



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0047341
(43) 공개일자 2019년05월08일

(51) 국제특허분류(Int. Cl.)
G06F 7/50 (2006.01) H04L 9/30 (2006.01)
(52) CPC특허분류
G06F 7/50 (2013.01)
H04L 9/302 (2013.01)
(21) 출원번호 10-2017-0141107
(22) 출원일자 2017년10월27일
심사청구일자 2017년10월27일

(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대 학교)
(72) 발명자
강진구
서울특별시 서초구 양재대로2길 109, 101동 1104호(우면동, 서초참누리에코리치)
허성화
인천광역시 남구 재념이길 105-15, 202호(용현동, 태일원룸)
(74) 대리인
양성보

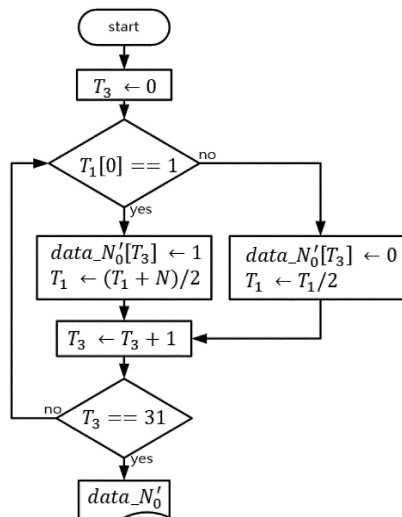
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법

(57) 요약

제안하는 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법이 제시된다. 본 발명에서 제안하는 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법은 확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계, 확장 이진 GCD 알고리즘을 적용하여 상기 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계, 상기 R 및 N이 항상 홀수 이고, R은 2의 멱수라는 입력 값의 특성을 이용하여 상기 확장 이진 GCD 알고리즘에서 뺄셈연산 과정을 제거하는 단계 및 상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N_0' 을 구하는 단계를 포함한다.

대표도 - 도6



이 발명을 지원한 국가연구개발사업

과제고유번호 IITP-2017-2012-0-00641

부처명 미래창조과학부

연구관리전문기관 정보통신산업진흥원

연구사업명 대학ICT연구센터육성지원사업

연구과제명 정보기기용 시스템반도체 핵심 설계 기술 개발 및 인력양성

기 여 율 1/1

주관기관 서울시립대학교 산학협력단

연구기간 2017.01.01 ~ 2017.12.31

명세서

청구범위

청구항 1

R 및 N이 항상 홀수 이고, R은 2의 멱수라는 입력 값의 특성을 이용하여 확장 이진 GCD 알고리즘에서 뺄셈연산 과정을 제거하는 단계; 및

상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계를 포함하는 몽고메리 알고리즘 파라미터 계산 방법.

청구항 2

제1항에 있어서,

확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계; 및

확장 이진 GCD 알고리즘을 적용하여 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계

를 더 포함하는 몽고메리 알고리즘 파라미터 계산 방법.

청구항 3

제2항에 있어서,

상기 확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계는,

R^{-1} 및 N' 의 두 식에서 N의 부호가 반대이므로 실제 N' 이 음수가 나올 경우 양수로 처리하고 양수일 경우 음수를 취하고, 모듈러 연산에서 음수는 나오지 않으므로 음수를 취한 후 R을 더하여 원하는 값을 구하는

몽고메리 알고리즘 파라미터 계산 방법.

청구항 4

제2항에 있어서,

상기 확장 이진 GCD 알고리즘을 적용하여 상기 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계는,

확장 이진 GCD 알고리즘은 3개의 덧셈기와 3개의 뺄셈기, 3개의 쉬프터가 사용되고, 9개의 레지스터가 포함되며, 확장 이진 GCD 알고리즘을 적용하여 구한 변수들의 관계식은 하기 식과 같은

$$u_3 = uu_1 + vu_2$$

$$v_3 = uv_1 + vv_2$$

$$t_3 = ut_1 + vt_2$$

몽고메리 알고리즘 파라미터 계산 방법.

청구항 5

제1항에 있어서,

상기 R 및 N이 항상 홀수 이고, R은 2의 멱수라는 입력 값의 특성을 이용하여 상기 확장 이진 GCD 알고리즘에

서 뺄셈연산 과정을 제거하는 단계는,

t_2 는 항상 음수이고, t_2 의 연산은 기존의 t_2 에서 R 을 빼는 뺄셈을 진행하게 되므로, 덧셈을 수행하고 앞에 음수를 붙이며, 값이 음수이면 양수로 취하면 되므로 덧셈으로 진행하는

몽고메리 알고리즘 파라미터 계산 방법.

청구항 6

제1항에 있어서,

상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계는,

초기 입력 값인 N 이 항상 홀수이기 때문에 k 를 구하는 과정을 필요로 하지 않고, t_1, t_2, t_3 만 있어도 원하는 값을 얻을 수 있기 때문에 u_1, u_2, u_3 와 v_1, v_2, v_3 값을 필요로 하지 않으며, t_3 은 루프 카운터 역할을 하는 숫자이기 때문에 연산을 필요로 하지 않는

몽고메리 알고리즘 파라미터 계산 방법.

청구항 7

제1항에 있어서,

상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계는,

2개의 덧셈기, 2개의 쉬프트, 2개의 레지스터 및 1개의 카운터만을 필요로 하는

몽고메리 알고리즘 파라미터 계산 방법.

청구항 8

제1항에 있어서,

상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계는,

N' 의 최하위 비트 값부터 구하고, 32회의 루프만 수행하며, 1개의 덧셈기, 1개의 쉬프트, 1개의 레지스터, 1개의 카운터만을 필요로 하는

몽고메리 알고리즘 파라미터 계산 방법.

발명의 설명

기술 분야

[0001] 본 발명은 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법에 관한 것이다.

배경 기술

[0002] RSA 알고리즘은 공개키 암호 중 주로 사용되는 알고리즘이다. RSA 알고리즘의 주요 핵심 연산은 모듈러(modular) 연산이다. 모듈러 연산을 그대로 하드웨어적으로 설계하기에는 어려움이 있기 때문에 몽고메리 알고리즘 몽고메리 알고리즘(Montgomery algorithm)을 이용한다. 여기서 또 하나의 문제점은 몽고메리 알고리즘을 사용하기 위해 미리 계산해야 하는 파라미터 값이다. 따라서, 그 파라미터 값을 하드웨어적으로 효과적인 연산을 진행할 수 있는 알고리즘을 필요로 한다.

발명의 내용

해결하려는 과제

[0003] 본 발명이 이루고자 하는 기술적 과제는 확장 유클리디안 알고리즘(Extended Euclidean algorithm)을 적용하여

N' 값을 구하기 위해 하드웨어 적으로 구현하는데 어려움을 해결하기 위해 확장 이진 GCD 알고리즘(Extended binary GCD algorithm)을 적용하고, RSA 알고리즘에 맞게 수정함으로써 하드웨어적으로 구현하기 용이하고 연산 과정이 단순화된 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법을 제공하는데 있다.

과제의 해결 수단

[0004] 일 측면에 있어서, 본 발명에서 제안하는 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법은 확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계, 확장 이진 GCD 알고리즘을 적용하여 상기 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계, 상기 R 및 N이 항상 홀수 이고, R은 2의 멱수라는 입력 값의 특성을 이용하여 상기 확장 이진 GCD 알고리즘에서 뺄셈연산 과정을 제거하는 단계 및 상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계를 포함한다.

[0005] 상기 확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계는 R^{-1} 및 N' 의 두 식에서 N 의 부호가 반대이므로 실제 N' 이 음수가 나올 경우 양수로 처리하고 양수일 경우 음수를 취하고, 모듈러 연산에서 음수는 나오지 않으므로 음수를 취한 후 R 을 더하여 원하는 값을 구한다.

[0006] 상기 확장 이진 GCD 알고리즘을 적용하여 상기 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계는 확장 이진 GCD 알고리즘은 3개의 덧셈기와 3개의 뺄셈기, 3개의 쉬프터가 사용되고, 9개의 레지스터가 포함되며, 확장 이진 GCD 알고리즘을 적용하여 구한 변수들의 관계식은 하기 식과 같다:

$$u_3 = uu_1 + vu_2$$

$$v_3 = uv_1 + vv_2$$

$$t_3 = ut_1 + vt_2$$

[0007]

[0008] 상기 R 및 N 이 항상 홀수 이고, R 은 2의 멱수라는 입력 값의 특성을 이용하여 상기 확장 이진 GCD 알고리즘에서 뺄셈연산 과정을 제거하는 단계는 t_2 는 항상 음수이고, t_2 의 연산은 기존의 t_2 에서 R 을 빼는 뺄셈을 진행하게 되므로, 덧셈을 수행하고 앞에 음수를 붙이며, 값이 음수이면 양수로 취하면 되므로 덧셈으로 진행한다.

[0009] 상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계는 초기 입력 값인 N 이 항상 홀수이기 때문에 k 를 구하는 과정을 필요로 하지 않고, t_1, t_2, t_3 만 있어도 원하는 값을 얻을 수 있기 때문에 u_1, u_2, u_3 와 v_1, v_2, v_3 값은 필요로 하지 않으며, t_3 은 루프 카운터 역할을 하는 숫자이기 때문에 연산을 필요로 하지 않는다.

[0010] 상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N'_0 을 구하는 단계는 2개의 덧셈기, 2개의 쉬프터, 2개의 레지스터 및 1개의 카운터만을 필요로 하고, N' 의 최하위 비트 값부터 구하고, 32회의 루프만 수행하며, 1개의 덧셈기, 1개의 쉬프터, 1개의 레지스터, 1개의 카운터만을 필요로 한다.

발명의 효과

[0011] 본 발명의 실시예들에 따르면 확장 유클리디안 알고리즘(Extended Euclidean algorithm)을 적용하여 N' 값을 구하기 위해 하드웨어 적으로 구현하는데 어려움을 해결하기 위해 확장 이진 GCD 알고리즘(Extended binary GCD algorithm)을 적용하고, RSA 알고리즘에 맞게 수정함으로써 하드웨어적으로 구현하기 용이하고 연산과정이 단순화된 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법을 제공한다.

도면의 간단한 설명

- [0012] 도 1은 본 발명의 일 실시예에 따른 워드기반 몽고메리 알고리즘을 나타내는 도면이다.
- 도 2는 본 발명의 일 실시예에 따른 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법을 설명하기 위한 흐름도이다.
- 도 3은 본 발명의 일 실시예에 따른 확장 유클리디안 알고리즘의 순서도이다.
- 도 4는 본 발명의 일 실시예에 따른 확장 이진 GCD 알고리즘의 순서도이다.
- 도 5는 본 발명의 일 실시예에 따른 수정된 확장 이진 GCD 알고리즘의 순서도이다.
- 도 6은 본 발명의 일 실시예에 따른 RSA 알고리즘 용 몽고메리 알고리즘을 적용한 N' 의 연산 순서도이다.

발명을 실시하기 위한 구체적인 내용

- [0013] 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.
- [0014] 도 1은 본 발명의 일 실시예에 따른 워드기반 몽고메리 알고리즘을 나타내는 도면이다.
- [0015] 도 1의 몽고메리 알고리즘은 워드 기반 몽고메리 알고리즘(Word based Montgomery algorithm)이다. 이러한 알고리즘에서 s 는 N 값의 워드 개수를 의미한다. 여기서 사용하는 N 값은 1024bit이므로 s 는 32가 된다. 몽고메리 알고리즘을 사용하기 위해서는 미리 구해야 하는 N_0' 값이 있다. N_0' 값은 $R \times R^{-1} - N \times N' = 1$ 이라는 식을 만족하는 N' 값의 하위 32bit값이다.
- [0016] 도 2는 본 발명의 일 실시예에 따른 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법을 설명하기 위한 흐름도이다.
- [0017] 제안하는 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법은 확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계(210), 확장 이진 GCD 알고리즘을 적용하여 상기 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계(220), 상기 R 및 N이 항상 홀수 이고, R은 2의 멱수라는 입력 값의 특성을 이용하여 상기 확장 이진 GCD 알고리즘에서 뺄셈연산 과정을 제거하는 단계(230) 및 상기 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N_0' 을 구하는 단계(240)를 포함한다.
- [0018] 확장 유클리디안 알고리즘을 적용하여 $R \times R^{-1} + N \times N' = 1$ 을 만족하는 R^{-1} 및 N' 을 구하는 단계(210)에서는 R^{-1} 및 N' 의 두 식에서 N 의 부호가 반대이므로 실제 N' 이 음수가 나올 경우 양수로 처리하고 양수일 경우 음수를 취하고, 모듈러 연산에서 음수는 나오지 않으므로 음수를 취한 후 R 을 더하여 원하는 값을 구한다.
- [0019] 이후, 확장 이진 GCD 알고리즘을 적용하여 상기 확장 유클리디안 알고리즘에서 나눗셈 과정을 제거하고, 덧셈, 뺄셈 및 쉬프트 연산만을 이용하여 R^{-1} 및 N' 을 구하는 단계(220)에서는 확장 이진 GCD 알고리즘은 3개의 덧셈기와 3개의 뺄셈기, 3개의 쉬프터가 사용되고, 9개의 레지스터가 포함된다.
- [0020] R 및 N이 항상 홀수 이고, R은 2의 멱수라는 입력 값의 특성을 이용하여 상기 확장 이진 GCD 알고리즘에서 뺄셈 연산 과정을 제거하는 단계(230)에서는 t_2 는 항상 음수이고, t_2 의 연산은 기존의 t_2 에서 R 을 빼는 뺄셈을 진행하게 되므로, 덧셈을 수행하고 앞에 음수를 붙이며, 값이 음수이면 양수로 취하면 되므로 덧셈으로 진행한다.
- [0021] 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N_0' 을 구하는 단계(240)에서는 초기 입력 값인 N 이 항상 홀수이기 때문에 k 를 구하는 과정을 필요로 하지 않고, t_1, t_2, t_3 만 있어도 원하는 값을 얻을 수 있기 때문에 u_1, u_2, u_3 와 v_1, v_2, v_3 값은 필요로 하지 않으며, t_3 은 루프 카운터 역할을 하는 숫자이기 때문에 연산을 필요로 하지 않는다. 이때, 2개의 덧셈기, 2개의 쉬프터, 2개의 레지스터 및 1개의 카운터만을

필요로 한다.

[0022] 마지막으로, 확장 이진 GCD 알고리즘의 결과에 대하여 RSA 용 몽고메리 알고리즘을 적용하여 N' 을 구하는 단계 (240)에서는 N' 의 최하위 비트 값부터 구하고, 32회의 루프만 수행하며, 1개의 덧셈기, 1개의 쉬프트, 1개의 레지스터, 1개의 카운터만을 필요로 한다. 도 1 내지 도 6을 참조하여 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법에 대하여 더욱 상세히 설명한다.

[0023] 다시 도 1의 몽고메리 알고리즘을 참조하면, 이러한 알고리즘에서 s 는 N 값의 워드 개수를 의미한다. 여기서 사용하는 N 값은 1024bit이므로 s 는 32가 된다. 몽고메리 알고리즘을 사용하기 위해서는 미리 구해야 하는 N_0' 값이 있다. N_0' 값은 $R \times R^{-1} - N \times N' = 1$ 이라는 식을 만족하는 N' 값의 하위 32bit값이다. 다음 식에 만족하는 R^{-1} 값과 N' 값을 구하기 위해서는 확장 유클리디안 알고리즘(Extended Euclidean algorithm)을 사용해야 한다. 확장 유클리디안 알고리즘을 사용하면 $R \times R^{-1} + N \times N' = 1$ 식을 만족하는 R^{-1} 값과 N' 값을 구할 수 있다. 여기서 두식의 N 값 앞의 부호가 반대임으로 실제 N' 값이 음수가 나올 경우 양수로 처리하고 양수일 경우 음수를 취하면 된다. 하지만 모듈러(Modular) 연산에서 음수는 나오지 않으므로 음수를 취한 후 값을 더하면 원하는 R 값을 구할 수 있다.

[0024] 도 3은 본 발명의 일 실시예에 따른 확장 유클리디안 알고리즘의 순서도이다.

[0025] 도 3과 같이 진행되는 알고리즘을 그대로 하드웨어로 구현하기에는 어려운 점이 있다. 연산과정에 나눗셈과 뺄셈 등 하드웨어로 구현하기 비효율적인 연산이 포함되어 있다. 연산과정을 살펴보면 나눗셈을 수행하여 몫과 나머지를 구하고 그 결과 값을 이용하여 다시 x 값과 y 값을 구하는 과정을 수행해야 한다. 연산이 복잡하지는 않지만 나눗셈과 뺄셈이 포함되어 있어 하드웨어로 구현하면 면적이 커지게 된다. 따라서, 같은 기능을 하지만 비교적 하드웨어로 구현이 편한 확장 이진 GCD 알고리즘(Extended binary GCD algorithm)을 이용한다.

[0026] 도 4는 본 발명의 일 실시예에 따른 확장 이진 GCD 알고리즘의 순서도이다.

[0027] 도 4의 알고리즘을 살펴보면 도 3과 다르게 나눗셈 과정이 포함되어 있지 않다. 덧셈과 뺄셈, 그리고 쉬프트 연산만으로 구성되었다. 확장 이진 GCD 알고리즘에서 나오는 변수들의 관계식은 다음과 같다.

$$u_3 = uu_1 + vu_2$$

$$v_3 = uv_1 + vv_2$$

$$t_3 = ut_1 + vt_2$$

[0028]

[0029] 확장 이진 GCD 알고리즘은 총 3개의 덧셈기와 3개의 뺄셈기, 3개의 쉬프트(shifter)가 사용된다. 또한, 9개의 레지스터가 포함되어 있다. 나눗셈연산은 제거 되었지만 아직 뺄셈 연산이 남아 있고, 본 발명에서 제안하는 알고리즘은 RSA용으로 사용하기 위한 것이기 때문에 도 4의 알고리즘 그대로 사용하는 것은 비효율적이다. 따라서, 뺄셈을 제거하고 RSA 용으로 사용할 수 있도록 하는 최적화 작업이 필요하다. 최적화 작업은 N 값은 항상 홀수 이고 R 값은 2의 멍수 값이라는 입력 값의 특수성을 이용한다. 이러한 특수성을 이용하면 뺄셈연산 및 불필요한 연산과정을 뺄 수 있다. 우선, 확장 이진 GCD 알고리즘에 값이 3599이고 R 값이 $2^{12} = 4096$ 인 경우를 살펴보면 표 1과 같다. 표 1은 N 값이 3599일 때 확장 이진 GCD 알고리즘 연산 결과이다.

[0030] <표 1>

u_1	u_2	u_3	v_1	v_2	v_3	t_1	t_2	t_3
1	0	4096	3599	-4095	3599	1	0	4096
						1800	-2048	2048
						900	-1024	1024
						450	-512	512
						225	-256	256
						1912	-2176	128
						956	-1088	64
						478	-544	32
						239	-272	16
						1919	-2184	8
						2759	-3140	4
						3179	-3618	2
						3389	-3857	1
3389	-3857	1						
						-210	238	-3598
						3389	-3858	-3598
						3494	-3977	-1799
			105	-119	1799			
						3284	-3738	-1798
						1642	-1869	-899
			1957	-2227	899			
						1432	-1630	-898
						716	-815	-449
			2883	-3281	449			
						506	-576	-448
						253	-288	-224
						1926	-2192	-112
						963	-1096	-56
						2281	-2596	-28
						2940	-3346	-14
						1470	-1673	-7
			2129	-2423	7			
						1260	-1434	-6
						630	-717	-3
			2969	-3379	3			
						420	-478	-2
						210	-239	-1
			3389	-3857	1			
						0	0	0
						3599	-4096	0

[0031]

[0032] 연산결과 R^{-1} 값은 3389이고 N' 값은 3857인 것을 확인할 수 있다. N' 값이 음수로 나왔기 때문에 양수로 취하면 된다. 위 과정에서 보면 알고리즘이 끝나지 않았지만 중간에 이미 우리가 원하는 값이 출력됨을 알 수 있다. 또한 t_3 값을 보면 R 값(4096)으로 시작하여 1bit씩 오른쪽으로 쉬프트 연산이 진행되었다가 최종적으로 1이 되었을 때 원하는 값을 얻을 수 있는 것을 확인할 수 있다. 다시 말해, 전체 루프 횟수는 N 값의 bit 수와 일치하는 것을 알 수 있다. 여기서 N 값이 12bit임으로 루프를 12회 수행하였을 때 원하는 N' 값을 얻을 수 있었다. 실제 RSA에서 N 값은 1024bit임으로 루프를 1024회 수행하면 값을 얻을 수 있다.

[0033] 이때, t_2 값을 살펴보면 항상 음수인 것을 확인할 수 있다. t_2 연산을 보면, 기존의 t_2 값에 R 값을 빼고, 이러한 과정은 음수에 뺄셈을 진행하게 되므로 결국 덧셈을 수행하고 앞에 음수를 붙이면 된다. 또한 N' 값이 음수이면 양수로 취하면 되므로 결국 덧셈으로 진행하면 된다는 것을 확인할 수 있다. 이러한 형태를 이용하여 1차적으로 도 5의 수정된 확장 이진 GCD 알고리즘과 같이 최적화 하였다.

[0034] 도 5는 본 발명의 일 실시예에 따른 수정된 확장 이진 GCD 알고리즘의 순서도이다.

[0035] 초기 입력 값인 N 값이 항상 홀수이기 때문에 k 값을 구하는 과정은 필요가 없다. 또한 변수 t_1, t_2, t_3 값만

있어도 원하는 값을 얻을 수 있기 때문에 변수 u_1, u_2, u_3 와 v_1, v_2, v_3 값은 필요가 없다. 그리고 t_3 값은 경우 루프 카운터 역할을 하는 숫자로 되었기 때문에 연산이 필요 없다. 따라서, 2개의 덧셈기, 2개의 쉬프트, 2개의 레지스터, 1개의 카운터만 필요하게 된다. 여기서 좀 더 t_3 값이 R 값에서 1로 될 때의 t_1 과 t_2 값을 분석해보면 다음 표 2와 표 3과 같은 현상이 나타난다. 표 2는 t 값의 연산과정이고, 표 3은 이진수로 표현한 t_2 값이다.

<표 2>

t_1	t_2	t_3
1	0	4096
1800	-2048	2048
900	-1024	1024
450	-512	512
225	-256	256
1912	-2176	128
956	-1088	64
478	-544	32
239	-272	16
1919	-2184	8
2759	-3140	4
3179	-3618	2
3389	-3857	1

<표 3>

t_2	이진수 값
0	0000 0000 0000
2048	1000 0000 0000
1024	0100 0000 0000
512	0010 0000 0000
256	0001 0000 0000
2176	1000 1000 0000
1088	0100 0100 0000
544	0010 0010 0000
272	0001 0001 0000
2184	1000 1000 1000
3140	1100 0100 0100
3618	1110 0010 0010
3857	1111 0001 0001

표 2는 1차 수정한 알고리즘에 기존의 N 값 R 값을 넣어 계산한 값의 진행이다. 표 3은 표 2에서 t_2 값만 따로 이진수 형태로 표현한 것이다. 표 2 및 표 3에서 네모 상자에 들어간 숫자 상태를 보면 이전 t_1 값과 t_2 값이 둘 다 짝수가 아닐 때의 상태를 수행한 결과이다. 1차 수정한 알고리즘에서 값과 값 둘 다 짝수가 아닐 경우 t_1 값에 N 값을 더하게 되고 t_2 에는 R 값을 더하게 된다. 이때, 결과적으로 t_2 값 최상위 비트에 1이 들어

가는 것과 같게 된다. 그리고 t_1 값과 t_2 값 모두 짝수일 경우에는 t_1 값과 t_2 모두 우측으로 쉬프트 연산이 수행된다. 결과적으로 t_2 값 최상위 비트에 0이 들어가게 된다. 따라서, 표 3에 보이는 바와 같이, 네모상자에 있는 값의 이진수 값을 보면 최상위 비트에 1이 있는 것을 확인할 수 있고 네모상자가 없는 값에는 최상위 비트에 0이 있는 것을 확인할 수 있다.

[0041] 또한, 연산 초기에 t_1 값에 1을 넣고 t_2 값에 0을 넣게 된다. 그러면 t_1 값이 홀수이므로 덧셈 후 쉬프트 연산이 진행되는 상태가 수행된다. t_2 값에는 R 값을 더하고 1bit 쉬프트한 값인 2048이 되고 이때 붙은 1은 연산이 끝날 때 최상위 비트에서 최하위 비트 자리로 옮겨지게 되고 연산이 종료된다. 이후 t_2 에는 항상 짝수가 더해짐으로 처음에 최상위 비트에 붙었던 1이 최하위 비트가 될 때까지 홀수가 되지 않는다. 그러므로 t_2 가 음수인지 아닌지를 판단하지 않아도 된다.

[0042] t_1 값과 t_2 값의 연산 루프는 총 12회 진행되었다. 이때 연산 루프횟수는 N 값의 비트 수와 같다. 또한 t_1 값이 홀수이든 짝수이든 t_2 값은 항상 우측으로 쉬프트 연산이 수행된다. 그리고 t_2 값의 첫 번째 연산에서 붙은 최상위 비트 1은 12번 쉬프트되어 최하위 비트가 되고, 두 번째 연산에서 붙은 최상위 비트 0은 11번 쉬프트되어 하위 2번째 비트가 되었다. 세 번째 연산에서는 최상위 비트 0이었고 이 비트가 10번 쉬프트되고 네 번째 연산은 9번 쉬프트 된다. 다시 말해, 최상위 비트부터 원하는 비트 값을 채워 넣으면서 오른쪽으로 쉬프트하는 것과 같게 된다. 이렇게 진행되는 것을 역으로 생각하면 처음부터 t_2 값의 최하위비트부터 차례차례 원하는 비트 값을 채워 넣으면 원하는 값을 얻을 수 있다. t_1 에 덧셈을 진행하면 1을 넣고 덧셈을 진행하지 않으면 0을 넣으면 최종적으로 원하는 값을 구할 수 있다. 이와 관련된 내용을 표 4에 정리 하였다. 표 4는 상태 별 N' 값을 나타낸다.

[0043] <표 4>

t_2	state 별 N' 값
0	0
2048	1
1024	01
512	001
256	0001
2176	1 0001
1088	01 0001
544	001 0001
272	0001 0001
2184	1 0001 0001
3140	11 0001 0001
3618	111 0001 0001
3857	1111 0001 0001

[0044]

[0045] 표 4에서 알 수 있듯이 네모상자일 때는 그전의 t_1 값이 홀수이기 때문에 덧셈을 진행 후 쉬프트 연산을 할 경우이고, 원형상자는 쉬프트 연산만을 할 경우이다. t_2 값의 최하위 비트부터 덧셈 후 쉬프트 연산을 하면 1이 붙고 쉬프트 연산만 하면 0이 붙는 것을 확인할 수 있다. 이와 같이 진행하면서 최종적으로 루프를 다 돌게 되면 기존에 원하는 값과 같은 값을 구할 수 있다는 것을 확인할 수 있다. 이를 정리하여 최종적으로 알고리즘을

구현하면 도 6과 같다.

[0046] 도 6은 본 발명의 일 실시예에 따른 RSA 알고리즘 용 몽고메리 알고리즘을 적용한 N'_0 의 연산 순서도이다.

[0047] 최종적으로 도 6의 알고리즘을 적용하여 구하려는 값은 N' 의 하위 32bit인 N'_0 값이다. 위 알고리즘을 사용하면 N'_0 값의 최하위 비트 값부터 알 수 있다. 따라서, 루프를 1024회 수행할 필요가 없고 32회만 수행하면 된다. 이와 같이 수정함으로써 덧셈기가 하나만 필요하게 되고 루프는 32회로 줄어들게 되었다. 또한, t_2 값도 연산을 하지 않아도 되므로 최종적으로 1개의 덧셈기, 1개의 쉬프트, 1개의 레지스터, 1개의 카운터가 사용된다.

[0048] 본 발명의 실시예들에 따르면 RSA 알고리즘 용 몽고메리 알고리즘 파라미터 계산 방법을 이용하여 확장 유클리디안 알고리즘(Extended Euclidean algorithm)을 적용하여 N' 값을 구하기 위해 하드웨어 적으로 구현하는데 어려움을 해결하기 위해 확장 이진 GCD 알고리즘(Extended binary GCD algorithm)을 적용하고, RSA 알고리즘에 맞게 수정함으로써 하드웨어적으로 구현하기 용이하고 연산과정이 단순화될 수 있다.

[0049] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0050] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embodiment)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0051] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0052] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형

태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0053] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

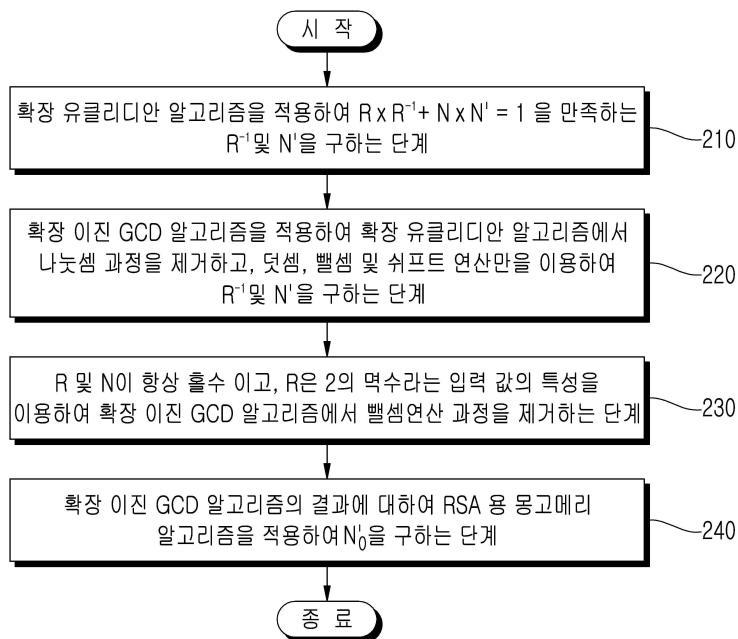
도면

도면1

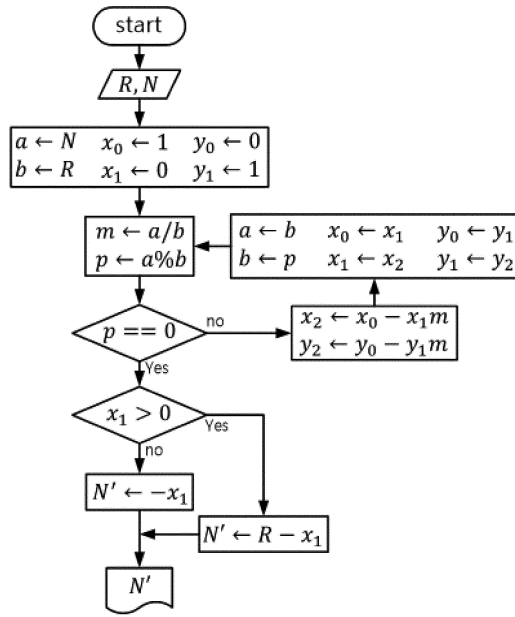
input : $A(a_{s-1}, a_{s-2}, \dots, a_0), B(b_{s-1}, b_{s-2}, \dots, b_0), N(n_{s-1}, n_{s-2}, \dots, n_0)$,
 $R = W^s, W = 2^{32}, R \times R^{-1} = N \times N^{-1} = 1, N'_0 = N' \bmod W$
 output : $T = ABW^{-s} \bmod N$

step1 : $T \leftarrow 0$
 step2 : for $i = 0$ to $s - 1$
 step2.1 : $m_i \leftarrow (T_0 + a_i b_0) N'_0 \bmod W$
 step2.2 : $T \leftarrow (T + a_i B + m_i N) / W$
 end for
 step3 : if $T \geq N$ then
 $T \leftarrow T - N$
 else
 $T \leftarrow T$
 return T

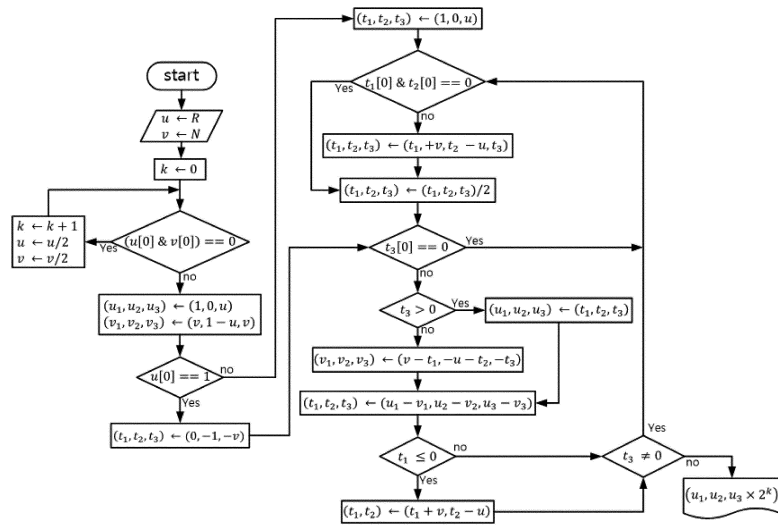
도면2



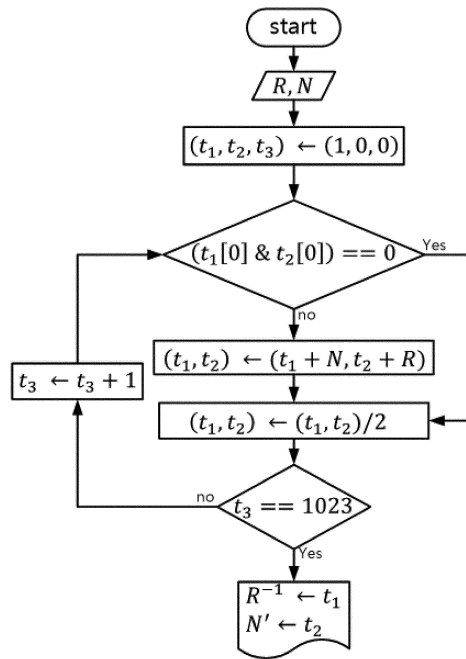
도면3



도면4



도면5



도면6

