RSA 일방향 어큐뮬레이터를 이용한 효율적이고 동적인 인증 딕셔너리 설계

김순석* · 이용희* · 이강우*

Efficient and Dynamic Authenticated Dictionary Design Using RSA One-way Accumulator

Soon-seok Kim* · Yong-hee Lee* · Kang-woo Lee*

요 약

현재 인터넷과 같이 널리 사용되는 공개된 네트워크를 통해 법률적으로 유효한 문서나 상거래 계약서와 같은 엄격한 보안이 요구되는 기밀정보가 교환되고 있으며, 이와 동시에 이러한 기밀정보의 교환에 있어서 정보의 무결성과 인증을 위해 공개키 인증서가 사용되고 있다. 그러나 현 공개키 기반구조 환경에서는 공개키 인증서의 유효성을 확인하는데 있어서 전송용량 면이나 안전성면에서 여러 가지 문제점들을 내포하고 있다. 본 논문은 RSA 일방향 어큐뮬레이터를 사용하여 믿을 수 있는 정보제공자에 의해 유지되는 집합에 대한 사용자의 멤버쉽 질의에 대해 공모 가능한 디렉토리가 암호학적으로 검증된 응답을 제공하는 효율적이고 동적인 인증된 딕셔너리를 실현하였다.

ABSTRACT

The widespread use of public networks, such as the Internet, for the exchange of sensitive data that need a severe security, like legally valid documents and business transactions. At the same time public-key certificates used for sensitive data interchange form the viewpoint of data integrity and authentication. But there are some weakness of data transfer capacity and security in public key infrastructure(PKI) environment. This paper use the RSA one-way accumulator to realize an efficient and dynamic authenticated dictionary, where untrusted directories provide cryptographically verifiable answers to membership queries on a set maintained by a trusted source.

키워드

Authenticated Dictionary, PKI, Certificate, RSA one-way accumulator

Ⅰ. 서 론

현재 PKI(Public Key Infrastructure)환경에서의 공개키 인증서 인증을 위한 메커니즘은 소형기기들의 컴퓨팅 능력이나 통신네트워크상의 데이터전송능력에 비해 매우 비효율적으로 운영되고 있으며 향후 일어날 사용자의 증가로 인한 네트워크 과부하도 예상 된다. 따라서 이러한 소형기기들의 빠르고 안전한 전자상거래 및 전자금융거래를 위해서는 현 공개키 인증서의 인증 메커니즘을 더욱 효율적이고 안전하게 개선해 나가야 할 것이다.

알려진 바에 의하면 공개키 인증서의 인증 메카니즘은 일반적으로 다음 세 가지 요건들을 충족해야 한다.

첫 번째로 소형기기들의 계산량을 줄여야 한다. 계산

* 한라대학교 컴퓨터공학과

접수일자 : 2008. 2. 26

능력이 월등히 좋은 서버가 대부분의 계산을 하도록 한다던지 소형기기들이 해야 할 계산을 서버가 미리 계산함으로서 소형기기의 계산량을 최대한 줄여야 한다. 두 번째로 소형기기와의 데이터 전송량을 줄여야 한다. 무선인터넷이라든지 모뎀을 이용한 통신과 같은 데이터 전송능력이 부족한 상황에서는 데이터의 손실이나 위변조등이 쉽게 일어날 수 있다. 마지막으로 소형기기를 포함한 모든 전자상거래나 전자금융거래를 이용하는 서비스 사용자가 늘어날 경우 발생하는 네트워크 과부하를 막아야 한다. 이러한 경우에 서버는 자신이 제공해야하는 정보를 여러 개의 미러사이트를 통해 제공함으로써 사용자에게 좀 더 빠르고 안정된 서비스를 제공할 수 있다. 이는 PKI 환경이 갖는 특성상 공개키 인증서를 인증하기 위한 인증서 폐기 목록인 CRL(Certificate Revocation List)을 특정 데이터베이스에 저장하였다가 제공할 때 역시 적용해 볼 수 있다. 그러나 이러한 미러사이트들의 무분별한 증가로 인한 정보의 무결성이 불투명해지는 문제가 발생할 수도 있다. 예를 들어 전자상거래를 위한 인증정보 또는 주식거래에 필요한 주식의 가격정보 등이 미러사이트들의 조작으로 위변조 된다거나 인증서 폐기목록인 CRL이 조작되어 이미 폐기된 공개키 인증서가 계속 사용할 수 있게 되는 문제가 발생하면 사용자는 큰 피해를 입게 될 것이다. 따라서 이러한 미러사이트들이나 CRL을 저장하는 곳은 자신이 제공한 정보가 정보제공자로부터 받은 정보와 동일한 정보라는 것을 사용자에게 암호학적인 방법을 이용하여 최대한 효율적으로 검증해야한다.

본 논문의 2장에서는 기존에 제한된 방법들의 문제점들을 살펴보고 3장에서는 이러한 문제점들을 개선하는 새로운 방법을 제안한다. 4장에서는 제안한 방법을 실제 환경과 유사하게 시뮬레이션 하여 기존방법과 제안한 방법에 대한 성능을 분석한 다음, 5장을 끝으로 결론을 맺고자 한다.

Ⅱ. 기존 연구

2.1. 기존 연구

2.1.1 Goodrich 방법

2.1.1.1 스트레이트 포워드 방법

스트레이트 포워드 방법[1]은 집합 $S=\{e_1,e_2,\cdots,e_n\}$을 정보제공자가 가지고 있는 정보들의 집합이라 하고, 정보제공자는 서로 다른 큰 소수 p, q를 선택하여, $N=pq$를 계산하고 N과 서로소인 밑수 a를 고른다. 또한 원소 e_i에 대해 $h(x_i)=e_i$가 되는 식별자 x_i를 이용해 $A=a^{x_1 x_2 \cdots x_n} \bmod N$과 타임스탬프 t를 묶어 메시지 (A, t)를 모든 디렉토리에 전송한다. 이러한 초기화 과정이 끝나고 사용자가 원소 e_i가 집합 S에 속했는지에 대한 질의를 디렉토리에게 보내게 되면 디렉토리는 $A_i=a^{x_1 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n} \bmod N$ 값을 계산하여 $A_i, N, (A, t)$를 사용자에게 되돌려 준다. 사용자는 디렉토리로부터 받은 메시지 중 A_i에 x_i승을 해줌으로서 $A_i^{x_i}=A$임을 확인한다. 정보제공자는 집합 S에 대해 새로 추가되거나 삭제되는 정보가 있을 경우 디렉토리에게 업데이트 정보를 보내어 추가되는 경우는 기존의 A에 추가되는 정보 x_j승을 하여 새로운 A를 생성하고 삭제되는 경우는 해당되는 정보를 빼고 다시 A를 생성한다.

2.1.1.2 전처리 어큐뮬레이터 방법

앞서 스트레이트포워드 방법은 디렉토리가 질의를 받아서 해당되는 A_i를 만들어야하기 때문에 질의에 대한 응답시간이 늦어진다. 그러나 전처리 어큐뮬레이터[1]는 전 이진트리를 이용하여 집합 S에 속한 모든 정보들에 대하여 그들의 A_i값을 미리 계산해 놓음으로써 질의가 왔을 때 미리 계산된 A_i값을 전송하여 질의에 대한 응답시간을 단축시켰다. 이 방법은 다음의 두 단계로 나누어 수행된다.

[단계 1] Bottom up 계산

전 이진트리 T를 구성하여 그 트리의 종단노드들에 집합 S에 속한 원소들을 하나씩 배치한다. 그리고 트리 T를 후위순회로 운행하면서 각각의 노드 v에 대해서 $x(v)$를 계산한다. 만약 노드 v가 종단노드일 경우 $x(v)=x_i \bmod \Phi(N)$로 계산되어지고(x_i는 v에 배치된 값), v가 내부노드일 경우는 v의 왼쪽 자식노드 l와 오른쪽 자식노드 r에 대해 $x(v)=x(l)x(r) \bmod \Phi(N)$를 계산하여 트리 T의 모든 노드들을 계산한다.

[단계 2] Top down 계산

[단계 1]에서 계산된 전 이진트리 T를 이용하여 T의 루트노드 r에 대해서 $A(r)$=1로 정의한 후 전위순회로 트리 T를 운행하며 루트노드 r을 제외한 모든 노드들 v에 대하여 노드 v의 부모노드 z와 형제노드 w를 이용하여 $A(v) = A(z)^{x(w)} \bmod N$을 계산한다. 이렇게 계산된 트리 T의 종단 노드들에는 결국 각각의 원소들에 해당하는 A_i값들이 놓여지게 되고 사용자의 질의에 미리 계산된 A_i값을 보낼 수 있게 된다.

이러한 2단계 계산은 모두 정보제공자가 직접 계산하여야 하고 디렉토리는 정보제공자로부터 계산되어진 A_i값들만 갖게 된다. 또한 모든 업데이트 정보 역시 정보제공자가 계산하여 디렉토리에게 전송한다.

2.1.1.3 파라미터를 이용한 어큐뮬레이터 방법

이 방법은 파리미터값 p를 이용하여 집합 S를 p개의 그룹으로 나누어 계산하는 방법[1]으로 p값은 $1 \le p \le n$의 범위 내에서 정보제공자와 디렉토리의 계산 능력에 따라 유동적으로 조절될 수 있다는 것이 장점이다. 우선 집합 S를 총 p개의 부분집합 $Y_1, Y_2, \ldots, Y_p$로 나누고 각각의 집합 Y_j에 속한 원소인 e_i들은 2진 탐색 트리로 구성되며 B_j는 e_i를 포함하는 집합 Y_j의 계산된 루트값 y_j를 제외한 나머지 집합들의 루트 값들 $y_1, y_2 \ldots y_{j-1}\, y_{j+1} \ldots y_p$을 지수승한 $B_j = a^{y_1 y_2 \cdots y_{j-1} y_{j+1} \cdots y_p} \bmod N$를 계산하고 e_i에 해당하는 A_i값은 e_i가 속해있는 Y_j가 포함하는 범위를 $k \sim l$이라 하면 $A_i = B_j^{x_k x_{k+1} \cdots x_{i-1} x_{i+1} \cdots x_l} \bmod N$와 같이 계산할 수 있다. 이때 집합 Y_j의 원소들은 2진 탐색 트리로 저장되어 앞서 설명한 전처리 어큐뮬레이터의 첫 번째 단계와 같은 트리 연산을 통해 각각의 Y_j마다 개별적으로 트리를 구성하게 된다. 이러한 초기과정을 거쳐 사용자가 원소 e_i에 대해 질의를 보내게 되면 디렉토리는 e_i를 포함하는 Y_j의 트리에서 B_j를 계산하고 이를 이용해 A_i를 계산하여 기존의 방법과 같이 $A_i, N, (A, t)$를 사용자에게 되돌려 주게 된다. 또한 새로운 원소의 업데이트에 있어서 추가되거나 삭제되는 원소가 속한 부분집합 Y_j의 트리를 이용하여 B_j를 계산하고 총 p개의 부분집합들의 지수승을 통하여 A_i값을 계산하게 되므로 전 처리 어큐뮬레이터에 비해 업데이트 시간을 단축시킬 수 있다.

2.1.2 Faldella 방법

2.1.2.1 효율성

앞서 살펴본 Goodrich의 방법은 사용자의 질의에 대한 응답이나 새로운 원소에 대한 증거값을 업데이트 하는데 있어서 탁월한 성능을 갖고 있지만 원소 e_i가 집합 S에 속해 있지 않을 경우에 대한 증거 값을 제시해주기 위하여 원소들 사이의 범위값 L을 해쉬하여 Merkle hash tree[2]를 만들고 그 루트를 서명한 값과 루트로부터 종단 노드까지의 패스를 통해 증거를 제시해 주어야 한다. 이는 기본 트리 외에 Merkle hash tree를 따로 유지해야 하며 업데이트나 질의·응답시간이 어큐뮬레이터 방법에 비해 늦고 전송되는 데이터량도 많다.

Faldella의 방법[3]은 이러한 단점을 보완할 수 있는 방법으로 원소 e_i가 집합 S에 속해 있거나 그렇지 않을 경우에 상관없이 기존처럼 $A_i^{x_i} \bmod N = A$로 원소 e_i의 존재여부를 검증할 수 있다. 기존 Goodrich의 방법에서는 집합 S에 속해있는 모든 원소 e에 대해 각각의 식별자 값을 $h(x) = e$에 해당하는 적당한 큰 소수 x로 정하여 트리를 통해 미리 계산된 증거값들을 만들었으나 Faldella는 원소 e_j에 대해 각각의 식별자값 대신에 원소들 사이의 범위 값인 Statement를 유지하여 [e_i, e_{i+1}]을 SHA-1 해쉬 함수로 해쉬한 161bit값에 '1'을 덧붙여 홀수인 161bit 해쉬 값을 식별자 x_i로 사용한다. 이렇게 생성된 Statement를 증거 값과 함께 제시하면 증거 값을 검증하고 Statement값에 원소가 속해있는지 확인하면 Goodrich의 방법에서처럼 원소 e_i가 집합 S에 속해 있지 않음을 보이기 위해 추가적인 트리가 필요하지 않게 된다.

[그림 1]은 Statement에서 새로운 원소의 삽입과 삭제 과정을 그림으로 나타낸 것이다. 만약 기존에 없었던 원소 e_{r1}이 새로 삽입된다면 e_{r1}을 포함하는 범위인 e_{low1}과 e_{high1}을 찾아 그 Statement를 삭제하고 새로운 Statement인 [e_{low1}, e_{r1}]과 [e_{r1}, e_{high1}]를 생성한다. 또 존재하고 있는 원소 e_{r2}가 삭제된다면 e_{r2}를 포함하는 Statement [e_{low2}, e_{r1}]과 [e_{r1}, e_{high2}]를

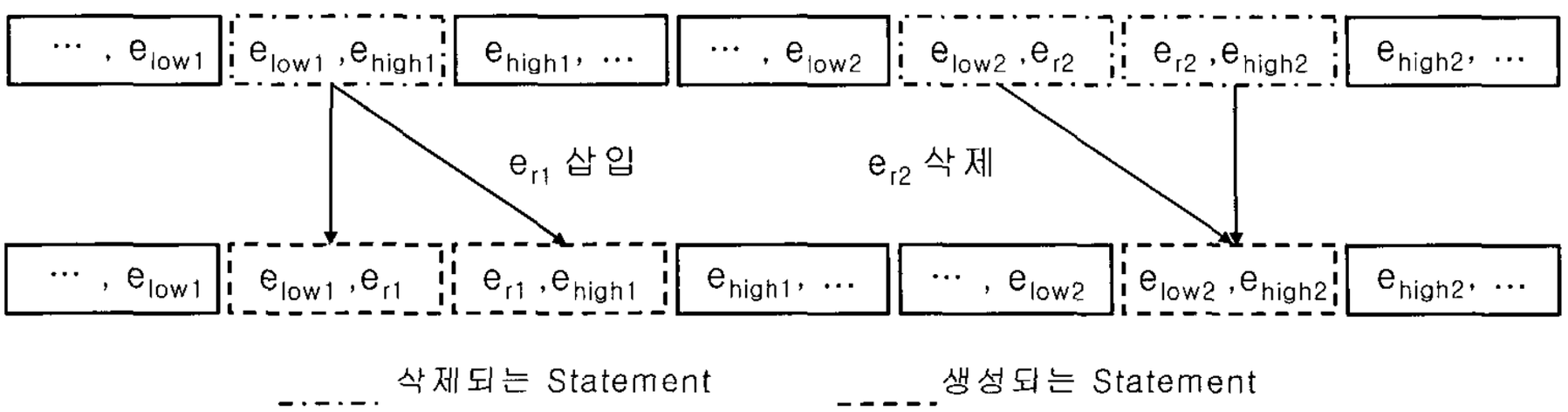


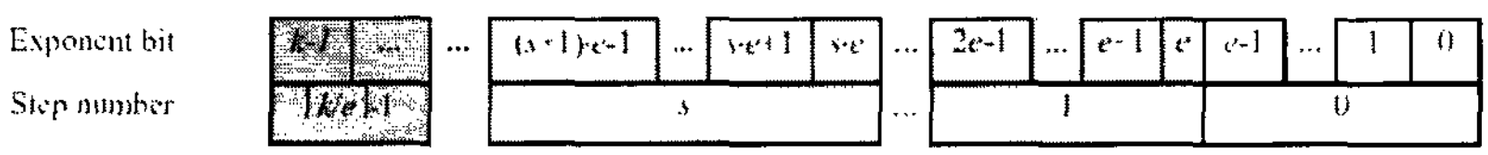
그림 1. Statement의 삽입, 삭제 과정
Fig. 1 Insert and Delete Process of Statement

삭제하고 그 양 끝 값을 하나의 Statement값인 [e_{low2}, e_{high2}]를 생성한다. 이렇게 삽입, 삭제되는 원소들은 즉시 Statement의 업데이트에 반영된다. 그러나 Faldella의 방법은 [그림 2]와 같이 멱지수의 각 bit마다 밑수 B에 지수승한 값을 미리 계산하여 테이블로 저장하여야 하는 저장 공간 면에서의 비효율성을 갖는다. 이 방법은 지수승 계산을 할 필요 없이 미리 계산된 테이블에서 해당 값을 찾아 쓰기 때문에 밑수 B에 지수승한 값을 계산하는데 걸리는 시간을 크게 단축시킬 수 있는 장점이 있으나 만약 멱지수가 N값으로 나눈 나머지 값, 즉 199bit이하라 한다면 2^{199}개의 미리 계산된 값을 보관하여야 하는 문제점을 갖고 있다. 이는 이론적으로만 가능한 계산이며 이를 실제 PKI환경에 적용한다는 것은 저장공간면에서 현실적으로 불가능하다고 볼 수 있다.

2.1.2.2 안전성

앞서 설명한바와 같이 일방향 어큐뮬레이터는 적절히 선택된 y_0와 소수 p와 q의 곱인 N값을 계수로 필요로 하는데, 이때 p와 q는 서로 다른 소수라 정의하고 있다.

Faldella 방법은 이 p와 q을 안전한 정수라 하여 소수 p'과 q'에 대해서 $2p'+1=p$, $2q'+1=q$도 소수인 소피제르맹(Sophie Germain) 소수를 사용하고 있다. 그러므로 공격자가 이미 알려진 N값을 이용하여 $\Phi(N)$을 알아내려 할 때 적절한 소수 p와 q만으로 생성된 N값보다 소피제르맹 소수를 이용하여 생성된 N값과 $\Phi(N)$이 더욱 안전하다고 할 수 있다. 또한 $N=(2p'+1)(2q'+1)$로 $\Phi(N)=4p'q'$로 표현이 가능하며 이를 소인수분해 했을 경우 각각 4, p', q'로 인수분해가 되고 따라서 각 원소의 식별자는 Statement값을 SHA-1 해쉬함수로 해쉬한 160bit 해쉬값에 1을 붙여 161bit 홀수를 만들면 $\Phi(N)$을 인수 분해한 4, p', q'값과 서로소이기 때문에 Goodrich의 방법처럼 각각의 원소 e에 대해 $h(x)=e$에 해당되는 큰 소수 x를 찾는 것



Value of the bit group		step s		step 1	step 0
0 (unused row)		1		1	1
1		$B^{(2^{se})}$		$B^{(2^{e})}$	B
2	…	$B^{(2^{se}\cdot 2)}$	…	$B^{(2^{e}\cdot 2)}$	B^2
3		$B^{(2^{se}\cdot 3)}$		$B^{(2^{e}\cdot 3)}$	B^3
…					…
2^e-1		$B^{(2^{se}\cdot(2^e-1))}$		$B^{(2^{e}\cdot(2^e-1))}$	$B^{(2^e-1)}$

그림 2. 미리 계산된 지수승 테이블
Fig. 2 Pre-computed Exponential Table

과 유사한 안전성을 갖으며 한번의 해쉬 계산만으로 식별자를 구하기 때문에 연산시간은 빠르다.

Ⅲ. 제안하는 방법

제안하는 방법은 기본적으로 미리 계산된 증거값을 이용하여 질의에 대한 응답시간을 최소화 하는데 그 목적이 있으며 파라미터를 이용한 어큐뮬레이터와 같이 파라미터 p를 이용하여 새로운 원소의 추가와 삭제에 소요되는 시간을 단축하고자 하였다. 또한 원소들의 범위값을 유지하고 그 범위값으로 식별자를 생성하여 증거값을 만들기 때문에 한번의 질의·응답으로 해당원소가 집합 S에 속하거나 그렇지 않은 경우 모두를 검증해낼 수 있다.

3.1. 식별자 생성

우선 전체 n개의 원소를 갖는 집합 $S=\{e_i \mid i=1,\ldots,n,$ 단 $e_1<e_2<\ldots<e_n\}$를 p개의 부분집합 $B_1, B_2, \ldots, B_p$로 나눈다. 다음으로 n/p개의 원소를 갖는 각각의 부분집합 B_j에 대해 그 원소들의 범위값 $[e_i, e_{i+1}]$들의 집합인 Statement를 생성한다. 생성된 Statement의 원소들인 범위값들은 각각 160-bit SHA-1 해쉬함수를 통해 해쉬된 후 마지막에 '1'을 덧붙여 홀수인 161bit 식별자 x_i를 생성하게 된다.

3.2. 초기화

각각의 부분집합 B_j들의 원소들에 대한 식별자 x_{ji}가 생성되면 앞서 말한 전처리 어큐뮬레이터와 같이 종단노드들에 부분집합의 원소들을 배열하고 종단노드일 경우 $x(v)=x_i \bmod \Phi(N)$, 내부노드일 경우는 왼쪽 자식노드 l과 오른쪽 자식노드 r를 이용하여 $x(v)=x(l)x(r) \bmod \Phi(N)$를 계산하여 트리 T_j를 구성하고 부모노드 z와 형제노드 w를 이용하여 $C(v)=C(z)^{x(w)} \bmod N$을 계산한다. 이렇게 되면 총 p개의 부분집합을 p개의 트리로 저장하게 된다. [그림 3]은 제안하는 방법의 구성도이다.

이렇게 계산된 C_{ji}들, 각 부분집합의 증거값, 그리고 타임스탬프 t를 묶은 $(C_1, t), (C_2, t), \ldots, (C_p, t)$ 값들을 디렉토리에 전송하고 디렉토리는 정보제공자가 모두 계산하여 보내준 값을 저장한다.

3.3. 질의 및 확인

사용자는 본인이 갖고 있는 공개키 인증서가 폐기목록인 집합 S에 속하는지를 확인하기 위하여 디렉토리에게 원소 e_i대한 질의를 보내고 디렉토리는 e_i가 포함되는 부분집합 B_j를 찾아 식별자 x_{ji}를 생성하여 x_{ji}가 포함되는 (C_j, t)값에 미리 계산되어진 C_{ji}값과 Statement값을 함께 해당 사용자에게 보내주게 된다. 사용자는 C_{ji}값에 x_{ji}승을 하여 C_j와 일치하는지 확인하고 Statement에 e_i이 포함되는지 확인한다. 이 방법은 p개의 부분집합에 속한 각각의 원소들에 대한 계산값이 미리 계산되어 있기 때문에 기존의 파라미터를 이용

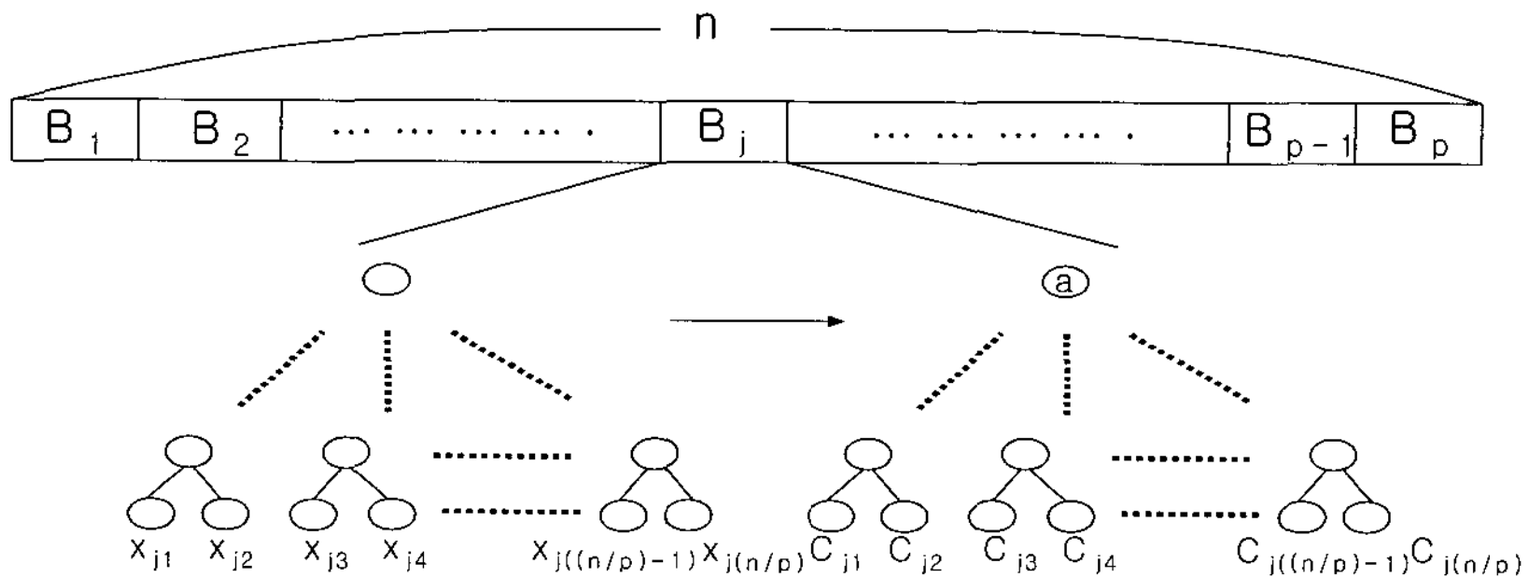


그림 3. 제안하는 방법
Fig. 3 Proposed Scheme

한 어큐뮬레이터 방법에 비해 매우 빠르게 질의에 대한 응답할 수 있게 해준다. 이러한 인증된 딕셔너리는 사용자가 질의를 보내 e_i가 집합 S에 속하는지, 다시 말해 폐기된 인증서인지를 확인하는 과정에서 e_i가 집합 S에 속해있지 않을 경우에도 Kocher[4]가 제안한 방법처럼 각 원소들간의 범위를 나타내는 $r_i=[x_i, x_{i+1}]$ 값으로 해쉬트리를 구성하고 이 트리의 루트값에 서명하여 질의에 응답해 주는 방법을 사용하지 않고도 전송된 Statement에서 원소 e_i가 범위값에 포함되는지 아니면 범위값의 맨 앞에 있는지 만을 판단하여 원소 e_i가 집합 S에 속해 있지 않음을 확인시켜줄 수 있다.

3.4. 업데이트

우선 새로운 원소가 삽입될 경우를 살펴보면 정보제공자는 자신이 갖고 있는 정렬된 p개의 부분집합 중에서 해당원소의 포함되는 부분집합을 찾아서 Statement를 갱신하고 새로운 식별자를 생성하여 해당되는 부분집합의 트리만을 재구성하면 된다. 이때 각각의 부분집합 원소들은 위의 3.1절 초기화에서 설명하였듯이 정수형으로 표현하였을 때의 크기순서로 정렬되어 있기 때문에 기존의 방법처럼 해당원소가 속한 부분집합과 또 그 해당 범위값을 찾는데 걸리는 시간이 크게 단축된다. 다음으로 원소가 삭제될 경우에는 삭제될 원소를 포함하고 있는 부분집합에서 해당원소를 삭제한 후 Statement를 갱신하여 식별자를 구하고 그 부분집합의 트리만을 재구성하면 된다. 이는 기존의 전처리 어큐뮬레이터가 전체 원소를 포함하는 트리를 재구성하는데 비해 매우 적은 계산량이 요구된다. 마지막으로 특정 부분집합에 새로운 원소가 계속적으로 삽입되거나 또는 원소들이 특정 부분집합에서 계속 삭제되어 부분집합의 크기가 매우 커지거나 또는 매우 작아지는 경우에는 해당 부분집합을 두개의 부분집합으로 나누거나 주변 부분집합에 합치는 방법으로 업데이트가 한곳에서만 집중적으로 일어나는 경우에 대비하였다.

3.5. 성능분석

기존에 어큐뮬레이터를 이용한 인증된 딕셔너리는 앞서 2장에서 밝힌바와 같이 Goodrich의 방법 세 가지와 Faldella의 방법으로 나누어 볼 수 있다. 이 방법들은 모두 사용자에게 적은 계산량을 할당하도록 설계되었으나 질의에 대한 응답이나 업데이트에 필요한 계산량이 상대적으로 많아지는 단점이 있다. 또한 미리 계산된 지수승 테이블이 너무 많은 저장공간을 차지하는 문제점도 있다. 제안하는 새로운 방법은 이러한 단점들을 좀더 효율적으로 개선하고자한 것으로 아래 [표 1]은 기존의 방식들과 제안하는 방식의 실행시간을 비교한 표이다.

[표 1]에서 보는 바와 같이 질의에 대한 실행시간은 $O(1)$로 파라미터를 이용한 어큐뮬레이터의 $O(n/p)$와 비교하였을 때 빠르게 실행되는 것을 확인할 수 있다. 이는 제안하는 방법이 열악한 네트워크 환경에서도 질의에 대한 빠른 응답이 가능하다는 것을 말하고 있다. 또한 삽입, 삭제와 같은 업데이트 시간은 파라미터를 이용한 어큐뮬레이터가 $O(p+\log(n/p))$이고 제안하는 방법이 $O(n/p)$으로 p값의 변화에 따라 유동적이다.

[표 2]는 파라미터를 이용한 어큐뮬레이터와 제안하는 방법에서의 업데이트 시간과 질의에 대한 응답시간

표 1. 실행시간 비교 (※m은 N값의 bit size)
Table. 1 Runtime Comparison

방 법	공간	삽 입	삭 제	업데이트 정보	질 의	질 의 정보	확 인
스트레이트 포워드	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(1)$
전처리 어큐뮬레이터	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(1)$
파라미터를 이용한 어큐뮬레이터	$O(n)$	$O(p+\log(n/p))$ $O(p+\log(n/p))$	$O(p+\log(n/p))$	$O(p)$	$O(n/p)$	$O(1)$	$O(1)$
Faldella 방법	$O(n+2^m)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
제안하는 방법	$O(n)$	$O(n/p)$	$O(n/p)$	$O(n/p)$	$O(1)$	$O(1)$	$O(1)$

표 2. p값에 따른 실행 시간비교
Table. 2 Runtime Comparison by Value P

방 법	업 데 이 트			질 의		
	$p=\sqrt{n}$	$p\to 1$	$p\to n$	$p=\sqrt{n}$	$p\to 1$	$p\to n$
파라미터를 이용한 어큐뮬레이터	$O(\sqrt{n}+\log n)$	$O(\log n)$	$O(n)$	$O(\sqrt{n})$	$O(n)$	$O(1)$
Faldella 방법	$O(1)$			$O(1)$		
제안하는 방법	$O(\sqrt{n})$	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$

을 p값의 변화에 따라 비교한 것이다. [표 2]에서는 p값이 $\sqrt{n}$을 기준으로 $p\geq\sqrt{n}$일 경우 파라미터를 이용한 어큐뮬레이터보다 제안하는 방법이 질의 시간과 업데이트 시간이 모두 월등히 빨라짐을 볼 수 있다. 또한 $p\leq\sqrt{n}$일지라도 질의 시간이 파라미터를 이용한 어큐뮬레이터보다 빠르기 때문에 제안하는 방법이 열악한 네트워크 환경을 갖는 소형기기에서는 더욱 안정적으로 동작할 수 있다.

Ⅳ. 시뮬레이션 및 성능분석

본 절에서는 앞서 3장에서 설명한 Goodrich와 Faldella의 방법과 4장에서 설명한 새롭게 제안하는 방법에 대해 실제 실험을 통해 그 결과를 분석해 보고자 한다.

4.1. 시뮬레이션 환경

먼저 본 시스템의 하드웨어 및 소프트웨어 환경은 [표 3] [표 4]와 같다.

표 3. 시뮬레이션 하드웨어 환경
Table. 3 Simulation Hardware Environment

개 체	CPU(RAM)	OS
사용자	notebook Pentium Ⅳ 700MHz (128M)	Windows XP
디렉토리	desktop Pentium Ⅳ 1.8GHz (512M)	Windows XP

[표 3]은 본 시뮬레이션에 사용된 하드웨어를 나열한 것이다. 사용자는 실제 소형기기나 열악한 네트워크 환경에 유사하게 하기 위하여 노트북을 사용하였으며 디렉토리는 계산량을 고려해 좀더 나은 성능을 갖는 데스크탑 PC를 사용하였다. OS는 Windows XP를 사용하여 XP자체에 기본적으로 내제되어있는 암호 모듈을 응용할 수 있도록 하였다. [표 4]는 시뮬레이션에 사용된 소프트웨어를 나열한 것이다. 언어는 Visual C++를 사용하여 코딩하였고 기타 데이터베이스, 큰 소수, microseconds 시간측정을 위해 필요한 함수들은 기존에 MFC로 작성되어 있는 모듈들을 사용하였으며 Microsoft Platform SDK를 최신버전으로 업데이트 하여 필요로 하는 암호 모듈을 사용할 수 있었다. 그 외에 원소들의 식별자인 해쉬값은 160-bit SHA-1 해쉬값에 '1'을 붙여 홀수인 161bit 정수값을 사용하였으며 RSA 일방향 어큐뮬레이터의 매개변수 N값은 199-bit 정수값을 사용하였다. 이때 N값을 정하는 p와 q값은 각각 소수 p'과 q'에 대해서 $2p'+1=p$, $2q'+1=q$도 소수인 소피제르맹(Sophie Germain) 소수를 사용하여 그 안전성을 더욱 높였다.

표 4. 시뮬레이션 소프트웨어 환경
Table. 4 Simulation Software Environment

소프트웨어	종 류	버 전
Visual C++	Microsoft Visual C++	6.0
Platform SDK	Microsoft Platform SDK February 2003	5.2.3790.0

4.2. 성능분석

본 연구의 주된 결과는 [그림 4][그림 5][그림6][그림 7]에서 확인할 수 있다. 여기서 x축은 총 1만개의 원소들을 p개의 부분집합으로 나눴을 경우 p값의 변화를 나타낸 것이고 y축은 주어진 연산을 수행하는데 걸리는 평균시간을 microseconds단위로 나타낸 것이다. [그림 4]는 사용자가 디렉토리에게 보낸 질의에 대한 증거

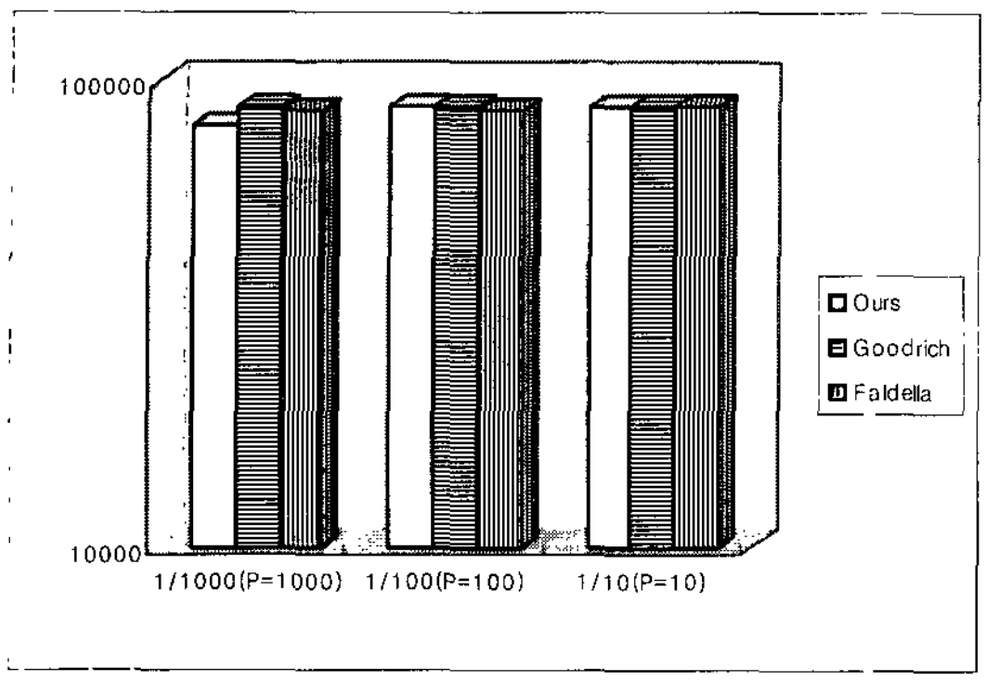


그림 4. 사용자의 증거값 검증 시간비교
Fig. 4 Verification Time Comparison by Proof Value of User

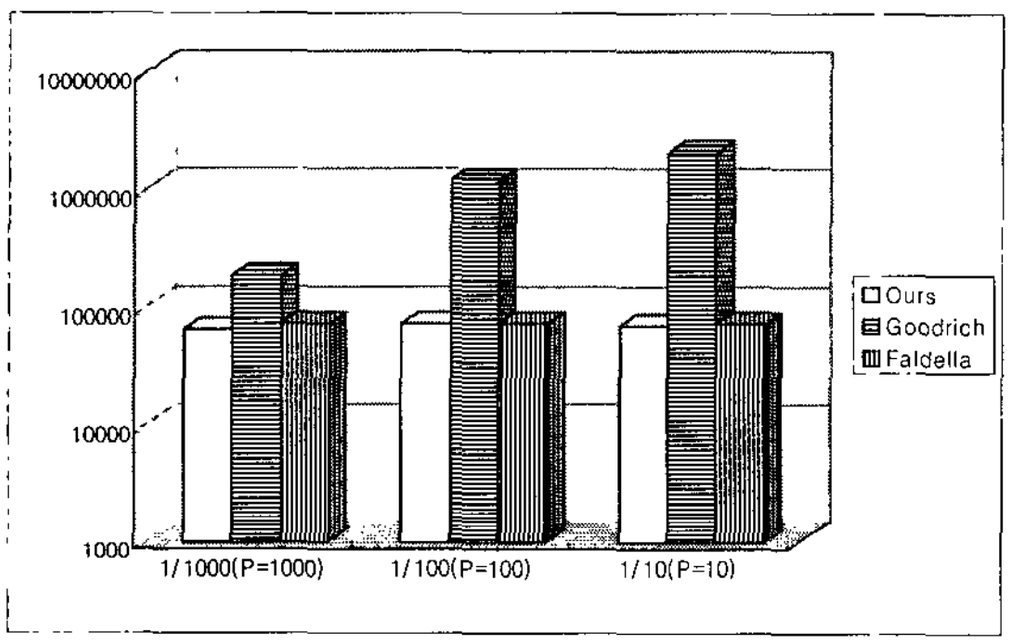


그림 5. 디렉토리의 증거값 생성 시간비교
Fig. 5 Generation Time Comparison by Proof Value of Directory

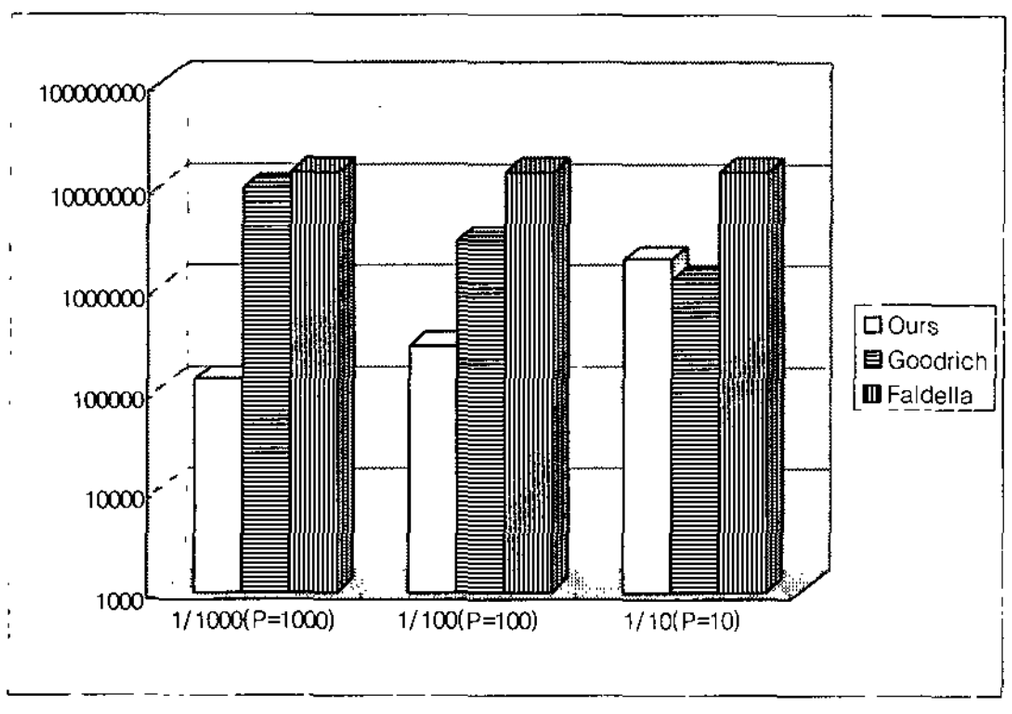


그림 6. 디렉토리의 업데이트 시간비교
Fig. 6 Update Time Comparison of Directory

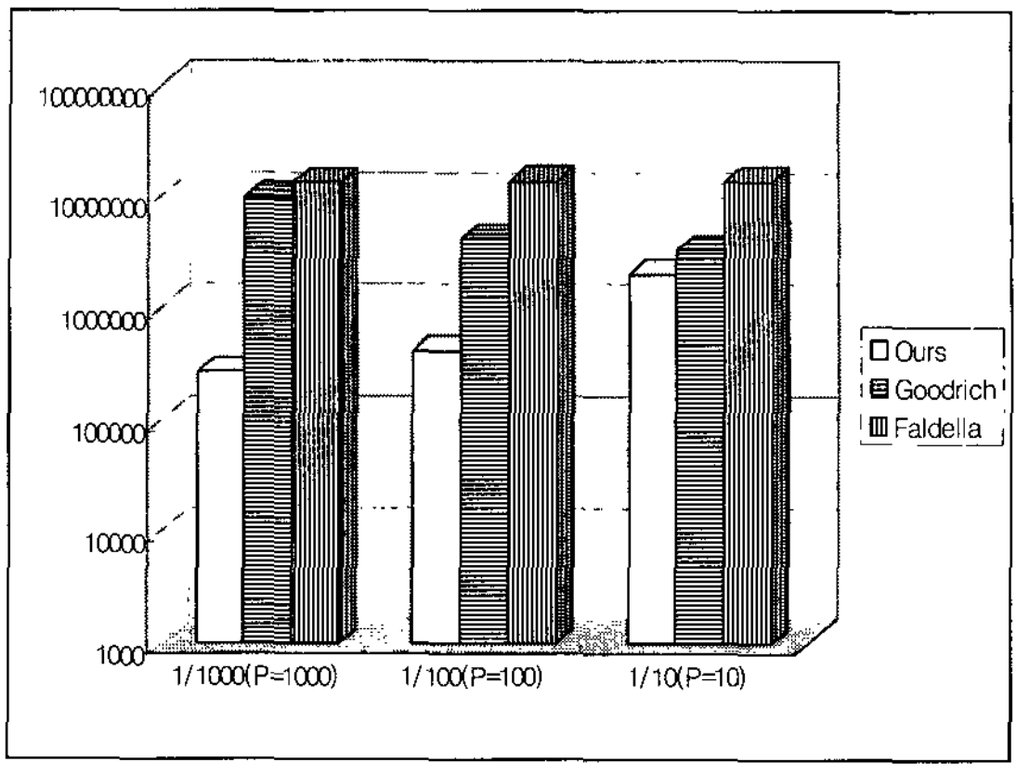


그림 7. 전체 수행시간비교
Fig. 7 Total Execution Time Comparison

값을 받아 해당 원소의 해쉬값이 **statement**에 속하는지 확인하고 $A_i^{x'} \bmod N = A$를 계산하여 해당 원소가 폐기되었지 또는 그렇지 않은지를 확인하고 질의에 대한 증거값을 검증하는데 걸린 시간을 측정한 것이다. 기존의 방법과 새로 제안하는 방법 모두 $O(1)$값을 갖고 있으며 실제 실험 결과에서도 **0.1**초 정도로 비슷한 결과를 보이고 있다.

[그림 5]는 디렉토리가 사용자로부터 받은 질의에 대해 증거값을 받는데 걸린 시간을 측정한 값이다. 이를 [표 4]에서 이론적으로 계산된 값과 비교 해보면 제안하는 방법은 p값의 변화와 상관없이 $O(1)$을 유지하고 있는 반면에 **Goodrich**의 방법은 p값이 감소함에 따라 즉 부분집합의 원소 수들이 증가함에 따라 $O(1)$에서 $O(n)$로 증가하고 있으며 이는 실제 실험상에서도 **185043 microseconds**에서 **2030527microseconds**로 증가하는 것을 볼 수 있다. **Faldella**의 방법은 전체 원소를 p개의 부분집합으로 나누지 않고 어큐뮬레이터 계산을 빠르게 실행시키기 위한 미리 계산된 테이블을 사용하기 때문에 p값의 변화에 상관없이 일정하게 유지된다.

[그림 6]은 정보제공자로부터 받은 삽입 또는 삭제 요청에 의해 디렉토리에서 새로운 원소를 업데이트하는데 걸린 시간을 측정한 값이다. **Faldella**의 방법은 역시 전체 원소를 p개의 부분집합으로 나누지 않기 때문에 일정한 값을 유지한다. [표 4]의 이론적으로 계산된 값과 비교 해보면 **Goodrich**방법은 p값이 작아짐에 따라 $O(n)$에서 $O(\log n)$로 연산시간이 줄어들 듯이 실제 실험에

서도 9752928microseconds에서 1190952 microseconds로 줄어든다. 제안하는 방법은 p값이 작아짐에 따라 $O(1)$에서 $O(n)$로 즉 125180 microseconds에서 191357microseconds로 증가하는 것을 볼 수 있다. 그러나 Goodrich방법은 원소의 식별자를 계산하는 시간이 상대적으로 오래 걸리기 때문에 전체적인 연산시간은 제안하는 방법에 비해 비효율적이다.

위의 그림들 외에 앞서 초기화에서 설명했던 원소들의 식별자인 해쉬값을 하나 생성하는데 걸린 시간은 평균 4milliseconds로 기존 Goodrich방법의 45millisecond[1]보다 약 1/10정도의 시간감소를 보였다.

[그림 4][그림 5][그림 6]을 통해 시뮬레이션 결과를 모두 종합해 보면 p값의 변화에 따라 각 방법의 성능은 조금씩 차이를 보이지만 결국 [그림 7]에서 보는 바와 같이 제안하는 방법이 p값과 상관없이 모두 좋은 성능을 보이고 있다. 단 전체 원소의 개수를 n이라 하였을 때 p값이 $\sqrt{n}$보다 커지면 제안하는 방법이 다른 방법들에 비해 월등히 좋은 성능을 나타내는 것을 볼 수 있다. p값이 커지게 되면 원소의 개수는 줄어들게 되고, 이는 전체 원소의 개수를 더욱 많은 개수의 부분집합으로 나눈다는 의미로 각각의 부분집합별로 계산된 증거 값을 제공해주는 제안하는 방법에서는 각 부분집합별 원소의 개수가 줄어들어 업데이트가 일어날 때마다 다시 계산되어야 하는 원소의 개수가 적어지고 이 때문에 전체적인 수행시간은 빨라지게 된다.

V. 결 론

본 본문에서 제안하는 바와 같이 질의에 대한 실행시간은 $O(1)$로 파라미터를 이용한 어큐뮬레이터 방식의 $O(n/p)$와 비교하였을 때 빠르게 실행되는 것을 확인할 수 있었다. 이는 제안하는 방법이 열악한 네트워크 환경에서도 질의에 대한 빠른 응답이 가능하다는 것을 말하고 있다. 또한 삽입, 삭제와 같은 업데이트 시간은 파라미터를 이용한 어큐뮬레이터가 $O(p+\log(n/p))$이고 제안하는 방법이 $O(n/p)$으로 p값의 변화에 따라 유동적이다. 또한 파라미터를 이용한 어큐뮬레이터 방법을 제안하는 방법에서의 업데이트 시간과 질의에 대한 응답시간을 p값의 변화에 따라 비교해 보면 p값이 $\sqrt{n}$을 기준으로 $p \geq \sqrt{n}$일 경우 파라미터를 이용한 어큐뮬레이터보다 제안하는 방법이 질의 시간과 업데이트 시간이 모두 월등히 빨라짐을 볼 수 있다. 결론적으로 말해 이것은 제안하는 방법이 파라미터를 이용한 어큐뮬레이터 방법보다 높은 효율성을 갖기 때문에 열악한 네트워크 환경을 갖는 소형 기기에서는 더욱 안정적으로 동작할 수 있음을 의미한다.

참고문헌

[1] P. C. Kocher. On certificate revocation and validation. In Proc. International Conference on Financial Cryptography, volume 1465 of Lecture Notes in Computer Science, 1998.

[2] M. T. Goodrich, R. Tamassia and J. Hasic. An Efficient Dynamic and Distributed Cryptographic Accumulator. Johns Hopkins Information Security Institute, 2002.

[3] E. Faldella. A Flexible Scheme for On-Line Public-key Certificate Status Updating and Verification, Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02), 2002.

[4] P. C. Kocher. On certificate revocation and validation. In Proc. International Conference on Financial Cryptography, volume 1465 of Lecture Notes in Computer Science, 1998.

저자소개

김 순 석(Soon-Seok Kim)

1997년 2월 진주산업대학교 컴퓨터공학과(공학사)
1999년 2월 중앙대학교 컴퓨터공학과 (공학석사)
2003년 2월 중앙대학교 컴퓨터공학과(공학박사)
2003년 3월 ~ 현재 한라대학교 컴퓨터공학과 조교수
※관심분야 : 정보보호, 암호응용, 생체보안

이 용 희(Yong-Hee Lee)

1991년 2월 한양대학교 전자공학과 (공학사)
1993년 2월 한양대학교 전자공학과 (공학석사)
1998년 2월 한양대학교 전자공학과(공학박사)
1999년 3월 ~ 현재 한라대학교 컴퓨터공학과 교수
※관심분야 : 의용전자, 임베디드시스템, U-헬스

이 강 우(Kang-Woo Lee)

1997년 2월 홍익대학교 전자계산학과 (이학박사)
1994년~2002년 서남대학교 정보통신공학부 교수
2002년~현재 한라대학교 컴퓨터공학과 교수
※관심분야 : 데이터베이스, 유비쿼터스컴퓨팅