GF(2^m)상의 타원곡선 암호 시스템의 객체지향적 설계 및 구현

이항록* ,류희수
한국 전자통신연구원
neoguass@etri.re.kr*, hsryu@etri.re.kr

Implementation ECC algorithm over GF(2^m) with Java

HangRok Lee*, Heuisu Ryu
Information Security Basic Research Team, ETRI

요 약

본 논문에서는 디자인패턴 개념을 이용하여 GF(2^m)상에서의 타원곡선 암호알고리즘을 객체지향적으로 설계하는 방법에 대해서 논해보고, 이를 이용하여 타원곡선 암호 라이브러리 구현에 핵심이 되는 연산 클래스에 대한 전체적인 framework 및 UML을 제시한다.

1. 서 론

자바(JAVA) 언어는 자바언어가 가지고 있는 장점 및 객체지향적언어라는 특성 때문에 웹 솔루션이나, 또는 광범위 서버시스템에서부터 작은 스마트카드에 이르기까지 어플리케이션의 구현에 이용되고 있다. 또한 현재 이런 다양한 시스템에서 필수요구사항으로는 보안 개념을 적용한 시스템의 개발을 필요로 하고 있다. 이렇게 자바를 이용하여 보안 시스템의 구현은 여러 업체에서 이루어지고 있는 반면에 기본적으로 Java Crypto API에서 지원해주고 있는 암호 알고리즘들이 제한되어 있다는 것이며, 그로 인해 자바로 개발된 많은 보안 시스템에서는 지원되는 몇 가지 알고리즘만을 이용하여 구현하거나 또는 필요로 하는 암호 알고리즘은 C 오픈 소스를 이용하고 이를 java에서 native method로써 호출하여 사용하는 등의 비합리적인 방법으로 시스템을 구현하고 있다. 따라서 자바를 이용하여 구현되어지는 여러 보안 시스템의 발전을 위해서는 최신의 암호 알고리즘 및 국산 암호 알고리즘들을 발 빠르게 자바 Crypto API로써 포함하여 널리 사용하게 하는 것이며 이를 위해서는 최신의 암호 알고리즘에 대한 Crytpo API 개발이 필요하다고 생각된다. 따라서 본 논문에서는 일반 Java Crypto API 에는 포함되어 있지 않은 타원곡선 암호 알고리즘(ECC)에 대한 기본적인 연산 클래스에대한 객체지향적 설계 및 구현하는 방법에 대해서 제시하고자 한다. 본 논문의 구성은 2장에서는 타원곡선 암호 알고리즘에 대해서 설명할 것이며 3장에서는 이 타원곡선 암호 알고리즘을 UML을 이용하여 객체지향적으로 구현하는 방법에 대해서 설명할 것이다. 4장은 결론으로 암호 연산 클래스로써의 적합성 및 알고리즘 performance에 대해서 논한다.

2. 타원곡선 암호법에 대한 개요

타원곡선 암호 시스템(ECC)은 유한체상에서 정의된 타원곡선 이산대수 문제의 어려움(ECDLP)에 기반한 공개키 암호 시스템이다. ECC 알고리즘은 RSA 알고르즘에 비해서 키생성이 매우 빠르고 키bit 길이가 매우 짧다는 장점을 가지고있기 때문에 현재 여러 모바일 환경과 같은 자원이 제한적인 환경에 적절한 보안 알고리즘으로써 채택이 되고있는 실정이다. 특히 타원곡선 암호 시스템은 여러가지 유한체 상에서 구현되어질 수 있다. 타원곡선 구현에 사용되는 유한체는 다음과 같다.

- GF(p) prime field
- GF(2^m) binary field
- GF(p^m) OEFs (p 는 mersenne prime)

이중에서 GF(2^m) binary field가 현재 타원곡선 구현에 가장 널리 사용되고 있는 유한체이며, 특히 binary field는 사용되는 기저(basis)에 따라서 polynomial basis 와 normal basis로 일반적으로 나뉜다. normal basis는 타원곡선 암호시스템의 하드웨어 구현에 많이 사용되며, polynomial basis는 소프트웨어로의 구현에 주로 사용되고 있다. 본 논문에서는 sec2 및 wtls 문서에 권고 타원곡선으로 제시되어있는 것들 중에서 polynomial basis를 사용하는 sect163r1, sect163r2, wtls_5_curve를 구현하였다. $f(x)$를 m차 irreducible polynomial(기약 다항식) 이라 하고, 두 원소 $\alpha, \beta \in$ GF(2^m) 이라고 하자. 타원곡선 연산 구현에 필요한 GF(2^m)유한체 연산들은 다음과 같다.

- Addition,Subtraction 연산
  $\alpha + \beta = \alpha \oplus \beta$ (bitwise xor)
- Multiplication & Reduction 연산(LR-comb width 4)
  $\alpha * \beta \equiv \gamma \bmod f(x)$ ($\gamma \in$GF(2^m))
- Square 연산 (precomputation 이용하는 방법 사용)
  $\alpha^2 \equiv \gamma \bmod f(x)$ ($\gamma \in$GF(2^m))
- Inversion 연산(EEA 또는 MAIA inversion 사용)
  $\alpha * \alpha^{-1} \equiv 1 \bmod f(x)$ ($\alpha^{-1} \in$GF(2^m))

위와 같이 네가지 유한체 연산을 구현해야하며 Multiplication,

Square, Reduction, Inversion 연산은 [1]을 참조하여 가장 빠른 알고리즘을 이용하여 구현해야 하며. 특히 reduction 연산은 $f(x)$ 가 pentinomial 또는 trinomial 인 경우에 빠른 구현이 가능하므로 이를 이용하여 빠른 reduction 연산을 구현해야 전체적은 performance 가 향상된다. 위의 유한체 연산을 이용한 타원곡선 연산은 다음과 같다.

$$E = \{(x,y) \mid y^2 + xy = x^3 + ax + b\} \cup O$$

(단 $x, y, a, b \in$ GF(2^m) & O는 infinity point)

- 타원곡선 Addition 연산 (projective 좌표계 사용)
  $P + Q = R$ ($P, Q, R \in E$ 타원곡선 위의 점)
- 타원곡선 Doubling 연산
  $2P = R$ (projective 좌표계 사용)
- 타원곡선 Scalar Multiplication 연산
  $kP = R$ (projective 좌표계 사용)
- 타원곡선 좌표계의 상호 변화 연산
  affine 좌표계 ←→ projective 좌표

특히 타원곡선 연산 구현상에있어서 affile 좌표계를 사용하여 Addition 연산과 Doubling연산을 구현하게 되면 연산 중간에 유한체 inversion 연산을 사용되므로 전체적인 performance 가 떨어진다. 따라서 projective 좌표계를 이용하여 구현하면 Addition 연산과 Doubling연산이 단순히 유한체 square 연산과 multiplication 만으로 구현되어질 수 있으므로 빠른 연산 속도를 낼 수 있다. 또한 Scalar Multiplication 연산은 일반적인 연산에는 Sliding Window method 또는 Montgomery method를 이용하여 계산하며, 키생성시에는 Fixed Based Comb method를 이용한다. 다음으로 타원곡선 암호 알고리즘들 중에서 ECDHC 과 ECDSA 대해서 간략하게 소개 하겠다.

- ECDHC 키교환 알고리즘 (서로 같은 타원곡선 선택)

A		B(b 개인키)
aG 전송	⟶	aG 수신
bG 수신	⟵	bG 전송
$habG$ 계산		$habG$ 계산

<a는 A의 개인키, b는 B의 개인키이며
h는 cofactor, G는 타원곡선의 Base Point이다.>

- ECDSA 서명 알고리즘 (서로 같은 타원곡선 선택)
  A의 서명 생성과정(a 개인키, $P_A = aG$ 공개키)
  1. random 값 k 생성 ($1 \le k \le n-1$)
  2. $kG = (x_1, y_1)$ 계산 $r = x_1 \bmod n$
  3. $k^{-1} \bmod n$ 계산
  4. $s = k^{-1}(e + ar)$ 계산
  5. 만약 $s = 0$ 이면 1.부터 다시 계산.

  <ECDSA 서명 검증과정 [2]ANSI X9.62참조>

3. 타원곡선 Crypto API 설계 및 구현

SUN 사에서 제공되어지는 jdk1.4 에 포함되어있는 공개키 알고리즘으로는 RSA 와 DSA알고리즘이다. 그나마 jdk1.3에서는 포함되어있지 않았던 RSA알고리즘이 default 알고리즘으로 제공되어지고 있기는 하나 현재 WPKI 시스템이나 모바일 환경에서 사용되어지고 있는 ECC 알고리즘에 관해서는 key interface 조차 수출제한으로 포함되어있지 않은 실정이다.

본 논문에서는 sec2 및 wtls 문서에 권고 타원곡선으로 제시되어있는 것들 중에서 모바일 환경에서 이용될 가능성이 높은 sect163r1 , sect163r2 , wtls5 커브를 구현하였다. 자바로 타원곡선 알고리즘을 구현하려면 먼저 타원곡선 암호 알고리즘별 Key 클래스 및 Key Spec클래스와 그리고 크게 네 가지의 연산 클래스 및 이러한 연산클래스를 중간에서 조율하는 provider class(직접적인 JCA Api 및 Spi클래스들을 상속 받아 구현해야 하는 클래스)들로 분류하여 구현할 수 있다. 이렇게 연산 클래스와 프로바이더 클래스를 분리하여 구현하는 이유는 각 클래스들의 기능을 세분화 및 분류화 함으로써 각 기능들의 행동모델에 기반한 객체지향적인 설계를 위한 것이며 또한 연산 클래스의 유지보수를 쉽게하기 위함이다. 먼저 연산 클래스는 아래와 같이 분류할 수 있다.

- GF2m(abstract) , GF163 , GF163_2 유한체 연산 클래스
- EC2m(interface), ECArithemtic(abstract), EC163, EC163_2 타원곡선 연산 클래스
- CommonKeyPairGenerator(interface), ECKCDSAKeyPairGenerator, ECKeyPairGenerator 알고리즘별 키 생성 연산 클래스
- CommonKeyAgreement(키 교환에 관련된 interface) ECDHCKeyAgreement, ECMQVKeyAgreement 클래스, CommonSignature(서명에 관련된 interface) ECDSASignature, ECKCDSASignature 서명 클래스

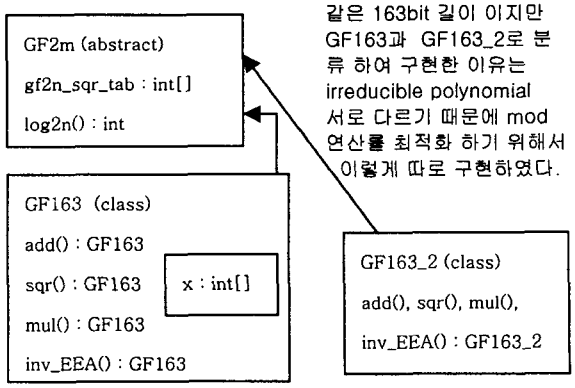
그리고 특히 여러 타원곡선 연산클래스에서 데이터(타원곡선 포인트) 저장에 사용되는 클래스(클래스 내부에는 int[] 에대한 참조변수 x,y,z만 선언되어있음) 및 여러 암호 연산 클래스에서 사용되는 타원곡선 파라메터를 담는데 사용되는 클래스는 다음과 같다.

- ECPointGF2m, EllipticCurveParam 클래스

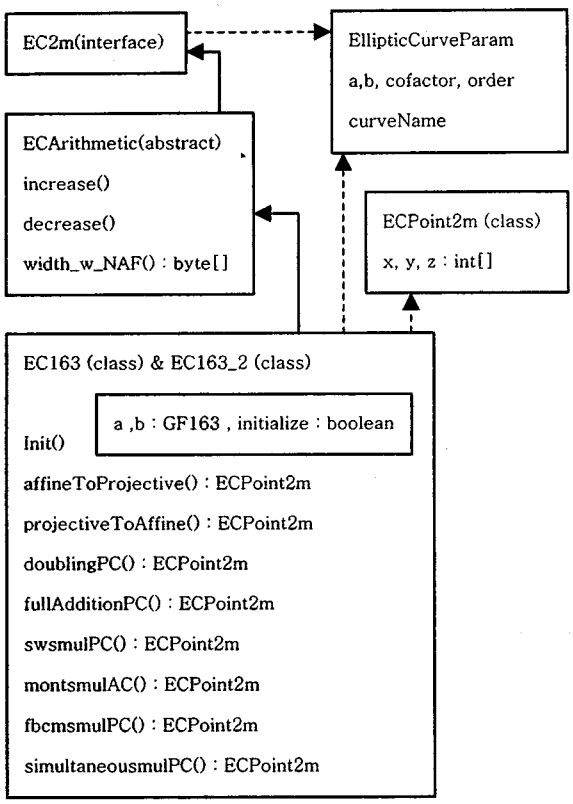
그리고 각 타원곡선 암호 알고리즘에 따른 키 및 구현상의 편리함을 위해서 키과 파라메터를 함께 갖고 있는 클래스는 다음과 같다.

- ETRICipherParam(interface), AsymKeyParam
- AsymKeyPair(interface), ECKeyGenerationParam
- AsymKeyParam, ECKeyParam, ECPublicKeyParam, ECPrivateKeyParam, ECMQVPrivateKeyParam, ECMQVPublicKeyParam 클래스

이러한 기본연산 클래스들 사이의 UML은 다음과 같다. 다음은 같은 163bit 길이 이지만 GF163과 GF163_2로 분류 하여 구현한 이유는 irreducible polynomial 서로 다르기 때문에 mod 연산를 최적화 하기 위해서 이렇게 따로 구현하였다.

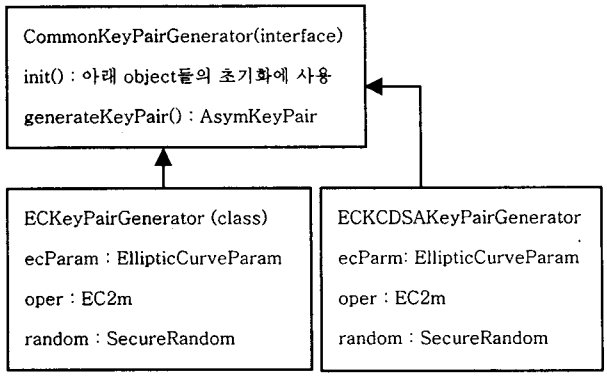


타원곡선 연산 클래스들에 대한 UML은 다음과 같다.

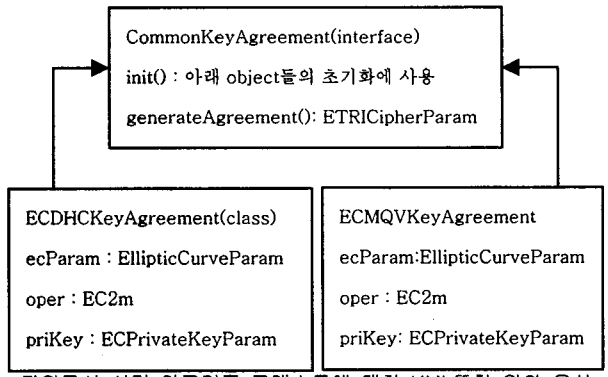


위에서 보는 것과 같이 EC2m이라는 interface를 선언함으로써 하나의 암호 알고리즘 연산 클래스에서 임의의 타원곡선 연산 객체를 호출해서 사용하기 위함이다. 즉 알고리즘 클래스들은 EC2m 형의 참조 변수만을 선언해 놓고서 필요에 따른 적당한 타원곡선 연산 클래스(EC163 & EC163_2)의 객체를 할당해주면 된다. 단 EC2m형 참조변수를 이용하여 EC163 과 EC163_2 클래스내에 존재하는 8개의 연산 메소드를 사용하기 위해서 EC2m interface 내에는 위 8개의 연산 메소드들에 대한 abstract method 형태로 prototype을 선언해줘야 한다. 이러한 자바의 다형성을 이용하여 구현한다면 각각의 타원곡선 연산 클래스에 따른 여러 개의 같은 형태의 암호 알고리즘 클래스들을 만들 필요 없이 하나의 암호 알고리즘 클래스의 구현 만으로도 여러 타원곡선 클래스를 이용할 수 있다는 장점이 있다. 그리고 여기서 타원곡선 연산 클래스를 이러한 다형성 특성을 이용하여 하나의 타원곡선 연산 클래스로 구현하지 않고 이렇게 EC163과 EC163_2로 분리하여 구현한 이유는 서로 다른 타원곡선 계수를 사용하여 fbcmsmulPC()이 메소드를 구현해야 하기 때문에 서로 다른 precomputation point들을 저장해야 하며 또한 타원곡선 계수에 따라서 연산의 구현을 간략화 할 수 있기 때문에 전체적인 연산의 performance를 위해서 타원곡선 별로 따로 클래스를 나누어서 각 타원곡선에 최적화한 연산을 구현한 것이다. 다음은 타원곡선 암호 알고리즘 연산 클래스들에 대한 UML이다.

<아래는 키 생성 연산 클래스들에 대한 UML>



<아래는 키 교환 알고리즘연산 클래스들에 대한 UML>



타원곡선 서명 알고리즘 클래스들에 대한 UML또한 위와 유사하여 여기서는 생략한다. 부록 참조

4. 결 론

본 논문에서는 타원곡선 암호 알고리즘을 Java언어를 이용하여 전반적인 타원곡선 암호 라이브러리에 필요한 핵심 연산 클래스들의 구조를 객체 지향적으로 설계하였으며 이에 대한 framework 을 제시 하였다. 논문에서 제시한 구조는 연산 클래스들이 암호 프로바이더 클래스들에 최대한 독립적으로 설계하였기 때문에 이러한 구조에 맞추어서 타원곡선 암호 알고리즘 연산 클래스을 구현한다면, JCA 및 JCE를 확장 구현한 임의의 암호 provider에 대해서도 타원곡선에 관련된 암호 프로바이더 부분만을 추가 구현해준다면 타원곡선 연산 클래스들을 쉽게 그 타원곡선 암호 부분에 붙여 넣을 수가 있다. 또한 실체 구현 performance에 있어서도 큰 문제가 되지 않는다. 예를 들어서 실제 구현 결과에 따른 테스트 결과는 P3 800Mhz, RAM 256, Windows 2000 운영체제하에서 1000의 키 생성에 소요된 시간은 대략 3.5초에 즉 3.5ms에 지나지 않으며, 전체 연산 클래스만을 jar 파일로 묶었을 경우의 크기가 107KB로써 상당히 lite하다. 따라서 모바일 환경에서도 충분이 적용 가능한 타원곡선 암호 알고리즘 클래스들에 대한 framework 라고 할 수 있다.

Reference

[1] Software implementation of Elliptic Curve over GF(2^m)
[2] ANSI X9.62 & X9.63
[3] SEC 2 : certicom
[4] 국내 인증서 기반 타원곡선 전자 서명 알고리즘: TTA
[5] Java Security : 정보 문화사 2001.11.27
[6] 자바 개발자를 위한 UML contact J : 대림