하둡 상에서 ARIA 알고리즘을 이용한 HDFS 데이터 암호화 기법의 설계 및 구현

송영호*, 신영성*, 윤민*, 장미영*, 장재우1)**
*전북대학교 컴퓨터공학부
**전북대학교 IT정보공학과
e-mail: (songyoungho, twotoma, myoon, brilliant, jwchang)@jbnu.ac.kr

Design and Implementation of HDFS data encryption scheme using ARIA algorithms on Hadoop

Youngho Song*, YoungSung Shin*, Min Yoon*, Miyoung Jang*, Jae-Woo Chang**
*Dept of Computer Engineering, Chonbuk National University
**Dept of IT Information Technology, Chonbuk National University

요 약

최근 스마트폰 기기의 보급 및 소셜 서비스 산업의 고도화로 인해, 빅데이터가 등장하였다. 한편 빅데이터에서 효율적으로 정보를 분석하는 대표적인 플랫폼으로 하둡이 존재한다. 하둡은 클러스터 환경에 기반한 우수한 확장성, 장애 복구 기능 및 사용자가 기능을 정의할 수 있는 맵리듀스 프레임워크 등을 지원한다. 아울러 하둡은 개인정보나 위치 데이터 등의 민감한 정보를 보호하기 위해 Kerberos를 통한 사용자 인증 기법을 제공하고, HDFS 압축 코덱을 활용한 AES 코덱 기반 데이터 암호화를 지원하고 있다. 그러나 하둡 기반 소프트웨어를 사용하고 있는 국내 기관 및 기업은 국내 ARIA 데이터 암호화를 적용하지 못하고 있다. 이를 해결하기 위해 본 논문에서는 하둡을 기반으로 ARIA 암호화를 지원하는 HDFS 데이터 암호화 기법을 제안한다.

1. 서론

최근 스마트폰 기기의 보급 및 소셜 서비스 산업의 고도화로 인해 소셜 네트워크 서비스(SNS) 등을 기반으로 사용자가 생성하는 데이터가 급증하고 있다. 사용자가 생성한 데이터는 텍스트, 사진, 동영상 등 다양한 형태의 멀티미디어 콘텐츠를 포함하는 빅데이터(Bigdata) 수준으로 발전하였다. 한편, 빅데이터를 분산 병렬 컴퓨팅 환경에서 처리하기 위한 대표적인 연구로, 맵리듀스(MapReduce) 프레임워크에 대한 연구가 활발히 진행되었다[1]. 현재 맵리듀스를 구현한 대표적인 대용량 분석 시스템으로 하둡(Hadoop)[2]이 존재하며, 하둡은 대용량 데이터 분산 병렬 처리 기법의 표준이 되었다. 따라서 하둡 기반 서비스는 기존 로그 분석 분야 뿐 아니라, 다양한 분야에서 수집된 개인 정보, 사용자 위치 데이터 등을 활용한 사업 마케팅 분야, 소비 패턴 분석 등에 활용되고 있다. 한편, 이러한 활용 분야의 확장은 회사에서 데이터 유출이 발생할 경우 민감한 개인 정보의 유출을 의미하기 때문에, 최근 하둡의 데이터 보안에 대한 연구가 활발히 진행되고 있다. 현재 하둡은 사용자 인증을 통해 네트워크 보안을 지원하여, 외부의 악의적인 공격자의 접근을 차단한다. 또한 내부에 의한 데이터 유출 방지를 위해 하둡의 HDFS 데이터 압축 기능을 이용하여 AES 등의 암호화 코덱을 적용한 HDFS 데이터 암호화 기능을 제공하고 있다.

한편, 개인정보보호법이 시행됨에 따라 최근 국내에서 데이터 보안 연구는 큰 관심을 받고 있으며, 정부에서는 국내 기술로 개발한 블록 암호화 알고리즘인 ARIA[7][8] 암호화 알고리즘을 표준으로 지정하여, 국공립 기관 및 기업에서 데이터 암호화에 사용하도록 권장하였다. 그러나 대표적인 빅데이터 관리 시스템인 하둡에서는 AES 데이터 암호화 만을 지원하고 있다. 따라서 본 논문에서는 하둡 상에서 ARIA 암호화를 지원하는 HDFS 데이터 암호화 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 암호화 기법에 대한 연구 배경을 소개하고, 3장에서는 제안하는 하둡 상에서 ARIA 알고리즘을 이용한 HDFS 데이터 암호화 기법에 대한 설계를 및 제안하는 기법의 구현 사항을 제시한다. 4장에서는 제안하는 기법의 성능평가를 제시하고, 마지막으로 5장에서 결론 및 향후 연구를 소개한다.

2. 연구 배경

2.1 하둡

구글에서 개발한 맵리듀스[1]는 맵(map)과 리듀스(reduce) 2개의 간단한 함수 형식으로 제공된다. 맵리듀스 프레임워크는 대용량 데이터를 이용한 정보 수집 및 분류

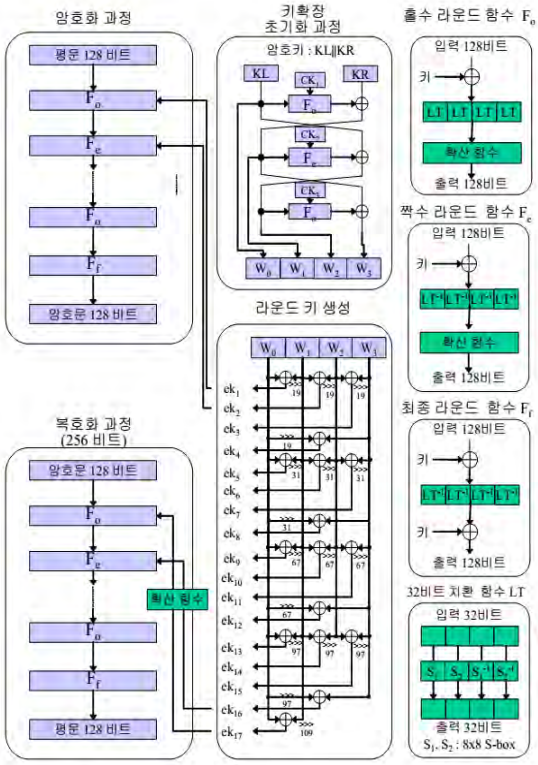
이 논문은 2014년 교육부와 한국연구재단의 지역혁신창의인력양성사업의 지원을 받아 수행된 연구임(NRF-2014H1C1A1065816)

1) 교신저자

기능을 효과적으로 제공하기 때문에, 대용량 데이터를 관리하는 기업의 시스템에 필수적인 요소이다. 한편, 맵리듀스 기술을 구현한 대표적인 대용량 분산 처리 시스템으로 하둡이 존재한다. 하둡은 클러스터 환경에 기반한 우수한 확장성, 장애 복구 기능 및 사용자가 기능을 정의할 수 있는 맵리듀스 프레임워크 등을 지원한다. 따라서 빅데이터를 보유하고 있는 기업 및 연구 기관에서 표준 플랫폼으로 사용되고 있다. 또한 하둡은 개인정보, 위치 데이터 등 민감한 정보를 보호하기 위해 Kerberos를 통한 사용자 인증 기법을 제공하고 있으며, 데이터 노드에서 HDFS로 결과를 전송하는 도중의 공격을 방지하기 위해 HDFS 데이터 압축 코덱 기능을 이용한, AES 암호화를 지원한다[5][6]. 한편, AES는 미국표준기술연구소(NIST)에서 연방정보처리 표준으로 발표한 대칭키 기반 암호하 알고리즘으로 키 값과 라운드 수가 블록 크기에 따라 가변적인 알고리즘으로 높은 안정성을 제공하고 성능의 요구에 따라 유연하게 사용할 수 있는 특징을 지니고 있다.

2.2 ARIA 알고리즘

ARIA 알고리즘[7][8]은 국가보안기술연구소 주도로 학계, 연구소, 정부기관 등의 암호 기술 전문가들이 개발한 국내 기술 암호화 알고리즘으로써, 128비트의 데이터 블록을 처리하고 AES와 동일한 128/192/256비트의 암호화 키를 사용한다. 대부분의 연산은 XOR과 같은 바이트 단위 연산으로 구성되어 있으며, 키 크기에 따라 12/14/16 라운드로 암호화를 수행한다. ARIA의 암호화 및 복호화 과정은 (그림 1)과 같다.

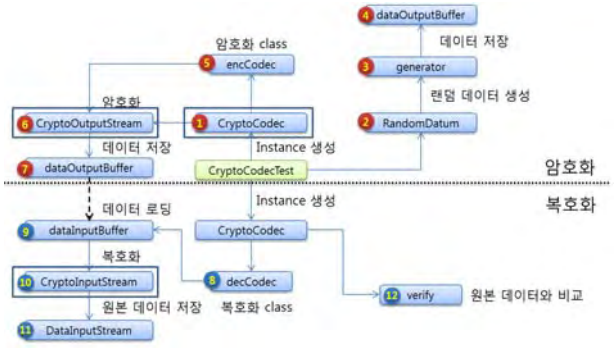


(그림 1) ARIA 암호화 및 복호화 과정

3. 제안하는 ARIA 암호화 기법의 설계 및 구현

3.1. 기존 하둡 기반 AES 코덱 분석

클라우데라(Cloudera) 및 하둡에서 제공하는 암호화 테스트 클래스인 TestCryptoCodec 클래스에서 HDFS 데이터를 암호화 및 복호화 과정 및 호출 클래스는 (그림 2)와 같다.



(그림 2) TestCryptoCodec의 HDFS 데이터 암/복호화

TestCryptoCodec 클래스를 통해 암호화된 HDFS 데이터 상에서 하둡 맵리듀스가 수행되는 과정을 분석하였다. 각 과정은 다음과 같다. i) 맵 함수를 실행하기 전에 입력 데이터로 들어온 암호화된 데이터를 복호화한다. ii) 맵 함수를 생성하여 데이터를 각 노드에 분배한다. iii) 맵이 종료되면 맵 출력으로 나온 결과를 암호화하여, HDFS에 저장한다. iv) 리듀스 함수가 실행되면 리듀스의 입력 데이터로 들어오는 암호화된 데이터를 복호화하여 리듀스 함수를 실행한다. v) 마지막으로 리듀스 함수가 끝나면, 최종 결과 데이터를 암호화하여 HDFS에 저장한다.

한편, HDFS Crypto 코덱 클래스는 사용자로부터 주어지는 암호화 코덱 클래스를 적용받아 암호화 및 복호화를 수행하는 클래스이며, 그 구성은 다음과 같다. i) 암호화 키 관리를 위한 클래스인 KeyProviderCryptoExtension 클래스, ii) 하둡에서 기본적으로 제공하는 AES 암호화 코덱 클래스인 JceAesCrypto Codec 클래스가 존재한다. HDFS Crypto 코덱 클래스는 최종적으로 AESCodec 클래스를 호출하여, HDFS 데이터에 대해 AES 기반 암호화 및 복호화를 수행한다. 암호화를 수행할 경우, AESCodec 클래스에서 Encryptor 클래스를 호출하고, 해당 클래스에서 cipher.writer() 메소드를 호출하여 암호화를 수행한다. 복호화를 수행할 경우, Decryptor 클래스를 호출하고, 해당 클래스에서 cipher.update() 메소드를 호출하여 복호화를 수행한다. Cipher 클래스는 JAVA에서 기본적으로 제공하는 라이브러리로 초기화 타입에 따라 AES, DBD, DES RSA 등 여러 암호화 알고리즘 중 하나를 선택하여 암호화, 복호화를 수행한다.

3.2. ARIA 암호화 기법의 설계

하둡 상에서 ARIA 알고리즘을 이용한 HDFS 암호화 둡의 AES 암호화 코덱을 기반으로 해당 코덱과 유사하게

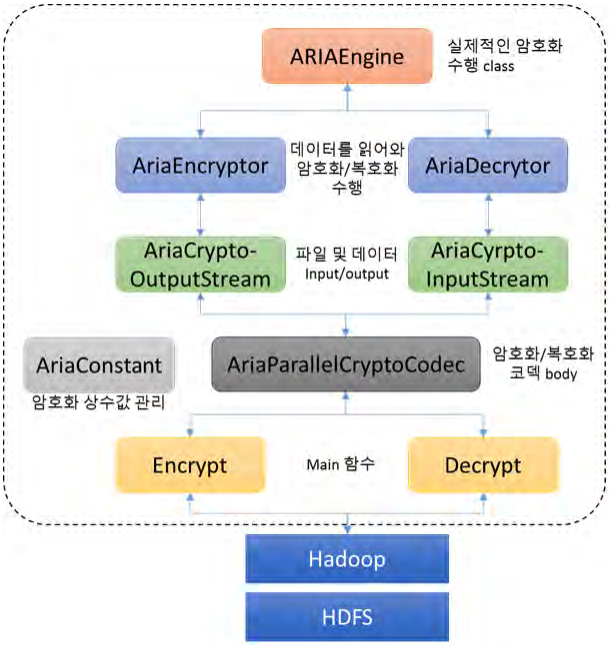
ARIA 코덱을 변경하는 방법이다. 이 방법은 Cipher 클래스에서 AES 호출 부분을 ARIA의 API로 변경한다. ARIA는 AES와 동일한 블록, 동일한 암호화 키 비트를 사용하기 때문에 AES 코덱을 호출하는 부분만 ARIA 코덱을 호출하는 것으로 교체가 가능하다. 이는 AES 코덱과 유사하기 때문에 개발이 용이하다는 장점이 존재한다. 둘째, 기존 AES 코덱을 기반으로 하는 것이 아닌 하둡 기반의 새로운 ARIA 암호화 코덱을 개발하는 것이다. 이는 AES코덱과 유사하게 개발하는 것이 불가능할 때 사용하는 방법이며, 개발비용이 증가한다.

두 가지 설계방안 중 하나의 설계방안을 선택하여 상세 설계를 수행하기 위해 실험을 수행하였다. 기존의 AES 코덱 기반의 암호화, 복호화 함수를 ARIA 코덱 기반의 암호화, 복호화 함수로 교체하여 텍스트파일에 대한 암호화, 복호화를 수행한 결과는 (그림 3)에서 확인하였다.

```
For key size of 256 bits, starting with the zero pla
buf len : 1048576, buf off 0
plaintext : 0a202020 20202020 20202020 20202020
ciphertext: 9f8ca4b3 3fb24eac d668eb59 67abe76e
buf len : 1048576, buf off 0
plaintext : 6e6f740a 20202020 20202020 20207065
ciphertext: 11d7fbf5 5b9a540c 32531ea3 96145cec
buf len : 1048576, buf off 0
plaintext : 64697469 6f6e7320 616e6420 74686520
ciphertext: bf864210 76c0c361 0578c90c cab04c71
buf len : 1048576, buf off 0
plaintext : 6f6e2069 6e636f72 706f7261 74656420
ciphertext: 6cb54615 29df1b85 896e2878 35510f91
buf len : 1048576, buf off 0
plaintext : 74792061 72636869 7665732e 0a0a2020
ciphertext: fdcf3074 f8f3d419 a3b47d39 9d101212
buf len : 1048576, buf off 0
```

(그림 3) ARIA 암호화 샘플 테스트 실행 결과

이 테스트를 통해, 첫번째 설계방안에 문제가 없다는 것을 판단하였다. 이에 따라 제안하는 ARIA 기반 HDFS 암호화 기법은 하둡에서 기존 AES 코덱을 기반으로 AES 암호화를 수행할 수 있도록 하는 CipherBuilder 클래스를 수정/확장하여 ARIA 암호화를 수행하도록 설계하였다. 제안하는 ARIA 암호화 기법의 상세 설계는 (그림 4)와 같다. 암호화 및 복호화 코덱 클래스를 호출하는 AriaParallelCryptoCodec이 존재하고 암호화 및 복호화를 위해 암호화된 파일 또는 데이터를 입/출력하는 AriaCryptoOutputStream과 AriaCryptoInputStream 클래스가 존재한다. 해당 스트림 클래스는 데이터를 읽어와 각각 AriaEncryptor, AriaDecryptor 클래스를 호출하여 암호화 및 복호화를 수행한다. 각 클래스는 내부에서 AriaEngine 클래스를 호출하여 ARIA 암호화를 수행한다. 한편, 암호화 키 관리를 포함한 상수값을 관리하는 클래스는 AriaConstant 클래스이다.



(그림 4) 제안하는 암호화 기법의 구조

3.3. ARIA 암호화 기법의 구현

원본 파일을 이용해 하둡 상에서 ARIA 알고리즘을 이용하여 HDFS 데이터를 암호화 한 후, 맵리듀스 응용을 실행하는 방법은 다음과 같다. 첫째, 하둡이 구현한 코덱을 이용하여 암호화를 수행하도록 설정한다. 둘째, 구현한 ARIA 암호화 코덱이 포함되어 있는 JAR 파일을 실행하여, 응용의 입력 파일을 ARIA 암호화를 수행한 후 HDFS에 삽입한다. 삽입된 암호화 데이터는 (그림 5)와 같이 .aira 확장자를 갖는다. 셋째, 하둡 응용을 실행한다. 이때, -libjars 파라메터를 이용하여 ARIA 압축 코덱을 라이브러리로 지정한다. 마지막으로, 실행 후 결과 데이터는 ARIA로 암호화 되어 (그림 6)과 같이 .aira 확장자로 출력된다.

```
lsr: DEPRECATED: Please use 'ls -R' instead.
15/09/13 23:32:18 WARN util.NativeCodeLoader: Unable to load native-ha
-rw-r--r--   3 dblab supergroup  107035964 2015-09-13 23:30 /test.aria
```

(그림 5) HDFS에 삽입된 /test.haes 파일

```
15/09/13 23:37:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
drwxr-xr-x   - dblab supergroup          0 2015-09-13 23:36 /home
drwxr-xr-x   - dblab supergroup          0 2015-09-13 23:36 /home/dblab
-rw-r--r--   3 dblab supergroup      32699 2015-09-13 23:36 /home/dblab/CNUenc.jar
drwxr-xr-x   - dblab supergroup          0 2015-09-13 23:37 /output
drwxr-xr-x   - dblab supergroup          0 2015-09-13 23:36 /output/_SUCCESS
-rw-r--r--   3 dblab supergroup    1048576 2015-09-13 23:37 /output/part-r-00000.aria
-rw-r--r--   3 dblab supergroup  107035964 2015-09-13 23:30 /test.aria
```

(그림 6) 실행 후 출력 데이터

4. 성능평가

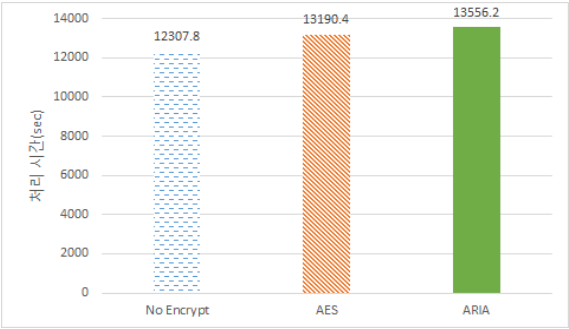
본 논문에서 제안하는 하둡 상에서 ARIA 알고리즘을 이용한 HDFS 데이터 암호화 기법에 대한 성능평가는 두 개의 맵리듀스 응용을 실행하여 측정하였다. 첫째, 텍스트 파일의 단어 수를 계산하는 단어계산(WordCount) 응용에 대한 성능을 측정하였다. 둘째, 텍스트파일에서 단어를 사전 순으로 정렬하는 정렬 알고리즘에 대한 성능평가를 수

행하였다.. 비교 대상은 No Encrypt, AES 암호화 알고리즘, ARIA 암호화 알고리즘이다. 실험 환경은 <표 1>과 같다.

<표 1> 실험 환경

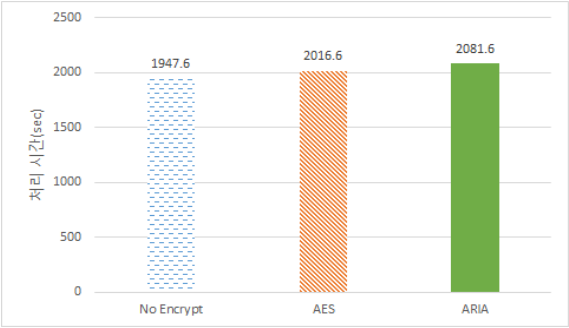
	성능	
CPU	Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz	
RAM	12GB	
응용	데이터	크기
단어계산	위키피디아 영문 07/25 덤프	86GB
정렬	메타위키 08/10 토막글 덤프	6.45GB

단어계산 응용의 성능평가 결과는 (그림 7)과 같다. 암호화를 하지 않은 상태인 NoEncrypt는 평균 12307.8초의 성능을 보였고, AES의 경우 암호화를 하지 않았을 경우보다 약 7.17% 느린 약 13190.4초의 성능을 보였다. 마지막으로 본 논문에서 제안하는 ARIA 알고리즘은 13556.2초의 성능을 보여 암호화를 하지 않았을 경우보다 약 10.14%, AES 암호화를 했을 경우보다 약 2.77% 느린 성능을 보였다.



(그림 7) 단어계산 응용의 성능평가

정렬 알고리즘의 성능평가 결과는 (그림 8)과 같다. 암호화를 하지 않은 상태인 No Encrypt는 평균 1947.6초의 성능을 보였고, AES의 경우 평균 2016.6초의 성능을 보여 암호화를 하지 않았을 경우보다 약 3.54% 느린 성능을 보였다. 마지막으로 제안하는 AIRA 알고리즘의 경우 평균 2081.6초의 성능을 보여 암호화를 하지 않았을 경우보다 약 6.88%, AES 암호화를 했을 경우보다 약 3.24% 느린 성능을 보였다.



(그림 8) 정렬 알고리즘의 성능평가

ARIA 알고리즘으로 암호화를 했을 경우, 원본 데이터만의 어플리케이션보다 10% 가량 느린 처리 성능을 보이지만, HDFS 데이터가 암호화되어 저장되기 때문에 데이터 유출을 방지하는 장점이 존재한다. 아울러 AES 암호화 알고리즘은 미국의 Intel사의 성능 최적화 및 암호화 기법 자체의 꾸준한 성능 개선으로 인해 현존 암호화 알고리즘 중 제일 빠른 것으로 평가되었다. 이에 비해 ARIA 알고리즘은 약 3% 가량의 성능 하락만으로 국내에서 권유한 데이터 암호화가 가능하단 점에서 개발 의의가 있다.

5. 결론 및 향후 연구

현재 하둡은 대용량 데이터 분산 병렬 처리 기법의 표준으로써 인정받게 되어 널리 사용되고 있다. 따라서 개인정보, 위치 데이터 등 민감한 정보 또한 하둡에서 처리하고 있기 때문에 하둡의 보안에 대한 연구가 활발히 진행되고 있다. 현재 하둡은 Kerberos 기반의 사용자 인증 기법을 통한 네트워크 보안을 지원하고, AES 등의 암호화 코덱을 활용한 HDFS 데이터 보안 기능을 제공하고 있다.

한편 정부에서는 국가 기관에서 자체 개발한 ARIA 기법을 적용한 소프트웨어를 권장함으로써, 하둡에 ARIA를 적용한 데이터 암호화 기법의 개발이 필요하였다. 따라서 본 논문에서는 하둡의 AES 암호화 코덱 클래스를 분석하고, ARIA 알고리즘을 적용한 HDFS 암호화 기법을 개발하였다. 이를 통해 하둡에서 HDFS 데이터를 ARIA 암호화 코덱을 기반으로 암호화 및 복호화를 수행할 수 있다. 아울러 단어계산 응용, 정렬알고리즘의 성능평가를 통해, AES 수준의 오버헤드만으로 빅데이터에 대한 암호화를 수행할 수 있음을 증명하였다.

향후 연구로는 산술 데이터 처리에 관한 성능평가를 수행하고 AES와 동일한 성능을 보이도록 성능을 개선하는 것이다.

참고문헌

[1] Jeffrey Dean, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[2] Apache Software Foundation, Apache Lucene: http://lucene.apache.org/.

[3] http://www.open-mpi.org/

[4] http://hortonworks.com/hadoop/tez/

[5] https://issues.apache.org/jira/secure/attachment/12571116/ Hadoop%20Crypto%20Design.pdf

[6] 박선영, 이영석. "HDFS 암호화 성능 분석." 정보과학회논문지: 데이타베이스 41.1 (2014): 21-27.

[7] Block Encryption Algorithm ARIA http://glukjeoluk.tistory.com/attachment/ok110000000002.pdf

[8] ARIA Algorithm Specification, May. 2004, available at:http://www.nsri.re.kr/ARIA/doc/ARIA-specification.pdf