Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.23, NO.6, Dec. 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.6.989

Pre_Buffered 키스트림을 이용한 효율적인 암호처리 구현*

강 철 오,[1†] 김 은 찬,[1] 박 재 민,[1] 류 재 철[2‡]
[1]한국전자통신연구원 부설연구소, [2]충남대학교

Efficient Implementation of Crypto Processing Based on Pre_Buffered Key Stream Method*

Cheol-oh Kang,[1†] Eun-chan Kim,[1] Jea-min Park,[1] Jea-cheol Ryou[2‡]
[1]The Attatched Institute of ETRI, [2]Choongnam University

요 약

모바일기기, 이동통신망, 유·무선 통신망의 발전으로 모바일 오피스 도입이 확산되고 있으나, 적절한 보안 대책 선행이 요구되고 있다. 본 논문은 모바일 환경의 통신 구간 보안을 위해 H/W 토큰을 이용한 IPSec VPN 솔루션을 구현함에 있어, H/W 토큰의 I/O 지연에 따른 성능 문제를 극복하는 방안으로 카운터 모드를 이용한 스트림 암호 방식을 적용하되 키스트림을 버퍼링하는 방안을 제안하고, 버퍼링 방식이 IP 패킷의 가변 크기, 유실 및 비순차적인 전송 환경에서도 적용 가능한 설계를 제시하고, 구현 결과를 버퍼링을 적용하지 않은 기존 방식과 성능 비교를 통해 제안한 방식의 효율성을 증명한다.

ABSTRACT

Mobile devices use VPN solution to transfer information securely through open network in mobile office environment. In this paper, we propose Pre_Buffered mechanism that improves the throughput of IPSec VPN using low performance H/W crypto Token. Pre_Buffered method precompute key stream, store them in Buffer and use them in IPSec engine for IP packet processing. Moreover, Design, analysis, and experimental results prove the efficiency and feasibility of our proposed method.

Keywords: H/W Token, IPSec, Pre_Buffer

I. 서 론

모바일 기기, 이동통신망, 유/무선 네트워크의 발전으로 활발하게 추진되는 모바일 오피스 구축은 전자메일, 전자결재, 게시판, 메모, 보고 자료 등을 포함하여 VoIP, 메신저, 동영상 교육 등 내부 인트라넷 서비스를 외부망(이동통신망, 인터넷)과 연동하여 언제 어디서든 제공받을 수 있는 환경을 제공한다. 그러나, 이동통신망과 유·무선망을 통해 유통되는 정보가 쉽게 제 3자에게 노출될 위험성이 있기 때문에 모바일 단말기는 물론 유·무선 네트워크 구간 및 모바일 서버에 대한 적절한 보안 대책이 필요하다.

이에 따라, 기존 유선망에서 많이 사용하는 가상사설망(VPN: Virtual Private Network) 보안솔루션을 모바일 단말과 서버(또는 내부 네트워크 진입점)

접수일(2013년 5월 20일), 수정일(2013년 10월 30일),게재확정일(2013년 11월 20일)
* 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10041579, 다양한 사용자 환경을 지원하기 위한 개인정보 대체기술 기반 개인정보보호 솔루션 개발]
† 주저자, cyberkan@ensec.re.kr
‡ 교신저자, jcryou@home.cnu.ac.kr(Corresponding author)

사이에 보안 대책으로 확대 적용하는 추세이다. 최근 VPN 솔루션은 전송 구간에서 통신 데이터의 기밀성을 제공하며 다양한 상위 응용 서비스를 수용하는 TCP/IP 프로토콜상의 IP 계층 보안 서비스를 제공하는 IPSec[1]과 TCP 계층의 보안 서비스를 제공하는 TLS[2]가 주요 솔루션이 되고 있다.

S/W로 구현된 단말용 VPN Client는 암호 알고리즘, 마스터 키, 세션 키 등 중요 정보들이 단말기의 저장 공간 및 실행 메모리상에 노출되는 위험성이 있다. 반면 암호 처리 전용 H/W 토큰을 사용하면, 토큰 내부에 키 및 알고리즘의 안전한 저장과 암호 연산을 수행할 수 있어[3] 호스트의 저장 공간 및 실행 메모리상에 키를 포함한 암호처리 관련 정보가 노출되는 위협을 제거할 수 있다.

스마트폰을 대상으로 범용성을 갖는 H/W 토큰으로 스마트카드 IC를 내장한 USIM 또는 MicroSD 형상을 갖는 제품들이 소개되고 있다. 이와 같은 H/W 토큰은 스마트카드 IC의 내부 CPU 성능, 발열 문제로 인한 Clock 제한, 제한된 메모리 등으로 암호 연산 성능을 개선하는데 한계가 있으며, 호스트 단말과 H/W 토큰과의 인터페이스로 인한 I/O 지연에 따라 성능 제약을 가질 수 밖에 없다.

본 논문에서는 이와 같은 제약 사항을 가진 저속의 H/W 토큰을 사용하여 IPSec 기반의 VPN 솔루션을 구현함에 있어, 전용 H/W 토큰 사용에 따른 VPN 솔루션의 안전성을 보장하며 H/W 토큰의 암호처리 성능 대비 고속 또는 실시간을 요구하는 응용서비스를 지원할 수 있는 Pre_buffered 메커니즘을 제안한다. 또한, 제안한 방안의 설계를 제시하고 구현 결과 성능 비교를 바탕으로 효율성을 증명한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴보고, 3장은 저속의 H/W 토큰을 사용하는 IPSec VPN 솔루션의 성능 개선을 위해 제안한 Pre_buffered 방식의 개념을 설명하고 아이디어의 타당성과 효율성을 고려한 설계를 제안한다. 4장에서는 제안한 방식의 분석 및 구현 결과 실험 데이터를 통해 제안한 방식의 효용성을 증명하고, 5장에서 결론을 맺는다.

II. 관련 연구

2.1 IPSec 개요

IPSec(Internet Protocol Security) 프로토콜은 TCP/IP 프로토콜의 보안 취약점을 해소하기 위해 IP 계층에서 데이터 무결성, 기밀성을 제공하는 표준 프로토콜이다.

IPSec은 IP Payload에 대한 인증 기능을 제공하는 AH(Authentication Header) 프로토콜[4], 데이터 암호화를 기본 기능으로 제공하는 ESP(Encapsulation Security Payload) 프로토콜[5], 키 교환 프로토콜인 IKE(Internet Key Exchange)[6]를 기반으로 각각의 프로토콜에 사용되는 인증, 암호 알고리즘의 운용 모드에 대한 표준들로 구성되어 있다.

ESP 프로토콜은 보안 터널 종단간에 IKE에 의해 협상된 알고리즘과 키를 이용하여 IP 패킷 단위의 암호화/복호화 및 무결성 서비스를 제공하며, 두 가지 모드(Transport/Tunnel)를 지원한다. 각 모드에 따른 ESP Payload는 Fig.1.과 같다. ESP 헤더는 각각 32-bit의 SPI(Security Parameter Index)와 SN(Sequence Number)로 구성된다. SPI는 ESP 패킷 수신자가 수신된 패킷에 매핑되는 SA(Security Association)를 식별하기 위해 사용되는 값으로 SA 협상 과정에서 각각 In/Out 패킷에 대한 SPI를 공유하게 된다.

SN 필드는 하나의 SA에 의해 만들어지는 ESP 패킷에 대해 일련번호와 같은 의미로 사용되며, 송신자와 수신자는 SA 협상 시 SN을 0으로 초기화하고 매 전송되는 패킷마다 1씩 증가시켜 사용한다. SN은 ESP 패킷에 대한 재전송 공격(replay attack)을 방지하는 용도로 사용되며, 구현 시 2^{32}번째 패킷 전송을 위해 SN이 1로 초기화되지 않도록 해야 한다. 즉, 2^{32}번째 패킷 처리 시 SA를 갱신하여 새로운 키의 사용과 SN을 초기화하도록 권고하고 있다. 고속의 IPSec 구현시에는 SN 필드로 64-bit를 사용하는 Extended SN(ESN)의 사용을 제시하고 있다.

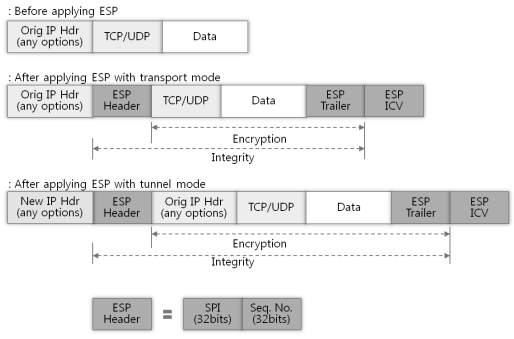


Fig.1. ESP Header & Payload Format

2.2 AES-CTR

IPSec ESP 프로토콜 구현 시 IP 패킷에 대한 암·복호용 블록 알고리즘으로 AES(Advanced Encryption Algorithm)[7]를 포함하여 표준에서 제시하는 다수의 알고리즘 지원을 강제화하고 있으며, 각 블록 알고리즘에 대해 운영 모드를 정의[8]하고 있다. CBC(Cipher Block Chaining) 모드를 포함한 블록 암호 방식은 평문 또는 비문 데이터를 암호 연산기에 전달하여 암호 연산기내에서 처리된 결과로 비문 또는 평문 데이터를 되돌려 받는 형태로 구현된다.

AES-CTR은 IPSec ESP 구현 시 AES 블록 암호를 카운터 모드로 운영하여 패킷 암·복호화를 수행한다[9]. 카운터 모드는 높은 수준의 기밀성을 제공하지만, 카운터(counter)의 올바르지 못한 사용은 쉽게 평문이 노출될 수 있는 위험을 안고 있다.

카운터 값이 반복 사용되거나 동일한 값이 사용되는 것을 방지하기 위해 128-bit의 카운터를 구성하는 Counter Block을 제시하고 각 필드는 다음과 같다.

- Nonce(32-bit) : SA 마다 새로운 값으로 갱신 사용하고, SA 성립 과정에서 SA 양단간에 공유
- IV(64-bit) : 송신자가 IV를 생성하여 수신측에 전달하는 값으로 동일 SA내에서는 매 패킷마다 다른 IV가 적용됨을 보장해야 함
- Block Counter(32-bit) : 매 패킷마다 초기값은 1로 설정하고 패킷의 블록 수에 따라 연속되는 블록에 대해 순차적으로 증가시켜 사용

AES-CTR과 같이 Counter Block을 구성하고 명시적인 IV를 사용하는 경우 송신자는 IV를 포함한 카운터 값을 미리 알 수 있으므로 키스트림을 먼저 생성할 수 있어 ESP 암호화에 따른 연산 지연을 줄일 수 있다. 그러나, 수신자는 수신된 패킷의 IV 값을 확인 한 후 카운터 값을 계산할 수 있으므로 키스트림을 미리 생성하여 사용하는 것은 불가능하다.

따라서, 명시적으로 IV를 공유하고, 키스트림을 실시간 생성하여 사용해야 하는 AES-CTR 방식은 본 논문에서 제한 사항으로 제시한 저속의 H/W 토큰을 암호연산기로 사용 시 I/O 지연으로 인한 성능 이슈가 여전히 존재한다.

2.3 스마트카드를 이용한 H/W 토큰 관련 연구

스마트카드 IC의 안전성을 기반으로 이를 활용한 H/W 토큰을 보안 통신에 적용하는 연구들이 꾸준히 진행되었다[3,10-14].

1990년대에는 스마트카드의 성장에 맞춰 암호연산기로서의 스마트카드의 성능 및 커스텀 알고리즘 사용 가능성 등에 대한 연구가 제안되었다[3]. 2000년에 스마트카드에 SSL을 탑재하는 최초의 연구 결과[10]가 제시되었으나, 당시 Java Card의 하드웨어 제약으로 인해 완전한 구현은 이루어지지 않았다.

2004년에는 [11]를 통해 SIM(Subscriber Identity Module) 스마트카드와 OTA(On-The-Air) 서버간 TCP를 이용한 SSL 핸드쉐이크 프로토콜을 제안하여 보안 세션을 제공하는 방안이 소개되었다. [12]는 EAP-TLS 스택 전체가 H/W 토큰에 구현되어 원격의 서버와 EAP-TLS 기반 보안 통신을 수행하는 연구가 진행되었으나 멀티미디어와 같은 데이터를 송수신하기에는 성능적인 제약이 수반될 수 밖에 없었다. [13]에서는 스마트카드 IC를 IPSec 프로토콜에서 요구되는 인증서와 암호화 키를 저장하는 H/W 토큰으로 FreeS/WAN Open 소스를 활용한 구현에 대한 연구를 진행하였다.

2008년에 발표된 [14]는 기존 연구의 성능 제약을 극복하기 위해 TLS-Tandem 스마트카드 아이디어가 발표되었다. TLS-Tandem 스마트카드는 [13]에서 제시된 EAP_TLS에 필요한 인증 및 키 공유 과정을 스마트카드를 이용하여 수행하고, 스마트카드가 삽입되는 호스트(PC 또는 모바일)에 SA를 전달하여 실질적인 데이터 암·복호화는 호스트에서 수행하는 방안이다. 하지만, [13]의 접근 방법과 유사하게 SA를 맺기 위한 마스터키 등의 보안 관리는 이루어지나 세션키를 포함해 실제 데이터를 암호화하는데 필요한 중요 정보는 호스트에 존재하는 보안 위험을 가지고 있다.

III. 제안 방안

본 장에서는 저속 H/W 토큰의 성능 제약을 극복할 수 있는 Pre-Buffered 메커니즘에 관해 설명한다. 먼저 저속 H/W 토큰 사용에 따른 문제점을 기술하고, 제안한 Pre-Buffered 방식의 설계 구조 및 구현 방법에 대해 설명한다.

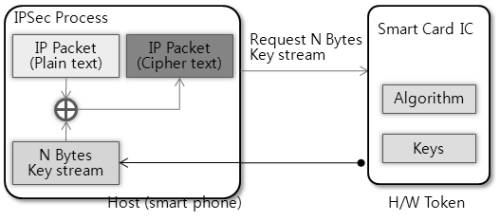
3.1 문제 제기

H/W 암호 연산 토큰을 이용한 IPSec 솔루션은

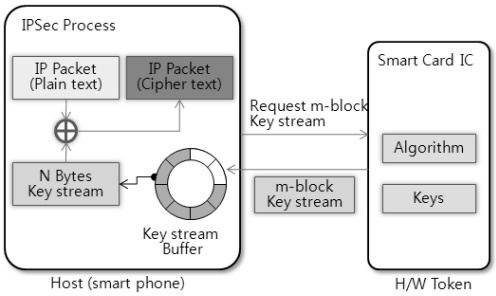
암호 연산과 관련된 중요 정보가 호스트의 저장 공간 또는 메모리상에 노출될 수 있는 위험을 제거하여 IPSec 솔루션의 안전성을 보장할 수 있다. 모바일 단말용 IPSec 솔루션에서 암호 연산기로 전용 H/W 토큰을 사용하는 경우, 단말기의 프로세싱 능력에 비해 H/W 토큰의 성능이 일반적으로 매우 낮아 IPSec 솔루션의 성능은 H/W 토큰의 성능에 의해 제한된다.

H/W 토큰의 성능을 결정하는 주요 2가지 요소는 H/W 토큰과 모바일 단말간의 I/O 성능, H/W 토큰의 암호 처리 성능이다. 스마트카드 IC를 사용하는 H/W 토큰의 암호 연산 성능은 IC 카드의 CPU, 내부 메모리 공간의 제약으로 암호연산 성능을 개선하는데 한계가 있으며, 저속의 I/O 인터페이스로 인해 I/O에 의한 성능이 제한 요소가 되는 경우 암호연산 성능 개선이 의미가 없게 된다.

Fig.2.(a)는 AES-CTR 방식과 같이 패킷마다 H/W 토큰의 I/O와 암호화가 순차적으로 발생하는 구조이다. 이는 패킷 단위 처리에 따라 H/W 토큰의 I/O 지연이 IPSec 솔루션 성능의 가장 큰 제약 사항이 되고 있음을 알 수 있다.



(a) H/W 토큰을 적용한 AES-CTR에서의 데이터 흐름



(b) Pre_Buffered 방식의 H/W 토큰과의 데이터 흐름

Fig.2. Packet Encryption Process with H/W Token

따라서 H/W 토큰의 I/O를 최소한으로 수행하거나, I/O 성능과 IPSec 처리 성능을 독립시킬 수 있는 방안이 필요하다. 패킷 단위 암·복호화가 필요한 IPSec 구현에 있어 H/W 토큰과의 I/O 횟수를 줄이는 것은 한계가 있으며, I/O 성능 제한을 극복하기 위해 Fig.2.(b)와 같이 H/W 토큰을 이용해 키 스트림을 미리 생성하여 호스트 단말에 일정 크기의 버퍼에 저장하고 있다가 IP 패킷 암·복호화 요청 시 버퍼에서 필요한 크기의 키 스트림을 추출하여 사용하는 방식으로 접근할 수 있다.

스트림 암호 방식은 암호처리부(H/W 또는 S/W)에서 키스트림을 생성하고 이를 평문과 Xor 등의 연산을 통해 암호문을 만드는 방식이다. 즉, 평문 데이터를 호스트 외부로 전달하기 전에 암호 데이터로 변환하는 방식으로 호스트의 동적 메모리에 일시적으로 키스트림이 노출된다. 단, 스트림 암호의 특징은 이러한 키 스트림으로 인해 다음 키 스트림이 유추될 수 없음이 안전성의 가장 큰 요소이다.

제안하는 방식은 기존 방식에 비해 메모리에 일시적으로 저장되는 키스트림이 버퍼 크기만큼 존재하며, 메모리에 존재하는 시간도 트래픽 량에 따라 가변적이다. 단, 키스트림을 제외한 모든 보안 기능은 H/W 토큰에서 이루어진다(알고리즘, 마스터키, 세션키, 세션키 공유 과정 등은 모두 토큰에서 이루어짐). 또한 저장되는 키스트림은 SA Lifetime 중에만 유효하다. 단말기의 전원을 껏다가 다시 켜거나, 동작중이더라도 SA가 갱신되면 메모리에 있던 키스트림은 더 이상 의미 없는 데이터가 된다. 즉, 단말기를 습득하여 메모리에 남아 있는 키스트림을 추출하는 사후 공격에 대해서는 고려할 필요가 없음을 의미한다. 동적 메모리에 저장되는 키 스트림이 메모리 해킹에 의해 노출될 수 있는 위험은 존재한다. 단, 이와 같은 가정은 SA가 맺어져있는 정상적인 운용 중 메모리 덤핑이 가능한 공격이 이루어질 때 유효한 것이다. 그러나, 이러한 공격의 가정은 단말기의 동적 메모리에 존재하는 키스트림을 획득하는 것보다, 암호화되기 전의 평문 자료를 획득하는 공격이 선행될 것으로 판단되어 의미 없는 공격 시나리오가 될 수 있다. 이와 같은 가정하에 본 논문에서는 키스트림 버퍼링에 의한 안전성에 대한 부분은 더 이상 고려하지 않으며, 버퍼링 방식을 IP 패킷 처리 환경에 효율적으로 적용 가능함을 제시하는데 중점을 둔다.

카운터 모드를 이용한 스트림 암호 방식은 고속의 처리를 요하는 다양한 구현에 활용되고 있으나, 기존의 구현은 동일한 크기의 데이터 또는 연속적인 데이터 처리에 사용되고 있다. 따라서, 성능 개선을 위해 키스트림을 버퍼링하여 사용하는 경우도 버퍼 관리를

위한 특별한 메커니즘이 요구되지 않는다.

IPSec 구현 시 IP 패킷 특성 상 가변 크기의 데이터, 유실된 패킷 또는 비순차 패킷의 발생에 따라 카운터 모드 기반의 스트림 암호 방식의 적용은 AES-CTR과 같이 각 패킷 단위의 키스트림 처리를 권장하고 있다.

제안하는 방식은 IPSec 구현에 스트림 암호를 적용하되, 키스트림의 버퍼링을 적용하는 방식으로 상기 IP 패킷 특성을 고려한 버퍼 관리, Counter Block, 키스트림의 En/Dequeue 메커니즘에 대한 설계 및 구현 결과를 제시함으로써, 기존 구현과 달리 가변 크기의 데이터, 비연속적인 패킷 기반 통신에 버퍼 방식을 적용하여 제한된 H/W 토큰의 성능을 개선할 수 있는 방안의 실효성을 제시한다.

3.2 Pre_Buffered AES-CTR 메커니즘

Pre_Buffered 방식은 AES 알고리즘을 CTR 모드로 운용하여 생성되는 키스트림을 Buffering하는 메커니즘, CTR 모드 운용에 따른 카운터 관리를 위한 Counter Block 설계, 패킷 스위칭 네트워크 특성 상 유실 또는 비순차(out-of-order) 패킷 처리를 위한 Buffer 관리 메커니즘 설계가 반영되어야 한다.

설계 및 구현에 사용된 H/W 토큰은 다음과 같은 조건을 가진다.

- 토큰 구성 : 스마트카드 IC를 내장한 MicroSD 형상(스마트카드 IC : 암호 연산)
- I/O 지연
  - 16 msec(기준 블록 크기 : 2048 Bytes)
  - 12 msec(768Bytes 이하)
- 알고리즘/모드 : AES-128/카운터 모드

3.2.1 Pre_Buffered 구조 설계

제안하는 Pre_Buffered 방식의 구성 모듈은 Fig.3.과 같이 설계한다.

미리 생성된 키스트림을 저장하는 Pre_Buffer는 각각 단방향 SA에 대응하는 복호용 키스트림, 암호용 키스트림을 관리할 수 있도록 Decrypt Buffer와 Encrypt Buffer로 따로 관리하며, Buffer_Manager는 Pre_Buffer를 할당하고 Make_ Enqueue 스레드를 생성한다. Make_Enqueue는 H/W 토큰으로부터 키스트림을 생성 받아 Pre_Buffer에 채워 넣는 역할을 수행하고, Make_Dequeue는 IPSec_

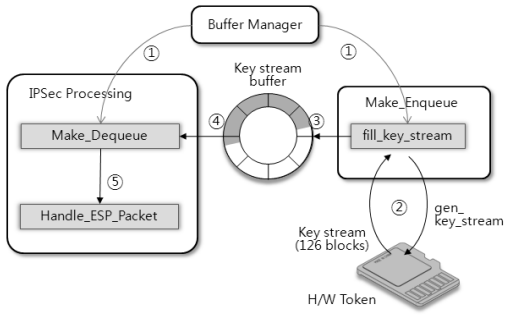


Fig.3. Architecture of the Pre-buffered method

Process의 함수로 구현하여 IPSec_Process의 요청에 따라 해당 버퍼에서 필요한 키스트림을 추출하는 기능을 수행한다.

각 구성 모듈의 키스트림 생성, 저장 및 추출, Pre_Buffer 관리는 다음과 같은 설계 개념이 반영되었다.

- En/Decrypt Buffer 구성
  - 원형 큐 구성
  - 큐의 한 블록 크기 : 16Bytes
  - Enqueue Pointer, Dequeue Pointer 사용
- 키스트림 생성 관리 이벤트
  - 키스트림 생성 (재)시작
  - 키스트림 생성 중지
- 버퍼 Empty/Full 및 En/Dequeue 관리
  - Empty_Block_Count : 전체 버퍼에서 키스트림이 채워지지 않은 블록 수를 저장하는 변수
  - Empty_Block_Count =〈 126 then Buffer Full
  - Empty_Block_Count 〉((size of buffer blocks) - 126)) then Buffer Empty
  - Start_Gen_Key 시그널 : 빈 블록 크기가 전체 블록 크기의 반 이상일 때
  - Sleep 시그널 : Buffer Full 일 때
  - Make_Enqueue : H/W 토큰에 키스트림 생성 명령을 전달하고, 수신된 키스트림을 버퍼에 채움
  - Make_Dequeue : IPSec_Process가 지정한 버퍼에서 요청된 블록 크기의 키스트림을 추출

3.2.2 Counter Block 설계

Pre_Buffered 방식은 암·복호화에 필요한 키스

트림을 미리 생성하여 사용하는 방식으로 AES-CTR에서 제시하는 명시적인 IV와 매 패킷마다 Block Counter를 증가시켜 사용하는 Counter Block을 사용할 수 없다. 따라서, Pre_Buffered 방식의 카운터 운용을 고려하여 Counter Block을 Fig.4.와 같이 구성한다. 32-bit Nonce 값은 AES-CTR과 동일한 목적으로 사용한다. AES-CTR의 64-bit IV와 32-bit Block Counter는 96-bit의 Block Counter로 구성하여 AES-CTR에서 매 패킷마다 적용한 Block Counter를 동일 SA내의 모든 패킷으로 확장하는 개념으로 변경 설계 하였다. 즉, 각 패킷마다 Block Counter를 리셋하지 않고 연속되는 스트림으로 처리하는 방식이다. 이는 2^{96}개 키스트림 블록에 대해 다른 카운터 사용을 보장할 수 있어 ESN을 사용하는 환경에서도 동일 SA내에서 중복된 카운터가 사용되는 경우를 피할 수 있다.

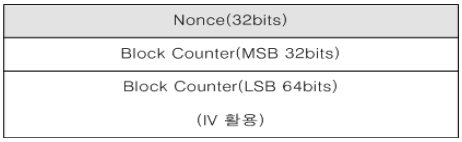


Fig.4. Counter Block of the Pre_Buffered method

96-bit Block Counter 필드는 상위 32-bit와 하위 64-bit로 나누어 구성하였다. 상위 32-bit는 0부터 $2^{32}-1$의 값을 가지며 명시적으로 상대에게 전달하지 않고 SA 양단간에 관리하는 값으로 하위 64-bit 값이 2^{64}이 되었을 때 1씩 증가시킨다. 하위 64-bit는 각 패킷마다 카운터 값에 대한 명시적인 공유를 위해 ESP 헤더에 포함하여 수신자에게 전달하는 값으로, 128-bit 카운터 공유를 위해 64-bit만 전달하도록 설계하여 ESP에 따른 오버헤드를 줄일 수 있다.

3.2.3 카운터와 버퍼 인덱스 관리 메커니즘

Pre_buffered 방식 적용 시 실 환경에서는 VoIP 패킷뿐만 아니라 다양한 응용 서비스(가변 길이의 Payload 데이터)를 위한 패킷이 서로 섞여서 수신되며, 유·무선, 이동통신망, 위성 등 다양한 네트워크가 융합되어 서비스되는 환경 특성 상 IP 패킷이 전송경로 상에서 유실, 재전송되는 경우, 백업 라인 등의 발달에 의해 비순차로 전송되는 경우 등 순차적으로 키스트림을 추출하는 방식을 적용할 수 없다.

카운터 모드를 사용하는 기존 구현(AES-CTR)에서는 카운터 값을 SA 양단간 명시적으로 공유하고, 수신된 패킷에 맞는 카운터를 입력으로 실시간 키스트림을 생성하여 사용하기 때문에 패킷 유실, 재전송, 비순차 패킷 등이 이슈가 되지 않았다.

Pre_buffered 방식에서 기존 구현과 같이 상기 문제가 발생하지 않도록 하는 방안으로 카운터를 공유하고, 수신측은 카운터에 맞는 키스트림을 비순차적으로 Pre_Buffer에서 추출할 수 있는 방안을 설계하였다. 카운터 값은 Fig.4.에 설계된 Counter Block을 이용하고, 카운터 공유를 위해 하위 64-bit의 Block Counter 값을 Fig.5.와 같이 ESP 패킷에 8 Bytes의 IV 필드로 수신측에 전달한다.

명시적 IV를 전달하는 것은 AES-CTR과 동일하지만, 수신측의 카운터 활용이 이를 이용하여 실시간 키스트림을 생성하는 방식이 아니라 Pre_Buffer에서 카운터 값에 해당하는 키스트림을 찾기 위한 값으로 활용되는 차이가 있다.

Fig.5. ESP Packet with IV

카운터 값과 매핑되는 키스트림을 버퍼에 저장 또는 추출하기 위해, Fig.6.과 같이 원형 큐로 구성된 버퍼의 start/end 인덱스를 관리하는 포인터로 Enqueue Pointer와 Dequeue Pointer 개념을 도입하여 버퍼 인덱스를 관리하고, 두 개의 포인터를 카운터와 매핑시키는 방법을 설계 적용하였다. Enqueue Pointer는 Pre_Buffer에 키스트림을 채우기 시작하는 블록을 지정하는 인덱스이며, Dequeue Pointer는 키스트림을 추출하기 시작하는 블록을 지정하는 인덱스이다.

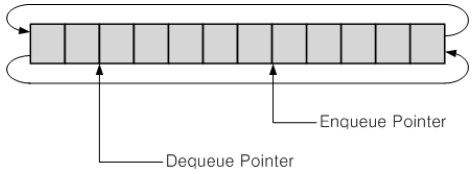


Fig.6. En/Dequque Pointer for Pre_Buffer

여기서 각각 En/Dequeue Pointer는 원형 버퍼 구현 시 사용되는 일반적인 인덱스 포인터와 다르다. 원형 버퍼 구현 시 요구되는 인덱스는 버퍼에 값을 채

우거나 추출하기 위해 지정된 포인터를 관리하는 개념으로 en/dequeue 위치를 미리 지정하고 있다. Pre_Buffered 방식 설계에서는 버퍼에 값을 채우거나 추출하기 위한 위치가 미리 지정된 방식이 아닌 버퍼에 데이터를 en/dequeue하는 시점에 카운터 값을 기준으로 실시간 계산하여 사용한다. 각 포인터는 다음과 같이 계산된다.

$$P_{enq} = CTR \, Mod \, N_{buf} \quad (1)$$

$$P_{deq} = CTR \, Mod \, N_{buf} \quad (2)$$

여기서 P_{enq}, P_{deq}, CTR, N_{buf}는 각각 Enqueue Pointer, Dequeue Pointer, counter, 버퍼 크기를 의미한다.

Enqueue Pointer는 키스트림 생성 시 사용된 카운터를 기준으로 계산되며, 한번의 enqueue 처리에 항상 126 블록을 채우게 된다. 따라서, 126 블록 이하의 빈 블록 존재 시 버퍼 Full로 처리해야 한다. 카운터는 연속된 값을 갖기 때문에 Enqueue Pointer의 증가를 추적하면 순차적으로 126씩 증가하는 결과를 나타낸다. enqueue 시에 Make_Enqueue 스레드에서 키 생성시마다 마지막 Enqueue Pointer 값에 126을 더한 값이 현재 Enqueue Pointer 값과 일치하는지 체크하는 루틴을 삽입하여 연속된 카운터가 오류 없이 사용되고 있음을 확인할 수 있다.

dequeue 시에는 Pre_Buffer에서 Dequeue Pointer를 기준으로 필요한 크기 블록의 키스트림이 존재하는지 확인해야 한다. 즉, 요청된 Dequeue Pointer를 기준으로 필요한 블록 수를 더한 값이 마지막 $P_{enq}+126$ 보다 작으면 키스트림 블록을 추출하고, 같거나 크면 새로운 키스트림이 생성되어 Enqueue Pointer가 추출할 최종 블록 인덱스보다 커질때까지 대기한다.

카운터와 Pre_Buffer의 인덱스를 매핑하여 관리하는 설계 개념에 따라 enqueue 시는 buffer에 순차적으로 키스트림이 채워질 수 있음을 보장하여 순차적으로 증가하는 카운터 값에 해당하는 모든 키스트림이 생성됨을 보장할 수 있다. dequeue 시에는 키스트림의 비순차적인 추출이 가능하여 수신 패킷의 순서와 상관없이 카운터에 해당하는 키스트림을 추출하여 사용한다.

이와 같이 카운터 값과 버퍼의 인덱스로 사용하는 Enqueue Pointer, Dequeue Pointer를 매핑시켜 관리함으로서 원형 큐 기반의 버퍼 관리를 간단하게 구현할 수 있으며, dequeue 메커니즘에 따라 유실된 패킷 및 비순차 패킷에 대한 이슈도 해결되었다.

패킷이 유실된 경우, 수신측에서는 dequeue 메커니즘에 의해 유실된 패킷에 대한 키스트림 블록은 사용되지 않으며, 원형 큐 개념에 따라 새로운 키스트림이 채워질 때 유실로 인해 사용되지 않은 블록을 덮어쓰게 한다. 단, 유실되는 패킷이 많은 경우 Empty_Block_Count에 의해 체크하는 키스트림 생성 이벤트가 발생하지 않을 수 있으나, 버퍼 크기를 아주 작게 설정한 경우가 아니라면 일반 통신에 대해 최소한의 품질을 보장하는 실 네트워크 환경에서는 발생할 수 없다.

비순차 패킷이 발생하는 경우, dequeue 메커니즘이 수신된 카운터를 기반으로 Dequeue Pointer를 계산하기 때문에 패킷이 도착하는 순서와 관계없이 정확한 키스트림을 추출하여 사용될 수 있음을 보장한다. 단, 버퍼 크기가 너무 작은 경우 순서가 바뀌어 나중에 도착하는 패킷이 원래 사용되어야 할 키스트림 대신에 새롭게 생성된 키스트림으로 대체되어 사용될 수 있는 문제가 발생할 수 있다.

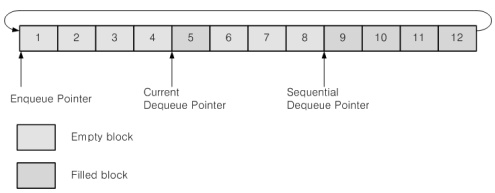


Fig.7. Example of Small Buffer size

버퍼 크기를 작게 설정한 경우의 예로, Fig.7. 같이 12개의 블록으로 구성된 버퍼가 키스트림으로 채워져 있는 상태에서 1에서 8번까지 데이터 중 5번 블록에 해당하는 패킷이 비순차로 도착하지 않은 상태가 발생할 수 있다. 전체 12개의 블록 중 7개가 사용되어 새로운 키를 생성하는 작업이 진행되며, 새로운 키스트림은 순차적으로 1에서 6번 블록까지 채우게 된다. 이때 지연된 5번 블록의 데이터가 도착하면 제시된 버퍼 관리 메커니즘에 의해 새로 생성된 키스트림을 사용하여 복호화를 수행하게 되고, 암호화 시 사용된 키스트림과 불일치를 발생시키는 문제가 발생한다. 비정상적으로 복호화된 데이터를 상위 계층에 올려 주었을 때 재전송을 요청하거나 폐기하는 데이터가 발생하는 문제로, 가능하면 IP 패킷을 처리하는데 있어 잘못된

복호화가 발생하지 않도록 하는 것이 바람직하다.

버퍼 크기를 호스트 메모리가 지원하는 한 최대한 크게 설정하는 경우, 고속의 트래픽 처리가 필요하지 않은 환경에서는 키스트림을 생성하여 버퍼를 채우는 연산에 호스트 프로세서를 포함한 호스트 자원의 불필요한 낭비를 유발하고, 고속의 트래픽 처리가 요구되는 환경에서는 버퍼에 키스트림이 누적되는 것보다 빠르게 추출되기 때문에 대부분 빈 블록만 존재하여 버퍼로 할당된 메모리 낭비만 초래하게 된다. 따라서 버퍼 크기는 트래픽 특성과 H/W 토큰의 성능을 주요 요소로 고려하여 최적의 크기를 선택해야 한다.

IV. 분석 및 실험

본 장에서는 제안한 Pre_Buffered 방식을 적용하여 저속의 H/W 토큰을 이용한 IPSec VPN을 구현하였을 때, 제안한 방식으로 얻을 수 있는 성능 개선 효과를 이론적 분석 결과와 구현 결과 실험 데이터의 비교를 통해 설명한다.

4.1 분석

4.1.1 VoIP 트래픽 특성에 따른 버퍼 크기 분석

Pre_Buffered 방안의 설계에 따라 최적의 버퍼 크기를 결정할 때, 트래픽 특성과 H/W 토큰의 키스트림 생성 성능이 가장 중요한 요소가 되며, 이동 단말기의 메모리 크기 및 키 생성, 버퍼 관리에 따른 호스트 프로세서의 오버헤드 등을 고려해야 한다.

본 절에서는 G.729 코덱을 사용하는 VoIP 서비스를 기준으로 트래픽 특성과 H/W 토큰 성능만을 고려한 최적의 버퍼 크기를 분석한 결과를 제시한다.

[15]에 따르면, G.729(8Kbps) 코덱을 사용하는 VoIP 서비스는 단방향 호(call)에 50개의 IP 패킷이 생성되고 각 IP 패킷의 크기는 60Bytes로 31.2Kbps의 이더넷 대역폭이 요구된다. 이는 IPSec 서비스가 VoIP 트래픽을 원활하게 처리하기 위해서 최소한 암호화 50PPS(Packet/Second), 복호화 50PPS 이상의 패킷 처리 성능을 제공해야 함을 의미한다.

VoIP 패킷 복호화에 필요한 초당 추출되는 블록 수는 200 BPS(Blocks/second)이다. Pre_Buffered 방식은 VoIP 패킷 복호화를 위해 20 msec마다 4 블록(64Bytes)의 키스트림을 버퍼에서 추출하고, 32 msec마다 126 블록의 키스트림을 버퍼에 저장한다. 1초에 3,937 블록의 키스트림을 버퍼에 누적할 수 있어 1초간의 키 생성으로 이후 19초 동안 새로운 키 생성 없이 VoIP 서비스 지원이 가능하다.

VoIP 특성인 실시간 대화형 서비스를 위해 실 구현 시 VoIP의 Packet Jitter를 최대 100 msec로 정의하고 있어, 비순차 패킷이 발생하더라도 지연 시간이 100msec 이내에 도착한 패킷만이 VoIP 응용에서 정상적으로 처리 가능하다. 이는 비순차 패킷이 최소한 이후 5번째 패킷이 도착하기 전에 도착해야 함을 의미하며, 버퍼에서는 최종 추출된 블록의 20번째 전 블록에 새로 생성된 키스트림이 덮어 쓰여지지 않도록 관리하면 정상적인 서비스가 가능함을 의미한다.

따라서, VoIP를 위한 버퍼 크기는 버퍼 관리를 위한 최소 크기인 256 블록(126 X 2) 이상만 할당하면 기능상의 문제가 없음을 알 수 있다. 단, 최소 크기로 할당 시 640 msec마다 키 생성 이벤트가 발생되고, 한 번의 키스트림 생성 후 바로 Buffer가 Full되어 Sleep 이벤트가 발생한다. 결론적으로 VoIP를 지원하기 위한 버퍼 크기는 잦은 키 생성 시작 및 중단에 의한 호스트 프로세서의 오버헤드를 줄이기 위해 256 블록의 N(5~10) 배수 정도로 할당하는 것이 바람직하다.

4.1.2 트래픽 특성(데이터 크기)에 따른 버퍼 관리

VoIP 같은 고정된 길이의 데이터가 일정한 주기로 발생하는 경우 버퍼의 크기 설정 및 관리는 앞서 분석 결과와 같이 상대적으로 간단하다. 하지만 실 환경에서는 가변 길이의 다양한 서비스 데이터가 임의의 시간차를 두고 암·복호화 요청을 하게 되어 모든 트래픽 특성 자체를 분석하여 최적의 버퍼 크기를 설정하기는 현실적으로 어렵다. 따라서 H/W 토큰 성능을 기반으로 IPSec 솔루션의 최대 성능을 설정하고, 이에 맞게 dequeue 메커니즘을 조정하여 최적의 버퍼 관리 방안을 접근할 수 있다.

호스트에서 IPSec ESP 서비스를 위해 IP 패킷을 처리하는 전 과정에 소요되는 연산 지연을 무시하면, 버퍼에 키스트림이 채워져 있다는 가정에서 IPSec 솔루션은 단말의 네트워크 성능과 동일한 서비스 속도를 제공할 수 있다.

예를 들어 630,000 블록(630,000 X 16Bytes = 약 10MBytes)의 버퍼에 키스트림이 미리 채워져 있다면, 네트워크 속도가 39,375 BPS(약 5Mbps)인

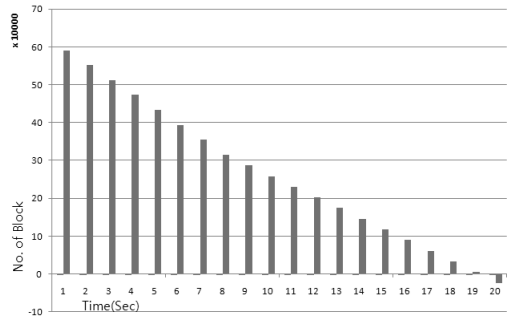


Fig.8. Change of a No. of filled Buffer blocks

단말에서는 16초 동안 H/W 토큰의 성능과 무관하게 5Mbps의 IPSec ESP 서비스를 제공할 수 있다.

앞의 예에서 Pre_Buffered 방식은 버퍼 관리 메커니즘에 따라 8초 후부터 버퍼에 키스트림을 채우는 키스트림 생성이 시작되며, 시간에 따라 버퍼에 남아 있는 키스트림 블록 수는 Fig.8. 같이 점차 감소하여 19초 후 버퍼는 Empty 상태가 된다.

Pre_Buffered 방식 적용 시 실 환경에서 다량의 파일을 ftp로 전송하는 환경에서는 앞서와 같이 성능이 급격히 떨어지는 문제가 발생될 수 있다.

실시간 서비스를 요구하는 VoIP 또는 화상 전화 등은 실제 데이터가 전달되는 패킷 사이즈는 크지 않으나 패킷에 대한 연산 지연을 최소화해야 하며, ftp 등 최대 IP 패킷 사이즈를 사용하는 서비스는 처리 속도가 느려도 사용자의 체감 속도에 차이를 줄 뿐 서비스 자체에는 문제가 없다. 따라서 트래픽 특성을 반영한 최적의 버퍼 크기 및 관리 메커니즘으로 버퍼가 Empty 상태가 되는 경우의 발생을 최소화하여 가능한 일정 서비스 성능을 제공할 수 있도록 IP 패킷 크기에 따라 패킷 당 임의의 지연 시간을 추가하는 접근

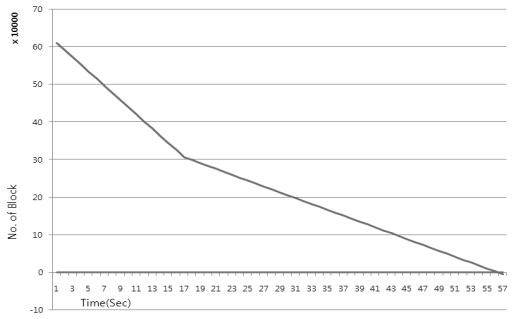


Fig.9. Change of a No. of filled Buffer blocks after delay added

을 시도할 수 있다.

임의 지연 시간은 패킷 크기를 다음과 같이 4가지로 나누어 패킷 크기에 따라 다르게 설계하였다.

① 1024 Bytes 이상의 패킷 : delay(5)
② 512 ~ 1020Bytes 패킷 : delay(3)
③ 256 ~ 508Bytes 패킷 : delay(2)
④ 256Bytes 미만의 패킷 : delay(0), Input PPS

위와 같은 설계에 따라 Fig.8.과 동일한 조건에서 한 패킷의 크기가 1,520Bytes인 서비스를 지원하는 경우 18초 후에 새로운 키스트림 생성이 시작된다. 버퍼 Empty는 Fig.9. 같이 56초 후 발생한다.

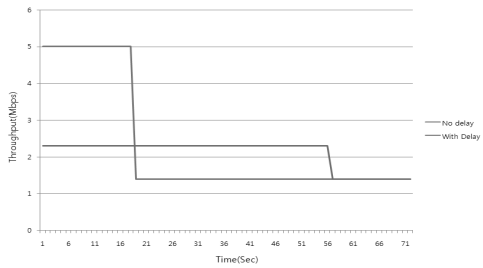


Fig.10. Buffer empty Time difference

Fig.10.은 임의 지연을 추가한 방안과 지연 없이 처리하는 방안의 시간에 따른 성능 차이를 보이고 있으며, 버퍼 Empty가 발생하는 시간 차이는 실제 트래픽 환경에서 IPSec 서비스 품질에 많은 차이를 나타낼 수 있다. 버퍼 Empty 발생 전까지는 다양한 서비스에 대해 지연 없이 서비스를 제공할 수 있음을 보장하는 것으로 실시간 서비스를 요하는 작은 크기의 패킷이 섞여 들어오는 환경에서는 서비스 품질 보장 시간을 늘릴 수 있다.

따라서 실 환경에서 Pre_Buffered 방식을 적용할 때, 사용하고자 하는 응용 서비스의 트래픽 특성을 파악하고 각 서비스 특성에 따라 패킷 처리에 임의 지연을 추가함으로서 모든 서비스를 수용할 수 있는 솔루션 제공도 가능할 수 있다.

4.2 성능 분석

제안 방식은 스마트폰(Galaxy S3, Android 4.2) 단말기에 오픈 소스인 Strongswan Client용 VPN 소스를 수정하여 H/W Token을 암호연산기로 사용하고 Pre_Buffer 크기를 16MBytes로 구현하였다. 구현 결과에 대한 성능은 IXIA사의 chariot

(v7.10) 툴을 이용하여 측정하였으며, 동일 시험 환경에서 AES-CTR 구현 결과와 비교하였다.

4.2.1 제안 방식의 IPSec VPN 성능 비교

제안한 Pre_Buffered 방식은 버퍼를 사용하지 않는 AES-CTR 방식과의 성능 비교를 통해 성능 개선 효과가 있음을 확인하였으며, 두 크기의 IP 패킷(128, 1024Bytes)에 대해 성능 측정 결과를 제시한다.

시험 환경은 VPN Gateway와 실험 대상 스마트폰간 VPN 터널을 구성하고, chariot 단말을 VPN 서버 뒤에 두어 AP와 연결된 스마트폰과의 IPSec 트래픽에 따른 성능을 측정할 수 있도록 구성하였다.

Table 1. Performance of Pre_Buffered Vs. AES-CTR

Packet Size		Pre_Buffered		AES-CTR
		Before Empty	After Empty	
1024 Bytes	Kbps	7,312	600	258
	PPS	652	73.2	31.5
128 Bytes	Kbps	7,271	341	55.4
	PPS	14,201	333	54

실험 결과는 Table 1.과 같은 결과를 나타내었다. 제안 방식은 Pre_Buffer의 Empty 전과 후로 나누어 성능을 제시하여 제안 방식의 최대 성능과 최소 성능을 확인 할 수 있다. 버퍼 Empty 전의 성능은 호스트의 연산 성능과 네트워크 속도에 좌우되는 값으로 제안한 아이디어의 타당성을 증명하는 데이터이다.

Table 1.의 결과는 4.1.2의 분석 결과와 유사한 성능패턴을 나타내고 있으며, 제안한 방식은 패킷 크기에 따라 성능 차이 1.8배에 불과하나 AES-CTR 방식은 4.7배의 성능 차이를 보이고 있다. 이는 제안한 방식이 AES-CTR 방식에 비해 작은 크기의 패킷에 대해 좀 더 효율적일 수 있음을 나타내며, 128-Byte 패킷을 기준으로 약 6배의 성능 향상을 보임을 알 수 있다.

AES-CTR의 결과에서 패킷 크기와 상관없이 동일한 PPS가 나올 것으로 판단하였으나, 실 구현 결과 H/W 토큰 내의 암호 연산 지연과 IPSec 프로세스의 연산 지연 등이 혼합되어 작은 패킷의 경우 PPS가 증가함을 확인하였다. AES-CTR 방식을 적용한 결과 물로 VoIP 통화 시 서비스가 안되는 것을 확인할 수 있었으며, 이는 I/O 지연만으로도 83PPS가 최대 성능인 H/W 토큰으로 인해 최소 100PPS를 요구하는 VoIP 서비스가 지원되지 못하는 것이다.

4.2.2 버퍼 크기에 따른 성능 평가

버퍼 크기에 따른 Pre_Buffered 방식의 성능 변화를 확인하기 위해 1,024-Byte 패킷을 연속해서 처리할 경우, 키스트림 버퍼 크기에 따른 성능 변화를 측정하였다.

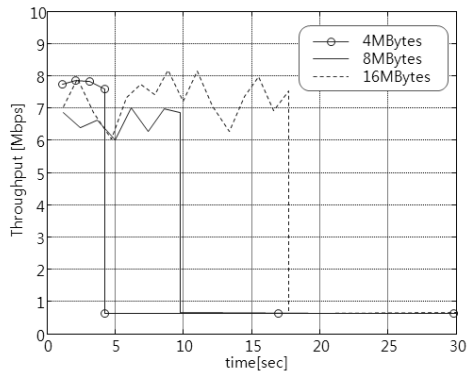


Fig.11. Throughput & Buffer size

Fig.11.은 버퍼 크기별 성능 변화를 보여주는 그래프로, 버퍼 Full 상태에서 약 7Mbps로 시작하여 4M/8M/16MBytes 버퍼 크기별로 4.2초, 9.7초, 17.7초 후에 버퍼 Empty가 발생하고 이후 약 600 Kbps의 성능을 제공함을 알 수 있다.

네트워크 트래픽 특성에 따라 버퍼의 크기를 결정하고 Fig.11.과 같이 트래픽 특성에 맞는 dequeue 메커니즘을 적용하면, 저속의 H/W 토큰을 활용하여 최적의 IPSec 솔루션을 제공 할 수 있다.

V. 결 론

모바일 서비스의 통신 데이터 보안 대책 중 하나인 IPSec VPN 솔루션은 암호 연산 H/W 토큰을 적용함으로서, 기존 S/W 솔루션이 갖는 위험 요소를 제거하여 안전성을 보장할 수 있다.

본 논문에서는 H/W 토큰과 호스트간의 I/O 인터페이스로 인한 제한된 성능을 제공하는 H/W 토큰을 적용하여 IPSec 솔루션 구현 시, H/W 토큰을 이용해 키스트림을 미리 생성하여 호스트 단말에 일정 크

기의 버퍼에 저장하고, IP 패킷 암·복호화 요청 시 버퍼에서 필요한 크기의 키스트림을 추출하여 사용하는 Pre_Buffered 방식을 제안하였다. 제안한 방식에 대해 Pre_Buffer, en/dequeue 메커니즘, 카운터 운용 방안을 설계/구현하여, AES-CTR 방식의 구현 결과와의 성능 비교를 통해 제안한 방식이 효율적이며, H/W 토큰의 성능 제약을 극복한 서비스 지원이 가능함을 증명하였다. 실험 결과에 따르면 본 제안 방식이 기존 방식과 비교하여 버퍼 Empty 상태에서도 최대 6배 이상의 성능 개선 효과를 나타냈다.

References

[1] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, Dec. 2005.

[2] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008.

[3] D. Naccache and D. M'Raihi, "Cryptographic smart cards," IEEE Micro, vol.16, Issue 3, pp. 16-24, June 1996.

[4] S. Kent, "IP Authentication Header (AH)," RFC 4302, Dec. 2005.

[5] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303, Dec. 2005.

[6] C. Kaufman and Ed., "The Internet Key Exchange (IKEv2) Protocol," RFC 4306, Dec. 2005.

[7] NIST. "Advanced Encryption Standard (AES)," FIPS PUB 197, Nov. 2001.

[8] Morris Dworkin, "Recommendation for Block Cipher Modes of Operation : Methods and Techniques," SP 800-38A, Dec. 2001.

[9] R. Housley, "Using AES Counter Mode With IPSec ESP," RFC 3686, Jan. 2004.

[10] P. Urien, H. Saleh and A. Tizraoui, "SSL in smart card," In Proc. of Networking and Computer Science PHD days ("Journees Doctorales Informatique et Reseaux"), JDIR 2000, Nov. 2000.

[11] M. Badra and P. Urien, "Toward SSL integration in SIM SmartCards," In Proc. of Wireless Communications and Networking Conference, pp. 889-893, March 2004.

[12] P. Urien, M. Badra and M. Dandjinou, "EAP-TLS smartcards, from dream to reality," In Proc. of Applications and Services in Wireless Networks, pp. 39-45, Aug. 2004,

[13] C.H. Yang, L.J. Lin and K. Sakurai, "An Integration of PKI and IC cards for IPSec," In Proc. of Sympoisum on Cryptography and Information Security, Jan. 2004.

[14] P. Urien and S. Elrharbi, "Tandem smart cards: enforcing trust for TLS-based network services," In Proc. of Applications and Services in Wireless Networks, pp. 96-104. Oct. 2008.

[15] CISCO Systems. "Voice Over IP - Per Call Bandwidth Consumption," Tech Notes, Document ID : 7934, Feb. 2006.

〈저자소개〉

사 진

강 철 오 (Cheol-Oh Kang) 정회원
1993년 2월: 인하대학교 전자계산학과 졸업
1995년 2월: 인하대학교 컴퓨터공학과 석사 졸업
2000년 2월~현재: 한국전자통신연구원 부설연구소 책임연구원
〈관심분야〉 네트워크 보안, 스마트폰 보안

사 진

김 은 찬 (Eun-Chan Kim) 정회원
1997년 2월: 숭실대학교 정보통신공학과 졸업
1999년 2월: 광주과학기술원 정보통신공학과 석사 졸업
2011년 2월: 광주과학기술원 정보통신공학과 박사 졸업
2010년 12월~현재: 한국전자통신연구원 부설연구소 선임연구원
〈관심분야〉 스마트폰 보안, 위치추적

사 진

박 재 민 (Jae-Min Park) 정회원
2004년 2월: 한동대학교 전산전자공학부 졸업
2006년 2월: 한국과학기술원 공학부 석사 졸업
2006년 1월~2011년 11월: KT 단말연구센터 연구원
2011년 12월~현재: 한국전자통신연구원 부설연구소 연구원
〈관심분야〉 스마트폰 보안, UICC 보안

류 재 철 (Jea-Cheol Ryou) 종신회원
1985년 2월: 한양대학교 산업공학과 졸업
1988년 2월: Iowa State University 전산학과 석사 졸업
1990년 2월: Northwestern University 전산학과 박사 졸업
1991년~현재: 충남대학교 전기정보통신공학부 교수
〈관심분야〉 인터넷 보안