情報保護學會論文誌
第14券 第5號, 2004. 10

보안 요구 수준에 근거한 효율적인 패킷 암호화 기법

노 지 명†, 양 정 민‡
대구가톨릭대학교

An Efficient Packet Encryption Scheme Based on Security Requirement Level

Ji-Myong Nho†, Jung-Min Yang‡
Catholic University of Daegu

요 약

온라인 게임과 같이 대규모로 데이터를 주고받는 클라이언트-서버 구조의 서비스에서 정보 보호를 위해 패킷을 암호화하면 암호화로 인해 서버에 과도한 부하가 유발되고 서비스에 심각한 문제가 생길 수 있다. 따라서 데이터의 보안 요구에 맞게 보다 효율적인 암호화 기법을 사용하여 암호화 부하를 되도록이면 줄이는 것이 필요하다. 본 논문에서는 암호화로 인한 부하를 줄이는 방법으로 복수의 암호 알고리즘을 이용하여 데이터의 보안 요구에 따라 암호화 체계를 달리하는 패킷 암호화 기법을 제안한다. 이러한 기법은 중요도가 서로 다른 다양한 종류의 데이터들이 끊임없이 전송되는 클라이언트-서버 구조의 인터넷 서비스에서 잘 적용될 수 있다. 실험을 통하여 본 논문에서 제안한 방법의 효율성을 정량적으로 검증한다.

ABSTRACT

Under a large-scale client-server service environment, e.g., online games, encrypting data for acquiring information security often causes overload to the server and hence degradation of the service itself. Therefore, for reducing encryption payload, it is necessary to use adequately an efficient encryption scheme with respect to the security requirements of transmission data. In this paper, we propose a packet encryption scheme using multiple cryptosystems to realize such capability, which assigns a different cryptosystem according to the security requirements level. The proposed encryption scheme is applicable to internet services with heavy traffic ratios in which different kinds of data packets are incessantly transmitted between clients and servers. To show its effectiveness and superiority, the performance of the proposed encryption scheme is verified by experiments.

Keywords : *Block Cypher, Multiciphering, Selective Encryption, Packet Encryption*

Ⅰ. 서 론

21세기에 들어서 인터넷의 규모가 급격히 확산되면서 네트워크의 국지적인 보안 문제가 예전처럼 쉽게 통제될 수 없으며, 어떤 보안 문제가 발생하면 인터넷을 통해 걷잡을 수 없이 퍼져 나가버려 더 큰 문제로 변하는 현상이 보편화되고 있다. 특히 온라인 게임같이 수많은 사용자가 접속하여 데이터를 주고받는 인터넷 서비스에서는 극소수의 사용자만이 해킹 등으로 불공정한 서비스 이용을 했다고 해도 해킹 도구의 온라인 전파, 보안 결함에 대한 대중적 여론 악화 등으로 해당 서비스는 치명적인 손실을 입게 된다[1].

이러한 인터넷 서비스에서 일어날 수 있는 보안

접수일 : 2004년 6월 28일 ; 채택일 : 2004년 10월 5일
† 주저자 : jmnho@cu.ac.kr
‡ 교신저자 : jmyang@cu.ac.kr

문제는 사용자 인증, 사용자 정보 등과 같은 기본적인 보안 정보의 유출뿐만 아니라 클라이언트 프로세스 해킹, 클라이언트-서버 간 네트워크 패킷 분석 및 조작, 서비스용 아이템(item)의 불공정한 거래[2] 등 다양하게 존재한다. 이와 같은 보안 문제를 해결하기 위한 가장 기본적인 대책은 다른 정보 보호 분야에서와 마찬가지로 클라이언트와 서버가 서로 주고받는 데이터를 암호화 하는 일이다[1,3]. 하지만 현재 실행중인 많은 인터넷 서비스가 데이터 암호화 등 네트워크 보안에 신경을 많이 쓰지 못하고 있다. 이러한 현상이 발생하게 된 가장 큰 이유는 데이터 패킷을 암호화하고 복호화 하는 모듈이 서버에 과도한 부하를 주어 서비스의 성능에 심각한 영향을 미치기 때문이다[3]. 제공되는 콘텐츠가 금융이나 쇼핑 같은 과금(課金) 서비스이면 속도보다 안전을 절대적으로 우선시할 수 있겠으나, 대규모 온라인 게임 같이 실행 속도가 중요시되고 보안 기준에 대한 제도적 장치가 아직까지 미비한 서비스에서는 서비스 제공자가 안전 확보를 위해 데이터 암호화를 수용하기가 어려운 실정이다[4]. 또한 가상사설망(VPN)[5]과 같은 보안망을 사용하는 경우 사용자 측에서 부가적인 하드웨어가 필요하게 되고 경우에 따라서 사용 제한이 가해지기 때문에 적절한 서비스 수익을 기대할 수 없다. 따라서 이러한 대규모 인터넷 서비스용 클라이언트-서버 구조에 적합한 네트워크 보안 모듈은 서비스 제공자 측이 수용할 수 있을 정도의 서비스 속도 및 성능을 최대한 보장하면서 동시에 적절한 수준의 패킷 암호화를 수행할 수 있어야 한다.

본 논문에서는 이러한 문제점을 해결하기 위한 한 방안으로서 복수의 암호화 체계를 이용한 패킷 암호화 기법과 이 기법을 이용한 새로운 네트워크 보안 모듈 구조를 제안한다. 본 논문에서 제안하는 암호화 기법은 전송되는 데이터의 보안 요구 수준에 따라서 우선순위를 결정하고 그 우선순위에 따라서 강도가 다른 암호 알고리즘으로 패킷을 암호화하는 방법이다. 중요한 패킷은 높은 암호화 강도를 가지는 암호 알고리즘으로 암호화하여 네트워크 상의 공격에 대해 안전도를 높이고, 상대적으로 중요도가 떨어지는 데이터는 비교적 처리 속도가 빠르고 암호화 강도가 낮은 알고리즘을 사용하여 암호화함으로써 암호화로 인한 부하를 줄이고 데이터의 보안은 요구 수준 이상으로 일정하게 유지한다. 이 기법은 중요도가 서로 다른 다양한 종류의 데이터들이 끊임없이 전송되는 클라이언트-서버 구조의 인터넷 서비스에서 잘 적용될 수 있다.

본 논문에서는 먼저 복수 개의 암호를 이용한 패킷 암호화 기법을 제안하고 이 기법을 구현한 네트워크 암호화 모듈의 상세 구조를 설명한다. 그리고 제안된 기법의 암호화 처리 부하 분석을 통해 암호화 알고리즘을 선택하는 기준에 대해 설명한다. 제안된 네트워크 보안 모듈은 데이터의 중요도를 저장하고 갱신하는 세부 모듈과 복수의 암호 알고리즘을 선택해서 사용하는 암호화 및 복호화 세부 모듈로 이루어진다. 마지막으로 본 논문에서 제안된 패킷 암호화 기법의 효용성을 실험을 통해 정량적으로 검증한다.

II. 관련 연구

복수 개의 암호를 이용한 암호화 개념은 Shannon이 여러 개의 보안 시스템들이 결합된 "암호들의 암호(Cipher of Ciphers)" 개념을 처음 제안한 것에서부터 시작되었다[6]. Shannon은 하나의 평문(plain text)을 서로 다른 암호 알고리즘으로 여러 번 암호화한 다중 암호화(multiple ciphering)와 여러 개의 암호들 중 랜덤(random)하게 하나를 골라서 암호화 하는 랜덤 암호화(random ciphering) 방법을 제안하였다. 이 중 다중 암호화 방식은 오늘날에도 3-DES 같이 동일한 블록 암호를 여러 번 적용하는 방법[7]이라든가 광영상 암호를 추가하여 사용자 인증을 구현하는 스마트카드[8] 등에서 사용되고 있다.

한편 Ritter는 현대의 암호 알고리즘 사용 체계에서 상용화된 기존 암호 알고리즘이 시간이 지나면 여러 가지 분석 및 공격(cryptanalysis)으로부터 노출이 되어 결국 그 안전성이 위협해져 새로운 암호 알고리즘이 개발되기까지 기다려야 하는 문제점을 말하였다[9]. Ritter는 또한 새로운 알고리즘이 개발되면 기존 알고리즘의 효용성이 급작스럽게 떨어져버리는 문제점도 지적하고 복수 개의 암호를 이용하는 방법을 그 대안으로 제시하였다[9]. 즉 생성된 키의 값에 따라서, 아니면 랜덤하게 복수 개의 암호 알고리즘 중 하나를 선택하여 암호화를 수행하면 암호 알고리즘 자체의 복잡도와 더불어 복수 개의 암호들 중 하나를 선택할 때 생기는 복잡도도 얻을 수 있다는 것이다. Ritter는 사용할 수 있는 암호 알고리즘의 개수에 따라서 얻어지는 실제 암호 스택(stack)의 개수는 지수함수적으로 늘어난다고 주장하였다[10]. Roellgen은 이러한 복수 개의 암호를 이용한 암호 알고리즘을 확장하여 다형성 암호(Polymorphic Encryption)를 개발하였다[11]. 다형성 암호는 기본적으로 스트림 암

호(stream cipher)와 동일한 구조를 가지는데 의사 난수(pseudo-random number)를 이용하여 키 스트림 생성자(key stream generator)를 여러 가지 암호로부터 다형적으로 만든다는 것이 알고리즘의 핵심이다. Roellgen은 랜덤하게 선택된 암호 알고리즘에 맞게 암호화 모듈을 실시간으로 컴파일하는 방법을 이용하여 다형성 암호의 수행 속도를 빠르게 할 수 있고, 여러 가지 암호 알고리즘을 사용함으로써 암호화 모듈의 엔트로피는 필연적으로 증가한다는 것을 밝혔다[12]. 이 밖에도 영상이나 음성 등 멀티미디어 데이터의 전송 분야에서는 복수 개의 암호를 사용하지는 않지만 데이터의 중요도에 따라서 일부 데이터만 선택해서 보내는 선택적 암호화(Selective Encryption) 방법이 최근 널리 쓰이고 있다[13,14].

본 논문에서 제안하고자 하는 암호화 기법은 Roellgen의 방법과 유사하나 의사 난수를 만들어 랜덤하게 암호 알고리즘을 사용하지 않고 데이터 패킷의 중요도에 따라서 강도가 다른 암호를 선택하는 방법을 쓴다. 즉 복수 개의 암호를 사용하는 "암호들의 암호" 기법과 멀티미디어 데이터 전송에서 사용되는 선택적 암호화 방법이 결합된 형태라고 말할 수 있다. 이 기법은 앞에서도 말했듯이 중요도가 서로 다른 다양한 종류의 데이터들이 끊임없이 전송되는 클라이언트-서버 구조의 인터넷 서비스에서 속도 면에서 효율적인 보안 모듈을 구현하는 데 잘 적용될 수 있다.

Ⅲ. 데이터의 보안 요구 수준에 근거한 암호화 모듈

3.1 암호화 모듈 구조

본 논문에서 제안하는 암호화 모듈의 구조는 그림 1과 같다. 암호화 모듈은 서버와 클라이언트에서 동일하게 패킷 우선순위 테이블(Packet Priority Table), 우선순위 매핑 테이블(Priority Mapping Table), 패킷 분석 및 암호화/복호화(Packet Analysis & Encryption/Decryption) 등 세 개의 세부 모듈을 가지며, 서버 부분에 트랙픽 모니터(Traffic Monitor)가 추가로 장착될 수 있다.

패킷 우선순위 테이블은 전송되는 패킷이 가지는 보안 요구 수준을 표시한 것으로서 각 등급마다 서로 다른 암호 알고리즘이 할당된다. 복수 개의 암호를 이용한 암호 알고리즘의 기본 아이디어는 데이터의 중요도에 따라서 보안 요구 수준을 달리 하고 이에 따라 서로 다른 암호 알고리즘을 사용하여 패킷을 암호화하는 것이기 때문에 패킷 우선순위 테이블은 전체 암호화 모듈에서 가장 핵심적인 역할을 담당한다고 말할 수 있다. 즉 중요도 등급이 높아질수록 높은 암호화 강도를 가지는 암호 알고리즘이 사용되며 등급이 낮은 데이터의 경우 비교적 처리 속도가 빠르고 암호화 강도도 낮은 알고리즘이 적용된다. 전송되는 데이터의 특성이 매우 다양한 인터넷 서비스의 경우 다른 사용자나 해커에게 공개되어도 전혀 문제가 없는 데이터가 있을 수 있으므로 가장 낮은 등급을 가지는 데이터 패킷은 암호화를 전혀 하지 않고 보낼 수도 있다. 또한 다른 사용자에게 공개되면 치명적인 손실을 가져올 수 있는 매우 중요한 데이터의 경우 암호화 처리 부하를 감수하면서 아주 높은 강도를 지니는 암호 알고리즘을 적용한다.

패킷 우선순위 테이블은 발생되는 패킷의 중요도 분석을 통해 결정되며, 패킷으로부터 우선순위를 식별하기 위한 정보와 우선순위에 따라 적용될 암호체계에 대한 정보를 담고 있다. 패킷 우선순위 테이블을 만드는 방법은 제안할 암호화 모듈을 설계 중인

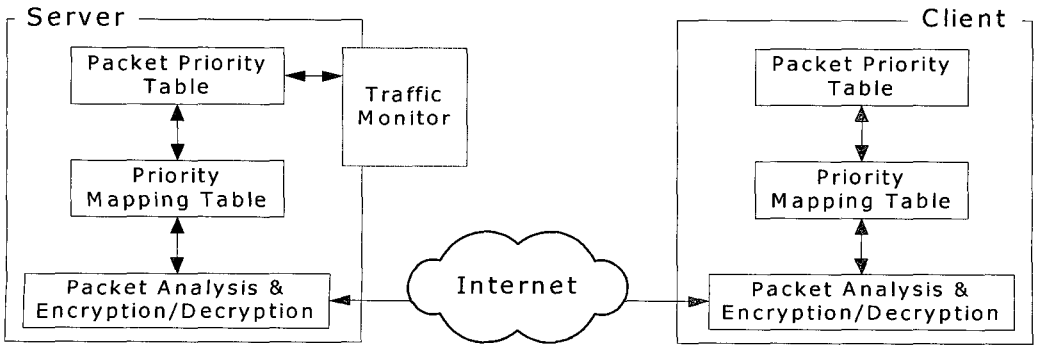


그림 1. 복수 개의 암호를 이용한 암호화 모듈 구조.

인터넷 서비스에 적용할 경우와 이미 운용되고 있는 서비스에 적용할 경우에 대해서 각각 다르게 설정된다. 설계 중인 서비스의 경우에는 패킷 헤더에 우선순위 식별자를 포함하여 패킷을 설계하고 식별자에 대한 정보를 우선순위 테이블에 반영하며, 이미 운용되고 있는 서비스의 경우에는 현존하는 패킷 헤더의 특성을 분석하여 우선순위 식별 기준을 정하고 그 기준 정보를 우선순위 테이블에 저장한다.

복수 개의 암호를 이용한 암호 알고리즘을 보다 능동적으로 사용하기 위해서는 데이터의 중요도와 더불어 온라인상에서 발생하는 패킷의 발생 빈도를 실시간으로 분석하는 방법이 필요하다. 패킷의 발생 빈도는 인터넷 서비스의 온라인 상태에 따라서 가변적이므로 우선순위 테이블에서 높은 등급을 가지는 패킷이 갑자기 많이 발생하면 암호화로 인한 처리 부하가 심각하게 증가하는 경우가 생길 수 있다. 이러한 상황을 막기 위해서는 패킷의 등급을 낮추어 보다 낮은 암호화 강도를 가지는 암호 알고리즘으로 암호화하거나 그 패킷이 속한 등급에 적용되는 암호 알고리즘 자체의 강도를 낮추어야 한다. 따라서 패킷 트래픽을 실시간으로 분석하고 그 결과를 우선순위 테이블에 피드백(feedback)해주는 트래픽 모니터(Traffic Monitor)[15,16] 기능이 암호화 모듈에 탑재되면 모듈 성능은 더욱 향상될 것이다. 본 논문은 이와 같은 트래픽 모니터를 이용한 능동적 대응에 대한 연구 결과는 포함하지 않는다.

우선순위 매핑 테이블은 패킷 우선순위 테이블에 있는 등급 순서를 치환(permutation)한 것으로서 암호 알고리즘의 혼돈성을 더욱 키우는 역할을 한다. 그림 1에서 볼 수 있듯이 데이터 패킷은 암호화되기 전에 패킷이 속한 중요도 등급 값을 패킷 우선순위 테이블로부터 직접 받지 않고 우선순위 매핑 테이블에 있는 치환된 등급 값으로 취한다. 우선순위 매핑 테이블을 사용하는 것은 암호 스택[8]에 대한 일종의 암호화로서 등급별 암호 알고리즘 테이블을 2단계 구조로 만드는 효과를 낸다. 즉 외부 공격자가 패킷을 가로채어 사용된 암호를 가리키는 헤더(header)값을 추출할 수 있다고 하더라도 그 정보는 우선순위 테이블이 다시 한 번 매핑된 값이므로 실제로 어떤 암호 알고리즘이 사용되었는지 찾아내기가 더욱 어렵게 된다. 다시 말하면 우선순위 매핑 테이블은 데이터 중요도 등급과 암호 스택에 대한 보안 기능을 추가함으로써 제안된 암호화 모듈의 안전성을 공고히 한다.

우선순위 매핑 테이블이 제 역할을 다 하기 위해서는 매핑 테이블을 주기적으로 갱신하여 한 종류의 데이터 패킷이 오랜 시간 동안 동일한 매핑 정보를 가지고 전송되는 것을 막아야 한다. 우선순위 테이블의 등급 정보가 치환되었다 하더라도 치환된 값을 고정하여 계속 사용하면 외부 공격자가 매핑 값을 알아낼 수도 있기 때문이다. 그림 2는 패킷 우선순위 테이블을 클라이언트에 설정하고 우선순위 매핑 테이블을 갱신하는 과정을 도시한 것이다. 서버는 데이터 패킷을 전송하기 전 패킷 우선순위 테이블과 최초로 설정된 우선순위 매핑 테이블을 클라이언트로 보내고 각각에 대해서 수신 승인을 받는다. 데이터 패킷을 전송하고 일정한 시간이 지나면 서버는 우선순위 매

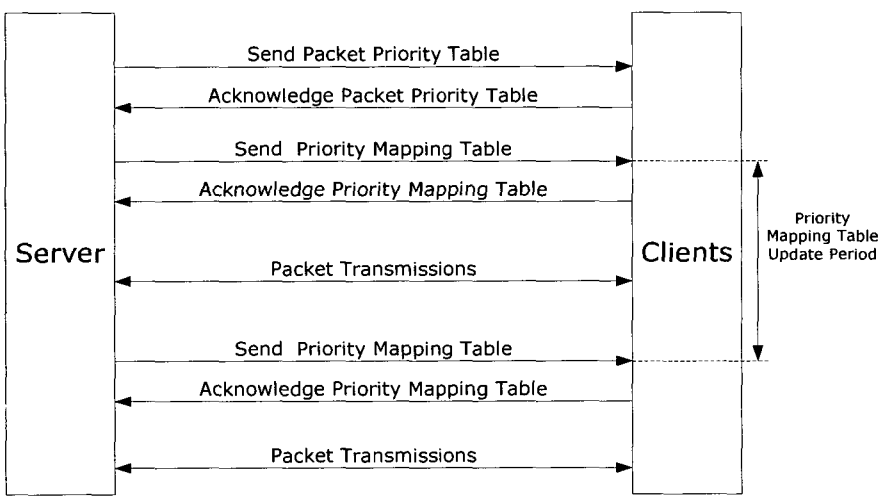


그림 2. 패킷 우선순위 테이블의 클라이언트 설정과 우선순위 매핑 테이블의 갱신.

핑 테이블을 다시 만들고 클라이언트에 새로운 테이블을 다시 보내어 갱신 과정을 완성한다. 우선순위 매핑 테이블 갱신 주기(Priority Mapping Table Update Period)는 인터넷 서비스의 특성과 패킷 우선순위 테이블의 복잡도에 따라서 서버가 정할 수 있다.

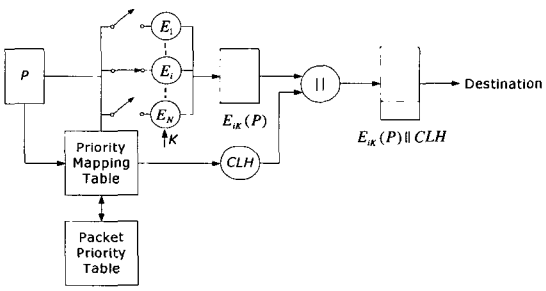


그림 3. 패킷 분석 및 암호화 과정.

그림 3은 본 논문에서 제안된 보안 요구 수준에 근거한 패킷 암호화 과정을 보여준다. 본 논문에서는 암호화 모듈이 사용자 인증, 전자서명 등의 보안 기능에는 사용되지 않고 데이터 은닉을 위한 암호화만 수행한다고 가정하여 대칭키 암호 알고리즘만 사용하기로 한다. 그림 3의 암호화 과정을 설명하면 다음과 같다.

1) 메시지 큐로부터 나온 패킷 P가 암호화 모듈에 입력되면 우선순위 매핑 테이블은 패킷 우선순위 테이블에서 P가 속해 있는 중요도 등급을 찾는다.
2) 매핑 우선순위 테이블은 암호 스택 $E_1, E_2, \cdots, E_N$ 중 P가 속한 등급이 사용하는 암호 $E_i (0 \le i \le N)$의 스위치를 닫는다. 또한 P의 암호화 등급이 매핑된 정보를 가지는 CLH (Cryptography Level Header) 헤더를 생성한다.
3) 비밀키 K와 암호 E_i를 이용하여 패킷 P에 대한 암호문 $E_{iK}(P)$를 생성한다.
4) 생성된 암호문 $E_{iK}(P)$와 CLH를 결합하여 $E_{iK}(P) \| CLH$를 만들어 목적지로 전송한다.

위 암호화 과정에서 사용되는 비밀키 K는 암호 스택에 있는 암호들이 사용하는 키 중 가장 긴 것과 동일한 길이를 가지도록 생성되어야 한다. K보다 짧은 키를 사용하는 암호가 선택되었을 때에는 선택된 암호의 키 길이만큼 K를 분할하여 사용한다. 예를 들어 K가 192 비트(bit)이고 선택된 암호가 64 비트의 키를 사용한다고 하면 K의 앞 64비트 블럭 K_{64}로 암호화를 수행한다. 이렇게 복수 개의 암호를 이용한 암호화 모듈은 항상 최대 길이의 키를 생성해야 한다는 단점이 있지만 중요도가 낮고 발생 빈도가 높은 패킷을 암호화할 때는 키의 길이를 줄여서 암호화를 수행하므로 암호화 모듈의 전체 처리 부하는 일정하게 유지할 수 있다.

그림 3에서 알 수 있듯이 암호 알고리즘 정보가 들어 있는 CLH는 암호화되지 않고 암호문에 결합되어 전송된다. 이것은 CLH마저 암호화되면 수신자가 패킷을 복호화할 때 어떤 암호 알고리즘이 적용되었는지 알 수 없기 때문이다. 따라서 외부 공격자가 전송되는 패킷을 가로채면 CLH의 모든 내용을 볼 수 있다. 앞에서도 기술했듯이 우선순위 매핑 테이블은 이러한 위험에 대비하는 역할을 한다. 매핑 테이블은 그림 2와 같이 패킷 우선순위 테이블을 치환하고 주기적으로 갱신함으로써 데이터의 중요도 등급 및 적용 알고리즘의 정보를 다시 암호화하여 외부 공격으로부터 암호화 모듈을 효과적으로 차단하는 일을 한다.

그림 3에 제시된 암호화 과정과 II장에서 기술한 Roellgen의 방법[11]의 가장 큰 차이점은 그림 3의 방법은 의사 난수를 쓰지 않고 미리 결정된 데이터 중요도에 따라서 강도가 다른 암호를 선택한다는 점이다. 따라서 암호화의 복잡성을 증대시키는 동시에 속도 면에서 효율성을 높이는 장점을 가진다. 또한 스트림 암호 형태로 되어 있는 Roellgen[11]의 방법과는 달리 그림 3의 방법은 블록 암호 형태로 구성되어 있으므로 대규모 패킷 암호화에 적용하기가 용이하다.

그림 4는 제안된 암호화 모듈의 복호화 과정이다. 우선순위 매핑 테이블을 이용하여 데이터 패킷 암호문에 적용된 암호 알고리즘을 찾아서 복호화를 수행

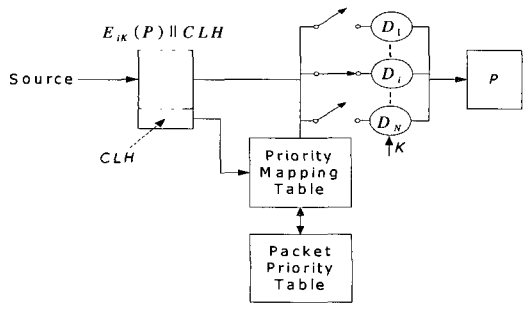


그림 4. 패킷 분석 및 복호화 과정.

하는 과정은 암호화 과정과 유사하므로 절차에 대한 자세한 설명은 생략한다.

3.2 암호 스택 설정 기준

본 논문에서 제안된 암호 알고리즘을 적용하기 위해서는 먼저 데이터 패킷의 종류에 따른 우선순위를 구해야 하는데 이는 서비스를 설계할 때 미리 결정하거나 설계 후 서비스를 직접 운용할 때 생성되는 패킷을 분석해서 결정할 수 있다. 우선순위를 결정한 후에는 등급에 따라 적용할 암호 알고리즘을 선택해야 한다. 각 우선순위에 적용할 암호 알고리즘을 무엇으로 할 것인지 결정하는 데 가장 중요한 기준은 암호 알고리즘의 강도와 암호화 처리 속도이다. 즉 우선순위가 낮을수록 암호화 강도가 낮고 암호화 처리 속도가 빠른 암호 알고리즘을 적용해야 하며, 우선순위가 높을수록 암호화 강도가 높은 암호 알고리즘을 적용해야 한다. 하지만 강도가 높은 암호 알고리즘은 일반적으로 암호화 처리 속도가 느리기 때문에 암호화 엔진 전체의 성능을 저하시킬 수 있다. 따라서 암호화 엔진의 성능과 사용되는 각 암호 알고리즘의 처리 속도 사이의 관계를 찾아서 암호화 처리 속도의 허용 범위를 정량적으로 규명해야 한다.

평문으로 된 패킷을 입력받아 본 논문에서 제안된 암호화 모듈을 이용하여 암호문 패킷을 생성하는 그림 5와 같은 암호화 엔진을 고려하자. 입력되는 패킷이 N 등급의 우선순위로 구분되어 있다고 한다면 이에 대응하는 암호 알고리즘도 N개가 있어야 한다 ($C_1, \cdots, C_N$). T시간 동안 암호화 엔진의 입출력 패킷을 분석한다고 할 때 $i\,(1 \le i \le N)$번째 우선순위를 가지는 패킷에 대해 다음과 같은 변수를 정의하자.

- L_i: i 번째 우선순위를 가지는 패킷에 적용할 암호 알고리즘 C_i의 처리 속도(bytes/sec)
- M_i: T 시간 동안에 발생하는 i 번째 우선순위를 가지는 패킷의 수
- $p_i^k\,(1 \le k \le M_i)$: i 번째 우선순위를 가지는 패킷의 크기(bytes)
- $t_i^k\,(1 \le k \le M_i)$: T 시간 동안 i 번째 우선순위를 가지는 패킷이 점유하는 시간(sec)
- M: T 시간 동안 발생하는 패킷의 총 수

단위 시간당 입력되는 패킷을 암호화하는 데 소요되는 암호화 엔진의 처리 속도 L(bytes/sec)은 각 암호 알고리즘 처리 속도의 선형 조합으로 계산되므로 다음과 같이 나타낼 수 있다.

$$L = \frac{1}{T}\sum_{i=1}^{N} L_i \sum_{k=1}^{M_i} t_i^k \tag{1}$$

식 (1)에서 M_i와 t_i^k는 패킷 분석 구간 T에 따라서 값이 변하는 확률 변수이므로 L의 평균값 $\overline{L}$은

$$\begin{aligned}\overline{L} &= E\{L\} = \frac{1}{T}\sum_{i=1}^{N} L_i E\left\{\sum_{k=1}^{M_i} t_i^k\right\} \\ &= \frac{1}{T}\sum_{i=1}^{N} L_i \overline{M_i}\ \overline{t_i}\end{aligned} \tag{2}$$

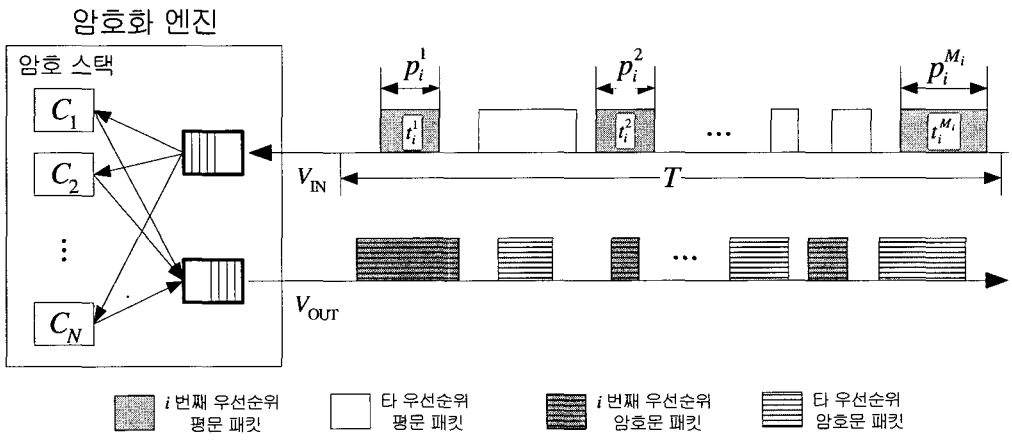


그림 5. 암호화 엔진의 패킷 흐름.

와 같이 된다. 여기서 $E\{\cdot\}$는 확률 변수의 평균값을 나타내며 $\overline{M_i}$는 i 번째 우선순위 패킷이 T 시간 동안 발생하는 평균 횟수이고 $\overline{t_i}$는 i 번째 우선순위 패킷의 평균 점유 시간을 나타낸다. $\overline{M_i}$와 $\overline{t_i}$는 측정하기 어려운 값이므로 측정하기 쉬운 매개 변수로 위 식을 다시 구해야 한다. 패킷의 평균 점유 시간 $\overline{t_i}$에 암호화 엔진의 데이터 입력 속도 V_{IN}을 곱하면 패킷의 평균 크기 $\overline{p_i}$가 나오므로 다음과 같은 식이 성립된다.

$$\overline{L} = \frac{1}{T}\sum_{i=1}^{N} L_i\left(\frac{\overline{M_i}\,\overline{t_i}V_{IN}}{V_{IN}}\right) = \frac{1}{V_{IN}}\left(\frac{M}{T}\right)\sum_{i=1}^{N} L_i\overline{p_i}\left(\frac{\overline{M_i}}{M}\right) \quad (3)$$

여기서 $\overline{M_i}/M$은 i 번째 우선순위를 가지는 패킷의 평균 발생 빈도 비율이며, M/T는 단위 시간당 입력되는 패킷의 수이다. $\overline{p_i}$와 $\overline{M_i}/M$, M/T는 모두 패킷의 발생 횟수와 크기만 알면 계산할 수 있으므로 $\overline{M_i}$와 $\overline{t_i}$보다 훨씬 더 쉽게 구할 수 있다.

식 (3)을 이용하여 암호 스택에 들어가는 암호 알고리즘들의 처리 속도를 결정하는 기준을 찾아야 하는데, 암호화 엔진을 하드웨어로 구현하는 경우와 소프트웨어로 구현하는 경우에 있어서 암호화 엔진 평균 처리 속도 $\overline{L}$가 만족해야 하는 기준이 각각 달라진다. 먼저 암호화 엔진을 하드웨어로 구현할 경우에는 $\overline{L}$가 암호화 엔진에 들어오는 패킷의 입력 속도 V_{IN}과 같거나 더 빨라야 모든 입력 패킷을 손실 없이 암호화 할 수 있다. 따라서 $\overline{L}$는 아래와 같은 범위 안에 있어야 한다.

$$\overline{L} \geq V_{IN} \quad (4)$$

식 (3)을 위 부등식에 대입하면 각 암호 알고리즘의 처리 속도 L_i가 만족해야 하는 조건은 다음과 같이 나온다.

$$\sum_{i=1}^{N} L_i\overline{p_i}\left(\frac{\overline{M_i}}{M}\right) \geq V_{IN}^2\left(\frac{T}{M}\right) \quad (5)$$

즉 위 조건이 성립하는 범위 내에서 암호 알고리즘의 조합을 선택해야 한다.

한편 하드웨어로 구현하는 경우와는 달리 암호화 엔진을 소프트웨어로 구현하는 경우에는 서버가 CPU 시간 중 일정 비율을 암호화 처리를 하는 데 할당해야만 한다. CPU 시간 중 암호화 처리 시간이 차지하는 비율을 $T_C\,(0 < T_C < 1)$라고 정의하면 서버 CPU가 실제로 가지는 암호화 처리 속도는 $T_C\overline{L}$가 되고 부등식 (4)는 아래와 같이 바뀐다.

$$T_C\overline{L} \geq V_{IN} \quad (6)$$

식 (3)을 위 조건식에 대입하면 암호화 엔진이 소프트웨어로 구현되는 경우에 암호 스택에 들어갈 각 암호 알고리즘의 처리 속도 L_i가 만족해야 하는 조건은 아래와 같이 유도된다.

$$\sum_{i=1}^{N} L_i\overline{p_i}\left(\frac{\overline{M_i}}{M}\right) \geq \frac{1}{T_C}V_{IN}^2\left(\frac{T}{M}\right) \quad (7)$$

식 (5)와 (7)을 비교하면 암호화 엔진이 소프트웨어로 구현되는 경우의 암호 알고리즘들이 가져야 할 처리 속도가 하드웨어로 구현되는 경우보다 더 빨라야 한다는 사실을 알 수 있다. 이것은 암호화 엔진이 소프트웨어 상에서 구현되는 경우에는 서버 CPU가 암호화 처리 외에도 다른 작업들을 처리하는 데 사용되어 구현되는 암호화 모듈의 실제 처리 속도가 느려지기 때문이다. 또한 T_C가 1이면 식 (7)은 (5)와 똑같아진다. 하드웨어로 암호화 엔진을 구현할 때에는 암호화 처리를 하는 데 프로세싱 시간(processing time)을 모두 소요하므로 $T_C = 1$이라고 말할 수 있다.

제안된 기법이 가지는 평균적인 암호화 계산량은 암호화 자체의 계산량과 암호 스택에서 적용할 암호를 찾는 데 소요되는 계산량의 합으로 구할 수 있다. 암호화 스택의 i 번째 암호 알고리즘 C_i가 기본 데이터 블록을 암호화하는 데 소요되는 계산량을 f_i라고 정의하면 복수 개의 암호를 우선순위에 따라서 사용하는 암호화 모듈의 평균 계산량 $\overline{f}$는 다음과 같이 표시된다.

$$\overline{f} = \sum_{i=1}^{N} f_i\frac{\overline{M_i}}{M} + g \quad (8)$$

위 식에서 $\overline{M_i}/M$는 앞에서 정의한 대로 i 번째 우선순위를 가지는 패킷의 평균 발생 빈도 비율이며 g는 암호 스택에서 적용할 암호를 찾는 데 소요되는 계산량을 가리킨다. 암호 스택에서 적용할 암호를 찾는 과정은 패킷 우선순위 테이블과 우선순위 매핑 테이블에서 패킷 헤더가 가리키는 암호 알고리즘을 탐색하는 과정이므로 $O(N)$의 계산량을 가진다고 말할 수 있다.

Ⅳ. 실험 결과

4.1 시스템 구성

제안한 암호화 기법을 검증하기 위해 클라이언트와 서버 간에 중요도가 다른 여러 형태의 데이터가 전달되는 네트워크 시스템을 설정하고 본 논문에서 제안된 암호화 모듈을 구현한 실험을 실시하였다. 실험에서 사용한 네트워크 시스템의 암호화 엔진은 OpenSSL에서 제공하는 EAY CSP (Cryptographic Service Provider)와 CSSM API[17]를 사용하여 구현하였다. 실험에서 사용할 서버와 클라이언트간의 암호화 키 및 데이터 전송 과정은 다음과 같다. 서버는 클라이언트의 접속을 허용할 때 현재 세션(session)의 암호화에 사용될 비밀키(secret key)를 공용키(public key) 방식으로 암호화하여 클라이언트에 보낸다. 클라이언트는 전송할 데이터의 중요도에 따라 상이한 발생 빈도로 랜덤하게 데이터를 생성하고 이를 전송 받은 비밀키를 사용하여 우선순위에 맞게 암호화하여 서버로 전송한다. 서버에서는 이를 수신하여 복호화한 후 다시 암호화하여 클라이언트로 보내며, 클라이언트에서는 서버로부터 수신한 데이터를 복호화한 후 서버로 송신한 데이터와 비교하여 확인하고 다음 데이터를 생성하여 보낸다.

본 시스템에서는 데이터 우선순위가 여섯 단계로 나뉜다고 가정한다. 이 중 우선순위가 가장 낮은 데이터는 암호화되지 않은 채 전송된다고 정한다. 이 데이터들은 클라이언트와 서버가 가장 빈번하게 주고받는 것들로서 생명주기(life time)가 매우 짧고 외부로 그 내용이 노출되어도 시스템 보안에 별 영향을 미치지 않는다. 온라인 게임 서비스에서 수행되는 단순 명령, 동적 위치 데이터 등이 이에 해당된다. 본 실험에서는 전체 데이터 패킷 중 86.72%가 이런 데이터라고 설정하였다.

우선순위가 가장 낮은 종류의 데이터를 제외하면 본 논문에서 제안된 암호화 기법을 사용하기 위해서는 강도가 다른 다섯 개의 암호 알고리즘이 필요하다. 암호 스택에 적용할 암호를 선택하기 위해서 대칭키 암호 알고리즘들의 암호화 강도의 측면을 고려해보자. 먼저 스트림 암호화 기법은 스트림 생성기에 대한 정보가 유출될 경우 쉽게 키를 찾아낼 수 있는 약점이 있으므로 암호화 강도가 블록 암호화에 비해 상대적으로 낮다고 정한다. 블록 암호화 기법은 사용되는 암호화 키의 길이가 길어질수록 암호 알고리즘의 강도가 강해지지만[5] 키 길이가 동일한 블록 암호화끼리는 암호화 강도의 단순 비교가 불가능하므로 암호화 처리 속도를 보고 우선순위를 정해야 한다. 본 연구에서도 암호 라이브러리가 제공하는 몇 가지 암호 알고리즘에 대해 처리 속도를 실험적으로 측정하여 우선순위를 결정하였다.

위에서 기술한 암호화 강도 결정 기준을 바탕으로 본 연구에서는 표 1에 나와 있는 암호 알고리즘들을 사용하였다. 먼저 192비트 키를 사용하는 3DES-3KEY 알고리즘이 64비트 키를 사용하는 다른 블록 암호나 스트림 암호에 비해 월등히 높은 암호화 강도를 가지므로 최우선순위를 가지는 패킷을 암호화 하는 데 사용된다. 32비트 키를 사용하는 스트림 암호 방식의 RC4 암호 알고리즘은 암호화 하는 등급 중 가장 낮은 우선순위로 할당하였다. 64비트 키를 사용하는 RC5 CBC(Cipher Block Chaining mode), DES CBC, RC2 CBC 등 세 개의 블록 암호는 앞에서 기술했듯이 실험적으로 측정된 처리 속도를 기준으로 각각 우선순위를 결정하였다.

표 1에 나와 있는 암호 알고리즘의 암호화 처리 속도는 일반적인 프로세서에서 암호화를 구현할 때 얻어지는 초당 수백 Kbyte 이상의 값들보다 더 느

표 1. 사용한 암호 알고리즘.

암호화 우선순위	암호 알고리즘	키 길이 (bits)	패킷 발생 빈도(%)	암호화 처리 속도(bytes/sec)
1	3DES-3KEY	192	0.20	32,942
2	RC5 CBC	64	0.25	35,942
3	DES CBC	64	0.33	37,609
4	RC2 CBC	64	2.50	40,589
5	RC4	32	10.00	41,753
6	암호화 안 함	-	86.72	225,033

린 값들이다. 이것은 앞에서 말한 대로 패킷 하나가 클라이언트와 서버에서 각각 한 번씩 암호화 및 복호화된 후 다음 패킷이 전송되는 방식으로 실험을 수행하기 때문이다. 따라서 하나의 패킷이 네 번의 암호화(또는 복호화) 과정과 네트워크 처리 과정을 거치기 때문에 표 1의 암호화 처리 속도는 일반적인 암호화 처리 속도의 1/4 이하의 값을 가지게 된다. 네트워크 처리 과정 중 라우터를 통과할 때 발생하는 가변적인 요인을 없앤 상태에서 처리 속도를 얻기 위해서 클라이언트와 서버 간 데이터 전송 구조를 동일한 컴퓨터상에서 로컬 호스트(local host) 서버와 클라이언트로 구성하고 TCP 프로토콜로 연결한 후 속도를 측정하였다.

한편 표 1에 나와 있는 암호 알고리즘들이 3.2절에서 제시한 암호 스택 설정 기준을 만족하는지 알아본다. 표 1의 암호 알고리즘 1~6번을 $C_1,\cdots,C_6$라고 하면(C_6은 암호화를 안 하는 모듈) 식 (7)의 $\overline{M_i}/M$와 L_i는 각각 표 1에 나와 있는 패킷 발생 빈도와 암호화 처리 속도와 같다. 우선순위 종류에 따라서 발생된 패킷의 평균 크기는 $p_1,\cdots,p_5=31$, $p_6=33$ 바이트로 나왔으므로 식 (7)의 좌변 값은 약 199.4 Mbps로 나온다. 단위 시간 당 발생한 모든 패킷의 평균 개수 M/T는 4593(sec^{-1})이고 암호화 엔진에 들어오는 패킷의 입력 속도 V_{IN}은 약 900Kbps로 측정되었으므로 암호화 처리 비율 T_C를 1이라 하고 식 (7)의 우변 값을 계산하면 176.4 Mbps이다. 따라서 표 1의 암호 알고리즘으로 구성된 암호화 모듈의 데이터 처리 속도는 패킷의 입력 속도보다 빠르므로 모든 패킷을 손실 없이 암호화한다고 말할 수 있다.

본 실험에서 구성한 클라이언트와 서버 간의 패킷 전송 시나리오는 다음과 같다.

1) 서버와 클라이언트는 각각 본 논문에서 제안된 암호화 기법을 구현한 암호화 모듈을 탑재하며, 클라이언트의 최초 접속 시에 서버가 암호화 키를 생성하여 클라이언트에 배포하고 사용할 암호화 방식을 선택한다. 암호화 키는 표 1에 나와 있는 모든 암호 알고리즘에서 사용될 수 있도록 최대 크기인 192비트로 생성된다. 암호 스택의 각 암호 알고리즘은 이 키의 일부를 이용하여 암호화 및 복호화를 수행한다.
2) 본 논문에서 제안된 기법이 가지는 성능을 정량적으로 규명하기 위해서 제안된 암호화 방식과 함께 아래와 같은 여러 가지 다른 암호화 방식을 서버가 선택할 수 있게 한다.
   a) 우선순위에 따라 암호 알고리즘을 달리하여 패킷을 암호화한 후 전송하는 방법 (본 논문에서 제안된 기법)
   b) 패킷을 암호화하지 않고 전송하는 방법
   c) 패킷을 모두 3DES-3KEY 암호 알고리즘으로 암호화하여 전송하는 방법
   d) 표 1에서 우선순위 1~5까지의 패킷을 모두 3DES-3KEY 암호 알고리즘으로 암호화하여 전송하는 방법

   c)와 d)에서 3DES-3KEY 암호 알고리즘을 적용한 이유는 암호화 강도를 높이기 위해서 가장 강력한 암호 알고리즘을 일괄적으로 사용하는 경우보다 본 논문에서 제안된 기법이 어느 정도 더 효율적인지를 보여주기 위함이다.
3) 클라이언트는 표 1에 나와 있는 우선순위 및 발생 빈도에 따라서 패킷을 생성하고 서버가 설정한 전송방식에 따라 패킷을 암호화한 후 서버로 전송한다.
4) 서버는 전송받은 패킷을 설정한 전송 방식에 따라서 복호화하고, 복호화한 데이터를 동일한 암호 알고리즘을 써서 다시 암호화한 후 클라이언트로 전송한다.
5) 클라이언트는 서버로부터 받은 패킷을 복호화한 후 앞서 전송한 패킷과 비교하여 서로 일치함을 확인한 다음 새로운 패킷을 전송한다.

4.2 결과 분석

제안된 암호화 기법을 적용하여 패킷을 전송하는 한 개의 클라이언트와 앞에서 기술한 패킷 전송 시나리오 2)에 나와 있는 암호화 방식 b)~d) 중 하나를 선택해서 사용하는 또 다른 클라이언트를 동일한 PC상에 동시에 구동시켜 실험을 실시하였다. 참고로 서버는 Pentium IV 3.0 GHz의 CPU를 사용하고 클라이언트 PC는 Pentium IV 1.7 GHz의 CPU를 가지며, 두 PC는 각각 100 Mbps와 10 Mbps 이더넷(Ethernet)에 연결된다. 본 실험에서 측정한 암호화 처리 속도는 클라이언트 컴퓨터로부터 전송받은 데이터를 서버 컴퓨터가 매 초당 복호화 및 암호화 처리를 하여 전송해주는 패킷의 바이트 수를 나타낸다. 이 경우는 패킷이 라우터를 통과하는 복잡한

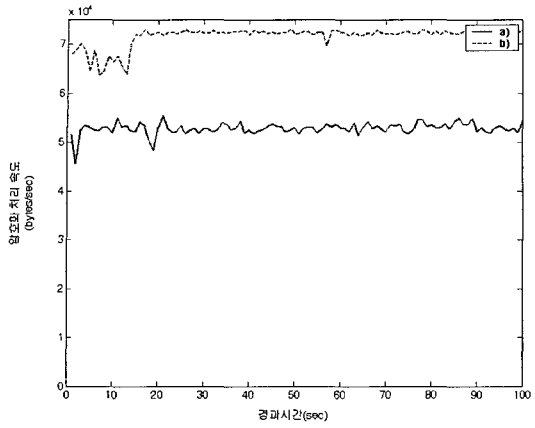


그림 6. 암호화 엔진 성능 비교 1 : a) 제안된 암호화 기법, b) 암호화 안 함.

과정을 거치게 되므로 암호화 처리 속도는 라우터의 영향이 없는 상태에서 측정한 표 1의 값보다 더 느리게 나온다.

그림 6은 본 논문에서 제안된 암호화 기법을 사용한 클라이언트의 암호화 처리 속도와 암호화를 하지 않고 데이터를 보내는 클라이언트의 패킷 전송 속도를 비교한 것이다. 제안된 암호화 기법을 사용한 클라이언트는 표 1에 나와 있듯이 전체 패킷 중 13.3%의 패킷을 우선순위 1~5에 따라서 암호화 하여 전송하였다. 제안된 암호화 기법을 사용한 클라이언트의 암호화 처리 속도는 암호화를 하지 않고 데이터를 보내는 클라이언트의 패킷 전송 속도에 비해 평균적으로 약 73.8%의 성능을 나타낸다.

그림 7은 제안된 암호화 기법과 모든 패킷에 대해 3DES-3KEY 암호 알고리즘을 동일하게 적용한 기법의 성능을 비교한 것이다. 단일 암호화로 표 1에 나와 있는 패킷의 보안 요구 수준을 모두 만족시키기 위해서는 우선순위가 가장 높은 패킷이 요구하는 암호 알고리즘 3DES-3KEY를 사용해야 한다. 그림에서 알 수 있듯이 제안된 기법을 이용한 암호화 처리 속도는 3DES-3KEY 암호 알고리즘을 적용한 경우에 비해서 2.83 배 정도 더 빠르게 나온다. 따라서 모든 패킷을 암호화 강도가 제일 높은 알고리즘으로 암호화하는 방법보다 본 논문에서 제안된 기법의 효율성이 훨씬 뛰어남을 알 수 있다.

마지막으로 그림 8은 암호화를 필요로 하는 우선순위(표 1의 우선순위 1~5)에 속한 패킷에 대해서만 동일한 암호 알고리즘 3DES-3KEY를 적용하고 나머지 패킷에 대해서는 암호화 처리를 하지 않는 방식과 본 논문에서 제안된 기법의 성능을 비교한 것이다. 제안된 암호화 기법의 암호화 처리 속도는 그림 8의 비교 대상에 비해 평균 4% 정도 더 우수한 성능을 보인다. 두 암호화 방식의 성능 차이가 별로 나지 않는 것은 비교 대상의 방법도 제안된 기법과 마찬가지로 표 1에 나와 있는 우선순위별 패킷 중 암호화를 요구하는 13.3%에 대해서만 암호화를 수행하기 때문이다. 하지만 이 기법은 제안된 기법과는 달리 암호화 대상 패킷을 무조건 가장 높은 강도를 가지는 3DES-3KEY 알고리즘으로 암호화하기 때문에 상대적으로 비효율적이라고 말할 수 있다. 소프트웨어로 암호화 엔진을 구현하는 경우 본질적으로 서로 다른 대칭키 암호 알고리즘 간의 암호화 처리 속도가 크게 차이나지 않는 것[7]도 그림 8의 두 암호화 기법의 처리 속도 차이가 적은 원인 중의 하나이다.

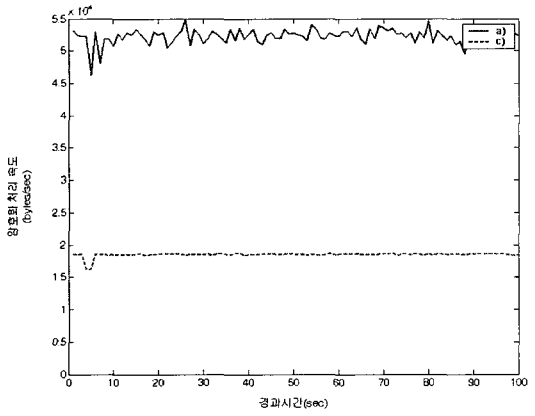


그림 7. 암호화 엔진 성능비교 2 : a) 제안된 암호화 기법, c) 모든 패킷에 대해 3DES-3KEY 적용.

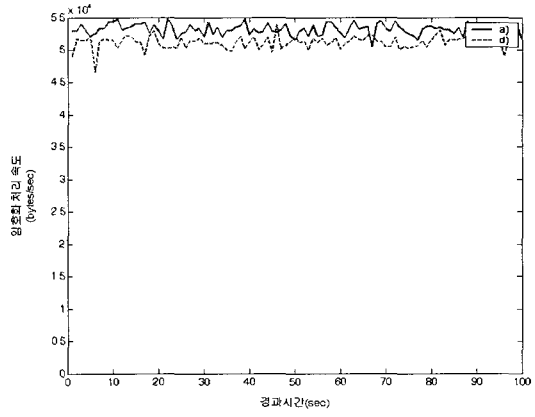


그림 8. 암호화 엔진 성능비교 3 : a) 제안된 암호화 기법, d) 암호화 요구 패킷에 대해 3DES-3KEY 적용.

그림 6~8의 결과는 제안된 암호화 기법이 패킷별로 요구되는 보안 수준을 모두 만족시키면서 동시에 암호화를 하지 않고 패킷을 전송하는 클라이언트에 비해 크게 뒤지지 않는 성능을 가진다는 사실을 보여준다. 참고로 본 연구의 결과는 한국전자통신연구원에서 개발한 Dream3D 게임 엔진을 탑재한 온라인 게임 베타 버전에 실제 적용되어 검증을 거치고 있는 중이다[18]. 분산 게임 서버 엔진을 사용하여 서버가 담당해야 하는 암호화 부하를 기술적으로 줄이면서 본 논문에서 개발한 암호화 모듈을 소프트웨어적으로 구현하여 처리 속도를 증가시키는 결과를 내었고 라우터에 암호화 모듈을 하드웨어적으로 구현하는 실험도 수행 중에 있다.

V. 결 론

본 논문에서는 다수의 사용자가 접속하여 대규모 데이터를 주고받는 클라이언트-서버 구조의 서비스에서 보다 효율적으로 패킷 암호화를 수행할 수 있는 암호화 기법을 제안하였다. 제안된 기법은 중요도에 따라서 데이터 패킷의 우선순위를 정한 후 우선순위 등급에 따라서 강도가 서로 다른 복수 개의 암호 알고리즘들을 적용하여 패킷을 암호화한다. 본 논문에서는 제안된 기법을 구현한 네트워크 암호화 모듈의 상세 구조와 동작 과정을 설명하고 암호화 엔진의 처리 속도를 분석하여 사용할 암호화 알고리즘을 선택하는 기준을 제시하였다. 또한 테스트 클라이언트와 서버간의 패킷 전송 실험을 통해 제안된 기법의 유용성을 정량적으로 검증하였다.

본 논문에서 제안된 암호화 기법은 소프트웨어에서 구현되면 암호화 처리 속도에 한계를 보이기 때문에 하드웨어로 구현하는 것이 더 유리하다. 각기 서로 다른 암호 알고리즘을 사용하는 본 연구의 결과를 하드웨어로 구현하는 방법에 대한 연구가 더 필요할 것이다. 또한 인터넷 서비스가 수행되고 있는 네트워크의 상황에 보다 능동적으로 대처하면서 제안된 기법을 적용하기 위해서는 실시간 패킷 발생 빈도를 고려하여 데이터의 우선순위를 조정하는 기법 등에 대한 연구가 선행되어야 할 것이다.

참 고 문 헌

[1] 정윤경, 기준백, 천정희, "온라인 게임 아이템의 안전한 전자 거래 시스템," 정보보호학회논문지, 제13권, 제3호, pp. 91-99, 2003.

[2] 김찬호, "MMORPG의 핫이슈 클라이언트 해킹과 보안," 마이크로소프트웨어, pp. 201-205, 3월, 2004.

[3] J. J. Yan and H. J Choi, "Security issues in online games," Electronic Library, vol. 20, no. 4, pp. 125-133, 2002.

[4] 유선실, "소프트웨어 및 인터넷 콘텐츠 편 : 온라인 게임", 정보통신산업동향-소프트웨어 및 인터넷 콘텐츠 편, 제1권, pp. 131-144, 2002.

[5] 윤재우, 이승형, "IP 기반 VPN 프로토콜의 연구동향 : 확장성과 보안성," 정보보호학회지, 제11권, 제6호, pp. 53-43, 2001.

[6] C. E. Shannon, "Communication theory of secrecy systems," Bell System Technical Journal, 28, pp. 656-715, 1949.

[7] W. Stallings, Cryptography and Network Security: Principles and Practices, 3rd ed., Prentice-Hall, 2003.

[8] 이성은, 장홍종, 박인재, 한선영, "다중 암호화 기법을 활용한 하이브리드 스마트카드 구현," 정보보호학회논문지, 제13권, 제2호, pp. 81-89, 2003.

[9] T. Ritter, "Cryptography: is staying with the herd really best?" IEEE Computer Society Magazine, vol.32, no. 8, pp. 94-95, 1999.

[10] T. Ritter, "Multiciphering and random cipher selection in shannon," http://www.ciphersbyritter.com/NEWS6/MULTSHAN.HTM

[11] C. B. Roellgen, "The polymorphic cipher," http://www.pmc-ciphers.com/

[12] C. B. Roellgen, "On the security of polymorphic encryption," http://www.pmc-ciphers.com/

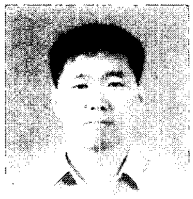
[13] H. Cheng and X. Li, "Partial encryption of compressed images and video," IEEE Transactions on Signal Processing, vol. 48, no. 8, pp. 2439-2451, 2000.

[14] X. Liu and A. M. Eskicioglu, "Selective encryption of multimedia content in distribution networks: challenges

and new directions," IASTED International Conference on Communications, Internet and Information Technology (CIIT 2003), November, 2003.
[15] R. Bolla, M. Marchese and S. Zappatore, "A congestion control scheme for multimedia traffic in packet switching best-effort networks," Lecture Notes in Computer Science, vol. 1242, pp. 523-536, 1997.
[16] R. Jain, "Congestion control and traffic management in ATM networks: recent advances and a survey," Computer Networks and ISDN Systems, vol. 28, no. 13, pp. 1723-1738, 1996.
[17] Technical Standard, "Common security : CDSA and CSSM, version 2," The Open Group, May, 2000 (http://www.opengroup.org/publications/catalog/c914.htm).
[18] 한국전자통신연구원, "온라인 게임 엔진 기술 개발," 연차보고서, 4월, 2004.

〈著者紹介〉

노 지 명 (Ji-Myong Nho) 정회원
1990년 2월 : 한국과학기술원 전자 및 전자공학과 졸업
1994년 2월 : 한국과학기술원 전자 및 전자공학과 석사
2000년 2월 : 한국과학기술원 전자 및 전자공학과 박사
2000년 3월~2002년 2월 : 한국전자통신연구원 선임연구원
2002년 3월~현재 : 대구가톨릭대학교 전자공학과 전임강사
〈관심분야〉 정보보호, 통신망 제어



양 정 민 (Jung-Min Yang) 정회원
1993년 2월 : 한국과학기술원 전자 및 전자공학과 졸업
1995년 2월 : 한국과학기술원 전자 및 전자공학과 석사
1999년 2월 : 한국과학기술원 전자 및 전자공학과 박사
1999년 3월~2001년 2월 : 한국전자통신연구원 선임연구원
2001년 3월~2003년 2월 : 대구가톨릭대학교 전자공학과 전임강사
2003년 3월~현재 : 대구가톨릭대학교 전자공학과 조교수
〈관심분야〉 온라인 게임 보안, 정보보호, 보행 로봇 제어