.

[이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2022-0-00979, 자율주행차량 데이터 및 V2X 통신 네트워크 보안성 평가 기술 및 시험기준 개발)]

참고문헌

[1] Taeyean Kwon, Jaehoon Lee, Hyunduk Choi, Okyeon Yi, Seongho Ju., “Efficiency of LEA compared with AES”, Journal of Convergence. Vol. 6. pp. 16 - 25.

[2] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu1, and Dong-Geon Lee, “LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors”, LNSC, Volume 8267. pp. 3 - 27.

[3] Bernstein, Daniel J., “ChaCha, a variant of Salsa20”, The State of the Art of Stream Ciphers. Vol. 8. pp. 3 - 5.