https://doi.org/10.7840/kics.2025.50.3.500

FlexMAC: 인공 신경망을 활용한 고유 해시 함수 생성 기반 보안성 강화 메시지 인증 코드 (MAC)

문 성 수◆, 박 태 준°

FlexMAC: Security-Enhanced Message Authentication Code (MAC) Based on Generating a Unique Hash Function with an Artificial Neural Network

Seongsu Moon◆, Taejune Park°

요 약

본 연구에서는 메시지 인증 코드(Message Authentication Code, MAC)에서 보안성을 강화하기 위한 새로운 시스템, FlexMAC을 제안한다. 이는 인공 신경망을 활용하여 고유하게 생성된 해시 함수를 통해 복제 또는 재현이 불가능한 고유한 인증키를 생성하는 기능을 포함하여 메시지 인증의 무결성을 강화할 뿐만 아니라 기존 MAC 시스템의 취약점들을 효과적으로 완화시킨다. 특히, 매 MAC세션마다 새로이 생성된 해시 함수를 통해 인증키를 생성함에 따라, 공격자가 이전 통신에서 사용된 인증키와 메시지를 저장해둔 후 이를 재전송하여 무결성 검증을 우회하려는 재전송 공격(Replay attack)을 효과적으로 막을 뿐만 아니라, 인증키를 파훼하기 위한 레인보우 테이블(Rainbow table) 또는 사전공격(Dictionary attack)에 필요한 비용을 기하급수적으로 증가시켜 공격을 매우 어렵게 만든다. 이러한 특성들로 인해, FlexMAC은 고도의 보안이 요구되는 환경에서 메시지의 무결성을 안전하게 보장할 수 있으며, MAC 시스템의 보안 수준을 획기적으로 향상시킬 수 있을 것으로 기대한다.

키워드 : 메시지 인증 코드, 해시 함수, 인공 신경망
key Words : Message Authentication Code (MAC), Hash function, Artificial Neural Network (ANN)

ABSTRACT

This paper proposes FlexMAC, which enhances security in a Message Authentication Code (MAC). FlexMAC generates an exclusive authentication key that cannot be duplicated or reproduced, utilizing a uniquely generated hash function with an artificial neural network, thus strengthening the integrity of message authentication and effectively mitigating vulnerabilities in traditional MAC systems. In particular, by generating a new authentication key with a newly created hash function in each MAC session, FlexMAC not only effectively prevents replay attacks—where an attacker attempts to bypass integrity verification by storing and retransmitting the authentication key and messages used in previous communications—but also exponentially increases the cost of rainbow table or dictionary attacks aimed at cracking the authentication key, making such attacks very difficult. These characteristics make FlexMAC capable of securely guaranteeing message integrity in environments requiring high security and are expected to dramatically improve the security level of MAC systems.

※ 본 논문은 2023년도 한국통신학회 동계학술발표회에서 우수논문상을 수상한 연구를 개선 및 확장한 것임.
※ 이 논문은 전남대학교 학술연구비(과제번호: 2023-0484) 지원에 의하여 연구되었음
◆ First Author : Chonnam National University Department of Artificial Intelligence, dalcw@jnu.ac.kr, 학생회원
° Corresponding Author : Chonnam National University Department of Artificial Intelligence, tajune.park@jnu.ac.kr, 종신회원
논문번호 : 202408-191-C-RN, Received August 27, 2024; Revised October 16, 2024; Accepted November 7, 2024

Ⅰ. Introduction

디지털 통신이 일상화된 현대 사회에서, 정보의 안전한 전송은 매우 중요한 과제로 자리 잡았다. 네트워크를 통해 수많은 데이터가 실시간으로 전송되면서, 이 정보들이 전송 과정에서 변조되거나 손상되지 않았음을 보장하는 것은 필수적이다. 이때 중요한 역할을 하는 것이 바로 메시지의 무결성 검증이다. 메시지의 무결성 검증은 데이터의 원본 상태를 유지하고, 이를 통해 통신의 신뢰성을 확보하는 핵심적인 매커니즘이다. 신뢰할 수 있는 데이터 전송이 보장되지 않는다면, 그 결과는 심각한 보안 위협으로 이어질 수 있다. 예를 들어, 금융 거래에서 메시지가 변조된다면, 자산 손실이나 사기 거래가 발생할 수 있으며, 이는 개인과 기업 모두에게 치명적인 영향을 미칠 수 있다. 또한, 의료 데이터나 정부 기밀 문서가 변조될 경우, 그 여파는 큰 사회적 혼란을 초래할 수 있다. 따라서, 메시지의 무결성 검증은 디지털 환경에서 안전한 정보 교환을 가능하게 하는 필수적인 보안 절차로 자리 잡고 있다.

전송된 메시지의 무결성을 확인하기 위한 방법으로는 MAC 시스템이 주로 사용된다[1]. MAC 시스템은 해시 함수를 사용해 전송할 메시지에 대한 고유 인증 코드인 MAC을 생성한 후, 이를 메시지와 함께 수신자에게 전송한다. 수신자는 전송받은 MAC을 활용해 메시지가 변조되지 않았는지 확인한다. MAC 시스템은 비교적 간단한 과정 덕분에, 무결성 검증을 위해 널리 사용되고 있다[2].

그러나 이와 같은 MAC 시스템에는 몇 가지 한계가 존재한다. 먼저, MAC 시스템이 해시 함수를 기반으로 동작하기 때문에 해시 함수 자체의 취약점이 MAC 시스템에도 영향을 미칠 수 있다. 해시 함수의 취약점으로는 서로 다른 입력값이 동일한 다이제스트로 매핑되게 하는 해시 충돌 공격과, 원하는 다이제스트가 나올 때까지 다양한 입력값을 무차별적으로 시도하는 무차별 대입 공격 등이 있다[3]. 이러한 해시 함수의 취약점은 이를 기반으로 하는 MAC 시스템의 전반적인 안전성을 위협할 수 있다. 또한, MAC 시스템 자체에도 고유한 취약점이 존재하는데, 그 중 대표적인 것이 재전송 공격(Replay attack)이다. 이는 공격자가 송신자와 수신자 사이에서 전송되는 MAC과 메시지를 가로채어 동일한 MAC과 메시지를 반복 전송함으로써 시스템을 속이는 방식이다[4]. 이러한 취약점들은 MAC 시스템의 안전성을 심각하게 저해할 수 있으며, 이로 인해 발생할 수 있는 보안 위협은 치명적인 결과를 초래할 수 있다.

본 연구에서는 기존 MAC 시스템의 취약점을 극복하고 보안성을 강화하기 위해, 인공 신경망을 활용해 해시 함수를 동적으로 생성할 수 있는 시스템을 MAC에 도입한 FlexMAC 시스템을 제안한다. FlexMAC 시스템은 각 세션마다 인공 신경망을 기반하여 매번 다른 해시 함수를 생성하고, 이를 송/수신자 간 공유하여 전송된 메시지의 무결성 검증을 수행하도록 설계되었다. 이러한 접근은 해시 충돌, 무차별 대입 공격, 그리고 재전송 공격과 같은 기존 MAC 시스템의 취약점을 효과적으로 완화할 수 있다.

해시 충돌이나 무차별 대입 공격을 수행하기 위해서는 대량의 입력값을 해시 함수에 대입하고 그 결과를 확인해야 하는 과정이 필요하다. 이때, 일반적으로 해시 함수의 출력이 가질 수 있는 경우의 수가 매우 크기 때문에(SHA-256의 경우 2^{256}의 경우의 수를 갖는다[5]), 이 과정에서 막대한 시간과 컴퓨팅 자원이 소모된다. 따라서, 공격자들은 이러한 공격을 빠르게 수행하기 위해 해시 함수의 입력과 출력의 매핑을 저장한 대규모 데이터셋(e.g., 레인보우 테이블 등)을 사전 정보로 활용한다[6]. 그러나 FlexMAC 시스템에서는 각 세션마다 무결성 검증에 사용되는 해시 함수가 교체되므로 기존 사전 정보는 무용지물이 되며, 공격 비용이 기하급수적으로 증가하게 된다. 또한 매 세션마다 사용되는 해시 함수가 교체되기 때문에, 공격자가 이전 세션에서 가로챈 메시지와 이에 대한 MAC이 이후 세션에서는 유효하지 않게 되며, 이로써 재전송 공격에 대응할 수 있다. 이와 같이, FlexMAC 시스템은 MAC을 위협하는 다양한 공격으로부터 시스템을 보호할 수 있으며, 차세대 무결성 검증 방법론으로 자리매김할 것으로 기대된다.

Ⅱ. Background & Motivation

2.1 MAC (Message Authentication Code)

MAC의 개념: MAC은 메시지의 무결성을 보장하기 위해 사용되는 중요한 암호학적 도구이다[1]. 디지털 통신 환경에서는 수많은 데이터가 실시간으로 전송되는 과정에서 메시지가 변조되거나 손상되지 않았음을 확인하는 무결성 검증이 필수적이다. 만약 메시지의 무결성 검증 과정이 제대로 이루어지지 않는다면, 공격자에 의해 조작된 메시지가 전송될 수 있으며, 이는 시스템을 심각한 보안 위협에 노출 시킬 수 있다. 이를 방지하기 위해 MAC 시스템은 송신자가 전송하는 메시지와 함께 고유한 인증 코드인 MAC을 생성하고, 수신자가 이를 통해 메시지의 무결성을 검증할 수 있도록 한다. 이처럼 MAC은 데이터 통신의 신뢰성을 보장하는 데

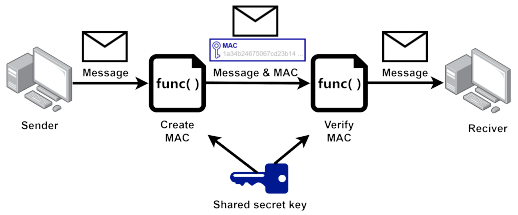

Fig. 1. MAC system operation process

핵심적인 역할을 수행하며, 다양한 보안 시스템에서 필수적인 요소로 자리 잡고 있다.

MAC의 동작 방식: 이처럼 무결성 검증에 주로 활용하는 MAC의 동작 과정은 Fig. 1.에서 볼 수 있듯이 비교적 간단한 절차로 이루어진다. 먼저, 송신자와 수신자는 비밀 키를 안전한 채널을 통해 공유하며, 이 비밀 키는 MAC을 생성하고 검증하는 데 사용된다. 키 공유가 완료되면, 송신자는 공유된 비밀 키와 전송할 메시지를 결합하여 해시 함수에 입력함으로써 MAC을 생성한다. 이렇게 생성된 MAC은 원본 메시지와 함께 수신자에게 전송된다. 수신자는 송신자로부터 전송된 메시지와 자신이 보유한 비밀 키를 이용하여 MAC을 다시 생성하고, 이를 송신자가 전송한 MAC과 비교한다. 이때, 두 MAC이 동일하다면, 메시지의 메시지의 무결성이 보장된 것이며, 두 MAC이 다르다면 메시지가 전송 중 변조되었거나 위조되었음을 의미한다. 이러한 과정을 통해 MAC 시스템은 메시지의 무결성을 보장하고, 송신자와 수신자 간의 안전한 통신을 가능하게 한다.

MAC의 활용 분야: MAC 시스템은 무결성 검증을 간단하면서도 효과적으로 수행할 수 있어 다양한 보안 응용 분야에서 널리 사용된다. 예를 들어, SWIFT (Society for Worldwide Interbank Financial Telecommunication)는 은행 간 거래 메시지의 무결성을 확인하기 위해 MAC 시스템을 활용한다[7]. 또한, IPSec, SSL/TLS와 같은 프로토콜에서도 통신 내용의 인증과 무결성을 보장하기 위해 MAC이 사용된다. VPN에서는 암호화와 MAC 시스템을 함께 사용하여 기밀성과 무결성을 동시에 달성한다[8-10]. 이처럼 MAC 시스템은 보안 응용 분야에서 중요한 역할을 담당하고 있다.

MAC의 취약점: 그러나 MAC 시스템에도 몇 가지 취약점이 존재한다. 그 중 대표적인 취약점으로는 재전송 공격이 있다. 이 공격은 송신자와 수신자 간의 통신을 감청할 수 있는 공격자가, 송신자가 전송한 MAC과 메시지를 기록해 둔 후, 이를 나중에 그대로 재전송하여 인증이나 무결성 검증을 우회하는 방식이다[4]. 공격자는 이전에 송신자가 보낸 MAC과 메시지를 그대로 수신자에게 다시 전송함으로써, 수신자가 해당 메시지를 진실된 것으로 오인하도록 만든다. 이는 무결성 검증을 우회할 수 있는 방법이기에 MAC 시스템이 보장하는 무결성과 신뢰성을 심각하게 저해할 수 있다.

2.2 Hash function

해시 함수의 개념: 해시 함수는 임의의 길이를 가진 입력 데이터를 고정된 길이의 다이제스트로 변환하는 암호학적 알고리즘이다[11]. 이 함수는 입력 데이터를 압축하여 고유한 비트열 형태인 다이제스트로 변환하며, 이를 통해 원본 데이터를 직접 비교하지 않고도 다이제스트 간의 비교만으로 데이터의 일치 여부를 확인할 수 있다[12]. 이러한 특성 덕분에 해시 함수는 메시지나 파일의 무결성 검증, 디지털 서명 등 데이터의 변조 여부를 확인해야 하는 다양한 보안 분야에서 핵심적인 역할을 수행한다.

특히, 해시 함수가 데이터의 변조를 확인하는 데 효과적인 이유는, 입력 데이터의 작은 변경에도 출력 다이제스트에 큰 변화를 일으키는 쇄도 효과(Avalanche effect)를 갖기 때문이다[12]. 이 효과로 인해 미세한 입력의 변화라도 완전히 다른 다이제스트를 생성하게 하여, 데이터의 변조 여부를 확실히 탐지할 수 있다. 이러한 이유로 해시 함수는 다양한 보안 시스템에서 무결성을 유지하는 데 필수적인 도구로 널리 사용되고 있다.

해시 함수의 취약점: 그러나 해시 함수의 중요성이 커짐에 따라, 이를 악용하고 무결성을 침해하고자 하는 공격 시도들이 점점 빈번해지고 있다. 대표적인 공격 방법으로는 서로 다른 입력이 동일한 다이제스트를 생성하도록 하는 해시 충돌 공격(Hash collision attack), 사전 해싱된 평문과 다이제스트의 매핑 테이블을 활용해 평문을 역산하는 레인보우 테이블 공격(Rainbow table attack), 그리고 가능한 모든 입력 조합을 시도하여 다이제스트와 일치하는 평문을 찾아내는 무차별 대입 공격(Brute force attack) 등이 있다. 이러한 공격들은 해시 함수를 기반으로 하는 보안 시스템에 심각한 위협을 가할 수 있다[3].

해시 함수의 충족 조건: 해시 함수가 이와 같은 공격에 견고하게 대응하기 위해 해시 함수는 세 가지 주요 보안 요건을 반드시 충족해야 한다. 첫 번째로는 역상 저항성(Preimage resistance)으로, 이는 주어진 다이제스트에서 원래의 입력값을 역추적하는 것이 수학적으로 불가능해야 한다는 조건이다. 두 번째로는 제 2 역상 저항성(Second preimage resistance)으로, 이는 주어진 입력값과 동일한 다이제스트를 생성하는 또 다른 입력값을 찾는 것은 매우 어려워야 한다는 조건이다. 마지막

으로는, 충돌 저항성(Collision resistance)이 요구되며, 이는 서로 다른 두 입력이 동일한 다이제스트를 생성할 가능성을 극도로 낮춰야 한다는 조건이다[13]. 이 세 가지 조건을 모두 충족하는 해시 함수는 다양한 공격에 대해 강력한 저항성을 지닌 것으로 간주된다.

해시 함수의 검증: 그러나 해시 함수가 이러한 요건들을 충족하는지 직접 검증하는 것은 매우 복잡하고, 시간이 많이 소요되는 과정이다. 그 예시로, MD5 해시 함수는 1991년에 상용화되었으나, 해당 해시 함수가 보안 요건을 충족하지 않아 안전하지 않다고 판단된 시기는 2005년에야 확인되었다[14]. 이러한 문제를 완화하기 위해, 해시 함수의 안전성을 신속하게 평가할 수 있는 수치적 방법들이 도입되었으며, 그 대표적인 지표로는 SAC (Strict Avalanche Criterion)와 BIC (Bit Independence Criterion)가 있다[13,15]. SAC는 입력 데이터에서 1비트가 변경되었을 때 다이제스트의 해밍 거리가 얼마나 큰지를 평가하는 지표로, 식 (1)로 정의된다. BIC는 입력과 출력 간의 독립성을 측정하는 지표로, 식 (2)와 같이 정의된다. 일반적으로 SAC의 값이 높고 BIC의 값이 낮을수록, 해시 함수는 충돌 저항성이 높고 보안적으로 안전하다고 평가된다[13,15].

$$\forall m, n | D(m, n) = 1, avg(D(H(m), H(n))) = \frac{1}{2}. \quad (1)$$

(D : Hammig distance, H : Hash function, s : Digest length)

$$\rho(P, Q) = \frac{cov(P, Q)}{SD(P)SD(Q)}. \quad (2)$$

(cov : correlation of P and Q, SD : Standar deviation)

2.3 Artificial Intelligence

인공 신경망의 개념: 인공 신경망은 복잡한 문제 해결과 패턴 인식을 위해 개발된 머신러닝 모델로, 뇌의 생물학적 뉴런 네트워크에서 영감을 받아 설계되었다[16]. 인공 신경망 모델은 각 노드에서 입력과 가중치의 선형 결합을 수행하고, 이를 활성화 함수에 적용하는 방식으로 정의된다. 이러한 계산 과정을 여러 계층에 걸쳐 반복함으로써, 인공 신경망은 복잡한 결정 경계(Decision boundary)를 형성할 수 있다.

인공 신경망의 학습 방식: 그러나 임의로 설정된 초기 가중치만으로는 목표하는 결정 경계를 형성하기 어렵다. 이때, 최적의 가중치를 찾기 위해 제공된 데이터를 기반으로 학습(Trainining)이 수행되며, 그 방법으로는 최소 제곱법과 경사 하강법이 있다. 이 중 최소 제곱법은 최적화 대상의 가중치가 증가할수록 연산량이 급격히 증가하는 문제가 있다[17]. 따라서 대부분의 인공 신경망 모델은 경사 하강법을 사용하여 가중치를 점진적으로 조정한다. 경사 하강법은 모델의 출력과 실제 정답 간의 차이를 계산하는 손실 함수에서, 가중치에 대한 그래디언트(Gradient)를 이용해 수행된다[18,19]. 이때, 그래디언트는 오차를 최대화하는 방향을 나타내므로, 오차를 최소화하기 위해서는 식 (3)과 같이 그래디언트의 반대 방향으로 가중치를 조정하게 된다. 이 과정을 반복함으로써 모델은 데이터의 특성을 잘 반영하는 결정 경계를 형성하며, 주어진 목적에 맞는 정확한 동작을 수행할 수 있게 된다.

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta J(\theta). \quad (3)$$

2.4 Motivation

디지털 통신의 발전과 함께 정보의 안전한 전송은 필수적인 과제가 되었으며, 메시지의 무결성을 보장하는 MAC 시스템은 중요한 역할을 담당하게 되었다. 그러나 기존 MAC 시스템은 해시 충돌 공격, 무차별 대입 공격과 같은 해시 함수의 취약점이 MAC 시스템에 전이될 수 있다는 문제와, 재전송 공격과 같은 MAC 자체의 취약점에 노출되는 한계가 있다.

본 연구에서는 이러한 기존 MAC 시스템의 취약점을 극복하고, 보다 안전한 방법으로 무결성 검증을 수행할 수 있는 FlexMAC 시스템을 제안한다. FlexMAC 시스템은 인공 신경망을 활용하여 매 세션마다 고유한 해시 함수를 생성하고 이를 통해 MAC을 생성 및 검증하는 방식으로 설계되었다. 이러한 접근은 해시 함수와 MAC 시스템의 취약점을 동시에 완화할 수 있다. 해시 충돌 공격이나 무차별 대입 공격을 통해 해시 함수의 안전성을 침해하기 위해서는, 일반적으로 공격자가 대량의 입력값을 해시 함수에 대입하고 그 결과를 확인해야 하며, 이 과정에서 막대한 시간과 컴퓨팅 자원이 소모된다[20]. 이러한 공격을 빠르게 수행하기 위해 공격자들은 해시 함수의 입력과 출력의 매핑을 축적한 대규모 데이터셋(e.g., 레인보우 테이블 등)을 사전 정보로 활용한다[6]. 그러나 FlexMAC 시스템에서는 매 세션마다 무결성 검증에 사용되는 해시 함수가 변경되므로, 기존에 축적된 사전 지식은 무용지물이 되다. 이로써 공격자가 해시 함수를 공격하기 위해서 소요되는 비용은 기하급수적으로 증가시키며, 해시 함수의 취약점으로부터 MAC 시스템을 안전하게 보호할 수 있다.

뿐만 아니라, MAC 자체의 취약점인 재전송 공격에 대한 방어력도 크게 강화된다. FlexMAC 시스템은 매 세션마다 사용되는 해시 함수가 변경되기 때문에, 공격

자가 이전 세션에서 획득한 MAC과 메시지는 이후 세션에서는 더 이상 유효하지 않게 된다. 따라서 공격자가 이를 재사용하려 해도 무효화되며, 이로 인해 FlexMAC 시스템은 다양한 위협으로부터 MAC 검증 절차를 안전하게 보호하고, 무결성 검증의 신뢰성을 한층 더 높일 수 있다.

Ⅲ. FlexMAC design

3.1 Overview

Fig. 2는 송신자(호스트 1)가 수신자(호스트 2)에게 메시지를 전송하는 과정에서 FlexMAC 시스템이 어떻게 작동하는지를 보여준다. FlexMAC 시스템은 두 가지 주요 단계로 구성된다: 준비 단계(Phase 1), 무결성 검증 단계(Phase 2)

Phase 1 - 준비 단계: 무결성 검증에 앞서 송신자(호스트 1)는 두 가지 사전 절차를 수행한다. 첫째, 인공신경망을 기반으로 현재 세션에서 사용할 새로운 해시 함수를 생성하고, 이를 수신자(호스트 2)와 교환한다. 이 해시 함수는 해당 세션에서만 유효하기에, 이후 세션에서는 새로운 해시 함수를 다시 생성한 후 교환해야 한다. 둘째, 비밀키를 생성하여 수신자(호스트 2)와 공유해야 한다. 이 비밀키는 송신자(호스트 1)에서 MAC을 생성하고, 수신자(호스트 2)에서 해당 MAC을 검증하는 데 사용된다.

Phase 2 - 무결성 검증 단계: 송신자(호스트 1)와 수신자(호스트 2)가 서로 동일한 해시 함수와 비밀 키를 교환한 후, 메시지 전송을 시작할 수 있다. 송신자(호스트 1)는 전송할 메시지와 비밀 키를 해시 함수의 입력으로 하여 MAC을 생성한다. 이후, 송신자(호스트 1)는 메시지의 원본 메시지와 생성된 MAC을 수신자(호스트 2)에게 전송한다. 이후 수신자(호스트 2)는 수신한 메시지와 MAC을 사용하여 메시지의 무결성을 검사한다. 이를 위해 수신자(호스트 2)는 수신한 메시지와 비밀 키를 해시 함수의 입력으로 사용하여 MAC을 생성한다. 이때, 수신자(호스트 2)에서 생성된 MAC이 송신자(호스트 1)로부터 수신한 MAC과 일치하면, 해당 메시지는 무결성이 보장된 것으로 간주할 수 있다. 반면, 두 MAC이 일치하지 않는 경우에는 외부 요인에 의해 메시지의 무결성이 손상되었음을 의미한다.

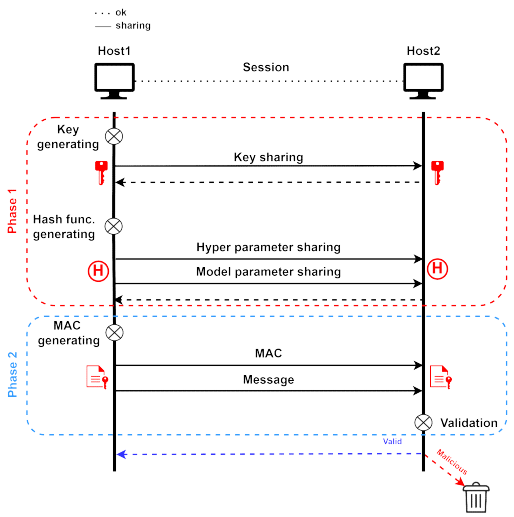

Fig. 2. FlexMAC overview

본 연구에서 제안하는 FlexMAC 시스템은 호스트들 간에 메시지 교환 과정에서 메시지의 무결성을 확인하기 위해 설계된 시스템으로, 매 세션마다 서로 다른 해시 함수를 활용하여 MAC을 생성하는 특징을 가진다. 이는 고정된 해시 함수를 사용하는 기존 MAC 시스템과 달리, 매 세션마다 해시 함수가 달라지기 때문에, 해시 함수의 취약점인 해시 충돌 공격, 무차별 대입 공격, 레인보우 테이블 공격 및 MAC 시스템에서의 취약점인 재전송 공격으로부터 시스템을 효과적으로 보호할 수 있다.

3.2 Phase 1: Preparation stage

준비 단계에서 송신자는 메시지의 무결성을 확인하기 위해 두 가지 사전 절차를 수행한다. 첫 번째로는 인공 신경망을 기반으로 고유한 해시 함수를 생성 및 공유 과정(3.2.1 절)이며, 두 번째로는 비밀 키를 생성 및 공유 과정(3.2.2 절)이다. 이후 송신자는 생성된 해시 함수와 비밀 키를 수신자와 공유하여 무결성 검증을 수행할 준비를 완료한다.

3.2.1 Hash function generation and sharing

하이퍼파라미터 설정: 생성할 해시 함수의 구조적 특징을 결정하기 위해, 먼저 파라미터를 설정한다. 이러한 하이퍼파라미터에는 다이제스트의 길이, 증폭기의 강도(추후 설명), 해시 함수를 구성하는 인공 신경망의 구조를 포함하며, 적용할 시스템의 사양과 목적에 따라 파라미터를 조정된다. 이를 통해 다양한 형태의 해시 함수를 생성할 수 있다.

학습 데이터 생성: 이 단계에서는 해시 함수를 구성하는 인공 신경망 블록을 학습시키기 위한 데이터를 생성한다. 전통적인 인공 신경망은 데이터 내의 패턴을 파악하는 데 사용되지만, FlexMAC 시스템의 인공 신경망 모델은 해시 함수를 구성 요소로 사용되기 때문에,

역상 저항성 및 충돌 저항성과 같은 해시 함수의 필수 요건을 충족해야 하며, 따라서 이 모델들은 특정 규칙이나 패턴을 학습하지 않도록 해야 한다. 이를 위해 모델 학습에 사용되는 입력 데이터와 정답 데이터는 0과 1 사이의 실숫값으로 균등 분포에서 생성된다. 이러한 방법을 통해 모델이 특정 패턴의 학습을 피하고, 해시 함수의 불확실성과 보안성을 강화할 수 있다. 학습 데이터를 생성하고, 이를 이용하여 학습을 수행하는 것과 관련된 내용은 Appendix B, C에서 다룬다.

해시 함수 생성: 해시 함수의 입력 메시지 길이는 다양할 수 있지만, 출력 다이제스트는 고정된 길이를 가져야 한다. 이를 위해 FlexMAC 시스템에서는 입력 메시지 처리 모듈과 압축 모듈을 사용한다. 입력 메시지 처리 모듈은 해시 함수의 입력을 특정 *크기*의 비트스트림 유닛으로 나눈다. 유닛의 *크기*는 처리 과정에서의 불확실성을 증가시키기 위해 다이제스트의 길이의 두 배로 설정된다. 만약 입력 메시지를 유닛 단위로 나눈 후 남은 부분이 유닛의 *크기*보다 작다면, 부족한 부분에 대해서는 0으로 패딩된다. 이 과정에서 발생할 수 있는 해시 충돌 문제(e.g., 유닛의 *크기*가 4라고 할때, 11110000과 11110은 1111, 0000이라는 동일한 두 유닛을 생성한다.)를 방지하기 위해, 입력 메시지 길이를 비트로 표현한 특수 유닛이 마지막에 추가된다.

압축 모듈은 입력 메시지 처리 모듈에 의해 나누어진 유닛들을 순차적으로 합병하여 최종 다이제스트를 생성한다. 이 모듈은 인공 신경망으로 구성된 합병 블록(Merge block)과 해시 블록(Hash block)으로 구성되며, 이들은 이전 단계에서 생성된 학습 데이터를 통해 학습됨으로써 모델 파라미터가 결정된다. 합병 블록은 입력 메시지 처리 모듈에 의해 생성된 유닛들을 하나의 벡터로 합병하는 과정에서 중간 생성 결과에 변형을 가하여, 불확실성과 쇄도 효과를 증대시키는 역할을 하는 인공 신경망 블록이다. 이는 입력과 출력의 차원이 유닛의 *크기*와 동일한 완전 연결 층의 구조로 구성된다. 해시 블록은 유닛들의 합병을 통해 생성된 최종 벡터를 다이제스트로 변환하는 역할을 하는 인공 신경망 블록이며, 입력 차원이 유닛의 *크기*와 동일하고 출력의 차원이 다이제스트의 *크기*로 설정된 완전 연결 층으로 구성되어 있다. 이와 같은 인공 신경망 블록을 통과한 이후 불확실성을 더욱 증대시켜 강한 쇄도 현상을 유발하기 위해, 각 블록의 출력 결과에 대해 큰 값(e.g., 10^n, 여기서 n은 증폭기의 강도로 정의됨)을 곱한 후, modulo 2 연산을 수행함으로써 각 블록의 최종 출력 결과물을 정의한다.

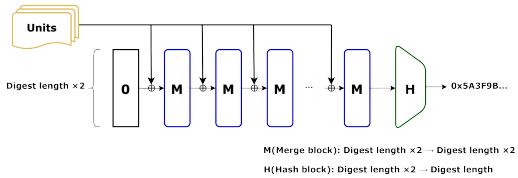

Fig. 3. Compression module

이와 같은 합병 블록과 해시 블록을 이용하여 다이제스트를 생성하는 과정은 Fig. 3 에 나타나 있다. 먼저, 다이제스트 길이의 두 배에 해당하는 0벡터를 기준 벡터로 초기화한 후, 각 유닛을 순차적으로 기준 벡터와 XOR 연산하여 병합한다. 이 과정에서 합병 블록은 XOR 연산 후 생성된 벡터에 변형을 가해 불확실성과 무작위성을 증가시킨다. 이 과정은 모든 유닛이 하나의 벡터로 통합될 때까지 반복되며, 이를 통해 해시 함수는 강력한 역상 저항성과 충돌 저항성을 갖추게 된다. 최종적으로 유닛이 합병되어 생성된 벡터는 다이제스트 길이의 두 배 *크기*를 가지며, 이 벡터가 해시 블록을 통과하면서 최종 다이제스트가 생성된다. 합병 블록과 해시 블록의 구조와 관련된 자세한 내용은 Appendix D, E에서 다룬다.

교환: 생성된 해시 함수는 설정된 하이퍼파라미터와 인공 신경망 블록의 모델 파라미터에 의해 그 고유성이 결정된다. 이는 하이퍼파라미터와 인공 신경망 블록의 모델 파라미터만 있으면, 동일한 해시 함수를 재구성할 수 있음을 의미한다. 따라서, 송신자와 수신자 양측만이 알고 있는 고유한 해시 함수를 정의하기 위해, 하이퍼파라미터와 인공 신경망 블록의 모델 파라미터를 안전한 채널을 통해 전송한다.

3.2.2 Secret key generation and sharing

FlexMAC 시스템에서 MAC을 생성하고 검증하기 위해서는 통신하는 호스트들 간에 공유하는 비밀 키가 정의 되어야한다. 따라서, 송신자는 64바이트 길이의 비밀 키를 랜덤으로 생성한 후, 이를 안전한 통신 채널을 통해 수신자에게 전달한다. 이 비밀 키는 이후 메시지와 결합하여 MAC을 생성하고 검증하는 데 사용된다.

3.3 Phase 2: Integrity verification stage

무결성 검증 단계에서는 Fig. 4와 같이 전송된 메시지가 손상된거나 위조되지 않았는지를 확인하는 절차이다. 이 단계는 두 가지 과정으로 나뉜다. 첫 번째 과정에서는 송신자가 공유한 비밀키와 해시 함수를 이용해 MAC을 생성한다. 두 번째 과정에서는 수신자가 수신

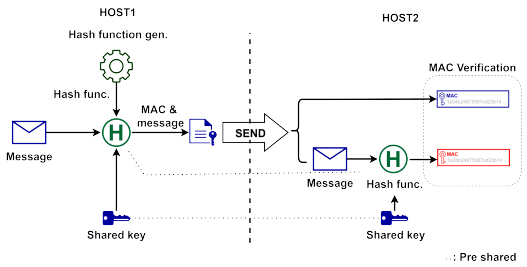

Fig. 4. Integrity verification stage

한 메시지와 동일한 비밀 키 및 해시 함수를 사용해 새로운 MAC을 생성하고, 이를 송신자가 보낸 MAC과 비교하여 동일성을 확인한다. 이 두 과정을 통해 메시지의 무결성을 확인하며, 데이터가 안전하게 전송되었음을 검증할 수 있다.

MAC 생성: MAC은 메시지의 무결성을 확인하기 위한 다이제스트의 일종으로, 공유된 비밀 키와 해시 함수를 이용하여 식 (4)와 같은 방식대로 생성된다. 먼저, 비밀 키에 대한 조정이 수행된다. 이때 비밀 키의 각 바이트에 대해 '0x5C'와 '0x36'을 XOR 연산하여 외부 패딩(outter padding)과 내부 패딩(inner padding) 값을 생성한다. 이후, 내부 패딩값과 메시지를 결합하여 해시 함수의 입력함으로써 다이제스트를 생성하며, 이를 "내부 해시값"이라고 한다. 그런 다음, 외부 패딩과 내부 해시값을 결합하여 다시 해시 함수에 입력하면, 최종적으로 생성된 값이 MAC이 된다. MAC을 생성하는 규칙과 관련해서는 기존 MAC 시스템의 동작 구조를 차용하였다[21]. 송신자는 생성된 MAC과 원문 메시지를 수신자에게 전송하며, 수신자는 이 MAC을 사용하여 메시지의 무결성을 검증한다.

$$\mathrm{MAC}_{k}(M) = \mathrm{Hash}[k \oplus 0x5C \| [\mathrm{Hash}[k \oplus 0x36] \| M]]. \quad (k: \text{secret key}, M: \text{message}) \tag{4}$$

MAC 동일성 확인: 수신자는 송신자로부터 받은 MAC과 원문 메시지를 이용하여 메시지의 무결성을 검증한다. 이를 위해 수신자는 공유된 비밀 키와 해시 함수를 사용하여 송신자가 MAC을 생성한 것과 동일한 방식으로 메시지에 대해 새로운 MAC을 생성한다. 이때, 생성된 MAC이 수신된 MAC과 일치하면, 메시지가 무결성을 유지한 것으로 간주할 수 있다. 반면, 두 MAC이 일치하지 않는 경우, 메시지가 손상되었음을 의미한다.

Ⅳ. Evaluation

본 연구에서 제안한 시스템의 개발 및 평가는 Intel Xeon Silver 4210R CPU, 64GB RAM, Geforce RTX 3090 (24GB VRAM) 2개, Ubuntu 22.04 운영체제를 갖춘 머신에서 수행되었다. FlexMAC 전체 시스템 구현은 PyTorch 라이브러리를 기반으로 작성되었다.

4.1 Attack resistance

공격 저항성 평가: 재전송 공격에 대한 저항성을 평가하기 위해 일반적인 MAC 시스템과 FlexMAC 시스템을 대상으로 각각 공격을 수행하였다. 공격자는 호스트 간 통신 과정에서 메시지와 MAC을 가로챈 후, 이를 다음 세션에서 재전송하여 공격이 성공하는지 확인하였다. Fig. 5는 이 공격의 결과를 나타낸다. 일반적인 MAC 시스템에 공격을 수행한 경우, 희생자가 메시지로부터 생성한 MAC과 공격자로부터 수신된 MAC이 동일하기 때문에 메시지에 대한 무결성 검증을 우회할 수 있다. 이는 무결성 검증에 사용되는 해시 함수가 매 세션마다 동일하기 때문에 발생하는 문제이다. 반면, FlexMAC 시스템에서는 공격자가 이전 세션에서의 MAC 값과 메시지를 재전송하더라도, 희생자가 생성한 MAC과 전송받은 MAC이 일치하지 않아 무결성 검증을 우회할 수 없다. 이는 매 세션마다 무결성 검증에 사용되는 해시 함수가 변경되기 때문이며, 이러한 결과는 FlexMAC 시스템이 재전송 공격을 효과적으로 완화할 수 있음을 입증한다.

Fig. 5. Result of replay attacks

4.2 Hash function stability

SAC & BIC 평가: 해시 함수의 안전성을 검증하기 위해, 전통적으로는 복잡한 수학적 분석이나 대량의 입력값을 이용한 전사적 검증 방식을 사용해 왔다. 그러나 이러한 방법은 상당한 시간과 자원을 요구하는 단점이 있어 매 통신마다 새로운 해시 함수를 도입하여 안정성을 높이고자 하는 FlexMAC에서 적용하기는 어렵다. 따라서, 이러한 문제를 극복하기 위해 FlexMAC에서는

해시 함수의 안전성을 빠르고 효과적으로 평가할 수 있는 시스템인 SAC 및 BIC 수치를 평가하는 방법을 활용한다. 이러한 지표들은 입력 데이터의 미세한 변화가 다이제스트에 미치는 영향을 측정하여, 해시 함수의 쇄도 효과를 평가하는 역할로, 측정된 쇄도 효과가 클수록 해시 함수의 충돌 저항성이 높으며 그 해시 함수가 안정적으로 사용 가능하다고 말할 수 있다. 이러한 방식은 해시 함수 (그리고 여타 암호 알고리즘)등의 평가와 검증에 에 널리 사용되는 방식이며 그 효용성이 충분히 입증된 방법이다[13].

FlexMAC 시스템에서는 생성된 해시 함수의 안전성을 평가하기 위해, 256비트 길이의 다이제스트를 생성하는 해시 함수 100개를 생성한 후, 이들의 SAC 및 BIC 값을 기존의 해시 함수(SHA2-256, SHA3-256)와 비교 분석 하였다. 이 테스트에 사용한 데이터는 1비트에서 8,000비트까지의 길이를 가진 랜덤하게 생성된 100,000개의 비트열로 구성되었다. Fig. 6에 나타난 결과에 따르면, 생성된 해시 함수가 SHA 계열의 해시 함수와 유사한 수준의 안전성을 가지고 있음을 확인할 수 있었다. 구체적으로, 생성된 해시 함수 중 약 74%는 SAC 평가에서, 약 98%는 BIC 평가에서 SHA 계열의 해시 함수보다 향상된 안전성을 보였다. 이러한 결과는 인공 신경망을 기반으로 생성된 동일한 256비트의 길이의 해시 함수가 기존에 널리 사용되는 해시 함수만큼 신뢰성 있는 해싱 성능을 지님을 보여준다. 나아가, 본 시스템을 통해 생성되는 해시 함수의 길이는 사용자 설정에 따라 가변적일 수 있으므로 필요에 따라 더욱 더 긴 비트의 다이제스트를 생성하는 해시함수를 설정함으로서 더욱더 향상된 안정성을 제공할 수 있을 것이다.

해시 함수의 고유성: FlexMAC 시스템에서 생성된 해시 함수의 고유성을 입증하기 위해, 동일한 매개 변수(e.g., 학습 횟수, 학습 데이터 셋 등)를 사용하여 256비트 길이의 다이제스트를 가진 해시 함수 100개를 생성하였다. 이후, 동일한 입력값을 각 해시 함수에 적용하여 생성된 100개의 다이제스트 간의 유사성을 Python difflib 모듈을 이용해 비교하였다[22]. Fig. 7에 나타난 결과에 따르면, 평균 유사성은 0.2로 나타났으며, 이는 동일한 학습 데이터 셋에서 파생된 해시 함수들이라 할

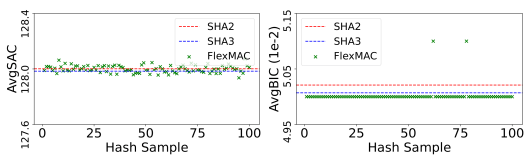

Fig. 6. SAC & BIC values (Compared with SHA-2/3)

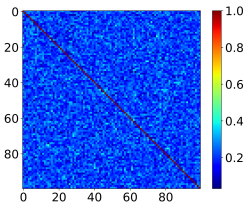

Fig. 7. Digest similarity by 100 hash functions

지라도 생성된 다이제스트가 상당히 상이하다는 것을 의미한다. 이 결과는 해시 함수 생성 과정에서 불확실성이 작용하여, 동일한 조건이라도 이전에 생성한 해시 함수를 재생성하는 것은 매우 어렵다는 점을 시사한다.

4.3 Performance evaluation

처리 시간: FlexMAC 시스템의 비밀 키 및 해시 함수 공유부터 MAC 검증까지 걸리는 시간을 현존하는 MAC 시스템과 비교하여 평가하였다. Fig. 8에서 보여지는 바와 같이, 본 연구에서 제안하는 FlexMAC 시스템은 평균적으로 약 10.72초가 소요된 반면, 기존 MAC 시스템은 평균 0.0009초 정도가 소요되었다. 이러한 차이는 해시 함수를 생성하는 과정에서 인공 신경망의 학습이 필요하기 때문이다. 비록 처리 시간이 상당히 길지만, 보안 측면에서의 높은 안전성을 고려할 때, FlexMAC 시스템은 충분히 실용적인 대안이 될 수 있을 것으로 판단된다.

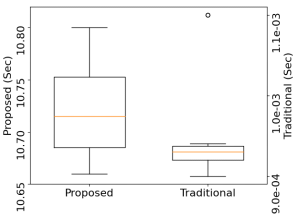

Fig. 8. Comparison of the speed of performance of the FlexMAC system with that of the traditional system

V. Conclusion

본 연구에서는 디지털 통신 환경에서 메시지의 무결성을 보장하기 위한 새로운 보안 메커니즘으로 FlexMAC 시스템을 제안하였다. FlexMAC 시스템은 인공 신경망을 기반으로 매 세션마다 고유한 해시 함수를 동적으로 생성하고 이를 활용하여 메시지의 무결성을 확인하는 방식으로 설계되었다. 이를 통해 기존 MAC 시스템이 직면했던 보안 취약점을 효과적으로 완

화할 수 있었다. 구체적으로 해시 함수의 취약점인 해시 충돌 및 무차별 대입 공격에 대해, FlexMAC 시스템은 공격자의 비용을 기하급수적으로 증가시켜 공격 난이도를 높이는 방향으로 문제를 해결하였다. 또한, MAC 시스템의 취약점인 재전송 공격에 대해서는 매 세션마다 다른 해시 함수를 사용함으로써 이전 세션의 MAC이 무효화 되도록 하여, MAC 시스템의 보안 수준을 한층 강화하였다. 이러한 접근은 고도로 민감한 데이터가 오가는 환경에서 메시지의 무결성을 안전하게 점검할 수 있는 강력한 보안 메커니즘을 제공한다.

그러나, FlexMAC 시스템에도 한계가 존재한다. FlexMAC 시스템의 해시 함수 생성하는 과정 자체는 수~수십초 이내로 빠르게 이루어지지만, 그에 대한 신뢰성 검증은 시간에 비례하여 증가하므로 어느 정도의 시간이 투입되어야 충분히 안정적인지에 대한 최적화 이슈가 남아있다. 이러한 문제는 필요한 해시 함수를 미리 생성하여 Pool을 구성하는 형태로 대응할 수 있을 것으로 기대하며, 향후 이를 해결하기 위해 알고리즘 개선 및 하드웨어 가속기 등을 도입함으로써 시간을 단축하고 실용성을 높이는 연구를 진행할 예정이다.

References

[1] J. M. Turner, "The keyed-hash message authentication code (HMAC)," *Federal Inf. Process. Standards Publication,* vol. 198.1 pp. 1-13, 2008. (https://doi.org/10.6028/nist.fips.198-1)

[2] S. M. S. Hussain, S. M. Farooq, and T. S. Ustun, "Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security," *IEEE Access,* vol. 7, pp. 80980-80984, 2019. (https://doi.org/10.1109/ACCESS.2019.2923728)

[3] R. Sobti and G. Ganesan, "Cryptographic hash functions: A review," *IJCSI*, vol. 9, no. 2, pp. 461-479, 2012.

[4] P. A. Grassi, et al., "Draft nist special publication 800-63b digital identity guidelines," *NIST,* vol. 27, 2016.

[5] H. Yoshida and A. Biryukov, "Analysis of a SHA-256 variant," *Int. Wkshp. Sel. Areas in Cryptography*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. (https://doi.org/10.1007/11693383_17)

[6] H. Kumar, et al., "Rainbow table to crack password using MD5 hashing algorithm," *2013 IEEE Conf. Inf. & Commun. Technol.,* IEEE, 2013. (https://doi.org/10.1109/cict.2013.6558135)

[7] S. V. Scott and M. Zachariadis, "The society for worldwide interbank financial telecommunication (SWIFT): Cooperative governance for network innovation, standards, and community," *Taylor & Francis*, 2014. (https://doi.org/10.4324/9781315849324)

[8] J. Deepakumara, H. M. Heys, and R. Venkatesan, "Performance comparison of message authentication code (MAC) algorithms for Internet protocol security (IPSEC)," in *Proc. Newfoundland Electr. and Comput. Eng. Conf.,* 2003.

[9] H. K. Lee, T. Malkin, and E. Nahum, "Cryptographic strength of SSL/TLS servers: Current and recent practices," in *Proc. 7th ACM SIGCOMM Conf. Internet Measurement*, 2007. (https://doi.org/10.1145/1298306.1298318)

[10] H. Tiwari and K. Asawa, "Cryptographic hash function: An elevated view," *Eur. J. Scientific Res.,* vol. 43, no. 4, pp. 452-465, 2010.

[11] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "*Cryptographic hash functions: A survey*," Technical Report 95-09, vol. 4, Department of Computer Science, University of Wollongong, 1995.

[12] G. K. Sodhi and G. S. Gaba, "An efficient hash algorithm to preserve data integrity," *J. Eng. Sci. and Technol.*, vol. 13, no. 3, pp. 778-789, 2018.

[13] D. Upadhyay, et al., "Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications," *IEEE Access,* vol. 10, pp. 112472-112486, 2022. (https://doi.org/10.1109/ACCESS.2022.3215778)

[14] X. Wang and H. Yu, "How to break MD5 and other hash functions," *Annual Int. Conf.*

Theory and Appl. of Cryptographic Techniques, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. (https://doi.org/10.1007/11426639_2)

[15] Y. Wang, et al., "A novel method to design S-box based on chaotic map and genetic algorithm," *Physics Lett. A*, vol. 376, no. 6-7, pp. 827-833, 2012. (https://doi.org/10.1016/j.physleta.2012.01.009)

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015. (https://doi.org/10.1038/nature14539)

[17] J. A. Belloch, et al., "Solving weighted least squares (WLS) problems on ARM-based architectures," *J. Supercomputing*, vol. 73, pp. 530-542, 2017. (https://doi.org/10.1007/s11227-016-1910-9)

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[19] S.-E. Moon, et al., "Trends in machine learning and deep learning technologies," *J. KICS*, vol. 33, no. 10, pp. 49-56, 2016.

[20] J. Katz and H. Shacham, et al., "Advances in Cryptology – CRYPTO 2017," *37th Annual Int. Cryptology Conf.*, Santa Barbara, CA, USA, Aug. 2017, Proceedings, Part III*, vol. 10403, Springer, 2017. (https://doi.org/10.1007/978-3-319-63697-9)

[21] Code, Authentication, "Keyed-hash message authentication code (HMAC)," 2002.

[22] "difflib — Helpers for Computing Deltas," *Python Documentation*, Python Software Foundation, https://docs.python.org/ko/3/library/difflib.html, Accessed 21 Aug. 2024.

문 성 수 (Seongsu Moon)

2021년 3월~현재 : 전남대학교 인공지능학부 재학

<관심분야> 인공지능, 정보보호

박 태 준 (Taejune Park)

2021년 2월 : 카이스트 공학박사

2021년 3월~8월 : 카이스트 연수연구원

2021년 9월~현재 : 전남대학교 인공지능학부 조교수

<관심분야> 네트워크 보안, 데이터평면

A. Appendix

FlexMAC 시스템에서 생성하는 해시 함수의 구조와 관련된. 본문에서 다루지 못한 세부 사항을 설명한다.

B. Generating training dataset details

해시 함수는 합병 블록(Merge block)과 해시 블록(Hash block)으로 구성되며, 이들은 인공 신경망을 기반으로 한다. 각 블록의 모델 파라미터는 해시 함수의 고유성을 결정하는 핵심 요소이다. 본 연구에서는 랜덤 데이터를 생성해 두 블록을 학습시켜 모델 파라미터를 결정하였다.

합병 블록은 다이제스트 길이의 두 배 크기의 벡터를 입력으로 받아 같은 크기의 벡터를 출력하며, 학습 데이터는 0과 1 사이의 실수 값으로 균등 분포에서 생성된다. 입력과 출력의 길이는 모두 다이제스트의 두 배로 설정된다. 해시 블록은 다이제스트 길이의 두 배 크기의 벡터를 입력으로 받아, 다이제스트 길이의 벡터를 출력한다. 이때 입력은 다이제스트의 두 배, 출력은 다이제스트 길이로 설정되며, 동일한 방식으로 0과 1 사이의 실수 값에서 생성된 데이터를 사용한다.

C. Training details

해시 함수를 구성하는 인공 신경망을 학습하기 위해서는 손실 함수와 최적화 방법이 정의되어야 한다. 본 연구에서는 손실 함수로는 평균 제곱 오차(MSE, Mean Squared Error)를 사용하였으며, 최적화 방법으로는 Adam(learning rate=0.001)을 사용하였다.

D. Merge block details

합병 블록은 입력 메시지 처리 모듈에 의해 생성된 유닛들을 하나의 벡터로 합병하는 과정에서 중간 결과에 변형을 가하여, 불확실성과 쇄도 효과를 증대시키는 역할을 하는 인공 신경망 블록이다. 합병 블록은 Fig. 9와 같이 이전 층의 모든 뉴런이 다음 층의 모든 뉴런과 연결되는 구조인 완전 연결 층(Fully Connected Layer)과, 이 완전 연결 층의 출력을 큰 폭으로 변형시키는 증폭기(Amplifier)로 구성되어 있다.

완전 연결 층은 학습 과정에서 초기화된 모델 파라미터와의 연산을 통해 합병 과정에서 입력 데이터를 변형시켜, 해시 함수의 쇄도 효과를 증대시킨다. 이때 완전 연결 층의 구조는 입력과 출력의 크기가 다이제스트 길이의 두 배와 같기만 하면 되며, 내부 은닉층의 구조는 자유롭게 조절할 수 있다.

증폭기는 해시 함수의 역상 저항성과 쇄도 효과를 증대시키기 위한 모듈이다. 이는 완전 연결 층을 통과한

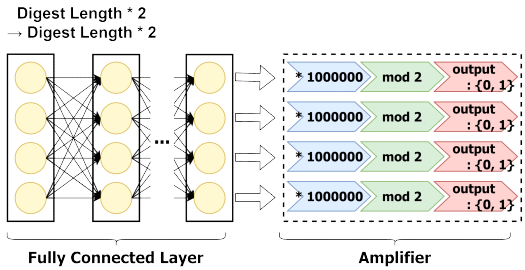

Fig. 9. Merge block structure

벡터에 10^n (이때, n은 증폭기의 강도로써 정의된다.) 형태의 큰 값을 곱 연산한 후, 해당 결과의 정수 부분에 대해 modulo 2 연산을 취하여 각 노드의 출력을 {0, 1}의 범위로 제한한다.

E. Hash block details

해시 블록은 유닛들의 합병을 통해 생성된 최종 벡터를 다이제스트로 변환하는 역할을 하는 인공 신경망 블록이다. Fig. 10과 같이 완전 연결 층과 증폭기로 구성되어 있으며, 완전 연결 층은 합병 블록에 의해 병합된 유닛 벡터를 입력으로 받아, 이를 다이제스트 크기에 맞게 변환한다. 이후, 완전 연결 층을 통과한 벡터에 대해 합병 블록에서와 동일한 증폭기를 적용하여 최종 다이제스트를 생성한다.

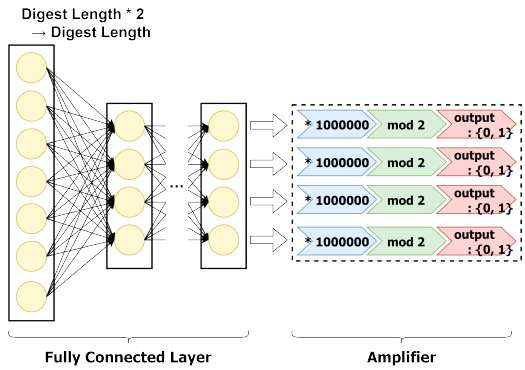

Fig. 10. Hash block structure