개인정보DB 암호화 관리 안내서

2009.10

방송통신위원회
한국인터넷진흥원
Korea Internet & Security Agency

Contents

1. 개요

최근 국내 정보통신망법 하위 고시인 '개인정보의 기술적 · 관리적 보호조치 기준' 개정으로 정보통신 서비스 제공자들의 고객 개인정보 암호화 저장이 의무화됨에 따라, 주민등록번호, 신용카드번호 및 계좌번호 등을 저장 · 관리하는 개인정보 DB 암호화에 대한 관심이 높아지고 있다. 공공부문의 경우, "공공기관의 개인정보보호에 관한 법률"에서 개인정보 DB 암호화 등을 명시적으로 규정하고 있고, 국외의 경우에도 의료정보 보호법(HIPPA) 등에서 민감한 개인정보에 대한 기술적 보호 조치로 암호화 규정을 두는 등 전 세계적으로 개인정보보호를 위한 법제도 마련의 움직임이 활발하다.

이에 따라 본 안내서에서는 개인정보보호를 위한 암호화 규정을 중심으로 국내 · 외 관련 법제도 재개정 현황을 파악해 보고, 해당 법률을 준수하기 위한 개인정보보호 대책의 하나로 개인정보 DB 암호화 관리 방안을 소개한다. 특히, 정보통신 서비스 제공자가 개인정보 암호화 적용에 참고할 수 있도록 국산 암호기술 SEED[1] 및 해쉬함수 SHA-256[2]을 이용해 개인정보 DB를 암 · 복호화하는 간단한 예를 소스코드 형태로 제공한다.

1) SEED는 128비트 크기의 블록 단위로 암 · 복호화를 수행하는 암호 알고리즘으로 1999 KISA를 중심으로 국내에서 개발된 ISO/IEC 국제 표준 블록암호 알고리즘
2) SHA-256은 입력 길이에 상관없이 일정한 길이(SHA-256의 경우, 256비트)의 메시지 압축 값을 출력하는 해쉬 알고리즘으로 미국 표준 해쉬 알고리즘(SHA, Secure Hash Algorithm)의 하나임

2. 국외 개인정보 암호화 관리 법제도 현황

국외 기업의 경우, 다음에서 소개하는 개인정보보호 관련 법률을 준수하기 위해 다양한 형태의 개인정보보호 대책을 수립 · 운영하고 있다. 특히, 최근 시행된 미국의 의료정보 보호법이나 전 세계적으로 기업 등에 적용된 사베인-옥슬리법 등은 기업의 개인정보보호에 대한 필요성 및 인식을 높이고, 이에 대한 대책을 수립하는데 중요한 동기가 되었다고 할 수 있다. 이에 따라, 본 장에서는 개인정보보호를 위한 대표적인 법제도 현황과 주요 규정, 특히 개인정보 암호화 관리 관련 규정 및 처벌사항 등을 소개한다.

2.1 캘리포니아 데이터베이스 보안 침해 고지 법(SB1386)

2003년 6월 발효된 캘리포니아 데이터베이스 보안 침해 고지 법(California Database Security Breach Notification Act)(일명, SB 1386)은 인가되지 않은 사람이 캘리포니아 주민의 암호화되지 않은 개인정보를 획득했거나, 또는 획득했다고 판단되어질 때 해당 개인정보를 관리하는 기업 또는 주 기관이 그 데이터의 보안 침해 사항을 당사자에게 공개하도록 규정하고 있다.

캘리포니아 SB1386은 데이터베이스 보안 및 침해 고지에 관해 처음으로 다루고 있는 법으로 제정 이후 캘리포니아에서 사업을 수행하는 모든 기업뿐만 아니라 캘리포니아에 위치하지 않더라도 캘리포니아 주민의 개인정보를 보유한 기업들에게 영향을 미쳤다. 또한, SB1386은 미국의 다른 주에도 영향을 미쳐 미국 내 반 이상의 주에서 이와 유사한 법과 규칙이 제정하였다.

2.1.1 적용 대상

캘리포니아에서 사업을 수행하고 개인정보를 소유하거나 소유를 허가 받은 모든 기관, 개인, 또는 회사는 SB1386을 따라야 한다. 즉, 주에서 고객 또는 종업원이 있는 모든 조직은 영향을 받는다. 또한 SB1386은 조직의 위치 또는 사업을 수행하는 위치에 관계없이 캘리포니아 주민의 개인정보를 소유한 모든 조직에 적용된다.

2.1.2 주요 내용

SB1386은 "보안 침해"의 정의를 캘리포니아 프라이버시 보호위원회(Office of Privacy Prevention)에 의해 정의된 "개인정보의 보안, 기밀성 또는 무결성을 침해하는 컴퓨터 처리되는 데이터의 비인가된 획득"으로 규정하고 있다. 또한, "특정 개인정보"는 이름과 다음의 정보를 포함하는 암호화되지 않은 컴퓨터로 처리되는 개인정보로 정의하고 있다.

- 사회보장 번호, 운전면허 또는 캘리포니아 개인식별 카드 번호
- 은행계좌 번호, PIN번호와 동반되는 신용카드 번호/직불카드 번호
- 계정 접근에 필요한 접근 코드 등

SB 1386은 조직이 "인가되지 않은 사람이 캘리포니아 주민의 암호화되지 않은 개인정보를 획득했거나 또는 획득했다고 합리적으로 생각되어질 때 시스템의 보안 침해를 캘리포니아 주민에게 공개"하도록 규정하고 있다. 특히, 도난당한 데이터는 도난당하지 않았다고 증명되지 않으면 도난당했다고 가정하는 것으로 규정하고 있다.

또한, SB 1386은 조직이 개인정보를 암호화하도록 규정하지는 않으나 그 법이 암호화된 데이터에는 적용되지 않음을 기술하고 있어 실제적으로 개인정보를 암호화하는 경우 본 법률에 따른 처벌 등을 받지 않음을 알 수 있다.

SB 1386에 의해 조직은 개인정보가 오용됨으로 인한 아이덴터티 도용을 방지하기 위해 온당한 책무(due diligence)를 다해야 한다. 이를 위해서는 적절한 암호가 필요하다. 만약 암호 솔루션이 정보의 라이프사이클을 통한 여러 형태(저장소, 전송 중, 서버 데이터베이스 또는 개인 컴퓨터)에서 개인정보를 적절하게 보호할 수 없다면 조직은 임무 태만이 되므로 결국 그에 대한 책임을 져야 한다.

2.2 의료정보 보호법(HIPAA)

미국 의료정보보호법(HIPAA, the Health Insurance Portability and Accountability Act of 1996, 공법 104-191)는 건강관리시스템의 효율성과 효과성을 개선하고 사기 발생률을 줄이는 것을 목적으로 제정되었다. 특히, 이 법은 환자 건강관리 정보의 안전한 전자적 이동에 중점을 두고 있어 건강관리 정보를 안전하게 송·수신하기 위한 시스템 보안과 정보 프라이버시에 대한 규정을 포함하고 있다.

2.2.1 적용 대상

사적인 건강 또는 환자와 관련된 정보를 통제하고 관리하고 저장하고 또는 교환하는 모든 객체는 규모에 상관없이 2006년 4월부터 HIPAA 요구사항을 따라야 한다. 이러한 객체에는 건강관리 조직, 건강 기록을 관리하는 고용주, 생명보험회사, 대부분의 의사, 간호사, 약사, 병원, 클리닉, 개인병원, 요양소 등이 포함된다. 또한, 의료 서비스와 관련된 전자적 기업 거래에 관한 계약을 하거나 해당 거래를 수행하는 회사도 적용 대상이 되며 그러한 거래는 배상 요구, 지불 통지, 적격성 요구, 전문의 허가 요구 등을 포함한다.

2.2.2 주요 내용

미국의 의료보장 프로그램을 관리하는 건강인권 서비스 부(DHHS, Department of Health and Human Services), 건강 계획을 운영 또는 건강 관리를 제공하는 다른 연방 기관, 주 의료 보조 기관, 민간 건강 기관, 건강관리 제공자, 건강관리 요양소 등은 HIPAA법에 따라 자신의 고객(예를 들어, 환자, 피보험자, 제공자, 건강 기관)에게 자신들이 수집, 관리 및 이용 · 전송하는 전자적 건강 정보의 무결성, 기밀성, 가용성이 보호되고 있음을 고객들에게 증명할 수 있어야 한다.

전자적으로 송 · 수신되는 건강 정보의 기밀성, 무결성 및 가용성을 보호하기 위해 HIPAA의 261항부터 264항까지의 II장 F절에서는 시스템 보안 관련 미국 국내 표준 채택을 요구하고 있다. 이러한 표준에는 HIPAA 적용 대상에 의해 보관 및 송 · 수신되는 동안 건강 정보를 안전하게 관리하기 위한 대책을 포함하고 있다.

HIPAA의 11장 Part.C는 미국 사회보장법(Social Security Act)의 1171항부터 1179항까지 내용으로 구성되는데, 이 항에서는 HHS, 건강기관, 건강관리 조직과 특정한 건강관리 제공자에 대한 요구사항 및 다양한 용어 정의가 포함되어 있다. 특히, 1173(d)항은 건강 정보를 저장 또는 전송하는 적용 대상이 정보의 무결성과 기밀성을 보장하고 예측되는 위협 · 위험에 대비할 수 있도록 적절한 기술적 · 관리적 · 물리적 대책을 강구하도록 규정하고 있다. 또한, 1173(d)항은 HHS 장관에게 건강 정보를 관리하기 위해 사용되는 기록 시스템의 기술적 기능과 보안 대책 비용, 건강 정보에 접근 하는 사람에 대한 훈련 · 교육, 기록 시스템에서의 감사 기록의 가치, 소규모 건강관리 제공자와 지방 건강관리 제공자의 수요와 능력 등을 고려하는 보안 표준을 채택하라고 요구하고 있다. 또한 1173(d)항은 표준을 통해 건강 관리 기관이 만약

보다 큰 조직의 일부라면, 그 기관의 활동을 독립화 시키는 정책과 보안 절차를 가지고 있어 보다 큰 조직이 건강정보에 인가를 받지 않고 접근하는 것을 막을 수 있음을 보장해야 함을 요구하고 있다.

HIPAA는 건강 관리 조직이 환자 정보의 기밀성을 보장하기 위해 프라이버시 규칙(Privacy Rule)과 보안 규칙(Security Rule)을 포함해 환자와 관련된 데이터의 보안을 증대하라고 규정하고 있다. 프라이버시 규칙은 건강 관리 조직과 관련 참여 객체들 간에 건강과 관련된 모든 정보를 안전하게 보호해야 한다고 규정하고 있으며, 이는 서면, 구두, 전자적인 것을 포함하는 모든 형태의 환자 건강 정보를 대상으로 하고 있다. 보안 규칙은 전자적으로 보호되는 건강정보(ePHI, electronic protected health information)의 기밀성, 무결성, 가용성을 보장하기 위한 것으로 "(a)전자적으로 저장되거나 전송되고, (b)개인과 관계되는 모든 건강 정보"에 대해 동일한 보안 수준의 인증, 데이터 무결성 · 기밀성 · 접근제어를 제공한다. 이러한 HIPAA의 보안 규칙을 따르기 위한 최적의 방법은 효과적인 네트워크 보안과 함께 데이터 암호화를 적용하는 것이라 할 수 있다.

2.2.3 처벌 사항

개인정보 소유자인 환자들은 해당 건강 관리 조직 등이 HIPAA를 따르지 않는다고 생각되면 건강 및 인권 서비스 부(DHHS)에 손해배상을 청구할 수 있다. HIPAA를 위반할 경우 위반 건당 100달러에서 위반 건 당 또는 위반된 금지 건 당 매년 25,000달러까지의 벌금형에 처해진다. 특히, 환자의 프라이버시를 침해하는 범죄는 최고 250,000달러의 벌금과 10년의 구속형에 처해질 수 있다.

2.3 사베인-옥슬리법(SOX)

사베인-옥슬리법(SOX, Sarbanes-Oxley Act)은 2001년 엔론사 사건을 계기로 주식회사 등의 재무 상태와 투명성, 책임성을 강화하기 위해 2002년 제정되었다. SOX은 소규모 회사도 재무 보고서에 대한 내부 통제의 평가에 관한 규정(404항)을 따라야 한다고 요구하고 있다.

2.3.1 적용 대상

미국 SEC(Securities and Exchange Commissions) 관할의 모든 기업은 SOX 요구사항을

따라야 한다. 이것은 기본적으로 모든 주식 회사는 SOX의 요구사항을 지켜야 한다는 것을 의미하는 것으로 SOX는 재무 서비스 기업, CPA와 CPA 기업뿐만 아니라 주식회사의 대리인에게까지 직접적으로 영향을 미친다. 또한 주식 상장에 관심 있거나 주식회사의 인수 또는 합병의 목표가 될 수 있는 개인 회사도 SOX를 따라야 한다.

2.3.2 주요 내용

SOX는 주로 기업의 재무 서비스와 관련된 규정이 대부분이나 특히,재무 상태의 투명성, 건전성 등을 보장하기 위한 방법의 하나로 재무 감사 기록을 보호하고 관리해야 함을 규정하고 있다(802항). 이 외에도 SOX에 따라 기업은 재무 보고에 대한 적절한 내부 통제 구조와 절차를 가질 책임이 있으며, 경영은 이러한 내부 통제를 평가해야 한다. 또한 기업은 기업의 주식 가격 또는 재무 실행에 영향을 주는 모든 사건을 48시간 내에 실시간으로 공개해야 하는데, 이 공개된 정보의 진위성 · 정확성을 보증해야 하는 등 재무 정보의 안전한 관리를 위한 다양한 요구사항을 규정하고 있다.

이렇게 다양한 SOX 요구사항을 따르기 위해 CEO와 CFO가 자신이 회사에서 적절한 "내부통제"가 데이터 변경으로부터 보호한다는 것을 다음의 항목 등을 통해 입증해야 한다.

- 정보에 대한 보증되고 정당한 접근
- 정보에의 접근을 보증하기 위한 프로세스와 통제
- 정보 공개 전의 정보 구분 및 보호

즉, SOX를 따르기 위해 기업은 데이터의 장시간 관리와 보안(무결성, 가용성)을 보증해야 한다. 또한 기업은 공개된 데이터의 진위성과 무결성을 보증해야 한다.

2.3.3 처벌 사항

SOX는 연방 조사와 파산 절차와 관련된 문서나 기록을 파괴하거나 위조하는 개인에게 100만 달러의 벌금형부터 20년이 넘지 않는 구금형을 규정하고 있다.

2.4 금융정보 보호법(GLBA)

금융 관련법의 여러 변화 중에서 GLBA(Graham-Leach-Bliley Act)는 고객의 사적인 금융정보를 보호하기 위해 설계된 중요하고 새로운 규정을 포함한다. GLBA는 금융기관이 비인가된 사용 · 접근으로부터 사적인 개인정보를 안전하게 하고 보호하도록 규정한다.

캘리포니아 SB1386과는 다르게 GLBA는 개인의 사적인 금융정보가 암호화되었다 할지라도 조직이 오용의 가능성을 합리적으로 방지하기 위해 효과적인 암호화 기법을 사용했음을 증명할 수 없다면 고객에게 이를 고지를 해야 한다

2.4.1 적용 대상

GLBA는 은행, 신용 조합, 증권 브로커, 보험 회사와 같은 미국 금융 기관에 영향을 미친다. 다른 형태의 금융 제품과 서비스를 고객에서 제공하는 기업 역시 영향을 받는다. 이러한 제품과 서비스는 대출, 특정한 형태의 고객 대부의 브로커나 서비스, 화폐 전송 또는 보호, 개인적 세금 반환을 준비하는 업무, 금융 자문 또는 신용 자문을 제공, 부동산 서비스 등을 포함한다.

2.4.2 주요 내용

고객의 사적인 금융정보에 대한 보호 규정은 GLBA의 5장에서 주로 다루고 있다. 다음은 5장의 주요 내용이다.

o 비공개 개인정보의 프라이버시 ➡ 금융 프라이버시 규칙

- 고객의 프라이버시를 고려해 고객의 비공개 개인 정보에 대한 보안과 기밀성을 보호하기 위해 확실하고 지속적인 의무를 다해야 함
- 금융 프라이버시 규칙은 고객의 비공개 개인 정보의 보호와 관련하여 회사와 고객 사이의 프라이버시 정책 협의를 제공

o 고객의 재무 데이터 보호 대책 구축 ➡ 보호 대책 규칙

- 기관은 고객 기록과 정보의 보안 · 무결성을 보증하고 그러한 기록의 보안 · 무결성에 대해 예측 가능한 위협이나 피해에 대비해야 함

– 고객 기록과 정보를 비인가된 접근 · 사용으로부터 보호하기 위해 자신의 관할권에 속하는 재무 기관에 대해 관리적, 기술적, 물리적 대책과 관련된 적절한 표준을 구축해야 함

대부분의 재무 서비스 기관이 고객에게 프라이버시 정책을 알려주는 반면 개인 정보를 안전하게 하기 위해 강한 데이터 보호 대책을 적절하게 위치시키는 기관은 별로 없다.

보호 대책 규칙은 기업에게 고객 정보를 보호하기 위한 자사의 프로그램을 설명하는 정보보호 계획을 개발하도록 요구한다. 미국 FTC (Federal Trade Commission)는 명시적으로 정보보호 계획의 일부에 "공공 네트워크를 통해 전자적으로 전송될 때 민감한 고객 정보를 암호화하는 것을" 포함해야 한다고 기술하고 있다.

2.4.3 처벌 대상

GLBA는 8개의 연방 기관과 주에 금융 프라이버시 규칙과 보안 대책 규칙을 관리하고 시행하도록 하는 권한을 주었다. GLBA를 따르지 않을 경우 다양한 벌금이나 각 위반에 대해 5년의 구금형에 처해질 수 있다. 특히, 2003년의 개정안에서는 금융 기관은 위반에 대해 100,000달러가 넘지 않는 벌금이, 금융 기관의 임직원에 대해서는 10,000달러가 넘지 않는 벌금이 부과되어야 한다고 기술하고 있다.

3. 국내 개인정보 암호화 관리 법제도 현황

국내 기업이 개인정보 암호화를 채택하는 이유는 외국 기업과 마찬가지로 주로 법적 규제 때문이라 할 수 있다. 국내 관련 법은 주로 개인정보보호 관련법이 있는데, 아직까지 개인정보를 수집하거나 이용하는 모든 영역의 조직이나 개인에게 적용되는 개인정보보호 관련 법률은 없으나 행정안전부가 공공 · 민간의 모든 개인정보 처리자에 대해 적용하는 개인정보보호법 제정을 추진하고 있다. 공공 부문은 "공공기관의 개인정보보호에 관한 법률"을 통해 규정하고 있고, 민간 부문은 정보통신망 이용촉진 및 정보보호 등에 관한 법률(정보통신망법), 의료법, 신용정보보호법 등을 적용하고 있는데, 이 법률 중에 민간 영역의 규제 범위가 가장 넓은 법률은 정보통신망법이다.

3.1 정보통신망 이용촉진 및 정보보호 등에 관한 법률

정보통신망법은 정보통신망의 이용을 촉진하고 정보통신서비스를 이용하는 자의 개인정보를 보호함과 아울러 정보통신망을 건전하고 안전하게 이용할 수 있는 환경을 조성하여 국민생활의 향상과 공공복리의 증진에 이바지함을 목적으로 한다. 정보통신망 이용 및 보호와 방송통신 관련 국제협력 등에 관한 업무를 체계적 · 종합적으로 수행할 수 있도록 하기 위해 유사 공공기관의 통합을 통해 공통 업무 부문을 축소하고 사업 분야의 지원을 증가시켜 기관 운영의 효율성을 제고하기 위해 2009년 4월 개정되었다.

3.1.1 적용 대상

이 법은 정보통신서비스제공자와 준용사업자를 대상으로 한다. 여기서 정보통신서비스제공자란 초고속인터넷 등 유 · 무선 통신망을 통하거나 컴퓨터 및 컴퓨터 이용기술을 활용하여 정보를 제공 · 매개하는 서비스로써 이동통신서비스, 초고속통신서비스, 인터넷 포탈 사이트, 쇼핑몰 등의 서비스를 말하는데, 이 중 영리 목적으로 정보통신서비스를 제공하는 사업자는 정보통신망법의 적용 대상이 된다. 이러한 정보통신서비스 제공자에는 초고속인터넷사업자, 이동통신사업자, 전화사업자, 웹호스팅 업체, 포털 사이트, 게임 사이트, 쇼핑몰, 경매 · 커뮤니티 · 미니홈피 · 블로그 서비스 제공자 등 인터넷 웹사이트 일반, P2P 사이트 등이 포함된다. 학술 · 종교 · 자선단체 등과 같은 비영리 목적으로 웹사이트를 운영하는 경우에는 정보통신망법의 적용대상에서 제외된다.

또한 정보통신서비스제공자 이외에 오프라인을 통해 개인정보를 수집하는 아래와 같은 사업자를 준용사업자라 하며, 정보통신서비스제공자와 함께 본 법률의 적용 대상에 포함된다.

- 여행 서비스 제공을 위해 고객의 개인정보를 수집하는 여행업자
- 피트니스 센터 등 멤버쉽 클럽 운영 및 객실 예약 · 이용 등을 위해 개인정보를 수집하는 호텔업자
- 마일리지 서비스 제공 등 회원제 서비스를 제공하는 항공운송 사업자, 휴양콘도미니엄업자, 할인점 · 백화점 · 쇼핑센터 운영사업자, 체인사업자
- 수강 신청, 등록을 위해 수강생 정보를 수집하는 학원 또는 교습소 운영자

3.1.2 주요 내용

정보통신망법의 시행규칙 제9조 (개인정보의 보호조치)에 개인정보의 안전성 확보에 필요한 기술적 조치의 하나로 암호화를 규정하고 있다.

정보통신망법 시행규칙

제9조 (개인정보의 보호조치)

① 법 제28조제1항 및 제67조제1항에 따른 개인정보의 안전성 확보에 필요한 관리적 조치는 다음 각 호와 같다.

1. 개인정보의 안전한 취급을 위한 내부관리계획의 수립 및 시행
2. 개인정보 관리책임자의 의무와 책임을 규정한 내부 지침 마련
3. 개인정보의 안전한 보관을 위한 잠금장치 등 물리적 접근방지 조치
4. 개인정보보호를 위한 정기적인 자체 감사 실시
5. 그 밖에 개인정보의 안전성 확보에 필요한 관리적 보호조치

② 법 제28조제1항 및 제67조제1항에 따른 개인정보의 안전성 확보에 필요한 기술적 조치는 다음 각 호와 같다.

1. 개인정보에 대한 접근 권한을 확인하기 위한 식별 및 인증 조치
2. 개인정보에 대한 권한 없는 접근을 차단하기 위한 암호화와 방화벽 설치 등의 조치
3. 접속기록의 위조 · 변조 방지를 위한 조치
4. 침해사고 방지를 위한 보안프로그램의 설치 및 운영
5. 그 밖에 개인정보의 안전성 확보에 필요한 기술적 보호조치

③ 행정안전부장관은 업종별 특성을 반영하여 제1항 및 제2항 각 호에 따른 보호조치의 구체적인 기준을 정하여 고시할 수 있다.

또한 정보통신망법 시행령 제15조제6항에 따라 방송통신위원회는 2009년 4월 개인정보의 기술적 · 관리적 보호조치 기준을 고시하였다. 이 기준의 제6조(개인정보의 암호화)에 "정보통신서비스 제공자등은 주민등록번호, 신용카드번호 및 계좌번호에 대해서는 안전한 암호알고리듬으로 암호화하여 저장한다."라고 규정하고 있다.

개인정보의 기술적 · 관리적 보호조치 기준

제6조(개인정보의 암호화)

① 정보통신서비스 제공자등은 비밀번호 및 바이오정보는 복호화 되지 아니하도록 일방향 암호화하여 저장한다.

② 정보통신서비스 제공자등은 주민등록번호, 신용카드번호 및 계좌번호에 대해서는 안전한 암호알고리듬으로 암호화하여 저장한다.

③ 정보통신서비스 제공자등은 정보통신망을 통해 이용자의 개인정보 및 인증정보를 송 · 수신할 때에는 안전한 보안서버 구축 등의 조치를 통해 이를 암호화해야 한다. 보안서버는 다음 각 호 중 하나의 기능을 갖추어야 한다.

1. 웹서버에 SSL(Secure Socket Layer) 인증서를 설치하여 전송하는 정보를 암호화하여 송 · 수신하는 기능
2. 웹서버에 암호화 응용프로그램을 설치하여 전송하는 정보를 암호화하여 송 · 수신하는 기능

④ 정보통신서비스 제공자등은 이용자의 개인정보를 개인용컴퓨터에 저장할 때에는 이를 암호화해야 한다.

3.1.3 처벌 사항

개인정보 관리적 · 기술적 조치를 제대로 하지 못했을 경우에는 개인정보관리책임자 지정 및 개인정보취급방침을 이용자에게 공개(과태료 2천만원 이하), 개인정보의 분실 · 도난 · 노출 · 변조 · 훼손 방지를 위해 기술적 · 관리적 조치 수행(과태료 3천만원 이하), 기술적 · 관리적 조치를 하지 않아 개인정보의 분실 · 도난 · 노출 · 변조 · 훼손 (2년 이하의 징역 또는 1천만원 이하의 벌금), 직무상 알게 된 개인정보를 훼손 · 침해 · 누설하거나 제공받아서는 안됨(5년 이하 징역 또는 5천만원 이하 벌금) 등의 관련 처벌규정을 받게 된다.

3.2 공공기관의 개인정보보호에 관한 법률

2008년 2월부터 시행된 이 법은 공공기관의 컴퓨터 · 폐쇄회로 텔레비전 등 정보의 처리 또

는 송 · 수신 기능을 가진 장치에 의하여 처리되는 개인정보의 보호를 위하여 그 취급에 관하여 필요한 사항을 정함으로써 공공업무의 적정한 수행을 도모함과 아울러 국민의 권리와 이익을 보호 하는 것을 목적으로 한다.

3.2.1 적용 대상

공공기관이라 함은 국가행정기관 · 지방자치간체 그 밖의 공공단체중 대통령령이 정하는 기관을 말한다. 즉, ① 국가행정기관, ② 지방자치단체, ③「초 · 중등교육법」 그 밖의 다른 법률에 따라 설치된 각급 학교, ④「공공기관의 운영에 관한법률」제4조제1항의 공공기관, ⑤ 특별법에 의하여 설립된 특수법인(다만, ④⑤에는「금융실명거래 및 비밀보장에 관한 법률」 제2조제1호에 따른 금융기관은 포함되지 않음), ⑥「지방공기업법」에 따른 지방공사 및 지방공단이 이 법의 적용 대상이라 할 수 있다.

3.2.2 주요 내용

이 법에는 암호화에 대해 규정하는 조항이 존재하지 않으나, 2008년 3월 행정안전부에서 발표한 “2008년 공공기관 개인정보보호 기본지침”의 "V. 기술 · 시스템적 보호장치 강화" 부분에 “(3) 각급기간은 오프라인으로 처리정보 이용 · 제공시 안정성 확보”에 암호화에 대한 사항을 규정하고 있다.

2008년도 공공기관 개인정보보호 기본지침

V. 기술.시스템적 보호장치 강화
1. 처리정보의 안전성 확보 조치
   (3) 각급기관은 오프라인으로 처리정보 이용.제공시 안정성 확보
      o 보안 USB 등 보안성이 높은 저장매체를 활용하고, 처리정보를 보호할 수 있도록 반드시 암호화
      o 처리정보를 안전하게 이동한 후 그 저장매체의 처리정보가 복구될 수 없도록 파기 조치
2. 개인정보 수집 · 저장 · 전송시 기술적 보호대책 강화
   (1) 개인정보의 안전한 처리를 위해 처리단계별 기술적 보호조치
      o 시스템에 대한 영역별 보안솔루션 도입 등의 조치

구 분	개인정보 유출 방지 솔루션
악성코드 예방 및 대응(PC 단계)	웜 · 바이러스 대책, 스파이웨어 차단 등
사용자 인증(PC 단계)	GPKI 도입, 생체인식 · OTP 도입 등
네트워크 접근통제	방화벽, 침입차단시스템 도입 등

구 분	개인정보 유출 방지 솔루션
서버, DB 접근통제	서버보안솔루션, DB접근제어 도입, 중요 정보 암호화 솔루션 도입 등
웹 어플리케이션	웹 방화벽 도입, 보안서버 도입, 키보드해킹방지 솔루션 도입, 게시판 필터링 솔루션 도입

- 사용자 PC에 웜바이러스 백신, 키보드해킹방지 프로그램 설치 등으로 PC에 보관하고 있는 개인정보 보호
- 네트워크 구간에서의 바이러스 차단, 침입방지 및 침입탐지 등을 통해 해킹 등 불법적인 접근을 차단하여 내부망 보호
- 서버와 DB 영역에 접근통제와 암호화 솔루션을 도입하여 비인가자의 접근 차단과 DB 암호화 등으로 개인정보 유출 방지

3.2.3 처벌 사항

공공기관의 개인정보처리업무를 방해할 목적으로 공공기관에서 처리하고 있는 개인정보를 변경 또는 말소한 자는 10년 이하의 징역에 처해지고 개인정보를 누설 또는 권한 없이 처리하거나 타인의 이용에 제공하는 등 부당한 목적으로 사용한 자는 3년 이하의 징역 또는 1천만원 이하의 벌금에 처한다.

4. 개인정보 암호화 관리

기업 및 기관은 개인정보 암호화 관리를 위해 먼저, 관리하고 있는 개인정보를 식별 가능성, 노출 민감도 등에 따라 분류한 후, 중요도가 높은 개인정보를 선별적으로 암호화하여 관리할 수 있다. 본 장에서는 기업 및 기관에서 개인정보의 중요도에 대한 분류 기준 수립 시 참조할 수 있도록 TTA 표준 및 NIST 가이드에서 제시하고 있는 개인정보 분류 기준을 소개한다. 또한, 중요도가 높은 개인정보 안전하게 관리하기 위해 암호기술을 이용하는 방법을 예로 제시한다.

4.1 개인정보 분류 기준

개인정보는 생존하는 개인을 식별하거나 식별할 수 있는 일체의 정보를 의미하는 것으로 이름, 주민등록번호, 여권번호 등 직접적으로 개인을 식별할 수 있는 정보에서부터 집 주소, 전화번호 등 추가적인 정보와의 연계를 통해 개인을 식별할 수 있는 정보들까지 다양한 형태의 개인정보가 존재한다. TTA 표준 및 NIST 가이드에서는 이렇게 다양한 개인정보를 분류하기 위한 다음의 기준을 제시하고 있다. 다만, 개인정보를 관리하는 기업 및 기관의 사업 환경 등에 따라 기준의 상세 정의 및 포함되는 개인정보의 형태는 달라 질 수 있다.

4.1.1 식별 가능성

개인정보는 단일 정보로 개인을 식별할 수 있는지 여부에 따라 분류될 수 있다. 예를 들어, 이름, 지문, 주민등록번호 등은 단일 정보만으로도 특정 개인에 대한 직접적 식별이 가능하지만 집 주소, 전화번호, 이메일 등은 간접적 개인 식별이 가능하도록 식별을 위한 부분 정보만을 제공하므로 다른 정보와의 연계를 통해 개인 식별이 가능하다. 또한, 직업, 나이, 성별 등은 직접적 식별 가능 정보와 연계되지 않는 경우 개인 식별이 불가능 하다. 다음은 식별 가능성에 따른 개인정보 분류의 예이다. 다만, 개인정보를 관리하는 기업 및 기관의 사업 환경 등에 따라 기준의 상세 정의 및 포함되는 개인정보의 형태는 달라 질 수 있다. 또한, 식별 불가능 개인 정보도 직접적 식별 가능 정보와 연계 시 개인식별에 이용 가능하므로 개인정보의 범위에 포함될 수 있다.

구분	정의 및 예
직접적 식별 가능	단일 정보로 개인에 대한 직접적 식별이 가능한 정보로 이름, 주민번호, 여권번호, 바이오 정보, (개인 식별 가능한) 사진 등이 있음
간접적 식별 가능	단일 정보로는 개인 식별이 불가능하며, 개인 식별을 위한 부분 정보만을 제공하여 다른 개인정보와의 연계를 통해야만 개인 식별이 가능한 정보로 집 주소, 전화번호, 이메일, 생일 등이 있음.
식별 불가능	직접적 식별 가능 개인정보와 연계되는 경우를 제외하고 개인 식별이 불가능한 정보로 직업, 나이, 성별, 종교 등이 있음

4.1.2 노출 민감도

개인정보는 개인에 대한 식별 가능성에 상관 없이 단일 개인정보가 노출 되었을 때 발생할 수 있는 피해에 따른 민감도에 따라 분류될 수 있다. 예를 들어 이름과 주민등록번호는 모두 직접적 개인 식별이 가능한 정보이나, 이름은 주민등록번호에 비해 노출 민감도가 낮다. 또한, 신용카드 번호 등은 이름에 비해 개인 식별의 가능성은 낮지만 노출 민감도는 높다고 할 수 있다. 이와 같이 기밀성 측면에서 정보 노출에 대한 민감도에 따라서도 개인정보를 분류할 수 있다. 다음은 노출 민감도에 따른 개인정보 분류의 예이다.

구분		정의 및 예
단일 정보 노출 민감도	높음	주민등록번호, 여권번호, 신용카드번호 등 (경우에 따라 종교 등도 민감)
	낮음	이름, (개인 식별 가능한) 사진, 이메일 등

또한, 단일 개인정보 각각의 노출 민감도와 별개로 단일 개인정보 간 결합하는 경우 노출 민감도가 증가할 수 있다. 예를 들어, 이름과 신용카드번호가 결합되는 경우가 이메일과 신용카드번호가 결합되는 경우에 비해 노출에 더욱 민감할 수 있다. 다만, 노출 민감도는 다른 분류 기준에 비해 개인 프라이버시와 사용자가 이용하는 서비스 형태, 지역, 문화 등에 따라 상당한 차이가 있을 수 있다.

4.1.3 변경에 따른 영향도

전화번호, 이메일, 집주소 등은 변경에 따른 영향도가 높지 않으나 주민번호, 이름 등 해당 기업 및 기관에서 고객에 대한 색인으로 사용되는 개인정보에 대해서는 해당 정보의 변경에 따른 영향이 클 수 밖에 없다. 정보에 대한 변경은 무결성 측면에서 불법적인 정보의 위 · 변조를 의미하며, 변경에 따른 영향도가 높은 개인정보일수록 안전하게 관리되어야 한다.

4.1.4 저장 및 관리 위치

개인정보의 저장 위치, 관리 방법 등 물리적 · 관리적 환경에 따라 개인정보에 대한 분류가 가능하다. 즉, 개인정보를 저장 · 관리하는 서버가 조직 내부에 있는지, 외부에 있는지에 따라 개인정보에 대한 분류가 가능하다. 역으로 개인정보의 중요도 등에 따라 내부 관리할 것인지 조직 외부에서 관리할 것인지를 판단할 수도 있다. 또한, 내부에서 관리하는 경우에도 내부 직원들에게 공개되는 범위에 따라 개인정보에 대한 분류가 가능하다. 예를 들어, 영업사원은 사내 저장된 고객 개인정보 DB에 접근하여 해당 고객의 연락처 정보 등을 획득할 수 있으나, 고객 주민등록번호 등과 같이 중요 정보에 대해서는 영업부서 관리자만이 접근할 수 있도록 저장 · 관리 되어야 한다.

4.1.5 접근 빈도

개인정보는 해당 정보의 접근빈도, 즉, 활용도에 따라 보호 수준이 다를 수 있다. 예를 들어, 콜센터 직원의 경우, 고객 이름과 전화번호에 대한 접근 빈도는 매우 높은데 반해 고객 이메일, 직업 등의 개인정보에 대한 접근 빈도는 낮을 수 있다. 접근 빈도가 높은 개인정보의 경우, 개인정보 유출 위협도 이에 비례할 수 있으므로 보다 높은 보호 수준이 요구된다.

4.1.6 정책 · 법적 근거

개인정보에 대한 기밀성, 무결성 측면에서 해당 조직 내에서 혹은 정부 등에서 정책적, 법적으로 보호해야 함을 명시하고 있는 개인정보와 그렇지 않은 정보 간에는 보호 수준의 차이가 발생할 수 있다. 즉, 주민등록번호 등과 같이 정부에서 정책적. 법적으로 보호를 명시하는 개인정보는 이메일 주소 등과 같이 관련 법적, 정책적 보호 규정이 없는 정보에 비해 높은 보호 수준이 요구된다.

4.2 개인정보 DB 보안 관리 방안

다음에서는 안전한 개인정보 DB관리를 위해 기본적인 기능별 보안설정 · 관리 방법을 소개하고, 이를 기반으로 DB의 설치 준비 단계에서부터 설치 시, 설치 이후 운영 단계에 이르기까지 각 단계에서 요구되는 보안 관리 방법을 재분류하여 제시한다. 다만, 제시하는 내용은 DB 보안을 위한 일반적인 방법으로 실제 DB에 적용하는 방법은 DB 종류 등에 따라 다를 수 있다.

4.2.1 기능별 DB 보안설정 및 관리

다음에서는 DB 보안을 위한 패스워드 및 계정 관리, DB 이용 및 조작 관련 권한과 DB 접근 권한 등에 대한 관리 방법, 기타 환경설정 방법을 제시한다.

구분	주요 내용
패스워드 관리	o 모든 계정에 패스워드를 설정해야 한다. o 패스워드는 평문으로 저장해서는 안되며, MD5, SHA1 등의 함수를 사용해서 패스워드 해쉬값으로 저장해야 한다 o 기본 설치 시, 디폴트로 설치되는 계정에 대한 패스워드가 디폴트값으로 부여되므로 디폴트 패스워드를 제거해야 한다. o NULL 패스워드를 사용하는 계정은 해당 계정을 삭제하거나 강력한 패스워드를 할당해야 한다.
계정 관리	o Guest 계정을 허용하지 않는다. o 사용하지 않는 계정은 삭제한다. o NULL 패스워드를 사용하는 계정은 삭제한다. ※ 주기적인으로 NULL 패스워드를 사용하는 계정을 점검하여 삭제하거나 강력한 패스워드를 할당해야 한다.
권한 관리	o 시스템 관리자와 DB 관리자를 분리한다. o DB 설치를 위한 기본 계정에 대해서는 설치 및 서비스 실행을 위한 최소한의 권한만을 부여해야 한다. o 사용자 계정 생성 시, 필요 이상의 권한을 부여하지 않는다. o 사용자에게 모든 호스트에 적용될 수 있는 권한은 부여하지 않는다. o 사용자별 역할을 주기적으로 점검하여 역할에 맞는 권한이 부여되었는지 확인하고, 해당 역할과 맞지 않는 경우 권한을 재설정한다. o 하나의 계정에 부여할 수 있는 권한의 수를 제한한다.
접근 관리	o DB에 대한 접근 허용을 위해서는 사용자 인증을 수행한다. 특히, 원격 접속자에 대해서는 강력한 인증을 수행한다. o 불필요한 네트워크 액세스를 제한한다. o 하나의 계정이 접속할 수 있는 수를 제한한다.
감사 관리	o 로그인, 암호변경, 시스템 관리자 업무 등 중요 이벤트에 대한 보안감사를 활성화 한다. o 로그인 감사수준을 사용자 및 이벤트, 서비스 등에 따라 적절히 설정한다. ※ DB관리자(DBA) 등 중요 사용자에 대한 감사수준은 매우 높게 설정
물리적 보안	o DB 서버는 물리적으로 안전한 환경에 설치 및 운영한다. o DB 서버와 인터넷 사이에 방화벽을 설치한다. o DB 설치 시 필요한 옵션만을 선택적으로 설치한다. o DB 설치 후, 이전 설정파일 및 불필요한 정보들은 삭제한다. ※ 기본 설치 시 함께 설치되는 예제 DB 등은 삭제 권고 o 항상 최신의 보안패치와 서비스팩을 설치한다. o 저장될 데이터의 목적, 내용에 따라 DB를 분리 설치 · 운영 한다.
기타	o DB와 응용프로그램 간에 전송되는 중요 데이터는 암호화한다. ※ 송수신 데이터의 암호화를 위해 일부 DBMS에서는 SSL 등의 보안 프로토콜 이용 o 패스워드 변경주기, DB 백업 정책, 보안 점검 주기 등 기타 보안 관련 정책을 사전에 설정하고 정책에 맞게 운영해야 한다.

4.2.2 설치 단계별 DB 보안설정 및 관리

다음에서는 기능별 보안관리 지침을 기반으로 DB 설치 과정에서의 각 단계별 요구되는 보안관리 방안을 재분류했다.

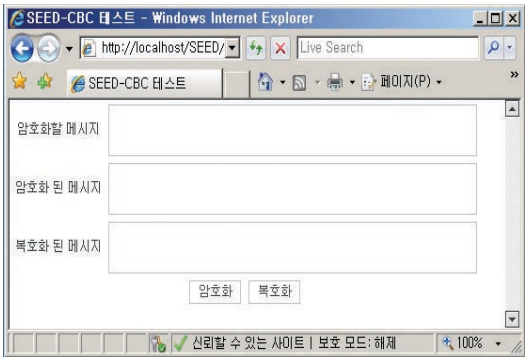
구분	주요 내용
설치 전 환경설정	o 물리적 보안 – DB 서버는 물리적으로 안전한 환경에 설치 및 운영되어야 한다. o DB 분리 – 저장될 데이터의 목적 및 내용에 따라 DB를 분리하여 설치 및 운영하거나, 혹은 도메인을 분리한다. ※ SQL Server의 경우, 개별 서비스를 개별 Windows 계정으로 실행 o 방화벽 설치 – DB 서버와 인터넷 사이에 방화벽을 설치해야 한다. – 다중 계층 환경에서는 여러 개의 방화벽을 사용해 서브넷을 차단해야 한다. o 서비스 계정 – DB 서비스를 설치 및 실행하기 위한 최소한의 권한을 가진 계정을 만든다.
설치 과정	o 선택적 옵션 설치 – DB 설치 시, 필요한 옵션을 선택적으로 설치해야 한다. o 설치를 위한 계정의 권한 최소화 – DB 설치를 위한 기본계정(해당 서버에 대한 계정)은 최소한의 권한으로 DB를 설치해야 한다. o 디폴트 계정 관리 – 기본 설치 시, 디폴트로 설치되는 계정에 대한 패스워드가 디폴트값으로 부여되는 경우가 있는데, 이러한 경우 디폴트 패스워드를 제거해야 한다. o 최신 버전 설치 – 항상 최신의 서비스팩과 보안 패치를 설치한다.
설치 후 환경설정	o 이전 설정 파일 등 삭제 – DB 설치 후, 이전 설정 파일 및 관련 정보들을 삭제해야 한다. – 기본 설치 시, 함께 설치되는 예제 DB 등 DB 서비스 제공에 불필요한 파일 및 서비스는 삭제한다. o 로그인 감사수준 설정 – 로그인 감사 수준을 적절히 설정한다. o 보안감사 활성화 – 로그인 관련 및 암호변경 관련 업무, 시스템 관리자 업무 등 주요 업무에 대한 보안감사를 활성화한다. o 기타 보안관련 설정 – 패스워드 변경 주기, DB 백업 정책, 보안 점검 주기 등 기타 보안관련 정책을 설정해야 한다.
운영 과정	o 패스워드 보안 – 모든 계정에 패스워드를 설정해야 하며, 디폴트 패스워드는 제거해야 한다. – 패스워드는 평문으로 저장해서는 안되며, MD5, SHA1 등의 함수를 사용해서 패스워드 해쉬값으로 저장하기를 권고한다. – NULL 패스워드를 사용하는 계정에 대해서는 해당 계정을 삭제하거나 강력한 패스워드를 할당해야 한다. o 최소권한 부여 – 사용자 계정 생성 시, 필요이상의 권한을 부여하지 않는다. – 사용자에게 모든 호스트에 대한 권한을 부여하지 않는다. 즉, 사용자별로 접근이 필요한 호스트에 대해서만 권한을 부여한다. – 일반 사용자에게 SUPER 등의 강력한 권한을 부여하지 않아야 한다. – 하나의 계정이 접속할 수 있는 수/권한의 수를 제한한다. 특히, 관리자 권한으로 접속하는 수를 줄어야 한다. – 사용자별 역할을 주기적으로 점검하여 권한이 역할의 범위에 맞게 할당 되었는지를 확인하고, 해당 역할과 맞지 않는 경우 권한을 재설정한다.

구분	주요 내용
운영 과정	o 관리자 분리 및 최소화 – 서버 관리자와 DB 관리자를 분리한다 – 보다 많은 권한을 갖는 DB 관리자의 수를 최소화해야 한다. o 효과적인 접근제어 – 사용자를 적절히 인증해야 한다. 특히, 원격 접속에 대한 제어를 해야 한다. – 호스트에 접근할 수 있는 최대 접근허가 수를 가능한 최소화해야 한다. o 계정관리 – Guest 계정을 사용하지 않으며, 사용하지 않는 계정은 삭제한다. – NULLL 암호를 사용하는 계정을 주기적으로 스캔하여 삭제하거나 강력한 패스워드를 할당한다. o 암호화 및 기타 – SSL 등을 이용하여 DB와 응용프로그램 간 송수신되는 주요 데이터를 암호화해야 한다. – 백업 정책, 보안 점검 주기 등 기타 보안관련 정책을 설정해야 한다.

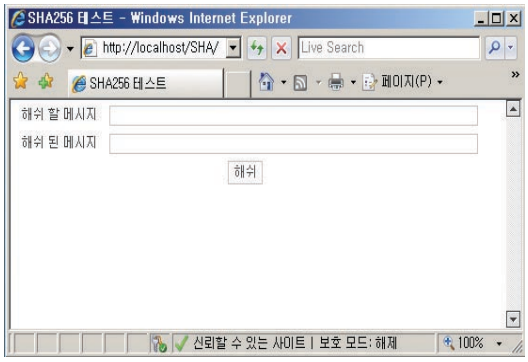
4.3 개인정보 암호화(예제)

다음에서는 국내에서 널리 활용되고 있는 국산 암호 알고리즘 SEED와 일방향 해쉬함수 SHA-256을 이용해 개인정보를 암호화하는 방법을 예제로 소개한다.

〈그림 1〉, 〈그림 2〉는 웹페이지 상에서 평문과 암호문을 입력받아 각각 SEED, SHA-256 알고리즘을 이용하여 입력 데이터를 암·복호화 및 해쉬한 결과를 보여주는 예제이다. 특히, SEED의 경우, 블록암호 운영모드 중 가장 널리 사용되고 있는 CBC 운영모드를 사용하고 있다.



〈그림 1〉 SEED-CBC 이용 예제 웹페이지



〈그림 2〉 SHA-256 이용 예제 웹페이지

다음에서 제시하는 예제는 암·복호화 및 해쉬한 결과를 직접적으로 보여주기 위해 결과를 웹페이지에 출력하는 형태로 구성하였으나, 실제 개인정보를 암호화 또는 해쉬하여 DB에 저장하기 위해서는 웹페이지 출력 명령어 부분을 DB에 저장하는 명령어로 변경해 주면 된다. 또한

암호화한 개인정보를 복호화하기 위해서는 암호문을 입력받는 명령어를 DB에서 해당 암호문을 호출하는 명령어로 변경해주면 된다.

또한, 본 예제에서 이용하는 암호 라이브러리(SEEDCBC.dll, 는 여기서 사용되는 암호 라이브러리(seedcbc.dll, seedcbc.class 등)는 KISA에서 개발한 공개용 라이브러리를 활용하고 있다.

4.3.1 SEED를 이용한 개인정보 DB 암호화 (CBC 운영모드)

4.3.1.1 SEED 및 CBC 운영모드

SEED(TTAS.KO-12.0004/R1)는 1999년 2월 한국인터넷진흥원을 중심으로 국내 암호 전문가들이 참여하여 순수 국내기술로 개발한 블록암호 알고리즘이다. SEED는 128비트의 비밀키를 사용하여 128비트 평문을 128비트 암호문으로 암호화하는 128비트 블록암호 알고리즘이다.

임의의 길이를 갖는 평문을 SEED로 암호화하기 위해서는 평문을 128비트 블록단위로 처리해주는 운영모드와 평문을 128비트의 배수로 만들기 위한 덧붙이기 방법을 사용하여야 한다. 본 예시에서는 SEED의 운영모드(TTAS.KO-12.0025)에 명시되어 있는 CBC 운영모드와 덧붙이기 방법 2를 사용한다.

4.3.1.2 SEED-CBC 암 · 복호화 예

다음은 SEED-CBC 암 · 복호화 소스이다.

o index.html

다음은 SEED-CBC 암 · 복호화 예시에 대한 index.html 소스이다. 본 소스에서는 ASP를 예시로 들고 있기 때문에, 암 · 복호화 버튼 클릭 시 실행되는 웹페이지가 SeedEncryption.asp, SeedDecryption.asp이다. JSP의 경우 이 부분을 SeedEncryption.jsp, SeedDecryption.jsp로 변경하여 사용한다.

```
<html>
<head>
   <title>SEED-CBC 테스트</title>
   <link rel="stylesheet" href="style.css" type="text/css">
   <meta http-equiv="content-type" content="text/html; charset=euc-kr" />
   <script src="./ajax.js" language="JavaScript"></script>
   <script language="JavaScript">
      function SeedCBC( URL, FORM, ACT )
      {
         if( ACT == "enc" )
                  FORM.oCipherText.value = ( GetHttpRequest( URL, "iPlainText=" +
encodeURIComponent(FORM.iPlainText.value)) ).trim();
         else if( ACT == "dec" )
                  FORM.oPlainText.value = ( GetHttpRequest( URL, "oCipherText=" +
encodeURIComponent(FORM.oCipherText.value)) ).trim();
      }
   </script>
</head>
<body>
<form name="seed_form">
   <table border="0" width="500" celpadding="5">
      <tr>
         <td width="100">암호화할 메시지</td>
         <td width="*">
            <textarea name="iPlainText" style="width:100%;" rows="3"></textarea>
         </td>
      </tr>
      <tr>
         <td>암호화 된 메시지</td>
         <td>
            <textarea name="oCipherText" style="width:100%;" rows="3" readonly="readonly"></textarea>
         </td>
      </tr>
      <tr>
         <td>복호화 된 메시지</td>
         <td>
            <textarea name="oPlainText" style="width:100%;" rows="3" readonly="readonly"></textarea>
         </td>
      </tr>
      <tr>
         <td colspan="2" align="center">
                  <input type="button" value="암호화" name="enc" onclick="JavaScript:SeedCBC(
'SeedEncryption.asp', document.seed_form, 'enc');" /> 
      ☞ JSP의 경우 SeedEncryption.asp를 SeedEncryption.jsp로 변경하여 사용
                  <input type="button" value="복호화" name="dec" onclick="JavaScript:SeedCBC(
'SeedDecryption.asp', document.seed_form, 'dec');" />
      ☞ JSP의 경우 SeedDecryption.asp를 SeedDecryption.jsp로 변경하여 사용
         </td>
      </tr>
   </table>
</form>
</body>
</html>
```

4.3.1.3 ASP 구현 예

다음은 ASP에서 SEED-CBC를 이용해 입력받은 개인정보를 암호화하기 위한 방법이다.

o seedcbc.dll의 레지스트리 등록

- seedcbc.dll 파일을 "%windir%\system32" 디렉토리에 저장
- SEED-CBC 모듈을 레지스트리에 등록하는 명령어(RegSvr32)

```
regsvr32.exe %windir%\system32\sha256.dll ☞ sha256.dll을 레지스트리에 등록
```

o config.asp

SEED-CBC 암·복호화에 사용할 128비트 초기값과 128비트 비밀키를 저장한 config 페이지의 ASP 소스이다. **초기값과 비밀키는 외부에 노출되면 안 되므로 별도의 키 저장 공간 또는 안전한 위치에 저장해야 한다.**

```
<%
   IVEC = "0000000000000000"
   ☞ 128비트 초기값
   ☞ DB 적용 시 : 외부에 노출되지 않도록 별도의 키 저장 공간 또는 안전한 위치에 저장
   MK = "1234567890123456"
   ☞ 128비트 비밀키
   ☞ DB 적용 시 : 외부에 노출되지 않도록 별도의 키 저장 공간 또는 안전한 위치에 저장
%>
```

o SeedEncryption.asp

SEED-CBC 암호화를 수행하는 SeedEncryption 페이지의 ASP 소스이다. 암호화 수행의 명령어는 오브젝트.Encrypt로 평문, 비밀키, 초기값을 입력으로 갖는다. DB 적용 시에는 Response.Write 명령어 대신 암호화된 결과값을 DB에 저장하도록 명령어를 수정하여 사용한다.

```
<%@ codepage="65001" language="VBScript" %>
<!--#include file='./config.asp'-->
<%
   '사용자키(MK)와 초기값 벡터(IVEC) include
   ' 유니코드로 파라미터가 전송되므로 값을 코드페이지를 유니코드로 설정
   ' 65001 : 유니코드 (UTF-8)
   sPlainText = Trim( Request("iPlainText") ) ☞ 웹페이지에서 평문을 가져오는 명령어
   set oSeed = Server.CreateObject( "seed.CBC" ) ☞ SEEDCBC.dll에 정의된 오브젝트 생성
   Response.Write( oSeed.Encrypt(sPlainText, MK, IVEC) )
   ☞ 웹페이지에 암호문을 출력하는 명령어
```

```
   ☞ DB 적용 시 : oSeed.Encrypt(sPlainText, MK, IVEC)의 결과를 DB에 저장하는 명령어로 수정
   ☞  ex) DBconn.Execute( "INSERT INTO 저장할 테이블(필드명) values ( '" &
          oSeed.Encrypt(sPlainText, MK, IVEC) & "' )" )
   set oSeed = nothing
%>
```

o SeedDecryption.asp

SEED-CBC 복호화를 수행하는 SeedEncryption 페이지의 ASP 소스이다. 복호화 수행의 명령어는 오브젝트.Decrypt로 암호문, 비밀키, 초기값을 입력으로 한다. DB 적용 시에는 Request 명령어 대신 암호문을 DB에서 불러오는 명령어로 수정하여 사용한다.

```
<%@ codepage="65001" language="VBScript" %>
<!--#include file='./config.asp'-->
<%
   '사용자키(MK)와 초기값 벡터(IVEC) include
   ' 유니코드로 파라미터가 전송되므로 값을 코드페이지를 유니코드로 설정
   ' 65001 : 유니코드 (UTF-8)
   sCipherText = Trim( Request("oCipherText") )
   ☞ 웹페이지에서 암호문을 가져오는 명령어
   ☞ DB 적용 시 : Trim( Request("oCipherText") ) 대신 DB에서 암호문을 호출하는 명령어로 수정
   ☞  ex) Set rs = DBconn.Execute( "SELECT 암호문 필드 FROM 해당테이블 [WHERE 조건문]"
         )sCipherText = rs( "암호문 필드" )
   set oSeed = Server.CreateObject( "seed.CBC" )  ☞ SEEDCBC.dll에 정의된 오브젝트 생성
   Response.Write( oSeed.Decrypt(sCipherText, MK, IVEC) ) ☞ 웹페이지에 평문을 출력하는 명령어
   set oSeed = nothing
%>
```

4.3.1.3 JSP 구현 예

다음은 JSP에서 SEED-CBC를 이용해 입력받은 개인정보를 암호화하기 위한 방법이다.

o SEEDCBC.class의 등록

- JSP에서 불러올 class 파일들이 위치하는 폴더에 "KISA" 폴더를 생성하고, SEEDCBC.class를 저장

```
c:\webapps\ROOT\WEB-INF\classes ☞ class 파일들이 위치하는 폴더
c:\webapps\ROOT\WEB-INF\classes\KISA ☞ KISA 폴더 생성 후 SEEDCBC.class 파일을 저장
```

o config.jsp

SEED-CBC 암·복호화에 사용할 128비트 초기값과 128비트 비밀키를 저장한 config 페이지의 JSP 소스이다. **초기값과 비밀키는 외부에 노출되면 안되기 때문에 별도의 키 저장 공간 또는 안전한 위치에 저장하여야 한다.**

```
<%
   IVEC = "0000000000000000"
   ☞ 128비트 초기값
   ☞ DB 적용 시 : 외부에 노출되지 않도록 별도의 키 저장 공간 또는 안전한 위치에 저장
   MK = "1234567890123456"
   ☞ 128비트 비밀키
   ☞ DB 적용 시 : 외부에 노출되지 않도록 별도의 키 저장 공간 또는 안전한 위치에 저장
%>
```

o SeedEncryption.jsp

SEED-CBC 암호화를 수행하는 SeedEncryption 페이지의 JSP 소스이다. 암호화 수행의 명령어는 오브젝트.Encryption으로 평문, 비밀키, 초기값을 입력으로 한다. DB 적용 시에는 out.print 명령어 대신 암호화된 결과값을 DB에 저장하도록 명령어를 수정하여 사용한다.

```
<%@page language="java" contentType="text/html;charset=EUC-KR" pageEncoding="EUC-KR"%>
<%@page import="java.io.*"%>
<%@page import="KISA.SeedCBC"%>
<%@include file="config.jsp"%>
<%
   SeedCBC s = new SeedCBC(); ☞ SEEDCBC.class에 정의된 오브젝트 생성
   String sPlainText = request.getParameter( "iPlainText" ); ☞ 웹페이지에서 평문을 가져오는 명령어
   out.print( s.Encryption(sPlainText.getBytes(), Key.getBytes(), IVEC.getBytes()) );
   ☞ 웹페이지에 암호문을 출력하는 명령어
    ☞ DB 적용 시 : s.Encryption(sPlainText.getBytes(), Key.getBytes(), IVEC.getBytes())의 결과를
       DB에 저장하는 명령어로 수정
     ☞ ex) DBconn.Execute( "INSERT INTO 저장할 테이블(필드명) values ( '" &
          s.Encryption(sPlainText.getBytes(), Key.getBytes(), IVEC.getBytes()) & "' )" )
%>
```

o SeedDecryption.jsp

SEED-CBC 복호화를 수행하는 SeedEncryption 페이지의 JSP 소스이다. 복호화 수행의 명령어는 오브젝트.Decryption으로 암호문, 비밀키, 초기값을 입력으로 한다. DB 적용 시에는 request.getParameter 명령어 대신 암호문을 DB에서 불러오는 명령어로 수정하여 사용한다.

```
<%@page language="java" contentType="text/html;charset=EUC-KR" pageEncoding="EUC-KR"%>
<%@page import="java.io.*"%>
<%@page import="KISA.SeedCBC"%>
<%@include file="config.jsp"%>
<%
   String sCipherText = request.getParameter( "oCipherText" ).trim();
   ☞ 웹페이지에서 암호문을 가져오는 명령어
   ☞ DB 적용 시 : request.getParameter( "oCipherText" ).trim() 대신 DB에서 암호문 호출 명령어로 수정
```

```
    ☞ ex) Set rs = DBconn.Execute( "SELECT 암호문 필드 FROM 해당테이블 [WHERE 조건문]"
          )sCipherText = rs( "암호문 필드" )
  SeedCBC s = new SeedCBC(); ? SEEDCBC.class에 정의된 오브젝트 생성
  out.print( s.Decryption(sCipherText, Key.getBytes(), IVEC.getBytes()) );
  ☞ 웹페이지에 평문을 출력하는 명령어
%>
```

4.3.2 SHA-256을 이용한 개인정보 DB 암호화

4.3.2.1 해쉬함수 SHA-256

SHA-256은 NIST에서 개발한 미국 표준 해쉬함수(FIPS 180-3) 중 하나이다. SHA-256은 임의의 길이를 가지는 입력 메시지를 512비트 블록 단위로 처리하여 256비트의 해쉬한 값을 출력한다.

SHA-256도 임의의 길이를 갖는 입력 메시지를 512비트의 배수로 만들기 위한 덧붙이기를 방법을 사용하는데, 해당 덧붙이기 방법은 FIPS 180-3에 명시되어 있다.

4.3.2.2 SHA-256 해쉬 예

다음은 SHA-256을 이용해 입력된 개인정보를 해쉬하는 소스이다.

o index.html

다음은 SHA-256 해쉬 예시에 대한 index.html 소스이다. 본 소스에서는 ASP를 예시로 들고 있기 때문에, 해쉬 버튼 클릭 시 실행되는 웹페이지가 sha256hash.asp이다. JSP의 경우 이 부분을 sha256hash.jsp로 변경하여 사용한다.

```
<html>
<head>
  <title>SHA256 테스트</title>
  <meta http-equiv="content-type" content="text/html; charset=euc-kr" />
  <link rel="stylesheet" href="style.css" type="text/css">
  <script src="ajax.js" language="JavaScript"></script>
  <script language="JavaScript">
    function sha256( URL )
    {
      var form = document.getElementById( "sha256_form" );
```

```
                form.oCipherText.value = (GetHttpRequest( URL, "iPlainText=" +
encodeURIComponent(form.iPlainText.value))).trim();
      }
   </script>
</head>
<body>
<form name="sha256_form">
   <table border="0" width="500" celpadding="5">
      <tr>
         <td width="100">해쉬 할 메시지</td>
         <td width="*"><input type="text" name="iPlainText" style="width:100%;" /></td>
      </tr>
      <tr>
         <td>해쉬 된 메시지</td>
            <td><input type="text" name="oCipherText" style="width:100%;" readonly="readonly"
/></td>
      </tr>
      <tr>
                  <td colspan="2"><input type="button" value="해 쉬 " name="enc"
onclick="JavaScript:sha256('sha256hash.asp');" /></td>
   ? JSP의 경우 sha256hash.asp를 sha256hash.jsp로 변경하여 사용
      </tr>
   </table>
</form>
</body>
</html>
```

4.3.2.3 ASP 구현 예

o sha256.dll의 레지스트리에 등록

- sha256.dll 파일을 "%windir%\system32" 디렉토리에 저장
- SHA-256 모듈을 레지스트리에 등록하는 명령어(RegSvr32)

```
regsvr32.exe %windir%\system32\sha256.dll ☞ sha256.dll을 레지스트리에 등록
```

o sha256hash.asp

SHA-256 해쉬를 수행하는 sha256hash 페이지의 ASP 소스이다. 해쉬 수행의 명령어는 오브젝트.hash로 입력 메시지를 입력으로 한다. DB 적용 시에는 Response.Write 명령어 대신 해쉬된 값을 DB에 저장하도록 명령어를 수정하여 사용한다.

```
<%@ codepage="65001" language="VBScript" %>
<%
   ' 유니코드로 파라미터가 전송되므로 값을 코드페이지를 유니코드로 설정
   ' 65001 : 유니코드 (UTF-8)
   sPlainText = Trim( Request("iPlainText") ) ☞ 웹페이지에서 메시지를 가져오는 명령어
   set oSha = Server.CreateObject( "SHA.sha256" ) ☞ SHA256.dll에 정의된 오브젝트 생성
   Response.Write( oSha.hash(sPlainText) )
   ☞ 웹페이지에 해쉬한 값을 출력하는 명령어
   ☞ DB 저장 : oSha.hash(sPlainText)의 결과를 DB에 저장하는 명령어로 수정
      ☞ ex) DBconn.Execute( "INSERT INTO 저장할 테이블(필드명) values ( '" &
oSha.hash(sPlainText) & "' )" )
   set oSha = nothing
%>
```

4.3.2.4 JSP 구현 예

o sha256.class의 등록

- JSP에서 불러올 class 파일들이 위치하는 폴더에 "KISA" 폴더를 생성하고, sha256.class를 저장

```
c:\webapps\ROOT\WEB-INF\classes ☞ class 파일들이 위치하는 폴더
c:\webapps\ROOT\WEB-INF\classes\KISA ☞ KISA 폴더 생성 후 SHA256.class 파일을 저장
```

o sha256hash.jsp

SHA-256 해쉬를 수행하는 sha256hash 페이지의 JSP 소스이다. 해쉬 수행의 명령어는 오브젝트.hash로 입력 메시지를 입력으로 한다. DB 적용 시에는 out.print 명령어 대신 해쉬된 값을 DB에 저장하도록 명령어를 수정하여 사용한다.

```
<%@page language="java" contentType="text/html;charset=EUC-KR"  pageEncoding="EUC-
KR"%>
<%@page import="java.io.*"%>
<%@page import="KISA.SHA256"%>
<%
   String sPlainText = request.getParameter( "iPlainText" ).trim();
   ☞ 웹페이지에서 메시지를 가져오는 명령어
   SHA256 s = new SHA256(); ☞ SHA256.class에 정의된 오브젝트 생성
   out.print( s.hash(sPlainText.getBytes()) );
   ☞ 웹페이지에 해쉬한 값을 출력하는 명령어
   ☞ DB 저장 : s.hash(sPlainText.getBytes())의 결과를 DB에 저장하는 명령어로 수정
   ☞  ex) DBconn.Execute( "INSERT INTO 저장할 테이블(필드명) values ( '" &
         s.hash(sPlainText.getBytes()) & "' )" )
%>
```

4.3.3 기타 사항

4.3.3.1 dll 및 class 파일의 구성

o SEEDCBC.dll의 암호화 인터페이스 함수

STDMETHODIMP CCBC::Encrypt(BSTR PlainText, BSTR MK, BSTR IVEC, BSTR *retCipherText)
- 인터페이스 함수명 : Encrypt
- 입력 : [문자열] BSTR PlainText – 평문
  [문자열] BSTR MK – 128비트 마스터 키
  [문자열] BSTR IVEC – CBC모드를 위한 128비트 초기값
- 출력 : [문자열] BSTR *retCipherText – 암호문 (문자열로 반환하기 위해 Base64로 인코딩됨)

o SEEDCBC.dll의 복호화 인터페이스 함수

STDMETHODIMP CCBC::Decrypt(BSTR CipherText, BSTR MK, BSTR IVEC, BSTR *retPlainText)
- 인터페이스 함수명 : Decrypt
- 입력 : [문자열] BSTR CipherText – 암호문 (Base64로 인코딩된 문자열)
  [문자열] BSTR MK – 128비트 마스터 키
  [문자열] BSTR IVEC – CBC모드를 위한 128비트 초기값
- 출력 : [문자열] BSTR *retPlainText – 평문

o SEEDCBC.class의 암호화 인터페이스 함수

public String Encryption(byte[] bPlainText, byte[] bKey, byte[] bIVEC)
- 인터페이스 함수명 : Encryption
- 입력 : [1BYTE 배열] byte[] bPlainText – 평문
  [1BYTE 배열] byte[] bKey – 128비트 마스터 키
  [1BYTE 배열] byte[] bIVEC – CBC모드를 위한 128비트 초기값
- 출력 : [문자열] String – 암호문 (문자열로 반환하기 위해 Base64로 인코딩됨)

o SEEDCBC.class의 복호화 인터페이스 함수

public String Decryption(String sCipherText, byte[] bKey, byte[] bIVEC)
- 인터페이스 함수명 : Decryption
- 입력 : [1BYTE 배열] byte[] bPlainText – 평문
  [1BYTE 배열] byte[] bKey – 128비트 마스터 키
  [1BYTE 배열] byte[] bIVEC – CBC모드를 위한 128비트 초기값
- 출력 : [문자열] String – 평문

o sha256.dll의 해쉬 인터페이스 함수

STDMETHODIMP Csha256::hash(BSTR sMessage, BSTR *retVal)
- 인터페이스 함수명 : hash
- 입력 : [문자열] BSTR sMessage - 메시지
- 출력 : [문자열] BSTR *retVal - 해쉬한 값 (문자열로 반환하기 위해 Base64로 인코딩됨)

o sha256.class의 해쉬 인터페이스 함수

public String hash(byte[] bpMessage)
- 인터페이스 함수명 : hash
- 입력 : [1Byte 배열] byte[] bpMessage - 메시지
- 출력 : [문자열] String - 해쉬한 값 (문자열로 반환하기 위해 Base64로 인코딩됨)

4.3.3.2 예시를 위한 공통 파일

o style.css

```
BODY
{
    FONT-FAMILY:  "굴림", "Tahoma", "Verdana", "MS Sans Serif", "Courier New";
    margin-top: 0px; margin-left: 0px; margin-right: 0px;
    /* border : Blue dotted f7f7f7;*/
    page-break-after : auto;
    page-break-before : auto;
    scrollbar-3dlight-color:#424142;
    scrollbar-arrow-color:#848284;
    scrollbar-base-color:#DEDFDE;
    scrollbar-darkshadow-color:white;
    scrollbar-face-color:#DEDFDE;
    scrollbar-highlight-color:white;
    scrollbar-shadow-color:#424142;
}
INPUT
{
    BORDER-RIGHT: #C2C2C2 1pt solid;
    BORDER-TOP: #C2C2C2 1pt solid;
    FONT-SIZE: 9pt;
    BORDER-LEFT: #C2C2C2 1pt solid;
    COLOR: #4F4E4E;
    BORDER-BOTTOM: #C2C2C2 1pt solid;
    FONT-FAMILY: 굴림, Arial;
    background-color: #FFFFFF;
    PADDING-RIGHT: 0px;
    MARGIN-TOP: 1pt;
    PADDING-LEFT: 2px;
    PADDING-TOP: 2px;
}
```

```
TEXTAREA
{
    BORDER-RIGHT: #C2C2C2 1px solid;
     BORDER-TOP: #C2C2C2 1px solid;
     FONT-SIZE: 9pt;
     OVERFLOW: auto;
     BORDER-LEFT: #C2C2C2 1px solid;
     COLOR: #4F4E4E;
     LINE-HEIGHT: 17px;
     BORDER-BOTTOM: #C2C2C2 1px solid;
     FONT-FAMILY: 굴림, Arial;
     BACKGROUND-COLOR: #FFFFFF;
}
TABLE
{
    BORDER : 0px solid #000000;
    TABLE-LAYOUT : fixed;
}
TD
{
    font-size: 12px;
    color: #3E3E3E;
    LINE-HEIGHT: 17px;
    TEXT-ALIGN: center;
}
```

o ajax.js

```
function GetHttpRequest( URL, Param )
{
    var xmlhttp = null;
    // FF일 경우 window.XMLHttpRequest 객체가 존재한다.
    if( window.XMLHttpRequest )
        // FF 로 객체선언
        xmlhttp = new XMLHttpRequest();
    else
        // IE 경우 객체선언
        xmlhttp = new ActiveXObject( "Microsoft.XMLHTTP" );

    // 값을 가져 왔을경우 호출할 메소드를 바로 선언
    xmlhttp.onreadystatechange = function()
    {
        // readyState 가 4 고 status 가 200 일 경우 올바로 가져옴
        if( xmlhttp.readyState == 4 && xmlhttp.status == 200 )
            // responseText 에 값을 저장
            var responseText = xmlhttp.responseText;
    }

    // POST 모드로 HTTP 요청을 전송함
    xmlhttp.open( "POST", URL, false );
    xmlhttp.setRequestHeader( "Content-Type", "application/x-www-form-urlencoded" );
    xmlhttp.send( Param );
    responseText = xmlhttp.responseText;
```

```
    // 가져온 xmlhttp 객체의 responseText 값을 반환
    return responseText;
}
String.prototype.trim = function()
{
    return this.replace( /(^\s*)|(\s*$)/gi, "" );
}
```

참고문헌

[1] California Senate Bill 1386, http://download.pgp.com/pdfs/regulations/SB_1386_Compliance_Brief-080618.pdf, 2008. 5

[2] http://www.dbguide.net/blog/post/post_view.jsp?urlid=webemotion&cnum=46440&pnum=13225

[3] Christian Kirsch, The role of encryption in database security, http:// www.net-security.org/article.php?id=1232

[4] NIST SP 800-122, "Guide to Protecting the Confidentiality of Personally Identifiable Information (PII) (Draft)", 2009. 1

개인정보DB 암호화 관리 안내서

2009년 12월 인쇄
2009년 12월 발행

발행인 : 김 희 정
발행처 : 한국인터넷진흥원
서울시 송파구 가락동 79-3 대동빌딩
TEL : 02-4055-114
인쇄처 : T&C
TEL : 02-2273-2607

- 본 안내서는 지속적으로 보완되어 홈페이지(http://seed.kisa.or.kr)에 게시될 예정이오니 참고바랍니다.
- 내용 중 오류 및 기타 의견이 있을 경우, 홈페이지(http://seed.kisa.or.kr) 또는 이메일(kisa@kisa.or.kr)로 문의하여 주시기 바랍니다.

국번없이

☎118

- 해킹·바이러스침해
- 개인정보침해
- 불법스팸신고

방송통신위원회

110-777 서울 종로구 세종로 20 방송통신위원회
대표전화 02-750-1114 | www.kcc.go.kr

한국인터넷진흥원
Korea Internet & Security Agency

138-950 서울 송파구 가락동 79-3 대동빌딩
대표전화 02-405-5014 | www.kisa.or.kr