

 (19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2014-0055423 (43) 공개일자 2014년05월09일
(51) 국제특허분류(Int. Cl.) <i>H04L 9/30</i> (2006.01) <i>H04L 9/28</i> (2006.01) (21) 출원번호 10-2012-0122317 (22) 출원일자 2012년10월31일 심사청구일자 2012년12월04일	(71) 출원인 고려대학교 산학협력단 서울특별시 성북구 안암로 145, 고려대학교 (안암동5가) (72) 발명자 홍석희 서울 도봉구 마들로 859-19, 103동 107호 (도봉동, 도봉한신아파트) 김현민 서울 성북구 길음로 118, 414동 1801호 (길음동, 대림e편한세상아파트) 권태웅 서울 중구 퇴계로88가길 48-26, (신당동) (74) 대리인 특허법인충현

전체 청구항 수 : 총 12 항

(54) 발명의 명칭 공개키 암호화 시스템 및 그 방법

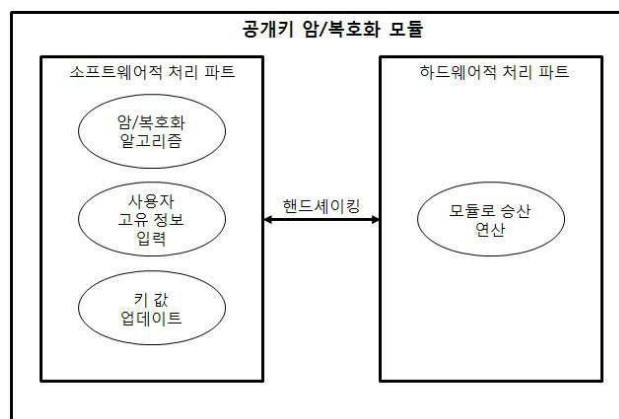
(57) 요약

본 발명은 스마트카드에서 요구되는 암호/복호화를 하드웨어적인 처리와 소프트웨어적인 처리를 통해 같이 구현함으로써 암호/복호화 처리에 소요되는 시간의 감소 및 보안성 확보를 동시에 도모할 수 있는 스마트카드에서의 암호화 방법에 관한 것이다.

개시되는 암호화 방법은 (a)공개키 암호화에 요구되는 키(key) 값의 업데이트와 사용자 고유 정보를 입력하기 위한 외부 인터페이스 및 상기 암호화의 구현을 위한 암호/복호화 알고리즘을 소프트웨어적인 처리로 구현하는 단계; 및 (b)상기 암호화 알고리즘에 포함된 모듈로 승산 연산(modulo multiplication operation)은 하드웨어적인 처리로 구현하는 단계를 포함한다.

본 발명은 소프트웨어적인 처리가 요구되며 아울러 소프트웨어적인 처리만으로도 성능(속도)에 지장이 없고 하드웨어적인 처리와는 무관한 키 값의 업데이트(저장) 및 암호/복호화 알고리즘은 소프트웨어적인 처리로 구현하고, 성능(속도)과 직접 관련되고 소프트웨어적인 처리와는 무관한 모듈로 승산 연산은 하드웨어적인 처리로 구현하여 두 처리 간 핸드셰이킹(handshaking)을 통해 스마트카드에 있어서의 암호/복호화 모듈을 구현하기 때문에 스마트카드에 있어서의 중요한 두 요소인 보안성 확보와 속도(성능)의 향상을 동시에 도모할 수 있는 이점을 갖는다.

대표도 - 도1



특허청구의 범위

청구항 1

(a)공개키 암호화에 요구되는 키(key) 값의 업데이트와 사용자 고유 정보를 입력하기 위한 외부 인터페이스 및 상기 암호화의 구현을 위한 암호/복호화 알고리즘을 소프트웨어적인 처리로 구현하는 단계; 및

(b)상기 알고리즘에 포함된 모듈로 승산 연산(modulo multiplication operation)은 하드웨어적인 처리로 구현하는 단계를 포함하는 것을 특징으로 하는 공개키 암호화 방법.

청구항 2

제 1 항에 있어서,

상기 암호화는 RSA(Rivest, Shamir, Adleman) 암호화 방식에 의한 것임을 특징으로 하는 공개키 암호화 방법.

청구항 3

제 1 항에 있어서,

상기 모듈로 승산 연산은 몽고메리 승산 연산(Montgomery multiplication operation)인 것을 특징으로 하는 공개키 암호화 방법.

청구항 4

제 1 항에 있어서,

상기 알고리즘을 상기 소프트웨어적인 처리에 의해 컴퓨터 프로그래밍 언어로 구현하는 경우, 상기 언어는 어셈블리어(assembly language)인 것을 특징으로 하는 공개키 암호화 방법.

청구항 5

제 1 항에 있어서,

상기 알고리즘의 소프트웨어적인 처리에 의한 프로그래밍은 loop unrolling 기법을 적용함을 특징으로 하는 공개키 암호화 방법.

청구항 6

제 3 항에 있어서, 상기 몽고메리 승산 연산에 의한 최종 결과 값(S_n)을 구하는 것은 n-비트 승수(multiplier)의 i번째 비트의 값(a_i) 또는 [i-1] 번째 비트에 대한 중간 결과 값(S_{i-1})의 최하위 비트(Least Significant Bit: LSB)의 값(s_0)이 0일 때에는 0이 아닌 이전 비트(prior bit)에 대한 연산 과정에서 얻어진 값을 그대로 사용하여 이루어지는 것을 특징으로 하는 공개키 암호화 방법.

청구항 7

공개키 암호화에 요구되는 키(key) 값의 업데이트와 사용자 고유 정보를 입력하기 위한 외부 인터페이스 및 상기 암호화의 구현을 위한 암호/복호화 알고리즘을 소프트웨어적인 처리로 구현하고, 상기 암호화 알고리즘에 포함된 모듈로 승산 연산(modulo multiplication operation)은 하드웨어적인 처리로 구현한 공개키 암호화 시스템.

청구항 8

제 7 항에 있어서,

상기 암호화는 RSA(Rivest, Shamir, Adleman) 암호화 방식에 의한 것임을 특징으로 하는 공개키 암호화 시스템.

청구항 9

제 7 항에 있어서,

상기 모듈로 승산 연산은 몽고메리 승산 연산(Montgomery multiplication operation)인 것을 특징으로 하는 공개키 암호화 시스템.

청구항 10

제 7 항에 있어서,

상기 알고리즘을 상기 소프트웨어적인 처리에 의해 컴퓨터 프로그래밍 언어로 구현하는 경우, 상기 언어는 어셈블리어(assembly language)인 것을 특징으로 하는 공개키 암호화 시스템.

청구항 11

제 7 항에 있어서,

상기 알고리즘의 소프트웨어적인 처리에 의한 프로그래밍은 loop unrolling 기법을 적용함을 특징으로 하는 공개키 암호화 시스템.

청구항 12

제 9 항에 있어서, 상기 몽고메리 승산 연산에 의한 최종 결과 값(S_n)을 구하는 것은 n-비트 승수(multiplier)의 i번째 비트의 값(a_i) 또는 [i-1] 번째 비트에 대한 중간 결과 값(S_{i-1})의 최하위 비트(Least Significant Bit: LSB)의 값(s_0)이 0일 때에는 0이 아닌 이전 비트(prior bit)에 대한 연산 과정에서 얻어진 값을 그대로 사용하여 이루어지는 것을 특징으로 하는 공개키 암호화 시스템.

명세서

기술분야

[0001] 본 발명은 공개키 암호화 시스템 및 그 방법에 관한 것으로, 보다 상세하게는 공개키 암호/복호화를 하드웨어적인 처리와 소프트웨어적인 처리를 통해 같이 구현함으로써 암호/복호화 처리에 소요되는 시간의 감소 및 보안성 확보를 동시에 도모할 수 있는 암호화 시스템 및 그 방법에 관한 것이다. 본 발명은 특히 스마트카드에서의 암호/복호화 모듈의 구현에 유용하도록 하기 위한 것이다.

배경기술

[0002] 오늘날 IT 산업이 급격히 발전하면서 전자기기는 점점 소형화될 뿐만 아니라 유비쿼터스(ubiquitous) 컴퓨팅 환경이 일반화되면서 모바일 스마트 기기의 사용도 일상화되었다. 이러한 환경에 수반하여 금융 거래 등도 개인 단말을 이용하여 인터넷을 통해 이루어지고 있으며, 모바일 어플리케이션을 통한 거래 또한 급격히 늘어나고 있다.

[0003] 한편 상기한 기능 등이 모바일 기기 등을 통해 이루어지는 것은 상기한 기능 등을 수행가능하게 하기 위한 '스마트카드(smart card)'가 기기에 삽입되어 이루어지고 있다. 그런데 스마트카드의 이용에 있어 특히 개인 정보의 외부 누출이 방지되도록 하는 보안성 확보가 전제되어야 하는데, 보안성 확보를 위한 솔루션(암호화 솔루션)에 관한 인식은 그 중요성에 비하여 많이 미비한 것이 실상이다. 그래서 아직까지는 스마트카드가 삽입되는 모바일 기기 같은 소형 전자 장비를 개발시에는 추가적인 비용이 많이 소요될 수밖에 없는 암호화 솔루션에 대한 신규 개발은 거의 도외시되고 있으며 기능(application) 및 성능(performance) 향상 측면에 더 많은 초점을 맞춰서 개발이 이루어지고 있다.

[0004] 그러나 이러한 장비의 사용이 증가할수록 스마트카드의 암호화 솔루션에 대한 중요성과 필요성은 점점 늘어날 수밖에 없음을 이미 충분히 경험하였으며(그간 스마트카드를 이용한 여러 응용에서 개인 정보의 누출 사고가 다수 발생), 따라서 스마트카드의 성능을 향상시킬 수 있음과 아울러 보안성의 확보도 동시에 도모할 수 있는 방안이 필요하다.

발명의 내용

해결하려는 과제

[0005] 본 발명은 상기와 같은 상황 및 필요성을 인식하여 창안된 것으로, 본 발명이 해결하고자 하는 과제는 스마트카

드를 이용시에 보안성을 확보할 수 있는 암호화 모듈 구현 방안을 제시함과 아울러, 이에 따라 야기될 수 있는 스마트카드의 성능 하락 문제에도 대처할 수 있는 스마트카드에서의 암호화 시스템 및 그 방법을 제시하는 것이다.

과제의 해결 수단

[0006] 상기와 같은 과제를 해결하기 위해 개시되는 암호화 방법은 (a)공개키 암호화에 요구되는 키(key) 값의 업데이트와 사용자 고유 정보를 입력하기 위한 외부 인터페이스 및 상기 암호화의 구현을 위한 암호/복호화 알고리즘을 소프트웨어적인 처리로 구현하는 단계; 및 (b)상기 암호화 알고리즘에 포함된 모듈로 승산 연산(modulo multiplication operation)은 하드웨어적인 처리로 구현하는 단계를 포함하여 상기한 과제를 해결한다.

[0007] 상기와 같은 과제를 해결하기 위해 개시되는 암호화 시스템은 공개키 암호화에 요구되는 키(key) 값의 업데이트와 사용자 고유 정보를 입력하기 위한 외부 인터페이스 및 상기 암호화의 구현을 위한 암호/복호화 알고리즘을 소프트웨어적인 처리로 구현하고, 상기 암호화 알고리즘에 포함된 모듈로 승산 연산은 하드웨어적인 처리로 구현하여 상기한 과제를 해결한다.

발명의 효과

[0008] 본 발명은 공개키 암호화 방식으로 암호/복호화 모듈이 구현되는 스마트카드에 있어서 소프트웨어적인 처리가 요구되며 아울러 소프트웨어적인 처리만으로도 성능(속도)에 지장이 없고 하드웨어적인 처리와는 무관한 키 값의 업데이트(저장) 및 암호/복호화 알고리즘은 소프트웨어적인 처리로 구현하고, 성능(속도)과 직접 관련되고 소프트웨어적인 처리와는 무관한 모듈로 승산 연산은 하드웨어적인 처리로 구현하여 두 처리 간 핸드셰이킹(handshaking)을 통해 스마트카드에 있어서의 암호/복호화 모듈을 구현하기 때문에 스마트카드에 있어서의 중요한 두 요소인 보안성 확보와 속도(성능)의 향상을 동시에 도모할 수 있는 이점을 갖는다.

도면의 간단한 설명

[0009] 도 1은 본 발명의 개념을 제시한 도면이다.

도 2는 본 발명의 흐름을 제시한 도면이다.

도 3은 본 발명에서의 몽고메리 승산 연산의 흐름을 제시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0010] 본 발명을 실시하기 위한 구체적인 내용의 설명에 앞서 이해의 편의를 위해 본 발명이 해결하려는 과제의 해결 방안의 개요를 우선 제시한다.

[0011] 스마트카드에서의 암호화에는 여러 가지 이유로 인해 비밀키 암호화 방식이 아닌 공개키 암호화 방식을 적용하는 것이 압도적이며, 공개키 암호화 방식에는 모듈로 승산 연산(modulo multiplication operation)이 반드시 포함된다. 공개키 암호화 방식은 공개키와 개인키에 대해 모듈로 멍승(modulo exponentiation)을 취하여 암호화 및 복호화가 이루어지며, 모듈로 멍승은 모듈로 승산을 기본 연산으로 하여 이루어지기 때문이다. 모듈로 승산 연산은 연속된 덧셈 연산으로부터 수행될 수 있으며, 연산 시에 캐리(carry)의 지연을 고려하지 않아도 되는 장점이 있는 몽고메리 승산 연산(Montgomery multiplication operation)을 주로 사용한다. 몽고메리 승산은 모듈로 승산을 가장 효율적으로 수행하는 알고리즘으로 알려져 있다.

[0012] 한편 스마트카드에서의 암호화 모듈(복호화 모듈도 병행됨)의 구현을 위한 기존의 방식은 모두 하드웨어적인 처리(하드웨어적인 구현 효율화) 또는 소프트웨어적인 처리(소프트웨어 최적화) 중 어느 한 처리 측면에서만 구현 내지 개선을 염두하고 있었으며, 각 처리에 관해서는 이미 다수의 기법들이 제안되거나 실시되고 있다.

[0013] 하드웨어적인 처리의 가장 큰 강점은 특히 암호화/복호화 속도(성능)의 향상을 도모할 수 있고, 이는 스마트카드를 이용한 응용 자체의 실행 속도 향상의 결과까지도 볼 수 있다는 것이다. 한편 암호화/복호화에는 소정의 키들(keys)이 요구되는데 이들 키는 보안성 확보를 위해 지속적으로 업데이트 될 필요가 있다. 그런데 하드웨어적인 처리에 의하면 이러한 업데이트의 융통성(flexibility)을 확보할 수 없어 보안성 확보 측면에서의 문제를 야기할 수 있는 가능성이 존재한다. 즉, 하드웨어적인 처리의 경우 상기한 업데이트를 위해서는 키 값을 저장하는 하드웨어 요소(예를 들어 스마트카드에 삽입되는 칩 셋(chip-set) 등)을 통째로 교환하거나 키 값을 하드웨어적으로 삭제하는 과정을 통해서 할 수밖에 없다. 따라서 이러한 업데이트 과정을 거치지 아니하는 한 키 값의

업데이트는 불가하고, 키 값의 업데이트가 되지 아니한 스마트카드의 지속적 사용에 따라 키 값의 누출(종국적으로는 개인 정보의 누출)이 발생할 수 있는 위험성이 상존한다.

- [0014] 상기한 업데이트의 융통성을 확보하기 위해 소프트웨어적인 처리가 그 대안으로 등장하였다. 소프트웨어적인 처리에 의하면 키 값의 업데이트를 위한 외부 인터페이스를 구현할 수 있어 업데이트가 지속적으로 가능하므로 보안성 확보 측면에서 하드웨어적인 처리보다 매우 우수하며, 아울러 암호/복호화 모듈을 모두 소프트웨어(소정의 프로그램 코드)로 구현하기 때문에 암호/복호화 모듈의 설계 및 관리가 하드웨어적인 처리에 비해 매우 용이하다. 그러나 소프트웨어적인 처리만에 의해서는 암호/복호화 속도(성능)의 적정 보장을 할 수 없는 단점을 가지며, 특히 상기한 모듈로 승산 연산을 소프트웨어적인 처리를 통해 구현하는 경우에는 계산량이 너무 많아 구현의 실효성이 없다. 이처럼 소프트웨어적인 처리는 하드웨어적인 처리와는 반대되는 장단점을 가진다.
- [0015] 스마트카드에서의 암호/복호화 모듈의 설계에 있어서의 성능(속도) 향상 및 보안성 확보는 모두 중시되어야 할 요소이며, 따라서 이들 두 요소를 모두 만족시킬 수 있는 방안이 필요한 바, 본 발명은 이에 착안하여 스마트카드에서의 암호/복호화 모듈을 하드웨어적인 처리와 소프트웨어적인 처리를 동시에 같이 이용하여 구현하고자 한다.
- [0016] 이하, 본 발명을 실시하기 위한 구체적인 내용을 본 발명의 바람직한 실시예에 근거하여 첨부 도면을 참조하여 상세히 설명하되, 도면의 구성요소들에 참조번호를 부여함에 있어서 동일 구성요소에 대해서는 비록 다른 도면상에 있더라도 동일 참조번호를 부여하였으며 당해 도면에 대한 설명시 필요한 경우 다른 도면의 구성요소를 인용할 수 있음을 미리 밝혀둔다. 아울러 본 발명과 관련된 공지 기능 혹은 구성에 대한 구체적인 설명 그리고 그 이외의 제반 사항이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우, 그 상세한 설명을 생략한다.
- [0017] 위에서 언급한 바와 같이 스마트카드의 암호/복호화 모듈은 공개키 암호화 방식에 의해 이루어짐이 압도적이며, 이 암호화 방식에는 모듈로 승산 연산이 반드시 수반된다. 한편 공개키 암호화와 관련된 제반 연산에 있어서 모듈로 승산 연산이 가장 많은 연산량을 차지하기 때문에 공개키 암호/복호화 방식의 소요 시간(암호/복호화 속도)은 모듈로 승산 연산에 소요되는 시간에 의해 좌우된다.
- [0018] 이러한 소요 시간을 감소시키기 위해서 소프트웨어적인 처리에 의해 모듈로 승산 연산의 효율적 구현을 상정해 볼 수도 있으나(예를 들어 모듈로 승산 연산 알고리즘의 최적화 등을 통해), 소프트웨어적인 처리에 의한 효율적 구현을 이루더라도 연산 속도(성능) 측면에서의 실효성은 없기 때문에 기존의 암호/복호화 모듈에서도 압도적으로 하드웨어적인 처리를 통해 모듈로 승산 연산을 수행토록 하고 있다. 아울러 위에서 언급한 바와 같이 기존의 암호/복호화 모듈은 두 처리 중 어느 한 처리만을 염두하고 있기 때문에 어느 한 처리에 대한 변경은 다른 처리가 이러한 변경에 대응할 수 있도록(compatible) 다른 처리에 대한 변경도 수반될 수밖에 없는 문제를 유발시킨다.
- [0019] 본 발명은 스마트카드에서의 암호/복호화 모듈의 구현을 하드웨어적인 처리와 소프트웨어적인 처리를 동시에 이용하여 실현하되, 성능(속도) 향상과 직접 관련된 측면에서는 하드웨어적인 처리로, 보안성 확보를 위한 측면에서는 소프트웨어적인 처리로 분리하여 암호/복호화 모듈을 구현하는 것이다. 아울러 이러한 분리 구현은 어느 한 측면에서의 변경이 발생하여도 나머지 측면에는 전혀 영향을 주지 않도록 구현한다.
- [0020] 이를 위해 본 발명은 공개키 암호화 방식으로 암호/복호화 모듈이 구현되는 스마트카드에 있어서 소프트웨어적인 처리가 요구되며 아울러 소프트웨어적인 처리만으로도 성능(속도)에 지장이 없고 하드웨어적인 처리와는 무관한 키 값의 업데이트(저장) 및 암호/복호화 알고리즘은 소프트웨어적인 처리로 구현하고, 성능(속도)과 직접 관련되고 소프트웨어적인 처리와는 무관한 모듈로 승산 연산은 하드웨어적인 처리로 구현하여 두 처리 간 핸드셰이킹(handshaking)을 통해 스마트카드에 있어서의 암호/복호화 모듈을 구현한다.
- [0021] 암호/복호화 알고리즘의 구현 및 설정, 키 값의 업데이트(저장)와 입력값(예를 들어 사용자 ID 등 스마트카드의 사용자와 관련된 고유 정보)의 입력을 위한 외부 인터페이스의 구현은 위에서 언급한 바와 같이 소프트웨어적인 처리에 의해 이루어진다. 외부 인터페이스는 소정의 소프트웨어 모듈로 구현되며 일종의 에뮬레이터를 통해 CPU와 통신한다. 외부 인터페이스를 통해 업데이트된 키 값 또는 입력값도 CPU의 내부 메모리에 또는 CPU를 통해 외부 메모리에 업로드된다.
- [0022] 암호/복호화 알고리즘으로는 공개키 암호화 방식을 이용할 수 있으며, 공개키 암호화 방식의 여러 방식 중에 RSA(Rivest, Shamir, Adleman) 암호화 방식을 이용할 수 있다. 암호/복호화 알고리즘의 소프트웨어적인 처리에 의한 구현은 효율성을 위해 컴퓨터 프로그래밍 언어의 일종인 어셈블리어(assembly language)에 의하는 것이 바람직하다. 왜냐하면 어셈블리어는 기계어(machine language)에 가장 근접한 프로그래밍 언어로 프로그래밍의 어려움은 다소 있으나, 처리 속도 상승의 효과를 가져올 수 있기 때문이다.

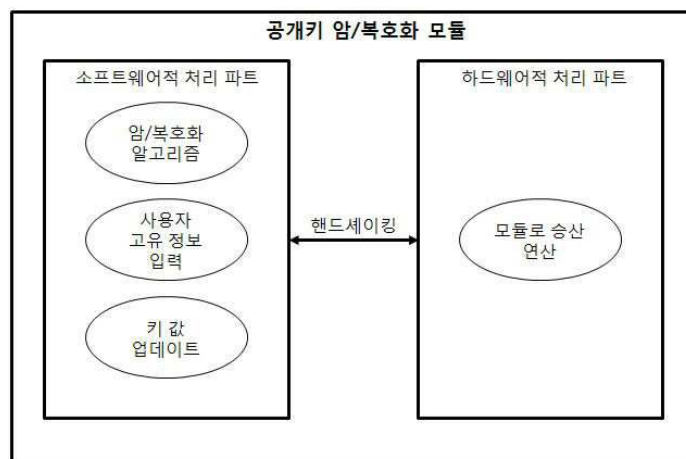
- [0023] 한편 암호/복호화 알고리즘의 소프트웨어적인 처리에 의한 구현의 효율성은 알고리즘의 프로그래밍 기법을 통해서도 이를 수 있는 바, 본 발명은 소위 loop unrolling 기법을 통해 암호/복호화 알고리즘의 프로그래밍을 한다. 암호/복호화 알고리즘의 경우 암호/복호화에 수반되는 소정의 연산을 행함에 있어 통상 루프 알고리즘을 수반하는데(C언어에서는 for문으로 표현됨), 루프 알고리즘은 암호/복호화에 수반되는 연산의 소요 시간을 증가시켜 암호/복호화 속도를 저하시키는 주된 원인으로 지목되어 왔으며 특히 소프트웨어적인 처리에만 의한 암호/복호화 모듈의 구현에 있어서는 더욱 그러했다. 이러한 루프 알고리즘의 문제를 완화하기 위한 프로그래밍 기법이 loop unrolling 기법이며, 이는 반복 처리가 요구되는 연산에 루프 처리가 아닌 순차적 처리(sequential process)를 통해 연산을 수행하는 기법이다. 이 기법은 순차적 처리에 근간을 두므로 프로그램 코드의 크기 상승을 가져오는 단점은 있으나, 처리 속도 향상의 측면에서 루프 처리에 비해 커다란 이점을 가져 빠른 처리 속도가 요구되는 응용에 흔히 사용하는 프로그래밍 기법이다.
- [0024] 공개키 암호화 방식에는 모듈로 승산 연산이 반드시 포함됨을 위에서 언급했는데, 모듈로 승산의 연산량은 공개키 암호화 방식에 의한 암호/복호화에 소요되는 연산량의 90% 이상을 차지하며, 연산량 자체가 너무 많아 이를 소프트웨어적인 처리를 통해 구현하는 것은 성능(속도) 향상 측면에서는 아무런 실효성이 없으므로, 본 발명에서는 공개키 암호화 방식(RSA 방식)에 의한 암호/복호화 알고리즘 중 모듈로 승산 연산 부분을 하드웨어적인 처리를 통해 구현한다. 하드웨어적인 처리란 일단 구현이 이루어지고 나면 변경이 되지 아니함(또는 변경이 불가능함)을 의미하며, 아울러 소프트웨어적인 처리 측면에서의 변경이 발생하더라도 그 영향을 받지 않는다. 즉, 소프트웨어적인 처리 측면에서의 변경이 발생하더라도 하드웨어적인 처리로 구현된 모듈로 승산 연산을 위한 입력값은 변하지 아니하므로 영향을 받지 아니한다는 의미이다.
- [0025] 한편 모듈로 승산 연산은 여러 방식이 존재하나, 몽고메리 승산 연산을 사용하는 것이 가장 바람직하다. 몽고메리 승산 연산(승산 알고리즘)은 승수(multiplier) A, 피승수(multiplicand) B 및 모듈러스(modulus) M을 입력으로 하여, 최종 합(S_n)을 계산하여 출력한다. 몽고메리 승산 연산의 알고리즘은 다음과 같다.
- [0026] 1. $S_0 = 0$
- [0027] 2. For $i = 1$ to n
- [0028] $S_i = S_{i-1} + a_i B$
- [0029] if $s_0 = 0$ then
- [0030] $S_i = S_{i-1} \text{ div } 2$
- [0031] else
- [0032] $S_i = (S_{i-1} + M) \text{ div } 2$
- [0033] 3. If $S_i \geq M$ then $S_i = S_{i-1} - m$
- [0034] 4. return S_n .
- [0035] 여기서 a_i 는 A가 n-비트로 표현될 경우의 i번째 비트의 값(0 또는 1)을, s_0 는 이전 비트(i-1 번째 비트)에 대한 결과 값(S_{i-1} , 중간 결과 값)의 최하위 비트(Least Significant Bit: LSB)의 값을, div는 쉬프트 라이트(shift right), m은 모듈로(modulo)를 의미한다.
- [0036] 몽고메리 승산 알고리즘은 그 특성상 쉬프트와 덧셈기(adder)를 사용해서 구현되며, 승산 연산시 항상 n-비트 승수의 i번째 비트 값(a_i)과 중간 결과 값(S_{i-1})의 비트 값을 체크한 후 다음 쉬프트(또는 덧셈)를 수행하게 되는 등 승수, 중간 결과 값의 모든 비트에 대해 동일한 연산 과정을 반복 수행한다(상기 과정 2의 전 과정이 어느 경우에도 수행됨). 그러나 이는 승산 연산의 리던던시(redundancy)를 야기하여 속도(성능)의 향상을 저해할 우려가 있는데, 특히 비트 값(a_i)이 0이거나 s_0 가 0인 경우에는 0이 아닌 이전 비트(prior bit)에 대한 연산 과정에서 얻어진 값(이전 루프에서 얻어진 값)을 그대로 사용하여도(일부 연산 과정을 생략(skip)하여도) 몽고메리 승산 연산의 최종 결과(S_n)에는 영향을 미치지 아니함을 위의 알고리즘을 통해 알 수 있다.
- [0037] 따라서 본 발명에서의 몽고메리 승산 연산은 a_i 와 s_0 의 값을 미리 체크하여 0인 경우에는 이전 루프에서 얻어진

값들을 그대로 사용하여 몽고메리 승산 연산의 속도를 더욱 향상시킬 수 있도록 한다.

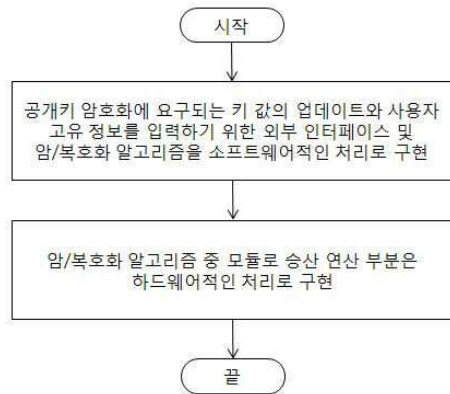
- [0038] 소프트웨어적인 처리와 하드웨어적인 처리에 의한 암호/복호화 모듈이 상기와 같이 구현되면 양 처리 파트 간의 핸드셰이킹을 통해 암호/복호화가 이루어진다. 핸드셰이킹은 양방향 데이터 교환이 일어나는 컴퓨팅이나 통신 등의 환경에서는 흔히 쓰이는 기법이며, 본 발명에 의한 핸드셰이킹도 기존의 기법에 따를 수 있다.
- [0039] 다만, 통상적인 핸드셰이킹은 소프트웨어적인 처리 파트가 하드웨어적인 처리 파트에 명령을 내리면, 하드웨어적인 처리 파트는 그 명령을 수행하며 그 시간에 소프트웨어적인 처리 파트는 대기한다. 하드웨어적인 처리 파트가 동작을 끝내면, 완료 응답을 소프트웨어적인 처리 파트에 보내고 소프트웨어적인 처리 파트는 다시 다음 명령을 하드웨어적인 처리 파트에 보내는 방식으로 구성되어 '하나의 명령' 단위로 이루어지는데 반해, 본 발명에서는 '두 개 이상의 명령'에 대해 동시에 이루어지도록 한다. 간단한 예를 들면 다음과 같다.
- [0040] a) 소프트웨어적인 처리 파트가 하드웨어적인 처리 파트에 첫 번째 명령(first que)과 두 번째 명령(Second que)을 같이 전달한다.
- [0041] b) 하드웨어적인 처리 파트는 전달받은 두 명령을 수행하고 다음 명령을 기다린다.
- [0042] c) 소프트웨어적인 처리 파트는 하드웨어적인 처리 파트가 이전에 보낸 두 명령을 실행하는 동안 다음 명령을 미리 준비하고 코프로세서에서 완료 응답이 오면 처음 두 명령을 캐쉬에서 삭제하고, 다음 명령을 하드웨어적인 처리 파트에 보낸다.
- [0043] 이렇게 다수의 명령 단위로 양 파트 간 핸드셰이킹을 수행하면 암호/복호화에 소요되는 시간을 더욱 줄일 수 있다.
- [0044] 본 방법발명은 또한 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 유무선 네트워크를 통한 전송)의 형태로 구현되는 것도 포함한다. 또한 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.
- [0045] 이제까지 본 발명에 대하여 그 바람직한 실시예를 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 본 명세서에서는 스마트카드를 중심으로 해서 발명이 개시되어 있지만 공개키 암호화 방식이 활용되는 어떠한 애플리케이션에 대해서도 본 발명의 적용은 가능하다. 그러므로 개시된 실시예는 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허 청구범위에 나타나 있으며, 그와 균등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

도면

도면1



도면2



도면3

