교 육 학 석 사 학 위 논 문

지문 데이터베이스의 무결성 검증에 관한 연구

2002年 8月

부 경 대 학 교 교 육 대 학 원

전 산 교 육 전 공

현 영 숙

교 육 학 석 사 학 위 논 문

지문 데이터베이스의 무결성 검증에 관한 연구

지도교수 김 창 수

이 논문을 교육학석사 학위논문으로 제출함

2002년 8월

부경대학교 교육대학원

전 산 교 육 전 공

현 영 숙

<차 례>

<표 차례>

<그림 차례>

A Study on the Integrity Verification for Fingerprint Database

Young-Sook Hyun

Graduate School of Education Pukyong National University

Abstract

Recently, Biometric Identification System to use the physical characteristics as technology of personal identification is actively proceeding. However, in the point of view that the stored personal information in the Biometric Identification System is people's own physical information, there is a matter of grave concern, when those information are opened or some disturbance is happened by the intentional change of data. To solve those problems fundamentally, the Database which has the information of the personal identification must be protected.

In this paper, we design and implement verification system for data integrity that is able to check illegal intrusion in fingerprint database system. This these provides functions which are conveniently capable to store and manage the results as a log file for verification of fingerprint system, and easily inform to manager for intentional update of fingerprint characteristics by providing the convenient user's interface.

I. 서론

생체 인식 기술은 개인의 독특한 특성을 측정해 그 결과를 사전에 측정한 특징과 비교함으로써 개인의 신원을 인증하는 방법이다. 오늘날 네트워크의 발달과 더불어 보안 및 개인 사생활 보호에 대한 관심이 높아지면서 이들의 연구가 활발히 진행되고 있다. 하지만 이들 생체 인식 시스템 내의 생체 정보가 개인의 고유한 신체 정보라는 점에서 정보가 유출되거나 의도적인 데이터의 변경으로 침해 발생 시 막대한 피해 및 경제적인 손실이 발생 할 수 있다. 그러므로 데이터의 일관성을 저해하는 우발적인 사고 등으로부터 데이터 혹은 데이터베이스를 보호해야 할 필요성이 있다.

본 논문에서는 생체 인식 시스템 중에서도 정확도 및 처리 속도 등의 성능, 이용의 편리성과 친밀성, 경제성 등이 높이 평가되어 최근 많은 연구와 활용이 구체화되고 있는 지문 인식 시스템을 사용하여 데이터가 수정 및 변조되었는지 확인하는 무결성 검증 시스템을 설계하고 구현 하고자 한다. 관리자가 쉽게 지문DB내 데이터의 변조를 확인할 수 있도록 편리한 사용자 인터페이스를 제공하고 결과를 로그 파일로 남기어 관리자가 편리하게 관리할 수 있도록 구현하였다. 본 논문의 구성은 다음과 같다. “2장 관련 연구” 에서는 지문 인식 시스템을 소개하고 지문 데이터의 표준화 동향을 고찰한다. 또한 데이터베이스의 보안 방법과 무결성 검증에 사용되는 해쉬 함수를 소개한다.

“3장 지문DB의 무결성 보안 설계”는 연구의 필요성과 2장에서 제시한 방법을 활용하여 설계하고 4장에서는 시스템 구현을 하였다. 5장에서는 시스템 구현 결과에 대한 분석 및 평가를 제시하였고 마지막으로 6장에서는 결론 및 향후 과제에 대해 설명한다.

Ⅱ. 관련 연구

2.1 지문 인식 시스템

사람의 지문은 평생 불변, 만인 부동의 특징을 가지고 있어서 사람 식별의 가장 확실한 수단으로 생각되어져 왔다. 범죄 현장에서의 용의자의 지문은 매우 중요한 단서로 활용되어지며 주민등록상에서도 우리의 지문은 중요한 개인의 식별을 위해 사용되어져 왔다. 이에 따른 자동화 연구가 활발히 진행되어 왔으며 지문을 이용한 개인 인증에 관한 연구는 많은 성과를 보이고 있고 현재에도 다양한 분야에 적용되고 있다[1].

2.1.1 지문의 형태적인 특성

지문 형상은 복합적인 곡선들의 모음이며 다른 손가락에 의한 두 지문은 같은 융선(Ridge)을 가지고 있지 않으며 그 형태는 평생 동안 변하지 않는다.

지문의 각 특성들은 [그림 2-1]에 나타나 있다.

[그림 2-1] 지문의 형태

지문의 여러 가지 형태의 선의 흐름을 융선(Ridge)이라고 하며 지문의 어두운 부분을 말한다. 융선과 융선 사이 즉, 밝은 부분은 골(valley)이라고 한다. Galton은 지문 융선의 일정한 흐름을 깨는 비연속점(local discontinuities)들이 지문 형상에 많이 존재한다고 밝혔으며 이를 특징점(minutiae)이라고 부른다. Galton feature란 지문의 융선 위에 나타난 특징 점을 뜻하는데 8가지를 분류했으나 현재 지문 인식 시스템에서는 모든 Galton feature들이 대부분 사용되지 않고 대표적인 2가지 특징 점인 끝점(ending point) 과 분기점(bifurcation) 만으로 나머지 Galton Feature를 표현할 수 있다[2]. 끝점(Ending point)은 융선이 부드럽게 흐르다가 끊어지는 점이며 분기점(Bifurcation point)은 융선이 부드럽게 흐르다가 두 갈래로 갈라지는 점을 말한다.

지문에는 특징 점 외에 특이점을 보유하고 있을 수 있는데 중심점(core)과 삼각주(delta)를 들 수 있다. 중심점은 지문의 융선 중 방향이 가장 급격하게 변하는 곳이며 삼각주는 융선의 흐름이 세 방향으로 나누어지는 곳을 말한다. 이 특이점은 지문의 기준 점을 잡거나 분류할 때 사용되는 요소들이다. 이에 반해 Henry는 지문의 전체적인 구조에 대해 연구했으며 기본적인 5가지 지문을 분류하고 그 분류는 arch, tented arch, left loop, right loop, whorl이다.

현재 지문 인식 시스템은 지문의 데이터베이스 량이 많을 때는 검색 시간과 계산의 복잡성을 줄여 효율적으로 하기 위해 Henry system을 통하여 지문을 분류하며 Galton Features들을 사용해 matching을 수행할 수 있다.

2.1.2 지문 인식 방법

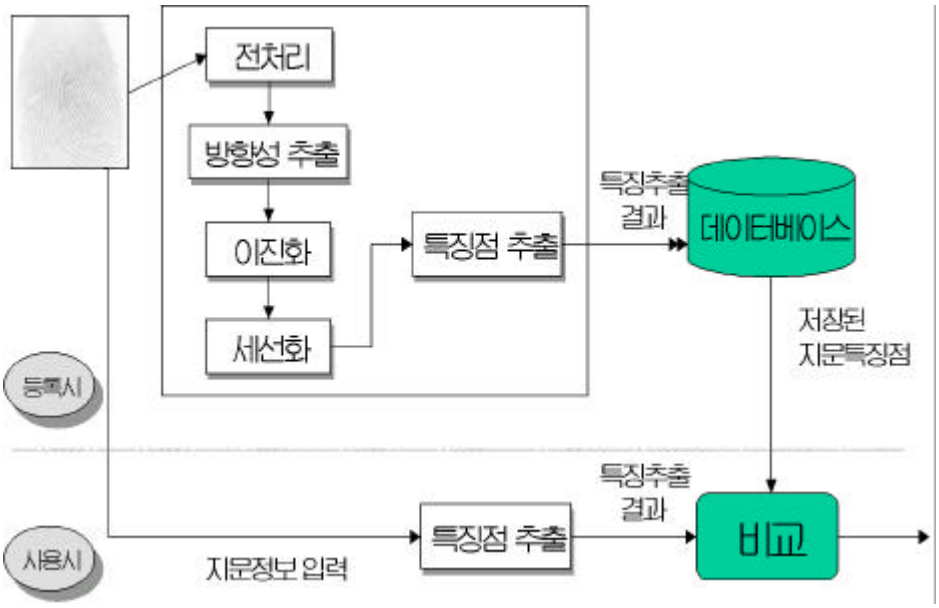
지문을 인식하는 방법은 크게 Galton Features를 이용한 특징점 기반 인식 시스템과 Galton features를 이용하지 않고 지문의 다른 특징들을 이용하는 시스템으로 나눌 수 있다[2].

특징점 기반 인식 시스템은 지문의 특징점과 주변의 특징점 분포를 이용

한 비교 방법을 사용하여 이루어지는데 먼저 특징점을 찾은 후 특징점의 속성과 그 상대적 위치, 방향의 계산으로 개인 식별을 할 수 있게 된다. 특징점 기반 인식 시스템은 요철의 전체적인 형태를 고려하지 않고 또 지문의 상태가 양호하지 않은 경우 특징점을 정확하게 추출하기 어려운 단점이 있다. 지문 융선의 특징점을 이용하는 방식이 아닌 다른 특징을 이용하는 시스템은 지문 상에 존재하는 땀샘의 분포를 이용하는 지문 인식 방법이 있기는 하지만 널리 사용되는 방식은 아니다.

2.1.3 지문 인식 시스템 처리 방법

지문 인식은 크게 두 단계의 프로세싱을 거친다. 입력된 지문 영상으로부터 특이점을 추출하는 추출 단계와 추출된 특이점을 비교하는 매칭 단계로 구분된다. 특이점의 추출은 다시 크게 4단계로 나누어지며 [그림 2-2]와 같은 방법으로 처리된다.



[그림 2-2] 특징점 추출 과정

[그림 2-2]는 원시 입력 지문 화상 감도를 향상시키는 전처리 과정과 각 융선의 방향 성분을 찾아내는 작업, 융선과 골을 구분하여 이를 이진화 하는 단계 및 각 융선의 굵기를 판단하여 이를 1 포인트의 선으로 세선화 하는 과정이다. 전처리 과정에서는 명암의 구분을 높이고 앞에서 언급한 잡음들을 제거하는 과정이다. 전처리 과정을 거친 영상은 다시 방향 성분을 추출하는 단계를 거친다. 추출된 방향 성분은 영상으로 나타낸 것이다 추출의 다음 단계는 원 화상을 융선과 배경으로 구분하는 이진화 단계이다. 이진화를 거친 영상은 지문 영상의 굵기를 판단하여, 이를 1 포인트의 선으로 세선화 하는 과정을 거친다. 세선화된 영상은 다시 한번 잡음 제거 과정을 거치며 이후 지문에서의 방향과 좌표 등으로 그 특이점들이 추출된다. 추출되는 특이점들은 앞에서 언급한 바와 같다. 추출된 특이점들은 차후의 인증을 위하여 데이터베이스 등으로 저장되며, 특이점의 크기는 각 알고리즘마다 다르다. 일대다 시스템일 경우 지문 분류 작업도 이루어지나, 일대일 및 일대 소수의 시스템에서는 지문의 분류 작업은 생략된다. 추출 과정은 위에서 언급한 각 단계별로 다시 몇 단계로 세분화되며, 각 단계가 차후 프로세스에 미치는 영향이 매우 커서, 지문 인식 기술을 개발하여 상품화하기에는 많은 시간 및 금전적인 투자가 필요하다. 현재 국내에서 극히 일부 기업만이 지문 인식에 대한 원천 기술을 보유하고 있는 이유도 여기에 있을 것이다. 일반적으로 매칭 알고리즘은 특이점간의 기하 적으로 구성된 그래프 패턴의 비교로 이루어진다. 각각의 일치 여부는 점수로 산출되며, 산출된 점수가 적정 기준을 초과할 경우 이를 동일인으로 인식하게 된다. 일반적으로 지문 인식 시스템들은 이 매칭 점수의 수준 조절로 그 인증 레벨을 결정하게 된다. 지문 인식에 있어서 매칭은 사용자의 지문 입력 위치가 항상 일정하지 않음으로써 절대적인 위치의 편차가 발생하고, 피부의 특성으로 인한 일그러짐 현상 및 전체적인 영상의 회전등 그 변수가 많아서 이를 검증하는 알고리즘의 성능에 따라 그 시스템의 성능이 결정될 수 있을 것이다.

2.2 표준화

생체 인식을 이용한 기술들은 기존의 ID나 패스워드에 비해서 사용상 장점이 여러 가지가 있으나 특히 자신의 신체 일부를 활용하는 것이므로 실존에 의한 인증이라는 것이 기존의 인증 방법에 비하여 가장 큰 장점이라 할 수 있다. 즉 해당자가 실제 이어야만 본인인지 확인이 가능하므로 분실이나 대여로 인한 오용을 방지 할 수 있게 되는 것이다. 그리고 기존의 방식과 비교해 사용상 편리성도 빼 놓을 수 가 없다

이런 이유로 오래 전부터 국내 및 국외의 많은 회사들이 생체 인식을 이용한 다양한 제품들을 시장에 내놓았고 사용자들은 큰 문제없이 사용한 것으로 생각된다. 그러나 이러한 제품들은 양적으로 많아 졌을지 모르지만 이들 제품간에 호환성이 없다는 문제점들이 야기되었고 그로 인해 시스템 개발자들은 자신들이 필요로 하는 각각의 생체 인식 장비를 위해서 장비 제공 회사인 software developer's kit(SDK)에 맞추어 적응해야하는 비효율적인 면들이 나타나게 되었다. 생체 인식 기술의 표준화를 통해서 장비 개발자 입장에서는 신속한 시장 확대와 제품 판매 기회를 얻을 수 있고, 제품 사용자 입장에서는 제품의 신제품 및 보다 나은 타사의 제품을 선택하더라도 보유한 application의 재 작성 및 구매 없이 진보된 생체 인식 기술을 최소의 비용으로 이용할 수 있다. 응용 프로그램 개발자 입장에서는 개별 생체 인식 장비의 특성을 파악할 필요 없이 신속하고 안정적인 응용 프로그램 개발이 가능하다. 마지막으로 생체 인식 서비스 제공 입장에서는 개별 장비간의 호환성 확보를 바탕으로 본격적인 개인 인증 서비스를 Internet, 금융, 사회복지 등의 거시적인 관점에서 시작할 수 있다.

외국의 경우를 살펴보면 표준화 부분에 있어서 상당한 진척을 보이고 있는 반면, 국내에서는 아직 생체 인식에 관련된 표준화에 관한 작업은 거의 진행되지 않은 것으로 알고 있다.

외국의 표준화 연구에 의한 결과물 중 몇몇 단체에서 공개한 specification은 BAPI, HA-API, BioAPI, CBEFF, EFTS등이 가장 대표적이다[9].

2.2.1 BioAPI

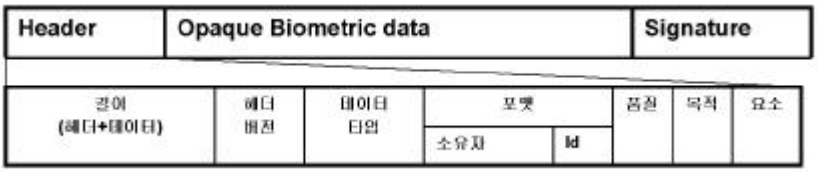
2000년 3월 30일에 version 1.0을 발표하였고 민관 주도로 활동이 이루어지고 있다 specification의 low level로 통합 결정하고 1999년 3월에 NIST의 ITL(Information Technology Laboratory)와 US Biometric Consortium이 주최한 단일화(unification)회의에서 BioAPI Consortium과 HA-API working group의 활동을 통합하기로 동의하였다. 동의 내용 중에 BioAPI Consortium의 조직을 다시 재구성할 계획도 포함되어 있었다. BioAPI Consortium 에는 생체 인식 기술, 통합 회사 및 이용자 집단 등 약 45개 회사들이 포함되어 있고 4개의 working group으로 구성되어 있다.

이 표준서는 이용자에 대한 생체 인식 등록 데이터에 대해 template이라는 용어를 이용하여 이용자가 인증을 받기 위해서는 자신들에게서 얻은 샘플이 template와 일치해야 한다.

BIR은 원 데이터(raw sample data), 중간 데이터(intermediate data), 처리된 샘플(completely processed data), 등록 데이터(enrollment data)를 포함하여 application으로 돌려보내는 어떤 생체 측정 데이터를 포괄적으로 지칭한다.

BIR 구조는 <표 2-1>과 같다

<표 2-1> BIR 구조



2.2.2 BAPI

BAPI(Biometric Application Programming Interface)는 민간 업체 주도로 1998년 9월 30일에 version 1.2를 발표하였다 이 표준화의 주요 응용 목표는 스마트 카드 소유자 확인, 물리적 접근 통제, 안정적 전자 상거래, 금융 거래, 워크스테이션 보안(사용자 인증), 파일이나 데이터베이스 암호화 등이다. 수록 내용으로는 BAPI를 3개의 레벨로 분류하였다.

2.2.3 EFTS

EFTS(Electronic Fingerprint Transmission Specification)은 1999년 1월 29일 version 7.0이 발표되었고 관 주도 및 독자 영역 내 활동을 하고 있다. 기타의 API와 달리 전적으로 범죄 수사 업무 지원을 목적으로 하고 있다. 거의 전 분야의 생체 정보와 멀티미디어 정보를 포함하는 자료 수록을 범위로 삼고 있으며, FBI 작업/납품 규격에서 ANSI로 적용중이고, 향후 세계 표준인 ISO를 목표로 하고 있다. 또한 Hardware에 관련된 부분도 정의되어있다. 현재 생체 정보 및 문자 정보에 대한 보안 및 암호화 기능 지원까지 확장 계획 중에 있다. 주요 내용은 개인 신상 정보, 사건 의뢰/접수 등 일반 관리 기록, 범죄 경력, 잉크를 사용한 지문(지문 회전 압착 날인, 네 손가락 전체의 평면 압착 날인, 장문), 사건 현장의 유류 지문, 지문 특징점, 얼굴, 사진, 신체 흉터, 문신 등의 기록, 범죄 현장 사진, 총기류 사용 후 탄흔 및 탄도 사진, 홍채, 음성 기록 등에 관련된 기록들이 수록되어 있다.

2.3 데이터베이스 보안

데이터 베이스 환경에서의 보안 위협은 심각하고 아주 중대한 문제이다. 보안성을 성취한다는 것은 위협 요소를 확인하고 적당한 기술적 정책과 방법이 필요하다는 것이다. 데이터베이스를 위협하는 위반 사항은 데이터의 부당한 검색, 수정 혹은 삭제로 이루어지는데 이들 위협 요소를 세 부류로 나누어 볼 수 있다. 첫째, 부당한 사용자가 의도적이거나 혹은 우연히 접근하여 데이터를 판독(read) 함으로써 정보의 부당한 유출(improper release of information) 이 발생한다. 둘째, 데이터의 부당한 수정(improper modification of data) 이다. 이것은 부당한 데이터 취급이나 수정에 의한 데이터 무결성에 관련된 모든 위반이다. 부당한 수정은 비 권한 판독(read)과 반드시 관련되지 않는다. 왜냐하면 데이터를 판독하지 않고도 부당하게 변경할 수 있기 때문이다. 셋째, 서비스 거부(denial of service)이다. 이것은 사용자가 데이터를 접근하거나 자원을 이용하지 못하게 하는 행위와 관련되는 것이다[6]. 위 세 가지 위협 요소에 대한 데이터베이스 보안 사항은 무결성, 비밀성, 가용성 등을 보장해 주어야 한다. 비밀성이라 함은 정보의 부적당한 유출을 방지, 탐지, 제지하는 것을 의미한다. 정보 무결성을 보장한다는 것은 정보의 부적당한 수정을 방지, 탐지, 제지하는 것을 의미한다. 시스템 가용성을 보장한다는 것은 시스템이 제공하는 서비스 접근에 대한 부당한 거부를 방지, 탐지, 제지하는 것을 의미한다[6]. 즉, 정당한 사용자가 정보를 필요로 할 때 적시에 제공될 수 있도록 컴퓨터와 통신 시스템이 고장나지 않도록 주의하고 자료에 대한 손상 방지와 백업 시스템을 충분히 갖추어 데이터베이스 서비스가 지속적으로 제공될 수 있도록 하는 것이다. 본 논문에서는 데이터베이스 보안 사항에서 두 번째 제시한 데이터베이스의 무결성을 보장하기 위해 시스템을 설계하고 구현하고자 한다. 저장되어 있는 데이터가 위, 변조가 되었는지를 탐지하고 데이터 무결성 위반의 결과를 로그 파일로 남김으로써 데이터베이스를 보안하고자 한다.

2.4 해쉬함수

무결성을 검증하기 위한 방법으로는 해쉬 함수가 널리 사용된다. 해쉬 함수는 다양한 길이의 입력을 고정된 짧은 길이의 출력으로 변환하는 함수로써 (식 1)과 같이 표현되어 질 수 있다.

$$y = h(x) \quad \text{---------- (식 1)}$$

여기서 x는 가변 길이 데이터이고, y는 해쉬 함수 h를 통하여 생성되어지는 고정 길이의 해쉬값(hash code)이다. 함수가 단방향인 경우, 메시지 다이제스트라고도 불리며, 이 기능들은 메시지를 분석하여 일정한 길이의 사실상 유일한 다이제스트를 만들어 낸다. 해쉬 함수의 대표적인 응용 분야로는 디지털 서명(digital signature)의 효율성 증대와 중요 정보의 무결성 확인을 위한 메시지 서명을 생성하는데 사용한다.

(1) MD2, MD4 및 MD5

MD2, MD4 및 MD5는 RSA의 Ron Rivest가 개발한 해쉬 기능들로, 128bit 다이제스트를 만들어 낸다. MD2가 가장 느리고 , MD4가 가장 빠르며, MD5는 MD4보다 좀더 보수적으로 설계되었다. 둘 다 공개적으로 사용 가능하다[10].

(2) MD5

MD5는 사실상의 다이제스트를 위한 해쉬 표준으로 대부분의 플랫폼에 대해 인터넷에서 공개 버전을 사용할 수 있으며 무결성 검사 시스템에도 널리 사용되고 있다.

(3) SHA-1

SHA-1(Secure hashing algorithm)은 NIST가 후원하는 해쉬 함수로 미국 정부가 표준으로 받아들였다. 160비트의 해쉬를 생성하며, MD5보다

대략 25% 더 느리지만 대체로 MD5보다는 SHA-1이 권장되고 있다.

(4) 해쉬/메시지 다이제스트의 장단점

해쉬/메시지 다이제스트의 장점으로는 암호화보다 훨씬 빠르고 결과가 고정 길이를 가진다는 것이다. 따라서 아주 큰 파일이라도 동일한 크기의 다이제스트를 만들어내므로, 데이터 전송 시 훨씬 효율적인 반면, 무결성만 보장할 수 있다는 한계 또한 존재한다. 현재 많은 인터넷 서버들이 다운 로드 받을 수 있도록 되어 있는 중요한 파일들에 대해 MD5 다이제스트를 제공하고 있으며 대부분의 전자 서명 시스템과 보안 이 메일 시스템 또한 무결성 보장을 위해 다이제스트 기능을 사용하고 있다

Ⅲ. 지문DB의 무결성 보안 설계

3.1 연구의 필요성

데이터베이스 무결성은 정보의 내용이 의도적이든 비의도적이든 부당하게 데이터의 수정이나 변조가 일어나지 않도록 데이터의 정확성을 유지시켜 주는 것이다. 지문 데이터는 개인의 고유한 신체 정보라는 점에서 정보가 유출되거나 의도적인 데이터의 변경으로 침해 발생 시 막대한 피해가 우려된다.

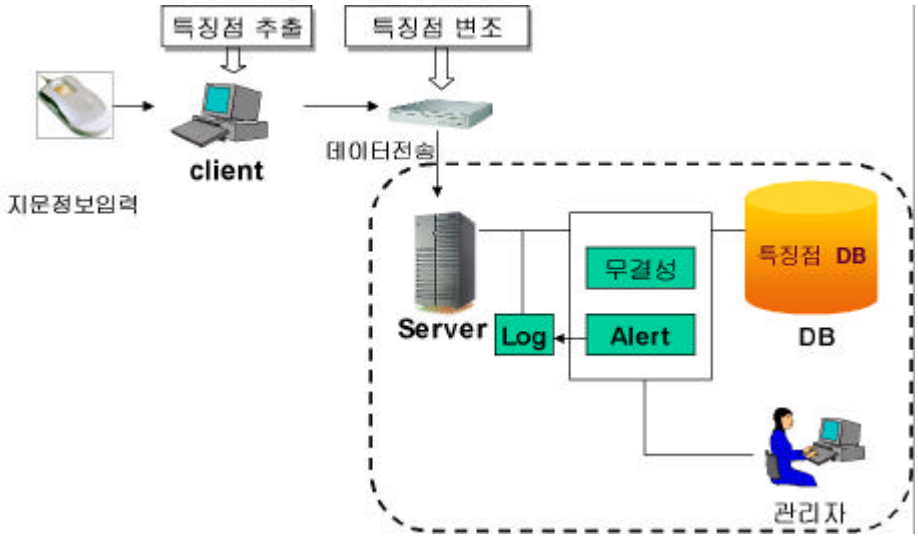
만일 A라는 사람이 의도적으로 B라는 지문 데이터에 자신의 지문 데이터를 복사하였고 B는 데이터가 변경되었는지 모른다고 가정하자. A는 B의 정보를 남용할 것이며 B는 자신의 정보가 유출되고 있는 지도 모르는 현상이 일어난다. 이로 인해 프라이버시 침해는 물론 컴퓨터 범죄 등 많은 문제를 발생시킬 수 있다.

현재 생체 인식 시스템의 개발 및 연구가 활발히 진행 중에 있다. 이들 정보는 개인의 고유한 정보라는 점에서 생체 인식 정보가 저장된 데이터베이스의 보호는 필수적이라 할 수 있다. 따라서 본 연구에서는 생체 인식 시스템 중에서도 정확도 및 처리속도, 이용의 편리성, 경제성 등이 높이 평가되고 있는 지문 인식 시스템을 대상으로 데이터의 위, 변조를 체크하여 신뢰성 있는 데이터를 관리할 수 있는 지문 데이터 베이스를 위한 무결성 검증 시스템을 설계 및 구현 하고자 한다.

3.2 시스템의 전체 구성

지문 데이터 베이스의 무결성을 보안하기 위한 전체적인 구성도는 [그림 3-1]과 같다.

지문 데이터는 본 연구에서는 지문 인식 방법 중 특징점 기반 인식 시스템에 필요한 지문 데이터들을 설계하여 이들의 중요한 데이터를 데이터 베이스에 저장하고자 할 때 해쉬 함수를 이용하여 중요한 정보에 대한 해쉬 값을 보안 필드로 추가한다. 그리고 데이터의 변조나 수정이 일어났을 경우 변경된 값에 관련된 정보를 로그 파일로 기록하여 관리자가 쉽게 데이터의 위, 변조를 파악할 수 있도록 하여 결과적으로는 데이터베이스 내 데이터의 정확성을 유지시키는 것을 목표로 한다. 무결성 검사 알고리즘은 관리자가 지문 데이터에 대한 무결성 검사를 위한 것으로 어떤 데이터가 변조되거나 수정이 되었는지 검사를 할 수 있는 소프트웨어적인 방법을 제시한다.



[그림 3-1] 지문DB 시스템 구성

3.2.1 지문 데이터베이스 보호의 위협 요소

지문 데이터의 보호는 데이터를 우발적 혹은 의도적으로 권한이 없는 사용자가 판독, 갱신하는 것을 방지하는 동시에 변조시키거나 변경하는 악의가 있는 행위 모두가 위협 요소이며 다음과 같은 세 가지 형태로 분류할 수 있다.

<표 3-1> 지문DB의 위협 요소

구분	무결성	기밀성	시스템 이상
위협	데이터의 부적절한 변조	데이터의 유출, 도난	디스크 오버플로우 접속 이상
결과	자료 손실 잘못된 정보 신뢰	정보의 손실 프라이버시 침해	정상 동작 및 서비스 방해

<표 3-1>의 위협 요소에는 무결성, 기밀성, 시스템 이상에 대한 위협을 나타낸다. 무결성에 대한 위협은 지문 데이터베이스 내의 중요 파일에 대해 데이터를 변경하거나 파괴함으로써 데이터의 일관성을 저해하고 자료가 손실된다. 그로 인해 잘못된 정보를 신뢰하게되는 결과를 얻게된다. 기밀성을 위협하는 요소로는 부적절한 접근으로부터 데이터를 유출하거나 도난 당함으로 중요 정보를 손실하게되고 프라이버시를 침해당하게 된다. 시스템을 관리하기 위해 기록되는 무결성 검사나 로그 파일들은 계속해서 기록하게된다. 이때 하드디스크의 여유 공간 부족으로 인하여 정상 동작 및 서비스를 방해받을 수 있다.

3.2.2 무결성 검증 시스템의 기능

본 논문에서는 데이터베이스 보안 사항 중 무결성 검증에 관한 시스템을 설계하고자 한다. 무결성 검증 시스템의 주요 기능은 <표 3-2>와 같다 시스템의 기능은 데이터 저장 기능, 데이터 변조 기능, 무결성 점검 기능, 로깅과 기록 기능 4가지이다. 데이터 저장 기능은 클라이언트에서 제공한 지문 특징점 데이터를 DB에 저장한다. 저장 시 중요 데이터에 대해 무결성을 검증하기 위한 해쉬 함수를 사용하여 데이터와 함께 계산된 해쉬 값을 보안 필드로 저장하게 된다. 데이터 변조 기능은 지문 데이터베이스의 무결성 여부를 검사하기 위한 전 단계로 사용자가 데이터를 의도적으로 변조하는 기능이다. 무결성 점검 기능에서는 데이터 변조 기능에서 변조된 내용들을 보안 사항으로 설정된 무결성 필드들에 대한 정보를 검사하여 지문 데이터의 정보 변경 유무를 판단하게 된다. 로깅과 기록에서는 지문 데이터의 변조에 관한 기록을 남기는 기능을 한다.

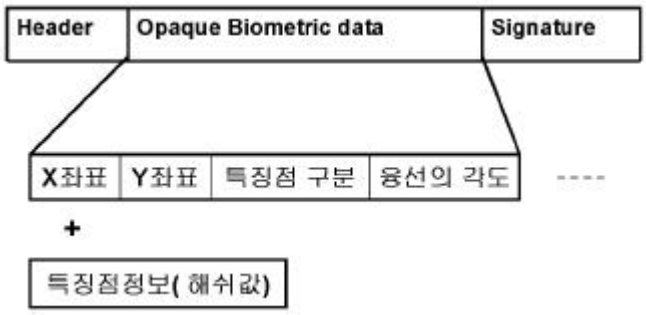
<표 3-2> 지문DB 무결성 검증 시스템의 기능

구 분	주 요 기 능
데이터 저장	지문 데이터 DB에 저장 해쉬 값을 이용한 데이터 저장
데이터 변조	지문 데이터 변조
무결성 점검	지문 데이터의 정보 변경 유무 판단
로깅과 기록	무결성 검사 결과 로그

3.3 지문 데이터 형식

지문 인식 시스템은 수치화 된 지문 정보를 미리 등록해 두고 필요시에 저장된 지문 정보를 검색하여 사용하게 된다. 이 때, 지문 영상을 저장하는 데는 많은 저장 공간을 필요로 하기 때문에 일반적으로 수치화 된 지문의 특징 정보를 저장하게 된다[3]. 일반적으로 하나의 지문 형상에 존재하는 특징점의 개수는 개인에 따라 차이가 있지만 약 50에서 150개의 특징점이 존재한다고 알려져 있다. 대부분의 지문 인식 시스템에서는 한사람의 지문 정보를 모두 저장하기 위해서는 손가락 10개의 지문 특징점들을 저장해야 한다. 이와 같이 대규모 지문 데이터베이스의 구축을 위해 특징점의 저장 구조 크기를 최소화하여 최대의 특징점 정보를 저장할 수 있는 효율적인 특징 저장 구조가 매우 중요하다.

본 연구에서는 현재 생체 측정 기술의 표준 생체 측정 API를 만들기 위해 6개 회사가 연합한 BioAPI Consortium에서 2000년에 발표한 BioAPI Specification Version 1.00에서 제안한 BIR구조를 기반으로 설계하고자 한다. BIR(Biometric Identification Record)의 지문 데이터의 구조는[그림 3-2]에 나타나며 원 데이터, 중간 데이터, 처리된 샘플, 등록 데이터를 포함한 생체 측정 데이터를 포괄적으로 지칭한다.



[그림 3-2] BIR의 지문 데이터 구조

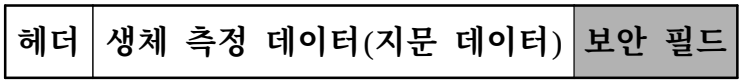
3.4 무결성 설정

지문 데이터베이스 내의 데이타의 변경 여부를 판단하기 위해서 관리자는 지문 데이터 베이스의 중요 데이터들에 대한 무결성 여부를 탐지하기 위한 항목을 설정하여 보안 필드에 저장하게 된다. 지문 인식 시스템은 수치화 된 지문 정보를 미리 등록해 두고 필요시에 저장된 지문 정보를 검색하여 사용하게 되며 특징점 기반 지문 인식 시스템에서는 일반적으로 수치화 된 지문의 특징 정보를 저장하게 된다[3]. 그리고 특징점 matching은 서로 다른 지문 이미지에서 추출된 특징점 고유의 속성과 특이점과 특징점 사이의 상대적 속성이 허용 오차 범위 내에서 서로 일치 할 때 서로 같은 특징점 이라 한다. 이렇게 볼 때 특징점과 특이점은 지문 인식 시스템에서 개인을 인식할 수 있는 중요한 정보가 되며 이 데이터의 무결성 검증은 필수적이라 할 수 있다. 특징점 정보는 여러 가지로 표현 할 수 있지만 본 논문에서는 x좌표, y 좌표, 특징점 구분, 방향성 정보 4가지를 사용한다. 이들은 데이터베이스에 저장 할 때 각 필드마다 해쉬 함수를 계산하여 해쉬 값을 저장하고 또한 해당 레코드 전체를 해쉬 함수를 이용하여 저장하게 된다.

3.5 무결성 검사

무결성 검사를 통하여 특정 데이터의 정보 변경 유무를 판단하는데 검사되는 정보로는 지문 데이터베이스의 중요 데이터 즉 지문의 특징점 정보와 한 특징점에 대한 전체 레코드의 변경 여부를 들 수 있다. 무결성 검사를 하기 위해서 지문 데이터베이스의 구조에 보안을 위한 항목이 필요하다. 본 논문에서는 BioAPI Specification Version 1.00에서 제안한 BIR구조[7] 기반에 무결성 보안을 위한 필드를 추가한다. 지문 데이터 구조는 [그림 3-3]과 같으며 헤더와 지문데이터 그리고 무결성 검사를

위한 보안필드로 구성된다. 보안 필드의 속성은 <표 3-5>에 표시되어 있다.



[그림 3-3] 지문 데이터 구조

BioAPI에서 제시한 생체 측정 데이터 레코드는 헤더와 데이터 블록으로 구성되어 있으며 일반적인 생체 측정 데이터 헤더는 데이터 내용을 나타내기 위한 최소한의 영역을 포함한다. 지문 인식 시스템에서의 헤더 정보로는 헤더의 크기 값, 지문 데이터 크기 값, 데이터 형태, 버전 정보 등이 있을 수 있다. 생체 측정 데이터는 헤더의 포맷 필드에 의해 표시되며 실제 지문 데이타를 포함하는 영역으로 가변 길이를 가진다. 특징점 기반 지문 인식 시스템에서 사용하는 일반적인 특징점의 속성 정보들은 <표 3-3>과 같이 주어진다.

<표 3-3> 일반적인 특징점 속성 정보

속성 정보	설명
특징점	끝점(Ridge ending) 분기점(Bifuracation)
기준점	중심점(core) 삼각주(delta point)
특징점의 위치	이미지 상의 절대적인 위치 특이점(core, delta point)으로 부터의 상대적인 위치
특징점이 이루는 각	끝점과 분기점이 이루는 각

이들 속성에 의해 하나의 특징점 정보를 저장하는데 본 논문에서는 기준점은 특징점 정보로 저장하지 않고, 특징점의 위치 정보 즉 특징점의 좌

표와 특징점의 종류, 특징점의 방향 정보만 저장 하고자 한다. 저장하기 위한 특징점 정보의 구조는 <표 3-4>와 같으며 [3]에서도 이와 비슷한 방법으로 설계되어 있다. 특징점 위치 좌표는 x좌표와 y좌표를 0에서 255 까지 표현하고 특징점 정보는 단점인지 분기점인지를 구분하는 것으로 특징 구분이 0이면 단점이고 1이면 분기점을 나타낸다. 융선의 각도는 특징점에서 융선의 방향을 나타내는데 360도 방향의 융선 방향 정보는 1바이트로 22.5。 간격으로 8단계를 표현한다.

<표 3-4> 특징점 정보

정보	x좌표	y좌표	특징점,	융선의 각도
크기	1byte	1byte	1byte	1byte
설명	0 255	0 255	0:단점 1:분기점	0 7 (22.5 ° 간격)

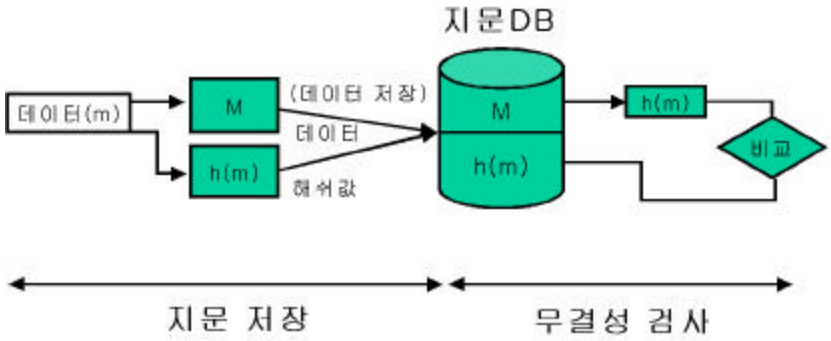
<표 3-5>는 보안을 위한 각 필드들에 대한 정보로 지문 데이터의 가장 중요한 특징점에 대해 보안 필드를 추가하는데 x좌표, y좌표, 방향성, 특징점과 특징점 전체에 대한 보안 필드를 추가하여 5가지 정보이다 이들 필드에는 해쉬 함수를 이용하여 얻어 낸 해쉬 값을 저장하게 되며 해쉬 함수 중 MD5을 이용하므로 각 필드의 크기는 16바이트를 차지하게 된다.

<표 3-5> 보안 필드

정보	각 특징점 해쉬값				특징점 전체에 대한 해쉬값
	X좌표	Y좌표	Direct	Minutiae	
크기 (바이트)	16	16	16	16	16

각 특징점의 보안을 위해 추가된 필드는 무결성 검사를 위해 사용 된다. 무결성 검사를 하는 방법은 [그림 3-4]에 나타나는데 저장된 데이터를

동일한 해쉬 함수를 사용하여 해쉬 값을 생성한 후 저장된 해쉬 값과 비교한다. 만약 비교한 값이 같다면 그 데이터는 변조되지 않은 원래의 데이터이고 비교한 값이 다르면 데이터 무결성에 위배된다. 지문 데이터의 무결성을 위한 해쉬 함수 적용 시 특징점 전체에 해쉬 함수를 적용하여 해쉬 값을 추가할 수도 있고 각 속성 정보의 무결성 검증을 위해서는 각 필드마다 해쉬 값을 저장하는 필드를 추가 할 수도 있다.



[그림 3-4] 해쉬 함수를 이용한 지문 데이터 저장 및 무결성 검사

3.6 로그 메세지

무결성 검사를 하여 데이터가 변조되거나 수정되었을 때 무결성 검사에 관련된 정보를 로그 파일에 저장한다. 지문 데이터의 중요 정보가 수정되거나 변조되었을 때 변경된 정보를 로그 파일로 기록함으로서 어떤 데이터가 변조되었는지 보고 받을 수 있다.

로그(audit trails 또는 log)는 시스템에서 일어나는 여러 상태에 대한 기록을 나타내는 것으로 데이터베이스의 무결성을 유지하는데 필요하며, 데이터베이스 접근에 대한 후속 적인 분석을 가능하게 한다. 무결성 검사와 관련된 정보를 로그 정보로 남겨 문제의 원인을 확인하고 파악할 수 있다.

무결성 검사와 관련된 로그 정보는 [그림 3-5]에 나타나 있다. 무결성 검사 결과 위배 조건이 나타나면 해당 레코드의 Id, 특징점 정보(x, y좌표, 방향성, 특징점)가 기록되고 변경된 값이라고 판단되는 필드의 저장된 해쉬 값과 검사를 위해 계산된 해쉬 값이 기록된다. 그리고 마지막에는 어떤 필드의 값이 변경되었는지의 정보와 함께 저장된다. 이로 인해 관리자는 어떤 필드의 값이 변경되었는지 알 수 있고 후속 적인 작업이 용이하게 된다.

ID	x	y	Direct	Minutiae	저장된 해쉬 값	계산한 해쉬 값	변경된 정보

[그림 3-5] 로그 자료 구조

Ⅳ. 시스템 구현 및 결과

4.1 구현 환경

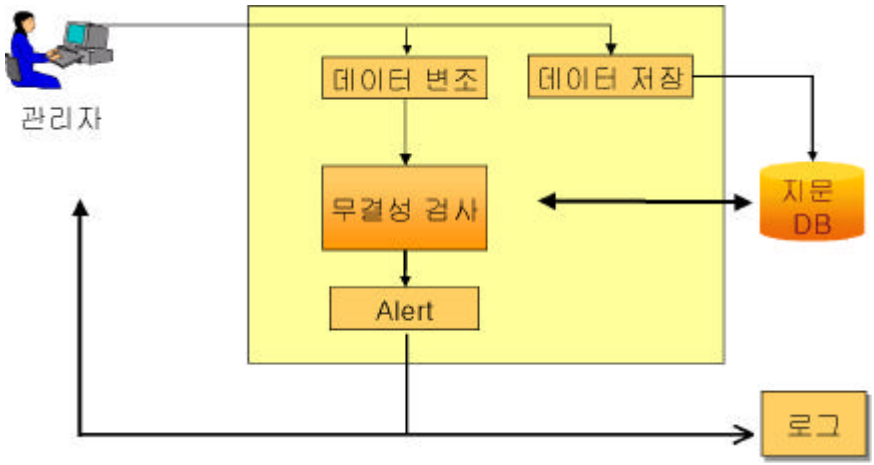
본 논문에서 설계하고 구현한 지문DB 무결성 보안 프로그램은 클라이언트-서버 환경에서 서버 시스템의 환경에 기반을 두었다. <표 4-1>은 지문DB 무결성 검사 시스템의 구현 환경을 나타낸다.

<표 4-1> 구현 환경

구 분	사 양
운영 체제	Windows2000 Server
개발 언어	Visual C++
데이터베이스	Microsoft Access
데이터베이스 연동	ODBC
해쉬 함수	MD5

지문 데이터 무결성 검증 프로그램은 사용자 편의를 위하여 GUI환경으로 구현하였고 Windows2000 운영체제에서 데이터베이스는 Microsoft Access DB를 사용하여 개발하였다. 개발 툴로는 Visual C++을 사용하고 데이터베이스와의 연동을 위해 ODBC드라이버를 사용하였다. 무결성 검사를 위해 사용되는 해쉬 함수는 MD5를 이용하였다.

4.2 무결성검사 프로그램 구현



[그림 4-1] 무결성 검사를 위한 시스템 구성도

[그림 4-1]은 무결성 검사를 위한 시스템 구성도 이다. 시스템의 구성은 특징점을 추출한 지문 데이터를 보안 필드와 함께 데이터베이스에 저장을 한다. 저장된 데이터의 무결성 검사를 위해 지문 데이터의 변조 과정을 거친다. 데이터 변조는 사용자가 원하는 레코드부터 원하는 간격, 원하는 개수만큼 레코드를 변조할 수 있다. 저장되어있는 데이터에 대한 무결성 검사는 전체적인 데이터에 대한 무결성 검사와 사용자가 원하는 사항만을 선택하여 검사하는 선택적 검사로 나뉘어 진다. 무결성 검사 후 변조 사실이 확인되면 로그 파일로 저장하고 관리자는 이를 관리할 수 있다.

4.2.1 데이터 저장 동작 구현

무결성 검사 프로그램의 사용을 위해서는 데이터를 데이터베이스에 저장하는 과정과 무결성 검사의 정확한 검증을 위한 변조 과정이 필요하다. 데이터베이스의 지문 데이터 테이블은 <표 4-2>과 같고 ODBC드라이버

를 이용하여 연결하게 된다.

<표 4-2> 지문 데이터의 테이블

serial	id	x	y	minutiae	…	made_time
순번	이름	x좌표	y좌표	특징점	…	생성 날짜

[그림 4-2]는 ODBC 드라이버를 이용하여 MS Access 데이터베이스와 연결시키기 위한 코드이다. CIntegritySet 클래스의 GetDefaultConnet라는 멤버 함수는 레코드 셋을 "FPDB"라는 데이터 소스에 연결하면 이 함수가 오버 라이딩 되는데 이 함수는 레코드 셋이 데이터 소스와 연결될 때 호출되어 레코드 셋이 연결 될 데이터 소스를 결정한다. GetDefaultSQL 멤버 함수는 레코드 셋이 테이블에 연결될 때 호출되어 레코드 셋이 연결될 테이블을 설정하는데 여기서는 “fingerprint"라는 테이블과 연결된다.

```
/*레코드 셋과 데이터 소스의 연결 */
CString CIntegritySet::GetDefaultConnect()
{
  return _T("ODBC;DSN=FPDB");
}
/*레코드 셋과 데이블의 연결 */
CString CIntegritySet::GetDefaultSQL()
{
  return _T("[fingerprint]");
}
```

[그림 4-2] 데이터 소스와 테이블 연결 설정

[그림 4-3]에 데이터베이스에 저장되는 레코드 구조가 나타나있으며 DoFieldExchange함수를 이용하여 레코드 셋의 멤버 변수와 데이터 소스의 레코드가 연결되게된다

```
Void CIntegritySet::DoFieldExchange(CFieldExchange* pFX)
{
        //{{AFX_FIELD_MAP(CIntegritySet)
        pFX->SetFieldType(CFieldExchange::outputColumn);
        RFX_Long(pFX, _T("[serial]"), m_serial);
        RFX_Text(pFX, _T("[id]"), m_id);
        RFX_Text(pFX, _T("[x]"), m_x);
        RFX_Text(pFX, _T("[y]"), m_y);
        RFX_Text(pFX, _T("[minutiae]"), m_minutiae);
        RFX_Text(pFX, _T("[direct]"), m_direct);
        RFX_Text(pFX, _T("[hash]"), m_hash);
        RFX_Text(pFX, _T("[extra]"), m_extra);
        RFX_Text(pFX, _T("[minutiae_hash]"), m_minutiae_hash);
        RFX_Text(pFX, _T("[direct_hash]"), m_direct_hash);
        RFX_Text(pFX, _T("[x_hash]"), m_x_hash);
        RFX_Text(pFX, _T("[y_hash]"), m_y_hash);
        RFX_Date(pFX, _T("[made_date]"), m_made_date);
        RFX_Date(pFX, _T("[made_time]"), m_made_time);
        //}}AFX_FIELD_MAP
```

[그림 4-3] 지문 데이터 레코드 구조

[그림 4-4]는 텍스트 데이터를 보안 필드와 함께 DB에 저장하는 source code이다. 지문 데이터의 저장은 텍스트로 저장된 지문 특징점 정보를 읽어서 순차적이며 가변적 길이로 저장한다. 텍스트 파일로 저장된 지문 특징점 정보의 각 필드는 Tab으로 구분되어 있다. 관리자로부터 ID를 입력받고 ①은 읽어들인 특징점 정보에 Tab을 구분하여 특징점 정보를 token배열에 필드로 구분하여 저장하는 부분이다. ②는 데이터베이스에

새로운 레코드를 추가하기 위한 명령을 내린다. ③은 각 필드에 해당하는 데이터를 데이터베이스에 저장하는 과정이다. ④은 x좌표에 대한 값에 대해 해쉬 함수를 적용하여 생성한 해쉬 값을 보안 필드에 저장하는 것으로 y좌표, 특징점, 위치, 전체 데이터의 대한 해쉬 값도 같은 방법으로 저장하게 된다.

```
/*지문데이터 읽기*/
token = strtok(line.GetBuffer(50),seps);
for (i = 0; token != NULL ; I++)        ---- ①
{
  sprintf (tmp_token[i], "%s", token);
  token = strtok(NULL,seps);
}
/*레코드 생성*/
m_pSet->AddNew();              ------------ ②
/* 데이터 저장 */
m_pSet->m_serial = j;
j++;
m_pSet->m_x = (CString)tmp_token[0]; --- ③
m_pSet->m_y = (CString)tmp_token[1];
m_pSet->m_minutiae=(CString)tmp_token[2];
m_pSet->m_direct=(CString)tmp_token[3];
//각 필드에 해쉬적용(x좌표)
t_eResult=m_pSet->m_x;                  --- ④
m_pSet->m_x_hash = pDoc->md5(t_eResult);
…
```

[그림 4-4] 텍스트 테이타 DB에 입력(보안 정보 포함)

4.2.2 데이터 변조 과정

지문 데이터베이스의 무결성을 점검하기 위해 데이터를 변조하는 과정이다. 변조는 원하는 레코드의 개수, 변조할 항목, 변조 시작 레코드, 변조 레코드 간격으로 구성되고 사용자가 선택할 수 있다. 변조 항목은 한 항목일 수도 있고 모든 항목을 선택해도 된다. 변조 시작 레코드와 레코드 간격은 사용자가 레코드 번호와 간격의 수를 지정할 수도 있고 Random하게 사용할 수 도 있다. [그림 4-5]는 데이터 변조를 위한 OnModulation()함수의 일부분이다. ①은 사용자가 입력한 레코드 번호로 변조를 시작하는 레코드이다. ②는 변조한 레코드의 개수만큼 수행되게 하는 문장이다. 사용자가 원하는 레코드의 개수를 입력하게 된다. ③은 선택된 레코드를 데이터베이스에서 검색하기 위한 코드이다. m_strFilter 멤버 변수에 검색 조건을 설정한 후 Requrey함수를 호출하면 데이터베이스를 검색할 수 있다. m_strFilter를 이용하면 SQL 문장을 만들어 이를 실행하게 되는데 SQL에서는 Where 키워드 다음에 검색 조건을 써서 데이터베이스를 검색할 수 있는데 m_strFilter에는 Where 키워드를 제외한 검색 조건을 SQL 문장과 똑같이 쓸 수 있다[11]. ③에서는 x좌표의 값을 변조를 하기 위한 소스 코드이다. ④는 변조된 레코드를 데이터베이스에 갱신하는 문장이다. ⑤는 사용자가 선택한 변조 간격만큼 레코드를 조절하여 다음에 변조시킬 레코드를 선택하는 문장이다. 이와 같이 ①에서 ⑤번의 과정을 원하는 레코드의 개수만큼 수행하게 된다.

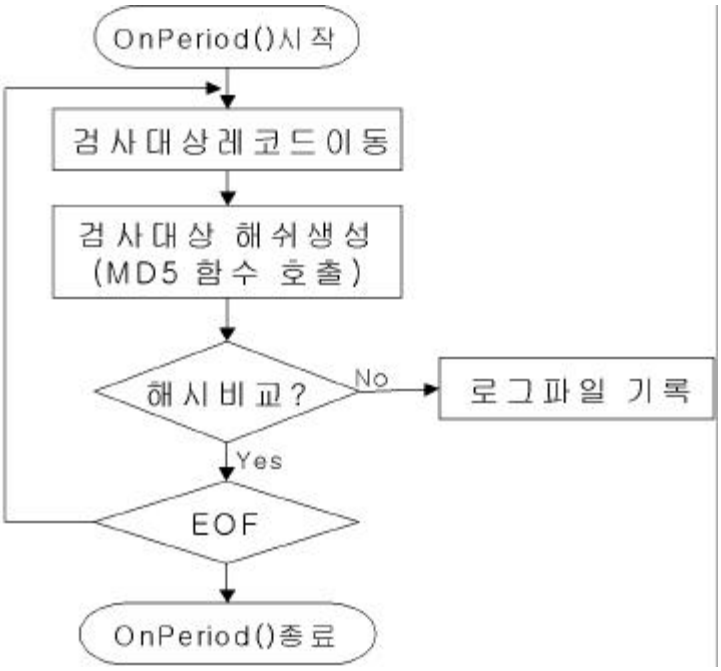
```
if(modulate_dlg.DoModal() == IDOK)
{
CString strOutText;
strOutText.Format("레코드갯수: %d, 시작레코드: %d, 레코드간격:
                    %d", modulate_dlg.m_RecCount,
                         modulate_dlg.m_start_rec,
                         modulate_dlg.m_modulate_gab);
MessageBox(strOutText, "변조 확인");
module_value=modulate_dlg.m_start_rec;  ----------------①
while(i<= modulate_dlg.m_RecCount)  ----------------②
{
  /* 데이터베이스 내의 레코드 검색*/
  m_pSet->m_strFilter.Format("serial= %d", module_value); -③
  m_pSet->Requery();
  /* x좌표 변조 원할 때 */
  if(modulate_dlg.m_modulate_x)
  {
    after_m=atoi(m_pSet->m_x);
    sprintf(temp, "%d", after_m+1);
    m_pSet->m_x=temp;
  }

.....
m_pSet->Update();/* 데이터 베이스 갱신 */  ----------④
/* 변조 레코드 간격 */
module_value=module_value+modulate_dlg.m_modulate_gab; --⑤
++i;
}
```

[그림 4-5] 데이터 변조를 위한 OnModulation() 함수의 일부

4.2.3 무결성검사 동작 구현

실질적인 무결성 검사를 처리하기 위한 부분은 전체적인 데이터를 검사하는 OnPeriod() 함수와 원하는 데이터만 선택하여 검사할 수 있는 NonPeriod() 함수를 작성하여 구성하였다.

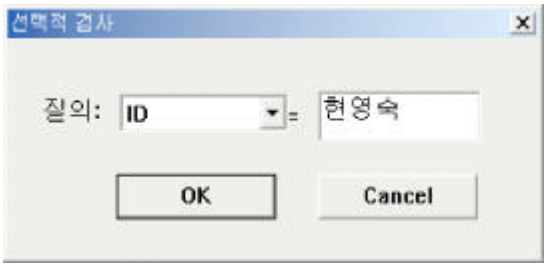


[그림 4-6] 무결성 검사 함수의 구조

OnPeriod() 함수의 구조는 [그림 4-6]과 같다. OnPeriod() 함수를 호출하면 레코드 검사를 위하여 데이터베이스의 레코드의 처음으로 이동한다. 첫 번째 레코드부터 시작해서 해시 값이 저장된 특징점의 각 필드와 전체 필드에 대해 데이터 입력하였을 때와 동일한 해시 함수를 적용하여 해시 값을 생성한다. x좌표, y좌표, 특징점, 방향성 필드, 전체 레코드에 대한 해쉬값 검사도 진행한다. 무결성 검사를 위해 새로 계산된 해시 값과 저장된 해시 값을 비교하여 결과 값이 같지 않으면 로그 파일에 저장하게 된다. 이때 로그 파일은 무결성 검사에 위배된 ID와 저장된 해쉬

값, 계산한 해쉬 값을 저장하고 어떤 필드에서 위배가 되었는지에 대한 정보도 포함 시킨다. 이와 같은 처리를 마지막 레코드까지 반복 수행하게 된다.

무결성 검사는 전체 검사와 선택적인 검사로 나누어지는데 선택적 검사를 위한 화면은 [그림 4-7]에 나타나 있다. 선택적인 검사는 ID, 날짜, 시간을 선택할 수 있다. ID는 사용자의 이름으로 원하는 사용자만 선택하여 무결성 검사를 하고 저장된 날짜와 시간별로 사용자가 선택하여 검사할 수 있다.



[그림 4-7] 선택적 검사 초기화면

4.2.4 로그 정보처리 과정 구현

지문 데이터베이스의 무결성 검사 후 보안 필드가 적용된 각 필드에 대해 데이터의 변경이 발생하면 로그 파일을 남기게 된다. 로그 파일의 정보로는 <표 4-3>에 나타나는데 로그 정보 파일의 구조는 무결성에 위배된 레코드의 ID, x좌표, y좌표, 특징점, 방향성을 표시해 주고 저장된 해시 값은 데이터 저장 시 계산하여 입력된 해쉬 값이다. 계산한 해쉬 값은 무결성 점검을 위해 새로 계산된 해쉬 값이다. 입력되어 있는 원래의 해쉬 값과 계산한 해쉬 값이 서로 다른 경우 데이터의 변조가 일어났다고 볼 수 있다. 해쉬 값은 MD5함수를 이용하여 적용한 것으로 16바이트의 해쉬 값이 나오며 16진수로 표시한 것이다. 변경된 정보의 내용으로는 어느 필드에서 데이터가 변경되었는지의 정보가 표현된다. <표 4-3>에서 로그 정보는 텍스트 파일로 저장이 되며 사용자 인터페이스에서나 메모장에서 불러 들여 읽을 수 있다.

<표 4-3> 로그 정보

ID	x	y	Dt	M	저장된해시값	계산한 해시값	변경된 정보
aaa…	19	70	0	0	6f4922f45568161a 8cdf4ad2299f6d23 ……	4173cb38f7f89ddb ebc2ac9128303f60 ……	x좌표 무결성 위배

데이터 변경이 발생했을 때 생성된 로그 파일을 생성하는 소스코드는 [그림 4-8]와 같다 로그 파일을 생성하기 위해서는 로그를 저장하기 위한 저장 파일을 생성한다. 그리고 각 각의 특징점 들을 대상으로 동일한 해쉬 함수를 적용한다. 적용한 해쉬 값과 입력되어 있는 해쉬 값을 비교하여 같지 않을 경우 로그 파일에 로그 정보를 저장한다. 로그 파일은 텍스트 파일로 저장이 되며 이를 메모장이나 다른 편집기에서도 읽어 들일 수 있다.

```
// 로그 저장 위한 구조 생성
CStdioFile file1;
file1.Open("integrity.log",CFile::modeCreate|
CFile::modeWrite);
while(!m_pSet->IsEOF())
{
  sw=0;
  /* 해쉬값 검사하는 부분 */
 // x좌표
 t_eResult= m_pSet->m_x;
 hash_result = pDoc->md5(t_eResult);
 if (hash_result.Compare(m_pSet->m_x_hash) != 0)
 {
  original_hash = m_pSet->m_x_hash;
  calculate_hash = hash_result;
  sw=1;
 }
 ......
/* 로그파일에 저장 */
CString log_write;
log_write = m_eResult;
file1.Write(log_write, strlen(log_write));
file1.Close();
UpdateData(FALSE);
}
```

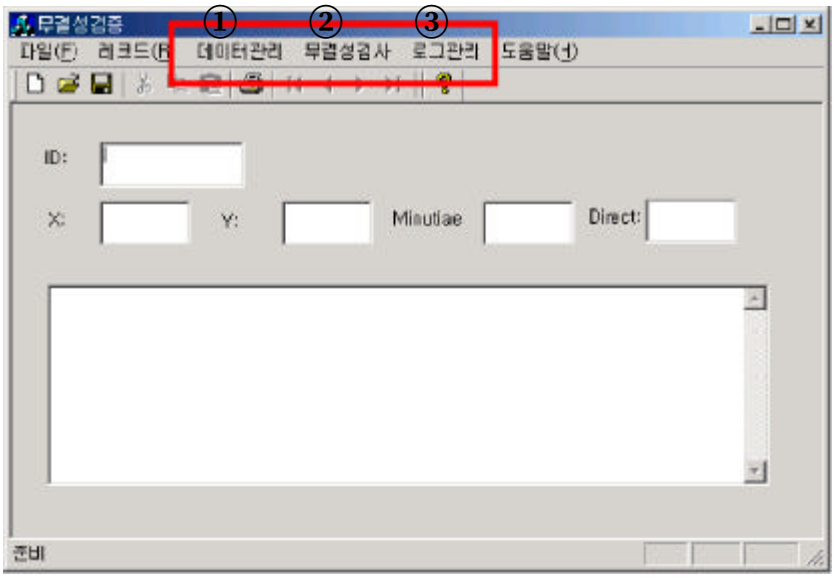
[그림 4-8] 로그 파일을 생성하는 source code

4.3 구현 결과

지문DB 무결성 보안 시스템은 지문 데이터에 보안 기능을 추가하여 정보의 내용이 함부로 변조나 위조가 되었는지 검사하는 무결성 검증 프로그램이다. 무결성 항목은 특징점 각 필드와 전체 특징점에 대해 보안 필드를 추가하였으며 테스트 결과 변조된 데이터가 검출됨이 확인되었으며 무결성 기능을 검증할 수 있었다.

[그림 4-9]은 무결성 검사 프로그램의 사용자 인터페이스이다. 사용자 인터페이스는 크게 3가지 메뉴로 구성되어 있으며 ① 데이터관리, ② 무결성 검사, ③ 로그 관리 메뉴이다.

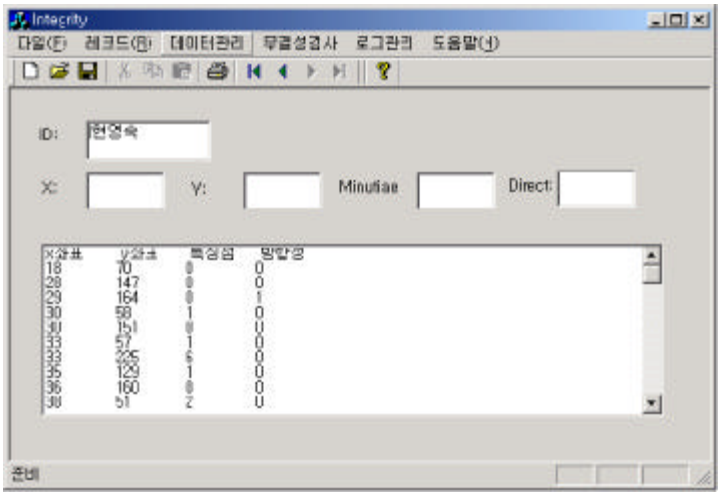
데이터 관리 메뉴에는 데이터를 저장하는 것과 무결성 검사를 위해 데이터를 변조하는 하위 메뉴가 있으며 무결성 검사 메뉴에는 검사 방법에 따라 전체 데이터를 검사하는 방법과 선택된 데이터만 검사를 하는 하위 메뉴를 가지고 있다. 로그 관리 메뉴는 무결성 검사 결과 후 무결성 위배 시 생성된 로그 파일을 볼 수 있는 메뉴가 있다.



[그림 4-9] 무결성 검사 모듈

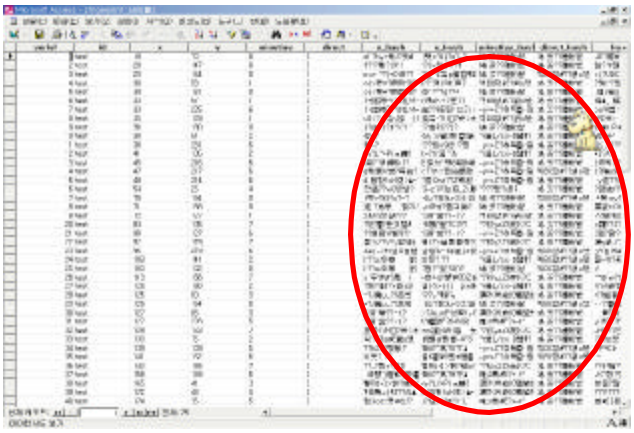
1) 데이터 저장

텍스트 파일로 된 지문 특징점 데이터를 데이터베이스에 저장한다. 이때 먼저 ID를 입력하고 데이터 저장 메뉴를 선택한다. 클라이언트에서 지문을 입력하여 지문 인식 시스템의 처리 결과에 따라 생성된 특징점 데이터에 무결성 보안 정보를 추가하여 데이터베이스에 저장하게 되는데 특징점 데이터의 내용은 x, y 좌표와 방향성, 특징점으로 텍스트 데이터로 저장되어 있고 저장된 특징점 정보는 tab으로 구분되어 있다. [그림 4-10]은 텍스트 데이터가 정상적으로 데이터베이스에 저장되었을 경우 나타난 결과이다. 데이터를 순차적으로 읽어들여 데이터 베이스에 각각의 필드에 저장을 한 후 성공적으로 저장이 되었을 때 그 결과를 사용자 인터페이스에 표시하여 준다. 사용자는 이들의 데이터를 확인 할 수 있다.



[그림 4-10] 지문 특징점 정보 데이터베이스에 저장 결과

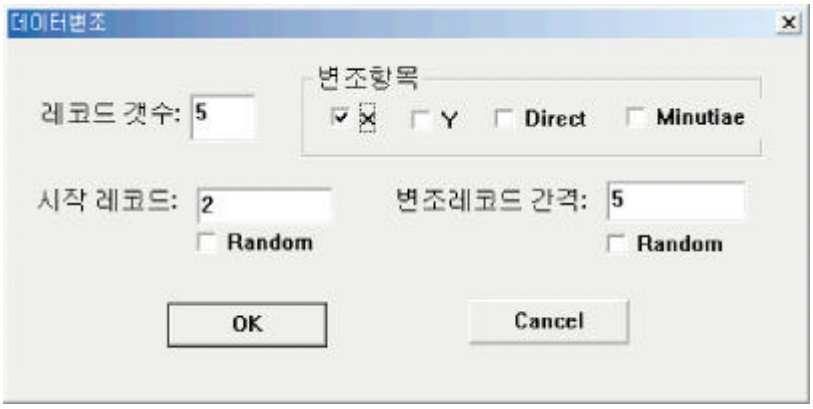
[그림 4-11]는 데이터베이스에 보안 필드와 함께 저장된 결과 화면을 표시한 것이다. 보안 필드는 무결성 검사를 위해 필요한 항목으로 보안을 위한 필드에 대해 해쉬 함수를 적용하여 계산된 해쉬 값이 보안 필드에 저장되어 있다.



[그림 4-11] 무결성이 보장된 지문 정보

2) 데이터 변조

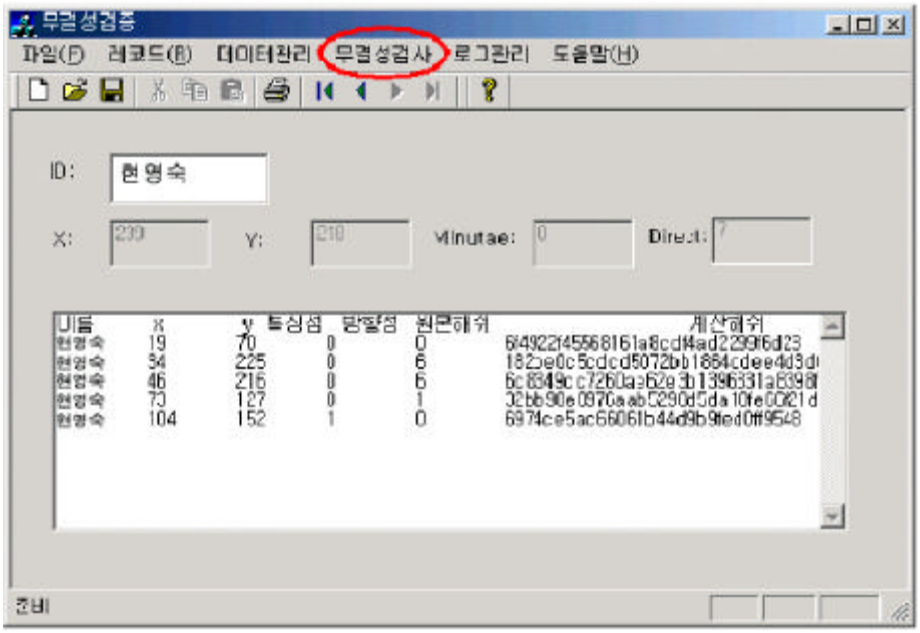
데이터 변조를 위한 화면은 [그림 4-12]과 같다 변조를 원하는 레코드 개수를 입력하고 변조할 시작 레코드와 레코드 간격을 선택하면 지문 데이터를 변조하게 된다. 시작 레코드와 변조 레코드 간격을 random으로 설정하면 시스템에서 시작 레코드와 변조 간격을 설정하여 준다. 변조 항목에는 x좌표, y좌표, 방향성, 특징점이 있다. 사용자는 변조 하고자 하는 항목을 선택한다. 복수 개의 변조 항목 선택이 가능하다.



[그림 4-12] 데이터 변조 화면

3) 무결성검사

데이터 변조 과정을 거친 후 무결성 검사를 실시하였을 때 [그림 4-13]는 무결성 검증에 위배되었을 때 표시되는 화면 결과이다. 검사 실시 후 사용자 인터페이스에 표시하며 로그 파일로 저장을 한다. 로그 파일은 메모장이나 텍스트 편집기에서도 볼 수 있으며 화면을 통하여서도 볼 수 있다.

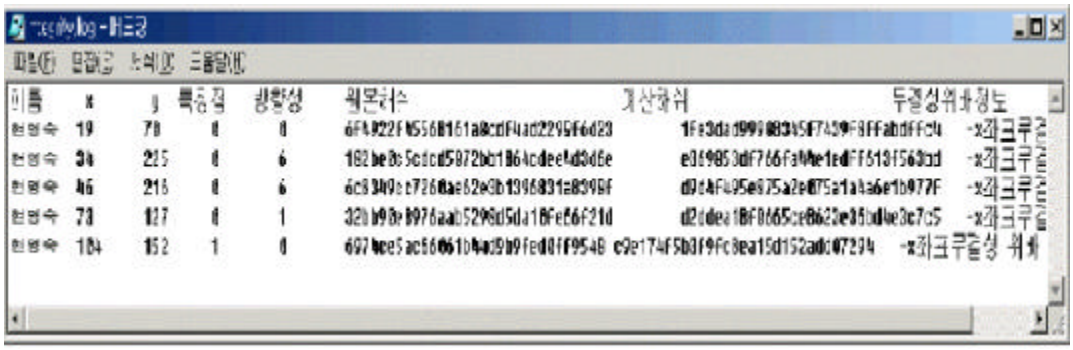


[그림 4-13] 주기적 검사 결과 화면

4) 로그 파일

로그 파일은 [그림4-14]에 나타나 있다. 데이터 변조 메뉴에서 선택한 레코드가 변조됨으로서 데이터의 정확성이 위배된다. 무결성 검사를 실시하면 무결성 위배 결과가 화면에 표시된 것이다. [그림 4-14]는 텍스트 편집기를 이용하여 로그 파일을 읽어 들인 것으로 변경된 레코드의 사용자 이름, x좌표, y좌표, 특징정, 방향성, 저장된 원본 해쉬 값, 검사를 위해 계산된 해쉬 값, 무결성 위배 정보가 표시되어 있다. 이를 통해 데이

터 베이스 내에 데이터의 변조가 일어나서 원래의 데이터의 값이 변경되었을 경우 무결성 검사 시스템을 이용하여 데이터의 무결성 검증이 됨을 알 수 있다



[그림 4-14] 무결성 위배시 생성된 로그 파일

.Ⅴ. 분석 및 평가

정보 통신의 발전이 급속하게 진행되고 있는 가운데 정보의 활용이 컴퓨터를 기반으로 한 정보를 중심으로 이루어져 있다. 그렇기 때문에 정보의 관리는 중요하고 이들 정보시스템에 대한 보안 수단으로 개인 신분 확인 기술이 적용되고 있다. 이들 생체 인식 시스템은 개인을 대상으로 한 정보이기 때문에 생체 데이터 정보의 보안은 더욱 더 중요하다고 볼 수 있다. 현재 생체 인식 분야에서는 신분 확인을 위한 프로그램의 개발이 산·학계에서 활발히 진행되어져 오고 있지만 생체 인식 데이터 자체의 보안에 관한 연구는 전무한 실정이다. 본 논문에서는 지문 데이터베이스의 보안 기능을 설정함에 있어 데이터 신뢰성을 위한 무결성 보안에 대해 소프트웨어적인 방법을 구현하였다. 무결성 검증을 하기 위해서는 보안 필드가 필연적으로 추가되어야 한다. 그리하여 본 장에서는 보안 필드를 추가하였을 경우와 보안 필드를 추가하지 않고 데이터를 저장할 때의 데이터 크기를 비교하고 저장되는 시간을 비교 분석하였다. 그리고 레코드의 개수가 증가함에 따른 무결성 검사 시간을 분석하였다. 본 검사는 다음과 같은 환경에서 실시하였다.

<표 5-1> 검사 환경

구분	사양
운영체제	Windows2000 Server
메모리	256M
CPU속도	펜티엄 1.7GHz

5.1 보안 필드저장과 비저장시 비교 분석

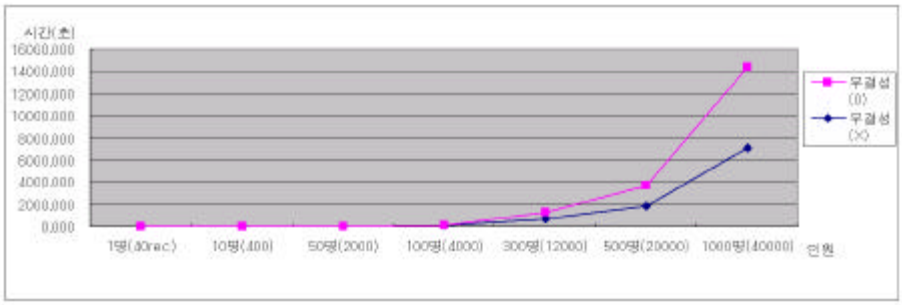
보안 필드를 적용한 지문 특징점 데이터의 크기와 저장 시간, 보안 필드를 적용하지 않은 지문 특징점 데이터의 크기와 저장 시간을 비교하였다. <표 5-2>에서는 인원을 1명, 10명, 50명, … ,1000명을 저장했을 때 시스템의 저장 속도를 서로 비교하였다. <표 5-2>의 결과를 보면 인원의 수가 증가함에 따라 저장 속도의 차가 조금씩 증가하고 있음을 알 수 있다. 인원이 500명인 경우까지는 10초 정도의 차이를 보이고 1000명인 경우에는 약 40초의 차가 발생한다. 보안 필드 추가 없이 저장한 속도에 대해 보안 필드를 추가로 저장한 속도의 비율을 살펴보면 데이터 양이 점점 증가하더라도 A에 대한 B의 증가율이 약 1% 정도로 일정하게 유지하고 있음을 알 수 있다. 즉 데이터가 증가하더라도 급격한 변화가 일어나거나 속도 차가 현저하게 늘어나지 않는다는 것을 알 수 있다.

<표 5-2> 인원의 증가에 따른 저장 실행 속도 비교(단위:초)

구분 / 인원	보안 필드 없이저장(A)	보안 필드 추가저장(B)	차이	증가율 (B/A)
1명	0.125	0.156	0.031	1.248
10명	1.641	1.781	0.140	1.085
50명	22.250	23.063	0.813	1.036
100명	78.109	79.922	1.813	1.023
300명	659.110	663.906	4.796	1.007
500명	1814.75	1824.75	9.985	1.005
1000명	7166.864	7206.297	39.433	1.005

[그림 5-1]은 보안 필드 저장과 비 저장 시의 인원의 증가에 따른 저장 실행 속도를 그래프로 표현 한 것으로 1000명 이하에서는 많은 차를 보

이고 있지 않지만 1000명 이상에서는 속도의 차가 점점 늘어나는 것을 볼 수 있다. 무결성 정보를 추가하여 저장한 속도는 저장인원 수가 많으면 많을수록 보안 필드 비 저장한 것과 비교해서 레코드 개수에 비례하여 차이가 점점 많아 질 것으로 예상된다.



[그림 5-1] 보안 필드 적용에 따른 속도 비교

<표 5-3>는 한 사람의 지문에 대해 보안 필드를 적용하지 않은 경우, 보안 필드를 특징점 데이터에 모두 적용한 경우, 레코드마다 적용한 경우, 1인당 한 개의 보안 필드를 적용한 경우에 대한 데이터 크기를 비교한 것이다. 본 논문은 특징점 필드마다 보안 필드를 적용한 것으로 1명당 데이터의 크기는 약 3360바이트를 필요로 한다. 보안 필드를 적용하지 않은 경우에 비해 21배 정도의 크기를 가지므로 많은 용량을 필요로 함을 알 수 있다. 레코드별로 보안이 추가된 경우는 한 레코드 당 16바이트를 추가함으로 1인당 40개의 레코드인 경우 800바이트가 필요함을 알 수 있다. 1인당 보안 필드를 추가하는 경우는 40개의 레코드 전체에 보안 필드 한 개를 추가하는 것으로 보안 필드를 적용하지 않고 저장하는 경우의 크기에 비해 16바이트를 추가로 필요로 한다. 그러므로 저장공간과 데이터의 보안 정도를 고려한다면 보안 필드를 구분하여 저장할 수 있을 것이다. 대규모 지문 DB 구축을 위한 특징점 저장 구조는 저장구조 크기를 최소화하여 최대의 특징점 정보를 저장하여야 하므로 한 사람의 전체 레코드에 대한 보안 필드를 적용하는 방법을 사용할 수 있을 것이고 아주 중요한 정보인 경우나 소규모 지문인 경우 본 논문에서와 같은 데이터 필드마다 보안 필드를 추가하여 저장하면 될 것이다. 사용

용도와 보안 정도에 따라 보안 필드의 추가 사항을 레코드별, 필드별, 1인별로 선택하여 사용하면 될 것이다.

.

<표 5-3> 레코드 개수별 저장 크기 비교(단위:바이트)

구분 / 레코드수	보안 필드 (X)	보안 필드 (필드)	보안 필드 (레코드)	보안 필드 (1인당)
1명(40 Record)	160	3360	800	176
10명(400 Record)	1600	33600	8000	1760
50명(2000 Record)	8000	168000	40000	8800
100명(4000 Record)	16000	336000	80000	17600
300명(12000 Record)	48000	1008000	240000	52800
500명(20000 Record)	80000	1680000	400000	88000
1000명(40000 Record)	160000	3360000	800000	176000

<표 5-2>과 <표 5-3>의 실험 결과에서 보면 시스템이 무결성을 가짐으로서 추가해야할 필드나 크기는 보안 필드를 적용하지 않은 시스템보다 필드는 많아지고 저장 크기는 많은 차이를 가짐을 알 수 있다. 하지만 속도는 완만한 곡선을 이루며 점차 증가하지만 보안 필드 미 저장에 비례하여 일정 비율로 증가함을 알 수 있다. 데이터의 중요도에 따라 보안 필드를 구분해서 전체 레코드에 대한 추가, 각 필드별 추가, 레코드별 추가 등 관리자의 사용 용도에 따라 보안 필드의 추가를 구분하여 저장하면 데이터 저장 공간은 유동성을 가진다 할 수 있다. 본 실험으로 보안 필드를 추가함으로써 속도와 데이터 크기는 다소 차이가 있지만 데이터의 위, 변조에 따른 무결성을 보장해 줌으로 데이터의 안전성은 보장된다고 할 수 있다.

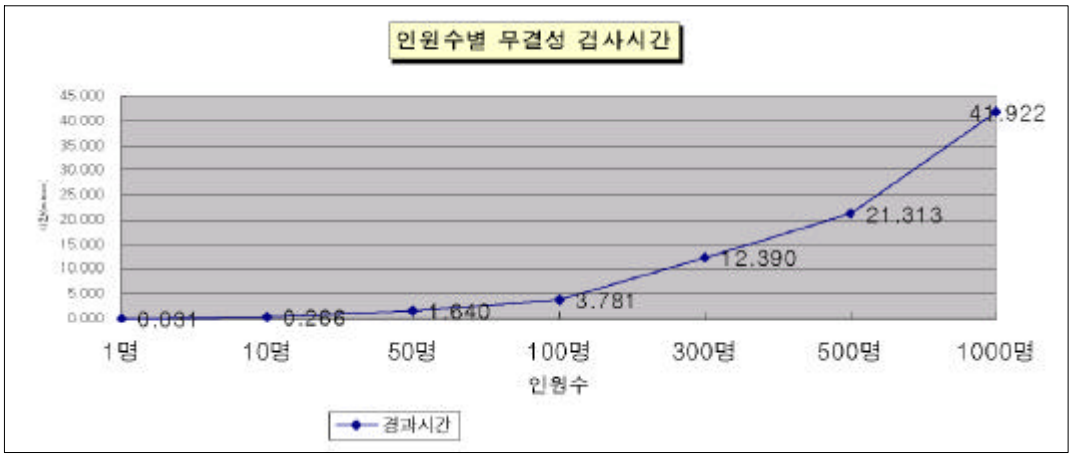
5.2 인원의 증가에 따른 무결성 검사 시간 비교

무결성 검사 시 대상 인원의 수가 증가하면 할수록 전체 데이터를 대상으로 하는 검사 시간은 점점 증가 할 것이다. 인원의 수가가 증가함에 따른 검사 수행 속도의 시간을 비교하여 속도의 증가율을 살펴보았다. <표 5-3>은 인원의 증가에 따른 무결성 검사 수행 속도를 나타낸 것이다. 인원이 1000명인 경우 약 42초의 시간이 들고 평균적으로 레코드 한 개의 계산 속도를 살펴보면 0.001059(초)의 시간이 필요함을 할 수 있다.

<표 5-4> 접근 회수에 따른 무결성 검사 수행 속도 비교(초)

인원	1명	10명	50명	100명	300명	500명	1000명
경과 시간	0.031	0.266	1.640	3.781	12.390	21.313	41.922

[그림 5-2]는 대상 인원의 증가에 따른 무결성 검사 시간을 비교한 것을 그래프로 나타낸 것이다. 검사대상 인원수가 많아짐에 따라 검사 시간은 점진적으로 증가함을 볼 수 있다. 인원수가 증가됨에 따른 그래프는 완만한 경사를 이루므로 급격한 속도의 차이는 발생하지 않는다.



[그림 5-2] 인원의 증가에 따른 무결성 검사시간 비교

Ⅵ. 결론 및 향후 과제

기존의 지문 인식 시스템들은 지문 데이터에 대한 매칭 방법들에 대해서는 활발히 연구가 진행되고 있다. 하지만 지문 데이터베이스의 공격으로 인한 지문 데이터의 보안에 관한 연구는 그렇지 않은 실정이다. 따라서 본 논문은 지문 데이터의 저장 구조나 무결성 정보에 있어서 부족한 점이 있지만 보안이 강화되어야 하는 지문 데이터베이스를 대상으로 무결성이 유지되는지 판단할 수 있는 지문 데이터베이스 무결성 검증 시스템을 구현하였다. 지문 데이터의 무결성을 유지하기 위하여 보안 필드를 추가하고 데이터의 변조나 수정이 발생하면 관련 정보를 로그로 남겨서 관리자가 관리 할 수 있도록 하는 무결성 검사 시스템의 모델을 제시하였다. 그리고 분석 및 평가에서는 인원의 증가와 보안 필드 적용에 따른 데이터의 크기와 저장 속도를 비교하였고 무결성 검사를 할 경우 대상 인원의 증가 따른 검사 수행 속도를 비교하였다. 그 결과 데이터의 크기와 저장 속도는 인원수의 증가에 따라 점차 증가하였지만 보안 필드 비 저장한 경우와의 저장 속도와 일정 비율로 증가하였으며 그다지 많은 차를 나타내지 않았다. 데이터의 크기는 사용 용도에 따라 보안 필드를 달리 지정함으로 크기의 조정이 가능하였다. 무결성 검사 시간도 인원수가 증가함에 따라 완만한 곡선을 이루므로 급속한 시간의 변화는 일어나지 않았다.

향후 연구 과제로는 다음과 같다 첫째, 본 논문은 무결성 검증에 관한 보안 사항만 검사하는 것으로 비밀성이나 가용성에 대한 보안 기법 연구가 계속 진행되어야 한다.

둘째, 지문 데이터베이스의 보안을 위한 데이터베이스 설계에 관한 연구가 계속 진행되어야 할 것이다.

참 고 문 헌

[1] 장동혁, 이동선, 이상범, “상관성이 적은 동일인 지문 영상에서의 지문 인식 알고리즘에 관한 연구” 한국 정보처리 학회 추계 학술 발표 논문집 제6권 제2호 ‘99.

[2] 황준호, “생체 측정을 이용한 안전한 인증 프로토콜 설계 및 효율적 지문 인식 알고리즘의 구현”, 포항 공과대 학교 석사 학위논문, 2001.

[3] 김현철, “특징점의 융선 연결 정보를 이용한 지문 인식”, 안동대학교 석사 학위논문, 2000.

[4] 정보 보호21C, “암호학의 이해와 응용”, 99.09.

[5] 김정덕, “데이타베이스 보안을 위한 통제 기술”, 데이터베이스 연구회지, 2000.8.

[6] 심갑식 편저, “데이터베이스 보안”, 다성 출판사, 2001.

[7] 한국 전자 통신 연구원, “생체 특정 시스템 기술/시장 보고서”

[8] 이만영,김지홍,류재철,송유진,염흥열,이임영, “전자 상거래 보안 기술”, 생능 출판사, 1999.

[9] www.kisa.or.kr/K_trend/KisaNews/200104/standardization6.html 정보 보호 뉴스

[10] 정보 보호21C, “보안 메커니즘(Security Mechanism)”, 2001.10.

[11] 김용성, “Visual C++ 6 완벽 가이드”, 영진 출판사, 2002.

[12] John Paul Mueller지음, 김형주박사 감수, “Visual C++6", 마이트 Press, 1998

[13] 구하성, “지문 인식 시스템”, 전자 공학 회지, 제26권, 제 11호, 1999년 11월, p1106-1113.

[14] 박정재, 구하성 “지문 인식을 이용한 Peer to Peer Network 보안 모델의 설계”, 2001년 한국 멀티미디어 학회 춘계 학술 발표 논문집, p481-487.

[15] 조상현, 차성덕, “시스템 보안 강화를 위한 로그 분석 도구 ILVA와

실제 적용 사례", 통신정보보호학회논문지, 제9권, 제3호, 1999.09, p13-25.
[16] 현영숙, 김창수, "지문 데이터베이스의 보안 관리 연구", 2001년 한국 멀티미디어 학회 추계 학술 발표 논문집, p686-689.
[17] http://cherup.yonsei.ac.kr/
[18] http://www.bioapi.org
[19] http://www.nist.gov

감사의 글

대학원 생활이 처음에는 막막하여 갈 길이 멀었으나 차츰 배우는 즐거움을 느끼면서 나름대로 행복한 시간들이었습니다. 이제 대학원 생활들을 접어야 하는 시간 앞에 그동안 학문의 결실을 맺을 수 있도록 도와주신 많은 분들에게 감사의 말씀을 전하고자 합니다. 본 논문이 완성되기까지 너무나 부족한 제게 열정을 다해 지도해 주시고 어려울 때면 언제나 용기를 불어넣어 주신 김창수 교수님께 진심으로 감사 드립니다. 또한 바쁘신 와중에서도 논문 심사를 위해서 세심한 관심과 격려를 아끼지 않으셨던 박만곤 교수님, 정목동 교수님께도 감사드립니다. 또한 여러 가지 면에서 지도해 주신 박승섭 교수님, 박지환 교수님, 윤성대 교수님, 이경현 교수님, 김영봉 교수님, 여정모 교수님, 박흥복 교수님, 정순호 교수님께도 감사드립니다. 사소한 일에서부터 큰 일에까지 항상 도움을 아끼지 않은 시스템 소프트웨어 및 이동 통신 연구실 선 후배님등께 감사의 마음을 전합니다. 그리고 2년 반동안 항상 똑같은 모습으로 가까이서 많은 도움을 주신 변옥남 선생님, 장용호 선생님, 정동호 선생님, 박철동 선생님에게도 감사드립니다.

바쁘다는 이유로 자주 찾아 뵙지 못해도 이해해 주시고 언제나 뒤에서 물심양면으로 도와주신 시아버님, 시어머님과 저를 믿고 항상 딸을 위해 기도해 주시며 용기를 주시는 아버님, 어머님께 진심으로 감사드립니다. 또한 배움의 길을 열어주시고 제자의 모자람을 채워주신 이재관 교수님, 김용덕 교수님, 이봉근 교수님, 김창식 교수님, 이재욱 교수님, 양황규 교수님, 박수현 교수님, 강석훈 교수님, 이준재 교수님께도 진심으로 감사드립니다.

끝으로 곁에서 싫은 내색하지 않고 끝까지 묵묵히 도와준 사랑하고 존경하는 남편과 엄마의 도움이 가장 필요한 시기인데도 잘 견디고 건강하게 자라준 나의 자랑 은지에게 미안한 마음과 함께 이 자그마한 기쁨을 함께 나누고자 합니다.

2002년 7월

현 영 숙 씀