



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2009-0097418
(43) 공개일자 2009년09월16일

(51) Int. Cl.

H04L 9/32 (2006.01) H04L 12/56 (2006.01)

(21) 출원번호 10-2008-0022542

(22) 출원일자 2008년03월11일

심사청구일자 2008년03월11일

(71) 출원인

인하대학교 산학협력단

인천 남구 용현동 253 인하대학교

(72) 발명자

이대성

인천 남구 학익동 713-4 대인파크빌 나동 202호

김기창

인천 연수구 동춘동 924 대림2차아파트 207동 404호

(74) 대리인

황광연, 강동호, 오정환

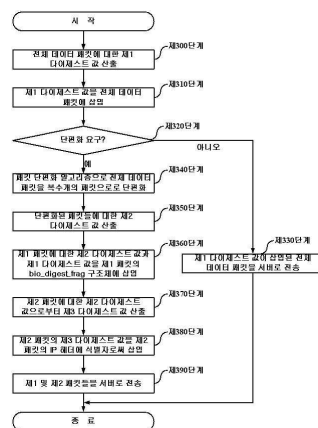
전체 청구항 수 : 총 5 항

(54) 생체정보가 삽입된 데이터 패킷의 단편화 방법 및 생체정보가 삽입된 단편화된 데이터 패킷의 인증 방법

(57) 요약

사용자의 생체정보를 이용하여 단편화된 패킷들을 인증할 수 있는 본 발명의 일 측면에 따른 생체정보가 삽입된 데이터 패킷의 단편화 방법은 (a) 서버에 의해 인증된 생체정보 중 일부를 이용하여 생성된 제1 암호키로 상기 데이터 패킷의 인증을 위한 제1 다이제스트 값을 산출하는 단계; (b) 상기 데이터 패킷의 단편화가 요구되는 경우 상기 데이터 패킷을 복수개의 패킷으로 단편화하고, 단편화된 각 패킷에 대해 상기 생체정보 중 일부를 이용하여 각 패킷 별로 생성된 제2 암호키로 각 패킷의 인증을 위한 제2 다이제스트 값을 산출하는 단계; (c) 상기 단편화된 패킷 중 최초 패킷인 제1 패킷의 IP헤더와 TCP헤더 사이의 영역에 상기 제1 다이제스트 값과 상기 제1 패킷의 제2 다이제스트 값을 삽입하는 단계; (d) 상기 단편화된 패킷 중 상기 제1 패킷을 제외한 패킷인 제2 패킷들의 제2 다이제스트 값으로부터 제3 다이제스트 값을 생성하여 상기 제2 패킷의 헤더 영역에 삽입하는 단계; 및 (e) 상기 제1 및 제2 패킷들을 상기 서버로 전송하는 단계를 포함한다.

대표도 - 도3



특허청구의 범위

청구항 1

생체정보가 삽입된 데이터 패킷의 단편화 방법으로서,

- (a) 서버에 의해 인증된 생체정보 중 일부를 이용하여 생성된 제1 암호키로 상기 데이터 패킷의 인증을 위한 제1 다이제스트 값을 산출하는 단계;
- (b) 상기 데이터 패킷의 단편화가 요구되는 경우 상기 데이터 패킷을 복수개의 패킷으로 단편화하고, 단편화된 각 패킷에 대해 상기 생체정보 중 일부를 이용하여 각 패킷 별로 생성된 제2 암호키로 각 패킷의 인증을 위한 제2 다이제스트 값을 산출하는 단계;
- (c) 상기 단편화된 패킷 중 최초 패킷인 제1 패킷의 IP헤더와 TCP헤더 사이의 영역에 상기 제1 다이제스트 값과 상기 제1 패킷의 제2 다이제스트 값을 삽입하는 단계;
- (d) 상기 단편화된 패킷 중 상기 제1 패킷을 제외한 패킷인 제2 패킷들의 제2 다이제스트 값으로부터 제3 다이제스트 값을 생성하여 상기 제2 패킷의 헤더 영역에 삽입하는 단계; 및
- (e) 상기 제1 및 제2 패킷들을 상기 서버로 전송하는 단계를 포함하는 것을 특징으로 하는 생체정보가 삽입된 데이터 패킷의 단편화 방법.

청구항 2

제1항에 있어서, 상기 (d)단계는,

- (d1) 상기 제2 패킷들의 제2 다이제스트 값으로부터 소정 비트 크기의 데이터를 추출하는 단계;
- (d2) 상기 추출된 데이터와 상기 제1 패킷의 IP 헤더 영역에 포함된 식별자 정보를 XOR연산함으로 상기 제3 다이제스트 값을 산출하는 단계; 및
- (d3) 상기 제3 다이제스트 값을 상기 제2 패킷들의 IP 헤더 중 식별자 영역에 상기 제2 패킷들의 식별자로 삽입하는 단계를 포함하는 것을 특징으로 하는 생체정보가 삽입된 데이터 패킷의 단편화 방법.

청구항 3

제1항에 있어서, 상기 (b)단계에서,

상기 제2 암호키는 상기 생체정보 중 일부와 상기 각 패킷들의 오프셋 정보를 이용하여 생성되는 것을 특징으로 하는 생체정보가 삽입된 데이터 패킷의 단편화 방법.

청구항 4

생체정보가 삽입된 단편화된 패킷의 인증방법으로서,

- (a) 클라이언트로부터 단편화된 패킷들을 수신하는 단계;
- (b) 상기 수신된 패킷이 단편화 된 패킷 중 최초 패킷인 제1 패킷인 경우, 상기 제1 패킷의 다이제스트 정보를 이용하여 상기 제1 패킷을 인증하는 단계;
- (c) 상기 수신된 패킷이 상기 제1 패킷을 제외한 패킷인 제2 패킷인 경우, 상기 제2 패킷의 식별자 정보와 상기 제1 패킷의 식별자 정보를 이용하여 상기 제2 패킷을 인증하는 단계;
- (d) 상기 제2 패킷이 인증되는 경우, 상기 제2 패킷의 식별자를 상기 제1 패킷의 식별자로 대체하는 단계; 및
- (e) 상기 제1 및 제2 패킷들을 재조립 알고리즘에 의해 재조립한 후 상기 제1 패킷에 포함된 전체 데이터 패킷의 다이제스트 정보를 이용하여 상기 전체 데이터 패킷을 인증하는 단계를 포함하는 것을 특징으로 하는 생체정보가 삽입된 단편화된 패킷의 인증방법.

청구항 5

제4항에 있어서, 상기 (d)단계는,

- (d1) 상기 생체정보 중 일부와 상기 제2 패킷의 오프셋 정보를 이용하여 상기 제2 패킷의 다이제스트 값을 산출하는 단계;
- (d2) 상기 제2 패킷들의 다이제스트 값 중 소정 비트 크기의 데이터를 추출하는 단계;
- (d3) 상기 추출된 데이터와 상기 제2 패킷의 식별자에 대해 XOR 연산을 수행하는 단계; 및
- (d4) 상기 XOR연산에 의해 산출된 값과 상기 제1 패킷의 식별자를 비교함으로써 상기 제2 패킷을 인증하는 단계를 포함하는 것을 특징으로 하는 생체정보가 삽입된 단편화된 패킷의 인증방법.

명세서

발명의 상세한 설명

기술 분야

- <1> 본 발명은 데이터 패킷의 인증에 관한 것으로서 보다 상세하게는 생체정보가 삽입된 단편화된 데이터 패킷의 인증에 관한 것이다.

배경 기술

- <2> 컴퓨터 시스템의 발달로 과거에 수작업으로 진행되던 많은 일들이 자동화되고 이로 인해 사용자는 시간의 절약은 물론 보다 편리한 생활을 영위하게 되었다. 특히 최근에는, 네트워크 시스템이 급속도로 성장하면서 언제 어디서든지 시간에 구애 받지 않고 온라인 상에 접속하여 작업을 처리할 수 있게 되었다.
- <3> 네트워크를 이용한 컴퓨터 시스템의 사용은 편리성을 제공하는 반면, 직접 대면하여 일을 처리하지 않는 특성으로 인하여 신분 증명의 문제를 부각시켰다. 이러한 신분 증명의 문제를 해결하기 위해 많은 연구들이 진행되어 왔다.
- <4> 특히, 최근에는 세션이 성립되는 초기 과정에서만 인증을 수행하는 일회성 인증 방법의 문제를 해결하기 위해, 전송되는 패킷마다 인증을 수행하는 방법인 유니캐스트 환경에서의 패킷 인증과 멀티캐스트 환경에서의 패킷 인증에 대한 연구들이 수행되고 있다.
- <5> 그러나, 이러한 유니캐스트 환경에서의 패킷 인증이나 멀티캐스트 환경에서의 패킷 인증은 모두 사용자 본인에 대한 인증이라기 보다는 컴퓨터 시스템이나 프로세스에 대한 인증이라고 볼 수 있어, 사용자 본인에 대한 인증이 요구되는 분야(의료분야, 군사분야 등)에서는 적용의 한계가 있다.
- <6> 또한, 최근에는 패킷 인증을 무력화하는 기법들의 발전은 물론, 단편화된 패킷을 이용한 공격 기법들이 꾸준히 발전하고 있지만 상술한 패킷 인증 기술들은 플로우(Flow) 내의 패킷 인증에 관해서만 고려하고 있으므로 단편화된 패킷을 이용한 공격에는 용이하게 대처할 수 없다는 문제점이 있다.

발명의 내용

해결 하고자하는 과제

- <7> 본 발명은 상술한 문제점을 해결하기 위한 것으로서, 사용자의 생체정보를 이용하여 단편화된 패킷들을 인증할 수 있는 생체정보가 삽입된 데이터 패킷의 단편화 방법 및 생체정보가 삽입된 단편화된 데이터 패킷의 인증 방법을 제공하는 것을 기술적 과제로 한다.
- <8> 또한, 본 발명은 단편화된 패킷을 인증하면서도 인증 정보를 각 패킷마다 별도로 삽입하지 않는 생체정보가 삽입된 데이터 패킷의 단편화 방법 및 생체정보가 삽입된 단편화된 데이터 패킷의 인증 방법을 제공하는 것을 다른 기술적 과제로 한다.

과제 해결수단

- <9> 상술한 목적을 달성하기 위한 본 발명의 일 측면에 따른 생체정보가 삽입된 데이터 패킷의 단편화 방법은 (a) 서버에 의해 인증된 생체정보 중 일부를 이용하여 생성된 제1 암호키로 상기 데이터 패킷의 인증을 위한 제1 다이제스트 값을 산출하는 단계; (b) 상기 데이터 패킷의 단편화가 요구되는 경우 상기 데이터 패킷을 복수개의 패킷으로 단편화하고, 단편화된 각 패킷에 대해 상기 생체정보 중 일부를 이용하여 각 패킷 별로 생성된 제2 암호

호기로 각 패킷의 인증을 위한 제2 다이제스트 값을 산출하는 단계; (c) 상기 단편화된 패킷 중 최초 패킷인 제1 패킷의 IP헤더와 TCP헤더 사이의 영역에 상기 제1 다이제스트 값과 상기 제1 패킷의 제2 다이제스트 값을 삽입하는 단계; (d) 상기 단편화된 패킷 중 상기 제1 패킷을 제외한 패킷인 제2 패킷들의 제2 다이제스트 값으로부터 제3 다이제스트 값을 생성하여 상기 제2 패킷의 헤더 영역에 삽입하는 단계; 및 (e) 상기 제1 및 제2 패킷들을 상기 서버로 전송하는 단계를 포함한다.

<10> 이때, 상기 (d)단계는, (d1) 상기 제2 패킷들의 제2 다이제스트 값으로부터 소정 비트 크기의 데이터를 추출하는 단계; (d2) 상기 추출된 데이터와 상기 제1 패킷의 IP 헤더 영역에 포함된 식별자 정보를 XOR연산함으로써 상기 제3 다이제스트 값을 산출하는 단계; 및 (d3) 상기 제3 다이제스트 값을 상기 제2 패킷들의 IP 헤더 중 식별자 영역에 상기 제2 패킷들의 식별자로 삽입하는 단계를 포함할 수 있다.

<11> 일 실시예에 있어서, 상기 (b)단계에서, 상기 제2 암호키는 상기 생체정보 중 일부와 상기 각 패킷들의 오프셋 정보를 이용하여 생성되는 것을 특징으로 한다.

<12> 상술한 다른 목적을 달성하기 위한 본 발명의 일 측면에 따른 생체정보가 삽입된 단편화된 패킷의 인증방법은, (a) 클라이언트로부터 단편화된 패킷들을 수신하는 단계; (b) 상기 수신된 패킷이 단편화된 패킷 중 최초 패킷인 제1 패킷인 경우, 상기 제1 패킷의 다이제스트 정보를 이용하여 상기 제1 패킷을 인증하는 단계; (c) 상기 수신된 패킷이 상기 제1 패킷을 제외한 패킷인 제2 패킷인 경우, 상기 제2 패킷의 식별자 정보와 상기 제1 패킷의 식별자 정보를 이용하여 상기 제2 패킷을 인증하는 단계; (d) 상기 제2 패킷이 인증되는 경우, 상기 제2 패킷의 식별자를 상기 제1 패킷의 식별자로 대체하는 단계; 및 (e) 상기 제1 및 제2 패킷들을 재조립 알고리즘에 의해 재조립한 후 상기 제1 패킷에 포함된 전체 데이터 패킷의 다이제스트 정보를 이용하여 상기 전체 데이터 패킷을 인증하는 단계를 포함한다.

<13> 이때, 상기 (d)단계는, (d1) 상기 생체정보 중 일부와 상기 제2 패킷의 오프셋 정보를 이용하여 상기 제2 패킷의 다이제스트 값을 산출하는 단계; (d2) 상기 제2 패킷들의 다이제스트 값 중 소정 비트 크기의 데이터를 추출하는 단계; (d3) 상기 추출된 데이터와 상기 제2 패킷의 식별자에 대해 XOR 연산을 수행하는 단계; 및 (d4) 상기 XOR연산에 의해 산출된 값과 상기 제1 패킷의 식별자를 비교함으로써 상기 제2 패킷을 인증하는 단계를 포함할 수 있다.

효 과

<14> 상술한 바와 같이 본 발명에 따르면, 단편화된 패킷 마다 사용자의 생체정보를 삽입함으로써 단편화된 패킷 공격에 대해 용이하게 대처함은 물론 사용자 본인에 대한 인증을 수행하기 때문에 인증의 신뢰성을 향상시킬 수 있다는 효과가 있다.

<15> 또한, 본 발명은 단편화된 패킷 마다 생체정보를 삽입하되, 생체정보 중 일부를 이용하여 생성된 인증정보만을 별도의 데이터로 삽입하는 것이 아니라 IP 헤더 내의 식별자 영역에 삽입하기 때문에 전송하는 패킷의 크기 감소는 물론 이로 인해 시스템의 부하 증가를 최소화할 수 있다는 효과가 있다.

발명의 실시를 위한 구체적인 내용

<16> 이하 첨부된 도면을 참조하여 본 발명의 실시예에 대해 상세히 설명한다. 먼저, 본 발명의 상세한 설명을 하기 위해, 도 1을 참조하여 본 발명에 따른 방법이 적용될 수 있는 인증 시스템에 대해 간략하게 설명한다.

<17> 도 1에 도시된 바와 같이, 본 발명에 따른 방법이 적용되는 인증 시스템(100)은 서버 리스트가 저장되는 커널 메모리(102)를 포함하는 클라이언트(104), 클라이언트 리스트가 저장되는 커널 메모리(106) 및 사용자의 생체정보가 저장되는 생체정보 DB(108)를 포함하는 서버(110)를 포함하고, 클라이언트(104) 및 서버(110)는 상호간에 데이터를 송수신할 수 있도록 통신망(112)으로 연결된다.

<18> 먼저, 클라이언트(104)는 생체인증이 요구되는 경우, 사용자로부터 실시간으로 획득되는 생체정보를 bio_crypto_info라는 구조체에 포함시켜 램포트(Lamport)방식에 의해 생성된 세션키를 이용하여 암호화한 후 SYN 패킷에 포함시켜 서버(110)로 전송한다. 이때 bio_crypto_info 구조체는 도 2a에 도시된 바와 같이 SYN 패킷의 IP 헤더 영역과 TCP 헤더 영역 사이의 영역에 삽입된다. 만약, 생체인증이 요구되지 않는 경우라면, 클라이언트(104)는 일반적인 TCP/IP프로토콜 흐름에 따라 SYN 패킷을 전송하게 된다.

<19> 또한, 클라이언트(104)는 서버(110)에 의해 생체정보가 인증되는 경우, 인증된 생체정보 중 일부를 추출하고, 추출된 생체정보의 일부를 이용하여 데이터 패킷의 인증을 위한 다이제스트 값을 산출하여 데이터 패킷 내에 삽

입한 후 서버(110)로 전송한다. 이때, 패킷의 인증을 위한 다이제스트 값은 도 2b에 도시된 바와 같이, 데이터 패킷 내의 IP 헤더와 TCP 헤더 사이의 영역에 삽입되는 bio_digest_info라는 구조체에 포함된다.

<20> 여기서, 생체정보 중 일부를 추출하기 위한 지시사항은 서버(110)로부터 전송되는 ACK|SYN 패킷에 삽입된 bio_crypto_info 구조체 내에 포함되되, 일회성 세션키를 통해 암호화되어 있다. 이하에서는 클라이언트(104)가 bio_crypto_info 구조체 내에 포함되어 있는 지시사항을 이용하여 생체정보 중 일부를 추출하고, 추출된 생체정보를 이용하여 데이터 패킷의 인증을 위한 다이제스트 값을 산출하는 방법에 대해 간략히 설명한다.

<21> bio_crypto_info 구조체 내에는 bio_data, seed_RNG, 및 mod_divisor이 포함되어 있는데, bio_data에는 클라이언트(104)에서 수집된 생체정보가 포함되어 있고, seed_RNG와 mod_divisor은 생체정보가 서버(110)에 의해 인증되었을 때 서버(110)에 의해 전송되는 것으로서, seed_RNG는 난수 생성 함수에 대한 seed값을 나타내고, mod_divisor은 생성된 난수로부터 나머지 연산 적용시에 제수(Divisor)를 나타낸다.

<22> 즉, 클라이언트(104)는 아래의 수학적 식 1에 도시된 바와 같이, 상술한 정보들을 이용하여 key_start_bit를 생성하고, 생체정보 중 이러한 key_start_bit를 기준으로 56비트의 생체정보를 추출하며, 추출된 생체정보를 암호키로 하여 데이터 패킷의 인증을 위한 다이제스트 값을 산출하게 되는 것이다.

수학적 식 1

<23> $key_start_bit = RNG(seed_RNG) \% mod_divisor$

<24> 서버(110)는 클라이언트(104)로부터 생체정보가 포함된 SYN 패킷이 수신되면, SYN 패킷에 포함되어 있는 생체정보를 클라이언트(104)와 공유하고 있는 세션키로 복호화한 후 복호된 생체정보를 생체정보 DB(108)에 저장되어 있는 사용자의 생체정보와 비교함으로써 생체정보를 인증한다. 이때, 인증된 생체정보는 클라이언트(104)와의 통신에 있어서 현재 세션의 여러 패킷들을 인증하기 위해 세션이 끝날 때까지 커널 메모리(102) 상에 상주된다.

<25> 또한, 서버(110)는 생체정보가 인증되면 ACK|SYN 패킷에 포함된 bio_crypto_info 구조체 내에 생체정보 중 일부를 추출하기 위한 지시사항을 포함시킨 후, 이를 암호화하여 클라이언트(104)로 전송한다.

<26> 이하, 상술한 클라이언트가 생체정보가 삽입된 데이터 패킷을 단편화하는 방법에 대해 도 3을 참조하여 구체적으로 설명한다.

<27> 먼저, 서버로 전송할 전체 데이터 패킷의 인증을 위한 제1 다이제스트 값을 산출한다(제300단계). 이때, 전체 데이터 패킷의 인증을 위한 제1 다이제스트 값은 상술한 수학적 식 1을 이용하여 산출된 key_start_bit를 기준으로 생체정보 중 일부인 56비트의 생체정보를 추출한 후, 추출된 생체정보를 암호키로 이용하여 산출된다.

<28> 이후, 산출된 전체 데이터 패킷에 대한 다이제스트 값(Digest T)을 도 4에 도시된 바와 같이, 전체 데이터 패킷의 bio_digest_frag라는 구조체에 삽입한다(제310단계).

<29> 다음으로, 생체정보가 삽입되어 있는 전체 데이터 패킷의 크기를 고려하여 전체 데이터 패킷에 대한 단편화가 요구되는지 여부를 판단한다(제320단계). 판단결과, 단편화가 요구되지 않는 경우 전체 데이터 패킷의 다이제스트 값이 삽입되어 있는 전체 데이터 패킷을 서버로 전송한다(제330단계).

<30> 한편, 제320단계의 판단결과 전체 데이터 패킷에 대한 단편화가 요구되는 경우, 패킷 단편화 알고리즘을 이용하여 전체 데이터 패킷을 소정 개수의 패킷들로 단편화한다(제340단계).

<31> 이후, 단편화된 패킷들의 인증을 위한 제2 다이제스트 값을 산출한다(제350단계). 일 실시예에 있어서, 각각의 단편화된 패킷에 대한 제2 다이제스트 값은 각 패킷들의 프래그먼트 오프셋(frag_offset)값과 서버로부터 전송된 bio_crypto_info 구조체 내에 포함되어 있는 지시사항 중 mod_divisor을 이용하여 산출된다.

<32> 여기서, 각 패킷들의 프래그먼트 오프셋은 도 5에 도시된 바와 같은 IP헤더(500) 중 프래그먼트 오프셋 필드(510)에 기록되어 있는 것으로서, IP계층에서 제공되는 단편화 알고리즘에 의해 자동으로 계산된다.

<33> 각 패킷에 대한 제2 다이제스트 값 산출 과정을 도 6을 참조하여 구체적으로 설명한다. 먼저, 아래의 수학적 식 2를 이용하여 각 패킷 별로 key_start_bit를 산출한다(제600단계).

수학적 식 2

<34> $key_start_bit = frag_offset \% mod_divisor$

- <35> 이후, 수집된 생체정보 중에서 key_start_bit로부터 56비트 영역을 추출하여 암호키를 생성한 후(제610단계), 생성된 암호키를 이용하여 각 패킷에 대해 160비트의 제2 다이제스트 값을 산출하게 된다(제620단계). 이때, 각 패킷들은 단편화 알고리즘에 의해 서로 다른 프래그먼트 오프셋 값을 갖기 때문에 제610단계에서 산출되는 암호키도 각 패킷 별로 다르게 되고, 따라서 제620단계에 산출되는 패킷들의 제2 다이제스트 값도 각 패킷마다 다르게 된다.
- <36> 다시 도 3을 참조하면, 단편화된 패킷 중 최초 패킷(이하, '제1 패킷'이라 함)에 대해, 제1 패킷의 제2 다이제스트 값(Digest F)과 전체 데이터 패킷에 대한 제1 다이제스트 값(Digest T)을 도 7에 도시된 바와 같은 자료구조를 가지는 bio_digest_frag라는 구조체 영역에 삽입한다(제360단계). 이때, bio_digest_frag 구조체 영역은 IP헤더 영역과 TCP 헤더 영역 사이에 삽입된다. 여기서, 제1 패킷 내에 전체 데이터 패킷에 대한 제1 다이제스트 값을 함께 포함시키는 것은 단편화된 패킷들이 서버로 전송되었을 때, 단편화된 패킷들이 재조립된 데이터 패킷을 정확하게 인증하기 위한 것이다.
- <37> 이러한 bio_digest_frag라는 구조체를 포함하는 제1 패킷의 구조가 도 4에 도시되어 있다. 제1 패킷은 IP헤더와 TCP헤더 사이에 bio_digest_frag라는 구조체 영역이 포함되어 있고, bio_digest_frag라는 구조체내에는 전체 데이터 패킷에 대한 제1 다이제스트 값(Digest T)와 제1 패킷에 대한 제2 다이제스트 값(Digest F)가 포함되어 있음을 알 수 있다.
- <38> 다시 도 3을 참조하면, 단편화된 패킷 중 최초 패킷을 제외한 나머지 패킷(이하, '제2 패킷'이라 함)에 대해, 각 패킷의 제2 다이제스트 값으로부터 제3 다이제스트 값을 산출한다(제370단계). 여기서, 제3 다이제스트 값을 산출하는 이유는, 단편화된 모든 패킷마다 160비트의 크기를 제2 다이제스트 값을 삽입하는 경우, 패킷의 크기가 너무 커지기 때문에 이를 방지하기 위한 것이다. 이러한 제3 다이제스트 값의 생성과정을 도 8을 참조하여 구체적으로 설명한다.
- <39> 먼저, 제2 패킷들의 제2 다이제스트 값으로부터 일부 비트 크기의 인증 정보를 추출한다(제800단계). 일 실시예에 있어서, 제2 다이제스트 값은 160비트의 크기이므로, 아래의 수학적 식 3을 이용하여 제2 다이제스트 값의 10%에 해당하는 크기인 16비트를 인증정보로써 제2 다이제스트 값으로부터 추출할 수 있다.

수학적 식 3

<40>
$$\text{digest_start_bit} = (\text{frag_offset} \% 10) \times 16$$

- <41> 수학적 식 3은 160비트의 제2 다이제스트 값으로부터 16비트의 인증정보를 추출하기 위한 비트 시작 위치를 구하는 식으로써, frag_offset는 단편화 알고리즘에 의해 제공되는 값이고, 계산의 편의를 위해 10으로 나머지 연산을 수행한다. 즉, 160비트의 제2 다이제스트 값 중에서 수학적 식 3을 이용하여 산출되는 digest_start_bit로부터 16비트를 추출하는 것이다.
- <42> 다음으로, 추출된 16비트의 인증정보와 제1 패킷의 IP헤더 중 식별자 필드에 기록되어 있는 제1 패킷의 식별자에 대해 XOR(eXclusive OR)연산을 수행함으로써 제3 다이제스트 값을 산출한다(제810단계).
- <43> 다시, 도 3을 참조하면, 제2 패킷들의 제3 다이제스트 값을 도 5에 도시된 바와 같은 패킷들의 IP헤더(500) 중 식별자 필드(520)에 각 패킷들의 식별자로써 삽입한다(제380단계). 즉, 제1 패킷의 경우에는 제1 패킷에 대한 제2 다이제스트 값 전체를 제1 패킷 중 IP헤더와 TCP헤더 사이에 삽입하지만, 제2 패킷들의 경우 제2 패킷들에 대한 제2 다이제스트 값의 일부를 이용하여 제3 다이제스트 값을 산출하고, 이러한 제3 다이제스트 값을 제2 패킷들의 IP헤더에 직접 삽입함으로써 인증정보를 모든 패킷마다 삽입하되, 패킷의 크기가 증가되지 않도록 하는 것이다.
- <44> 이러한 과정에서, 제2 패킷들의 제2 다이제스트 값으로부터 추출된 일부 비트의 값을 직접 제2 패킷의 IP헤더에 삽입하지 않는 것은 서버에서 단편화된 패킷들을 재조립(Reassembly)할 수 있도록 고려하여야 하기 때문이다.
- <45> 제2 패킷들의 제3 다이제스트 값 산출 과정 및 제3 다이제스트 값을 제2 패킷들의 IP 헤더 중 식별자 필드에 삽입하는 과정을 도식화하면 도 9에 도시된 바와 같다. 도 9에 도시된 바와 같이, k번째 패킷에 대해 상술한 수학적 식 2를 이용하여 추출된 암호키를 이용하여 k번째 패킷에 대한 160비트 크기의 제2 다이제스트 값을 산출하고, 160비트 크기의 제2 다이제스트 값에 상술한 수학적 식 3을 적용함으로써 16비트 크기의 인증정보를 추출한다. 이후, 추출된 16비트 크기의 인증정보와 제1 패킷의 식별자에 대해 XOR연산을 수행함으로써 제3 다이제스트 값을 산출하고, 산출된 제3 다이제스트 값을 k번째 패킷의 IP 헤더 중 식별자 영역에 삽입하는 것이다.

- <46> 다시, 도 3을 참조하면, 마지막으로 제1 패킷과 제2 패킷들을 서버로 전송한다(제390단계). 이때, 서버로 전송되는 제1 패킷과 제2 패킷의 구조는 도 10에 도시된 바와 같다. 도 10(a)에 도시된 포맷은 단편화된 패킷들이 조립되었을 때의 패킷 구조를 나타내고, 도 10(b)에 도시된 포맷은 단편화된 패킷 중 최초 패킷인 제1 패킷의 구조를 나타내고, 도 10(c) 및 (d)는 단편화된 패킷 중 최초 패킷을 제외한 패킷인 제2 패킷들을 나타내는 것으로서, 도 10(c)는 단편화된 패킷 중 두 번째 패킷의 구조를 나타내고, 도 10(d)는 단편화된 패킷 중 세 번째 패킷의 구조를 나타낸다.
- <47> 상술한 바와 같은 데이터 패킷의 단편화 과정을 알고리즘으로 표현하는 경우 도 11(a) 및 (b)에 도시된 바와 같이 표현될 수 있다. 도 11(a)는 전체 데이터 패킷이 단편화 되기 이전까지의 과정을 알고리즘으로 표현한 것이고, 도 11(b)는 전체 데이터 패킷을 단편화한 이후의 과정을 알고리즘으로 표현한 것이다.
- <48> 다음으로, 생체정보가 삽입된 단편화된 패킷을 인증하는 방법에 대해 도 12를 참조하여 구체적으로 설명한다.
- <49> 먼저, 클라이언트로부터 단편화된 패킷들을 수신한다(제1200단계). 수신된 패킷이 수신되면, 제1 패킷의 `bio_digest_frag` 구조체 영역에 포함되어 있는 제1 패킷에 대한 다이제스트 값을 이용하여 제1 패킷을 인증한다(제1220단계).
- <50> 한편, 제2 패킷이 수신되면 제2 패킷의 IP 헤더 중 식별자 영역에 기록되어 있는 식별자 정보와 제1 패킷의 IP 헤더 중 식별자 영역에 기록되어 있는 제1 패킷의 식별자 정보를 이용하여 제2 패킷을 인증한다(제1230단계). 이하에서는 도 13을 참조하여 제2 패킷의 인증과정을 보다 상세하게 설명한다.
- <51> 먼저, 제2 패킷의 IP 헤더 중 프래그먼트 오프셋 필드에 기록되어 있는 제2 패킷의 오프셋 정보와 클라이언트로 전송했던 생체정보 추출을 위한 지시사항에 포함된 `mod_divisor`를 이용하여 160비트 크기의 다이제스트 값을 산출한다(제1300단계). 이때, 160비트 크기의 다이제스트 값은 클라이언트로부터 전송된 생체정보 중 상술한 수학식 2를 이용하여 산출된 `key_start_bit`로부터 56비트 영역을 추출함으로써 생성된 암호키를 이용하여 생성된다.
- <52> 이후, 160 비트 크기의 다이제스트 값에 상술한 수학식 3을 적용함으로써 산출된 `digest_start_bit`를 이용하여 16비트 크기의 인증정보를 추출한다(제1310단계).
- <53> 이후, 16비트 크기의 인증정보와 제2 패킷의 IP 헤더 중 식별자 영역에 기록되어 있는 식별자 정보에 대해 XOR 연산을 수행하고(제1320단계), XOR연산에 따라 생성된 결과값을 제1 패킷의 IP 헤더 중 식별자 영역에 기록되어 있는 제1 패킷의 식별자 정보와 비교한다(제1330단계). 이때, XOR연산에 의해 생성된 결과값이 제1 패킷의 식별자 정보와 동일한 경우 해당 제2 패킷은 인증에 성공한 것이 되고, XOR연산에 의해 생성된 결과값이 제1 패킷의 식별자 정보와 동일하지 않은 경우 해당 제2 패킷의 인증은 실패한 것이 된다.
- <54> 이러한 제2 패킷의 인증 방법을 도식화하면 도 14에 도시된 바와 같다.
- <55> 다시 도 12를 참조하면, 제2 패킷의 인증이 성공하는 경우, 제2 패킷의 IP 헤더 중 식별자 영역에 기록되어 있는 식별자 정보를 제1 패킷의 식별자 정보로 대체한다(제1240단계). 이는 제2 패킷들의 경우, 제2 패킷에 대한 인증정보를 제2 패킷의 IP 헤더에 포함시키기 위해 제2 패킷들의 식별자 정보를 변경하였기 때문에, 단편화된 패킷들을 재조립하기 위해 제2 패킷들의 식별자 정보를 원래 값으로 복원시키기 위한 것이다.
- <56> 이후, 수신된 제1 패킷 및 제2 패킷들을 패킷 재조립 알고리즘을 이용하여 재조립한 후(제1250단계), 마지막으로 제1 패킷의 `bio_digest_frag` 구조체 내에 포함되어 있는 전체 데이터 패킷에 대한 다이제스트 값(Digest T)를 이용하여 재조립된 전체 데이터 패킷에 대한 인증을 수행한다(제1260단계).
- <57> 상술한 실시예에 있어서는 패킷들의 수신 과정에서 제1 패킷이 먼저 수신되는 것으로 가정하여 단편화된 패킷들을 인증하는 방법에 대해 기술하였다. 그러나, 항상 제1 패킷이 먼저 수신된다고 보장할 수는 없다. 따라서, 제2 패킷들이 제1 패킷보다 먼저 수신되는 경우를 대비하여 본 발명에서는 도 15에 도시된 바와 같은 자료구조를 추가로 정의한다.
- <58> 도 15를 참조하면, `frag_info` 구조체는 제1 패킷이 수신되지 않은 상황에서 제2 패킷들이 먼저 도착하는 경우를 처리하기 위해 추가된 자료구조이다. 일 실시예에 있어서, 단편화된 패킷들의 관리를 담당하는 `frag_info` 구조체는 빠른 검색을 위해 해쉬함수를 이용할 수 있다. 도 16은 이러한 `frag_info` 구조체에 사용되는 해쉬함수를 나타내는 알고리즘으로 해쉬에 사용되는 `id_or_XOR`값은 제1 패킷인 경우, 제1 패킷의 식별자 정보를 의미하고, 제2 패킷인 경우 제2 패킷으로부터 산출된 16비트 크기의 인증정보와 제2 패킷의 식별자 정보를 XOR연산하여 계

산된 값이다.

- <59> 도 15에 도시된 Frag_info 구조체 중 fragments 변수는 제1 패킷보다 먼저 수신된 제2 패킷들을 임시로 복사해 놓는 연결 리스트로 제1 패킷이 수신될 때까지 먼저 수신된 제2 패킷들을 대기시킨다.
- <60> 이러한 경우, 무한 시간 동안 대기하는 것이 아니고 frag_info 구조체의 timer 변수에 등록된 시간 동안만 대기 하게 된다. 즉, timer 변수에 등록된 시간이 만료되기 전에 제1 패킷이 도착하게 되면, fragments 변수에 대기 중인 제2 패킷들은 인증에 성공한 것이 되어 재조립을 위해 IP계층으로 전달된다. 그러나, 제1 패킷이 수신되기 전에 timer 변수에 등록된 시간이 만료된다면 fragments 변수에 대기중인 제2 패킷들은 인증에 실패한 것으로 판단되어 폐기된다.
- <61> 한편, frag_info 구조체의 first_frag_arrived 변수는 제1 패킷의 도착 여부를 나타낸다. 제1 패킷이 도착한 경우 first_frag_arrived 변수 값은 1로 설정되며, 이때 frag_info 구조체의 id 변수는 제1 패킷의 식별자 값을 의미한다. 또한, 제1 패킷이 도착하지 않은 경우 first_frag_arrived 변수 값은 0으로 설정되며, 이때 fragments 변수에는 먼저 도착한 제2 패킷들이 대기 중이며 이때의 id변수 값은 먼저 도착한 제2 패킷의 16비트 인증정보와 제2 패킷의 식별자 정보를 XOR 연산하여 계산된 값을 의미한다.
- <62> 도 17은 상술한 frag_info 구조체를 이용하여 제1 패킷 및 제2 패킷을 관리하는 방법을 도식화한 것이다. 도 17에 도시된 바와 같이, frag_info 구조체 B는 제1 패킷이 이미 도착하였기 때문에 first_frag_arrived 변수가 1로 설정되어 있으며, fragments 변수에는 대기중인 제2 패킷들이 존재하지 않는다. 그러나, frag_info구조체 A는 제1 패킷이 아직 도착하지 않은 상황에서 2개의 제2 패킷이 먼저 도착하였기 때문에 first_frag_arrived 변수는 0으로 설정되어 있음을 알 수 있다.
- <63> 제1 패킷이 먼저 수신되지 않은 경우 제1 패킷 및 제2 패킷을 인증하는 방법을 알고리즘으로 표현하면 도 18(a)에 도시된 바와 같고, 제1 패킷 및 제2 패킷이 모두 수신된 후 재조립과정을 거친 후 전체 데이터 패킷을 인증하는 방법을 알고리즘으로 표현하면 도 18(b)에 도시된 바와 같다.
- <64> 상술한 데이터 생체정보가 포함된 데이터 패킷의 단편화 방법 및 생체정보가 포함된 단편화된 패킷의 인증방법은 다양한 컴퓨터 수단을 이용하여 수행될 수 있는 프로그램 형태로도 구현될 수 있는데, 이때 생체정보가 포함된 데이터 패킷의 단편화 방법 및 생체정보가 포함된 단편화된 패킷의 인증방법을 수행하기 위한 프로그램은 하드 디스크, CD-ROM, DVD, 롬(ROM), 램, 또는 플래시 메모리와 같은 컴퓨터로 판독할 수 있는 기록 매체에 저장된다.
- <65> 본 발명이 속하는 기술분야의 당업자는 본 발명이 그 기술적 사상이나 필수적 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다.
- <66> 그러므로, 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적인 것이 아닌 것으로 이해해야만 한다. 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 등가 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

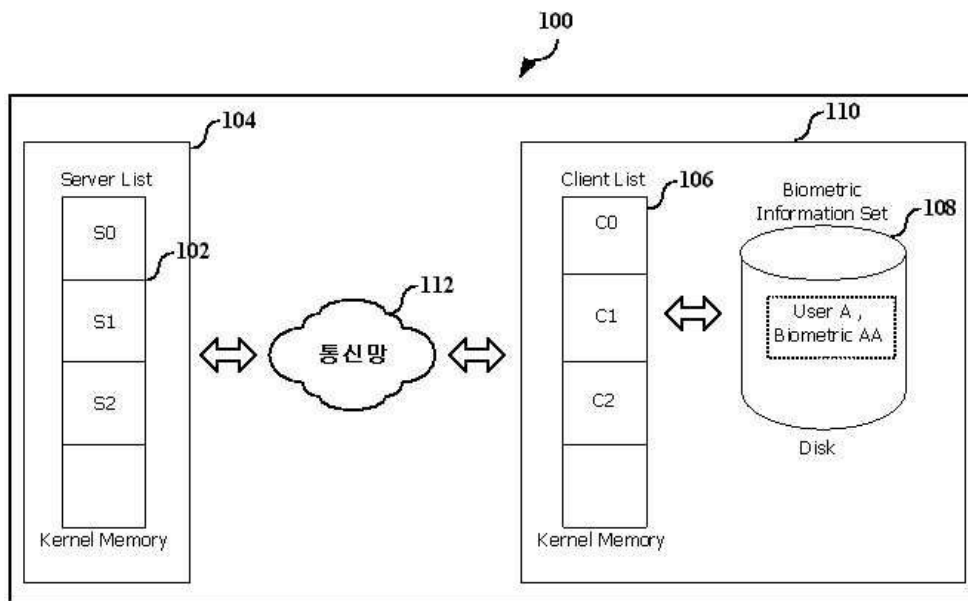
도면의 간단한 설명

- <67> 도 1은 본 발명의 일 실시예에 따른 인증 시스템의 네트워크 구성을 보여주는 도면.
- <68> 도 2는 본 발명의 일 실시예에 따라 생체정보가 삽입된 SYN 패킷 구조 및 데이터 패킷 구조를 보여주는 도면.
- <69> 도 3은 본 발명의 일 실시예에 따른 생체정보가 삽입된 데이터 패킷의 단편화 방법을 보여주는 플로우차트.
- <70> 도 4는 본 발명의 일 실시예에 따라 전체 데이터 패킷에 대한 다이제스트 값이 삽입된 데이터 패킷의 구조를 보여주는 도면.
- <71> 도 5는 각 패킷들의 IP 헤더의 구성을 보여주는 도면.
- <72> 도 6은 단편화된 패킷들에 대한 제2 다이제스트 값을 산출하는 방법을 보여주는 플로우차트.
- <73> 도 7은 bio_digest_flag 구조체의 자료구조를 보여주는 도면.
- <74> 도 8은 제2 패킷에 대한 제2 다이제스트 값으로부터 제3 다이제스트 값을 산출하는 방법을 보여주는 플로우차트.

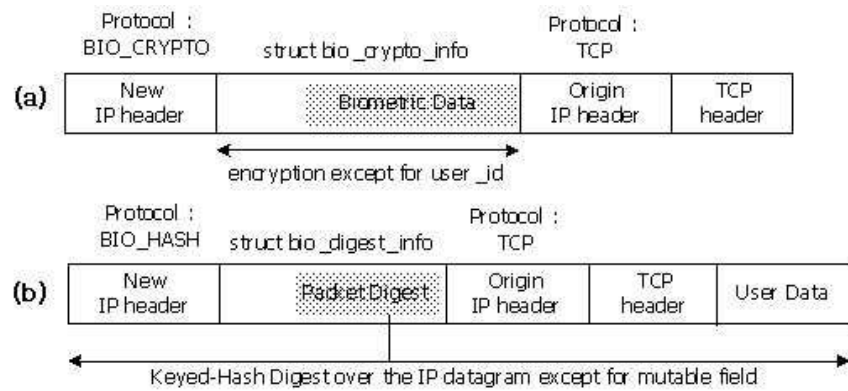
- <75> 도 9는 제2 패킷에 대한 제3 다이제스트 값의 산출과정 및 제3 다이제스트 값을 제2 패킷에 삽입하는 과정을 도식화하여 보여주는 도면.
- <76> 도 10은 단편화된 패킷들의 구조를 보여주는 도면.
- <77> 도 11은 도 3에 도시된 단편화 방법에 대한 알고리즘을 보여주는 도면.
- <78> 도 12는 본 발명의 일 실시예에 따른 생체정보가 삽입된 단편화된 패킷을 인증하는 방법을 보여주는 플로우차트.
- <79> 도 13은 제2 패킷의 인증방법을 구체적으로 보여주는 플로우차트.
- <80> 도 14는 제2 패킷의 인증방법을 도식화하여 보여주는 도면.
- <81> 도 15는 frag_info 구조체의 자료구조를 보여주는 도면.
- <82> 도 16은 frag_info 구조체에 이용되는 해쉬 알고리즘을 보여주는 도면.
- <83> 도 17은 frag_info 구조체를 이용하여 제1 패킷 및 제2 패킷을 관리하는 방법을 보여주는 도면.
- <84> 도 18은 제1 패킷이 먼저 수신되지 않은 경우 단편화된 패킷 인증을 위한 알고리즘을 보여주는 도면.

도면

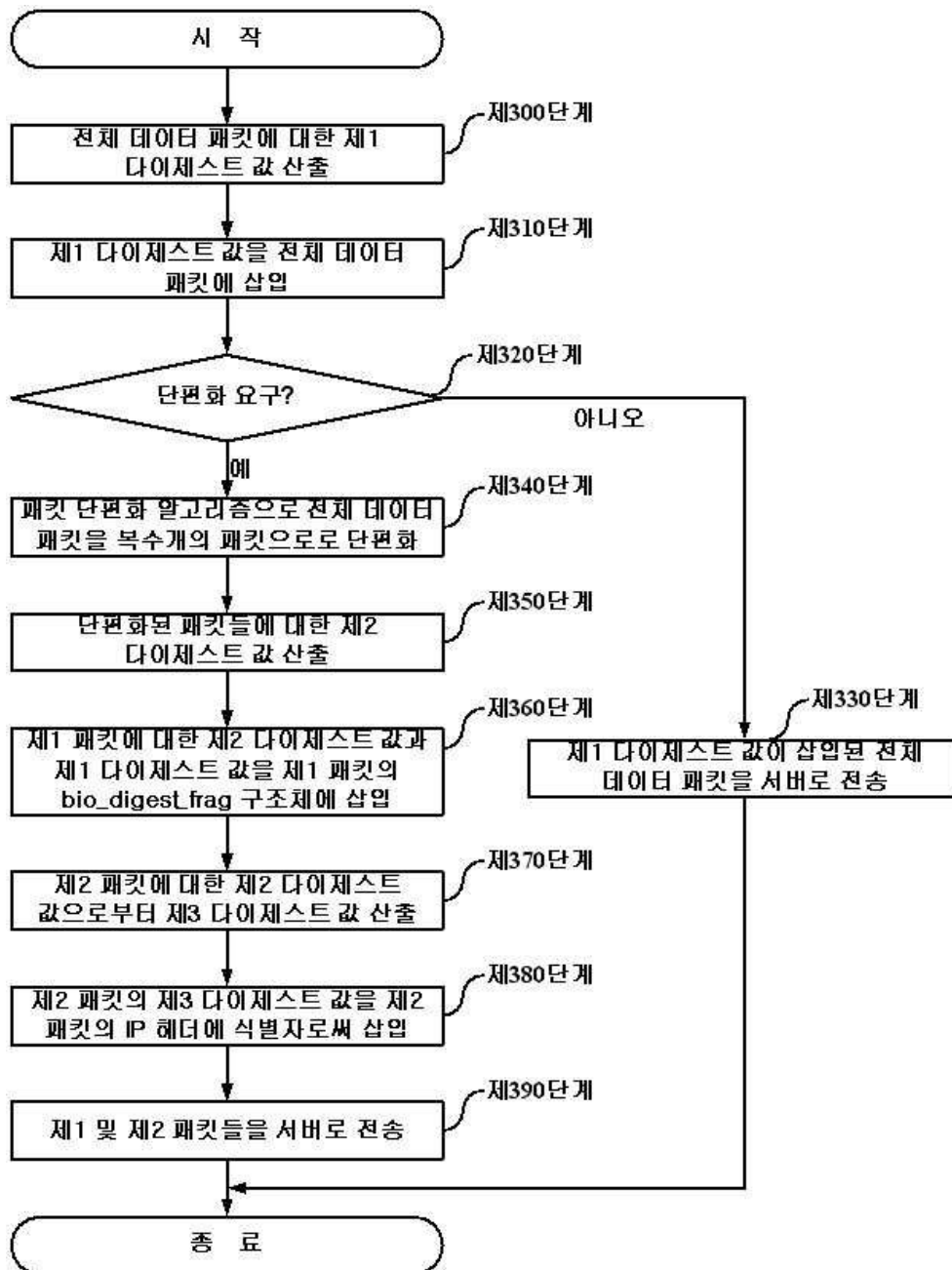
도면1



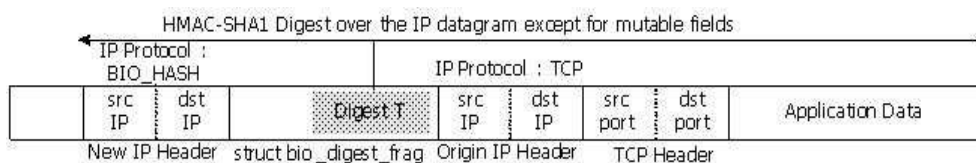
도면2



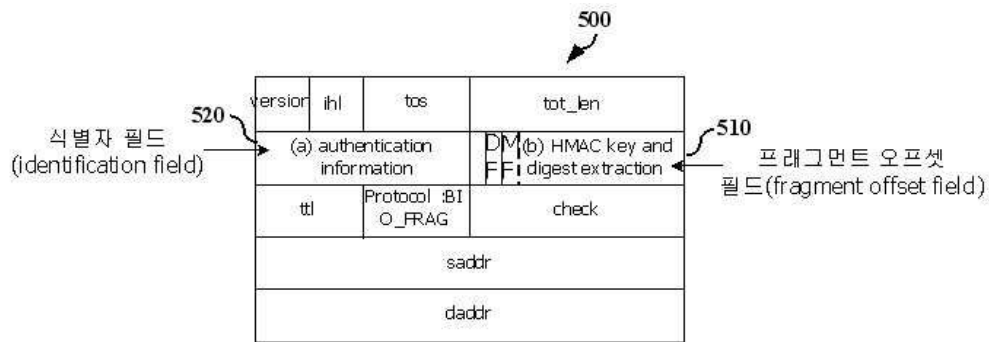
도면3



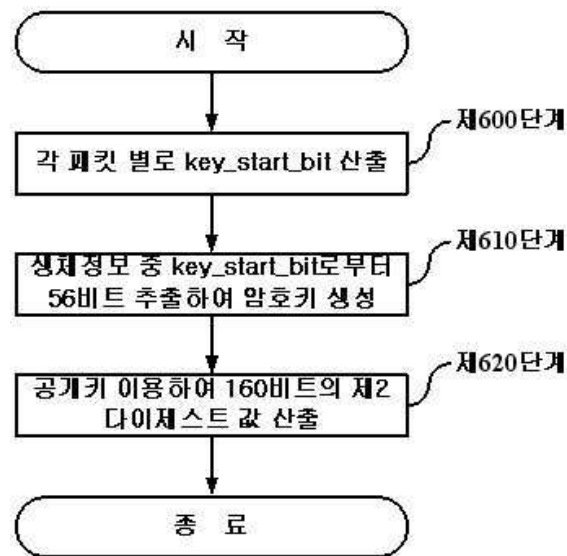
도면4



도면5



도면6

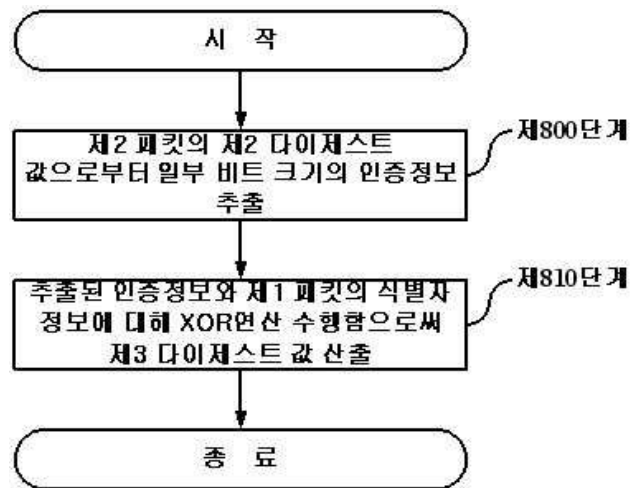


도면7

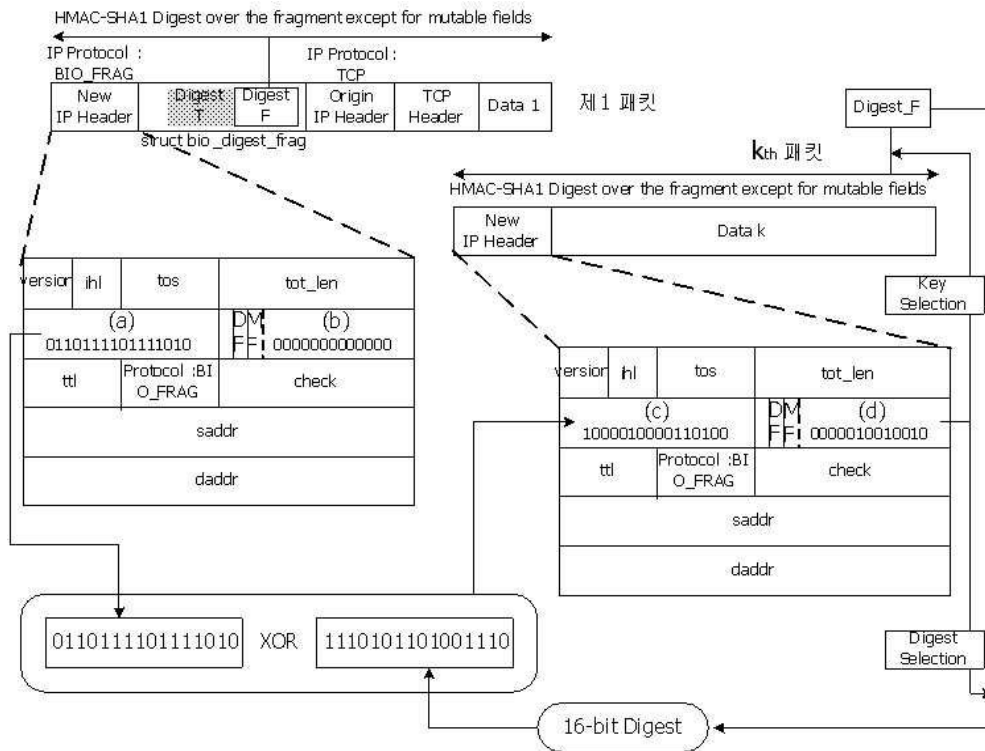
```

struct bio_digest_frag {
    struct bio_hash_header {
        unsigned long h_sequence; // sequence number for anti-replay attack
        unsigned int key_start_bit; // the start-bit number in the biometrics-data used as
                                   // per-packet symmetric-key for message authentication cod
    } h_header;
    struct bio_hash_data {
        unsigned int real_hash_len; // the size of a digested message
        unsigned char Digest_T[SHA1_DIGEST_SIZE]; // the digest over the IP datagram
        unsigned char Digest_F[SHA1_DIGEST_SIZE]; // the digest over the first fragmented packet
    } h_data;
};
    
```

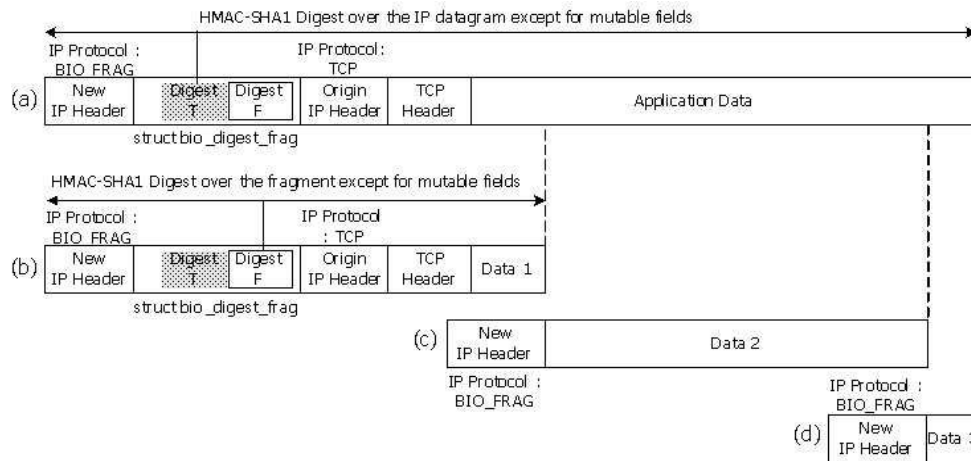
도면8



도면9



도면10



도면11a

입력 : 전송 계층으로부터 전달된 IP 패킷

출력 : BIO_FRAG 패킷, BIO_HASH 패킷

```

Outbound_BIOFRAGProcotol_BeforeFrag(P : packet) {
    let p_size be the size of P;
    if (p_size > PMTU) then { // PMTU : Path MTU
        calculate key_start_bit using;
        select a designated portion based on
        key_start_bit from the biometric data;
        digest IP_datagram using the selected portion as a key;
        copy the result into Digest_T
        of bio_digest_frag struct;
        add a new IP header with protocol BIO_FRAG;
        build an eIPBio packet and transmit;
    }
    else { // No fragmentation
        calculate key_start_bit using;
        select a designated portion based on
        key_start_bit from the biometric data;
        digest IP_datagram using the selected portion as a key;
        copy the result into HMAC_SHA1_result
        of bio_digest_info struct;
        add a new IP header with protocol BIO_HASH
        build an IPBio packet and transmit;
    }
}

```

도면11b

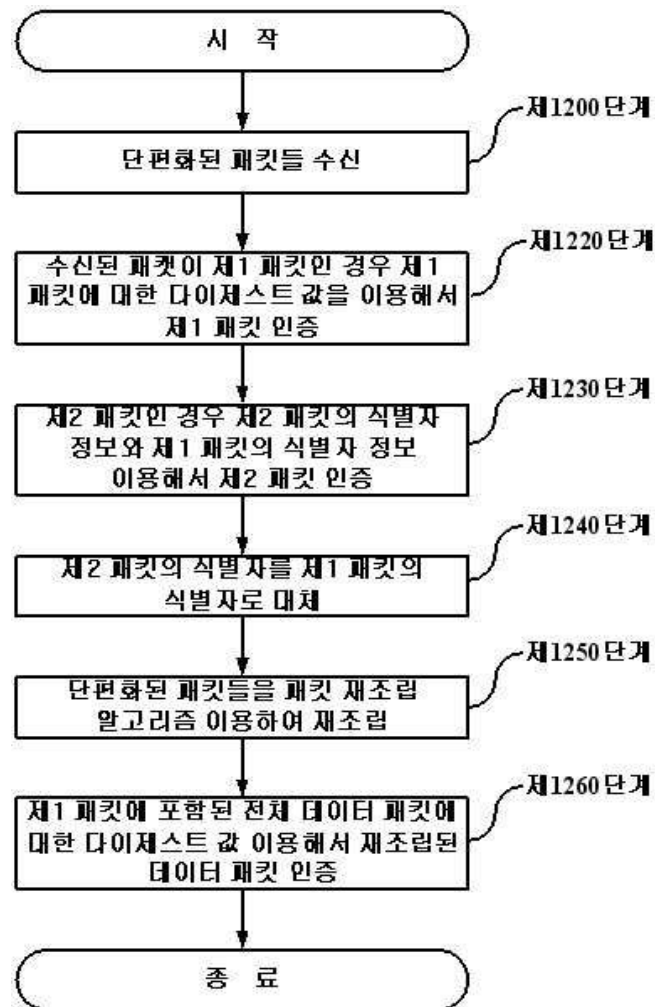
```

입력 : BIO_FRAG 패킷
출력 : BIO_FRAG 패킷

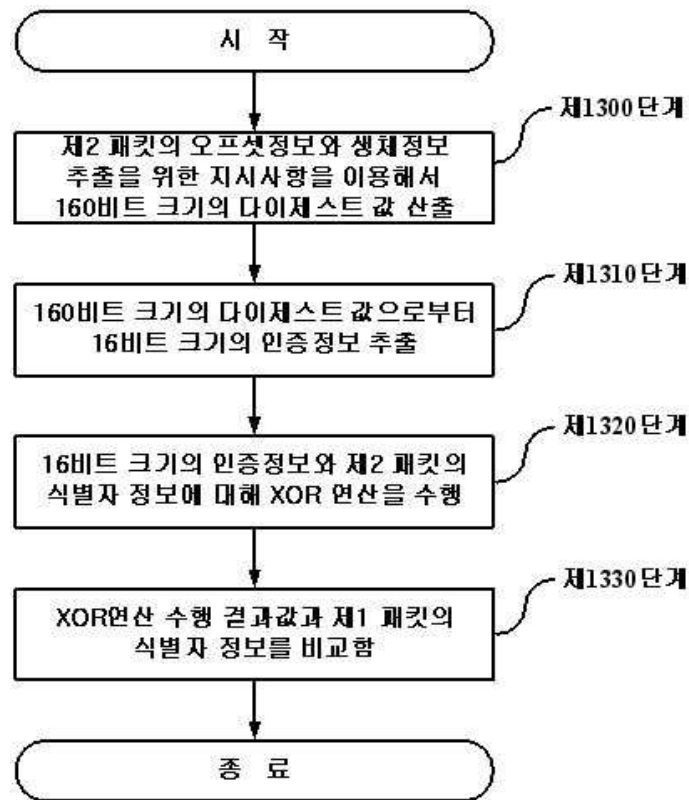
Outbound_BIOFRAGProcotol_AfterFrag(P : packet) {
  let p_type be the type of P;
  first_p = first fragmentd packet among Ps;
  if (p_type == BIO_FRAG) then {
    if (first_p == true) then {
      calculate key_start_bit using;
      select a designated portion based on
      key_start_bit from the biometric data;
      digest IP datagram using the selected portion
      as a key;
      copy the result into Digest_F
      of bio_digest_frag struct;
    }
    else {
      calculate key_start_bit using;
      select a designated portion based on
      key_start_bit from the biometric data;
      digest IP datagram using the selected portion
      as a key;
      select 16-bit portion from 160-bit digest result
      using (formula 4-2);
      XOR operation selected 16-bit portion and
      first fragmented packet's identification value;
      copy calculated XOR-value into identification field;
      recalculate IP header checksum and transmit;
    }
  }
  else
    follow normal IP protocol flow;
}

```

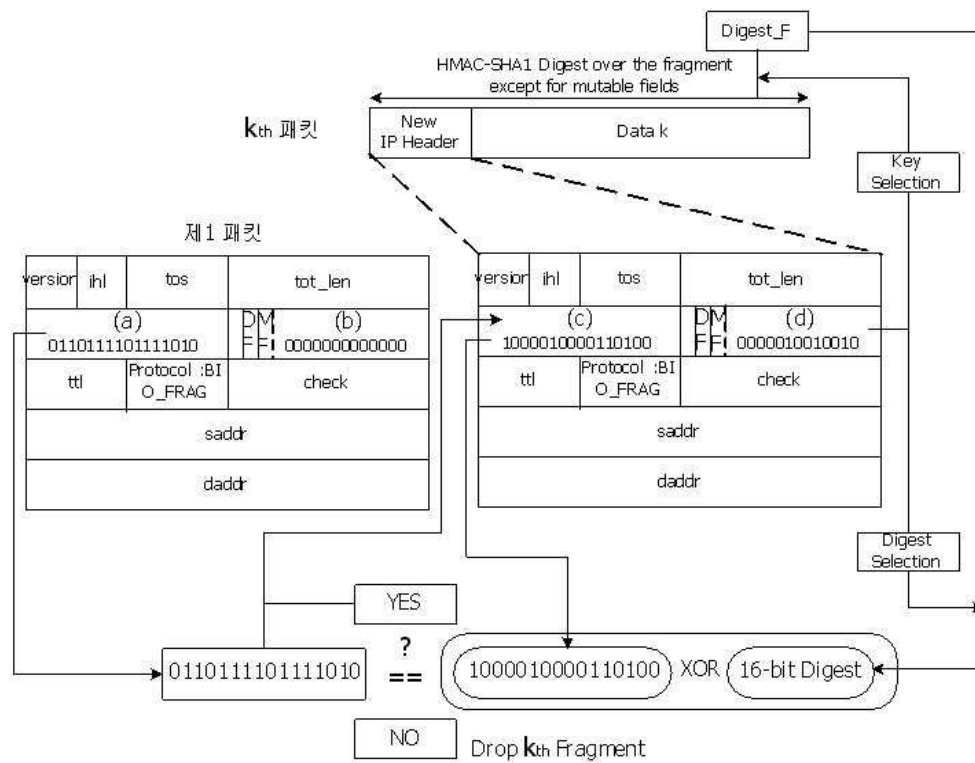
도면12



도면13



도면14



도면15

```
#define HASH_SZ 64

struct frag_info {
    struct frag_info * next;           // hash chain
    int first_frag_arrived;           // 1: arrived, 0: Not arrived
    unsigned int saddr;               // source address
    unsigned int daddr;               // destination address
    unsigned short id;                // first fragment id or calculated XOR-value
    struct sk_buff * fragments;        // linked list of received fragments
    struct timer_list timer;           // when will this frag_info expire?
};

struct frag_info * frag_hash[HASH_SZ]; // hash table
```

도면16

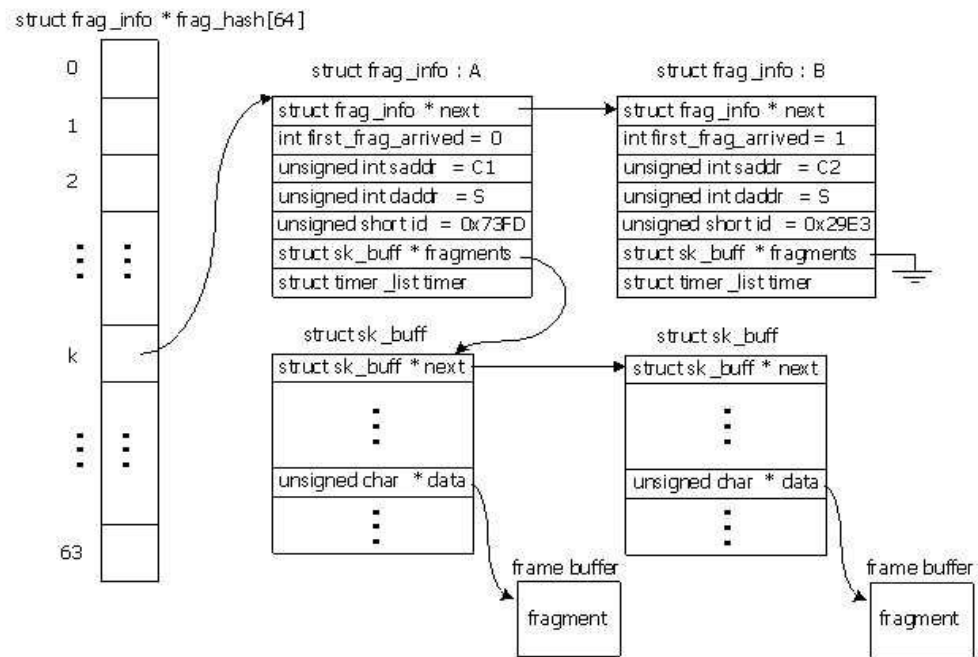
입력 : 송신자 IP 주소, 수신자 IP 주소, 16 비트 식별자 값 또는 인증 정보
출력 : 해쉬 테이블 엔트리(entry)

```
fraghashfn(saddr : source IP address,
           daddr : destination IP address,
           id_or_XOR : identification value or
                     authentication information) {

    HASH_SZ = 64;
    h = saddr ^ daddr;
    h ^= (h >> 16) ^ id_or_XOR;
    h ^= (h >> 8);

    return h & (HASH_SZ - 1);
}
```

도면17



도면18a

```

입력 : BIO_FRAG 패킷
출력 : 인증된 단편화된 BIO_FRAG 패킷

Inbound_BIOFRAGProtocol_BeforReass(P : packet) {
    let p_type is the type of P;
    first_p = first fragmentd packet among Ps;
    b_auth = false;
    if (p_type == BIO_FRAG) then {
        if (first_p == true) then {
            b_auth = verify(P, Digest_F);
            if (b_auth == true) then {
                look up hash table using source/destination
                address and identification value;
                if (pre-arrived fragments exist) then {
                    recover original IP datagram identification value;
                    relay them to IP layer for reassembly;
                    first_frag_arrived = 1;
                }
            }
            else {
                if (frag_info structure does not exist) then {
                    allocate a frag_info structure and
                    register it to hash table;
                    first_frag_arrived = 1;
                }
            }
        }
        else // b_auth == false
            drop this fragment;
    }
    else { // first_p == false
        calculate key_start_bit using (formula 4-1);
        select a designated portion based on
        key_start_bit from the biometric data;
        digest IP datagram using the selected portion
        as a key;
        select 16-bit portion from 160-bit digest result
        using (formula 4-2);
        XOR operation selected 16-bit portion and
        this fragment's identification value;
        look up hash table using source/destination address
        and calculated 16-bit XOR-value;
        if (first fragment packet already arrived) then {
            recover original IP datagram identification
            value;
            relay this fragment to IP layer for reassembly;
        }
        else if (another fragments exist) then
            link this fragment to "fragments" variable
            of frag_info struct;
        else
            allocate a frag_info struct
            and register it to hash table;
    }
}
else
    follow normal IP protocol flow;
}

```

도면18b

입력 : IP 패킷, BIO_CRYPTO 패킷, BIO_HASH 패킷, BIO_FRAG 패킷
출력 : IP 패킷, 인증된 IP 패킷

```

Inbound_eIPBioProtocol_AfterReass(P : packet) {
    let p_type is the type of P;
    b_auth = false;
    if (p_type == BIO_FRAG) then {
        b_auth = verify(P, Digest_T);
        if (b_auth == true) then
            relay this datagram to transport layer;
        else {
            free frag_info struct related to this datagram;
            drop this datagram;
        }
    }
    else if (p_type == BIO_CRYPTO || p_type = BIO_HASH) then
        [algorithm 3-2];
    else
        follow normal IP protocol flow;
}

```
