국내 암호이용 현황 및 암호구현 가이드

2011. 10. 12

전인경
(ikjeun@kisa.or.kr)

KISA 한국인터넷진흥원
Korea Internet & Security Agency

개발배경 및 목적

- 최근 개인정보 유출 사고 발생, 개인정보보호법 시행 등으로 **업체의 암호 기술 구현 필요성 증가**
  - 관련 규정, 지침 등에서 안전한 암호기술을 사용토록 규정하나,
  - **안전한 암호기술에 대한 구체적인 구현 방법이 없음**
- **암호 관련 가이드의 다양성 및 제한된 독자**
  - 사용자·관리자 중심의 암호 가이드
  - 암호이용 안내서, 암호정책 수립기준 설명서, 암호 알고리즘 및 키 길이 이용 안내서, 개인정보 DB암호화 관리 안내서, 패스워드 선택 및 이용 가이드 등
- 업체의 암호기술 구현 오류 예
  - 암호에 대한 잘못된 이해(치환≠?암호)
  - 취약성이 발표된 암호기술 사용
  - 암호키의 무분별한 복제·저장

목차

Ⅰ 암호적용 대상 및 기술

암호 적용 대상 및 기술

암호화가 필요한 정보의 종류

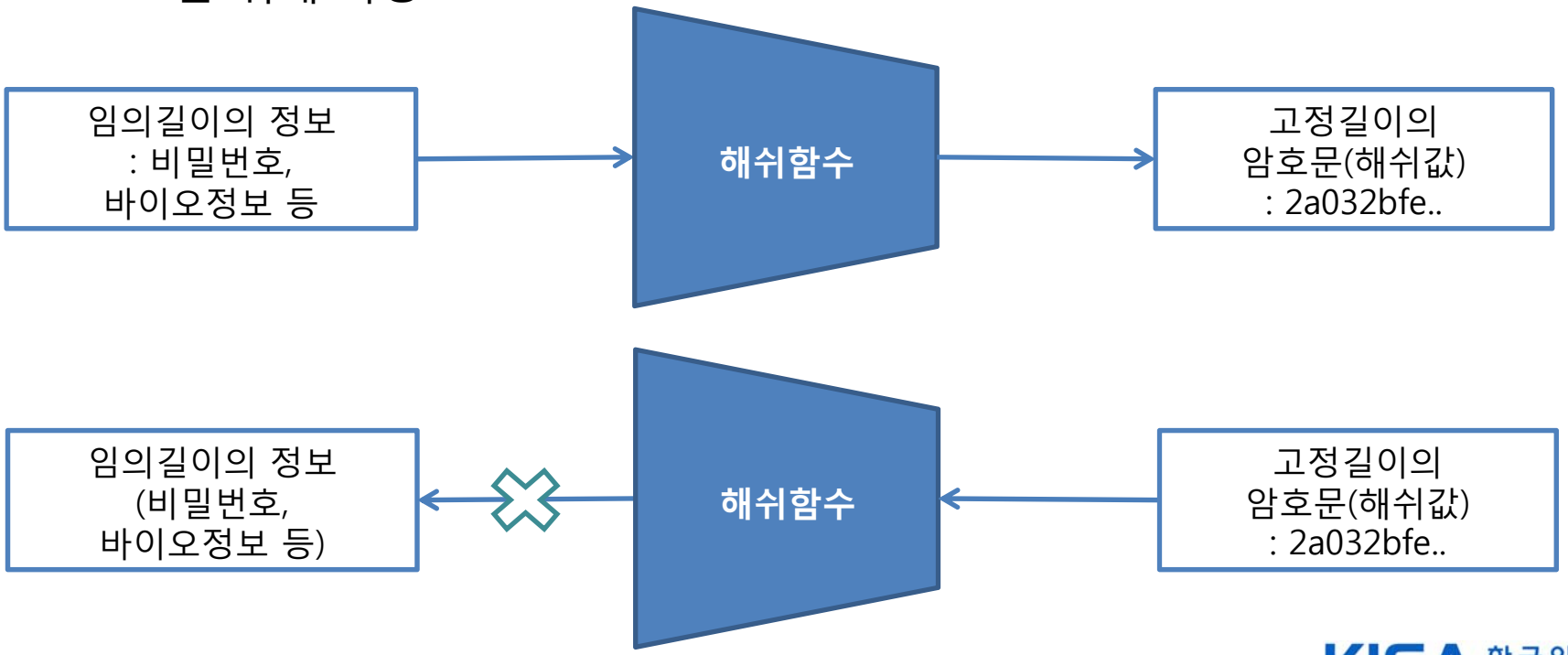
정보구분	정보통신망법	개인정보보호법	적용암호
비밀번호	○	○	해쉬함수
바이오정보	○	○	해쉬함수/블록암호
주민등록번호	○	○	블록암호
신용카드번호	○		블록암호
계좌번호	○		블록암호
여권번호		○	블록암호
운전면허번호		○	블록암호
외국인등록번호		○	블록암호

관련 법령

- 개인정보의 기술적 관리적 보호조치 기준 제6조 개인정보의 암호화
- 개인정보보호법 제24조 고유식별정보의 처리 제한

개인정보보호 저장시 활용되는 암호 기술

- 일방향 해쉬 함수
  - 임의길이의 정보들을 입력하여 고정된 길이의 암호문(해쉬값)을 출력하는 암호 기술
  - 입력을 가지고 암호문(해쉬값)을 생성해 낼 수는 있지만, 암호문을 가지고 입력값을 알아낼 수 없음
  - 사용자(정보소유자)를 제외하고 정보를 다루는 관리자조차 암호화된 정보의 원래 정보가 무엇인지 알 수 없어야 하는 정보(ex. 비밀번호, 바이오정보 등) 보호를 위해 사용



개인정보보호 저장시 활용되는 암호 기술

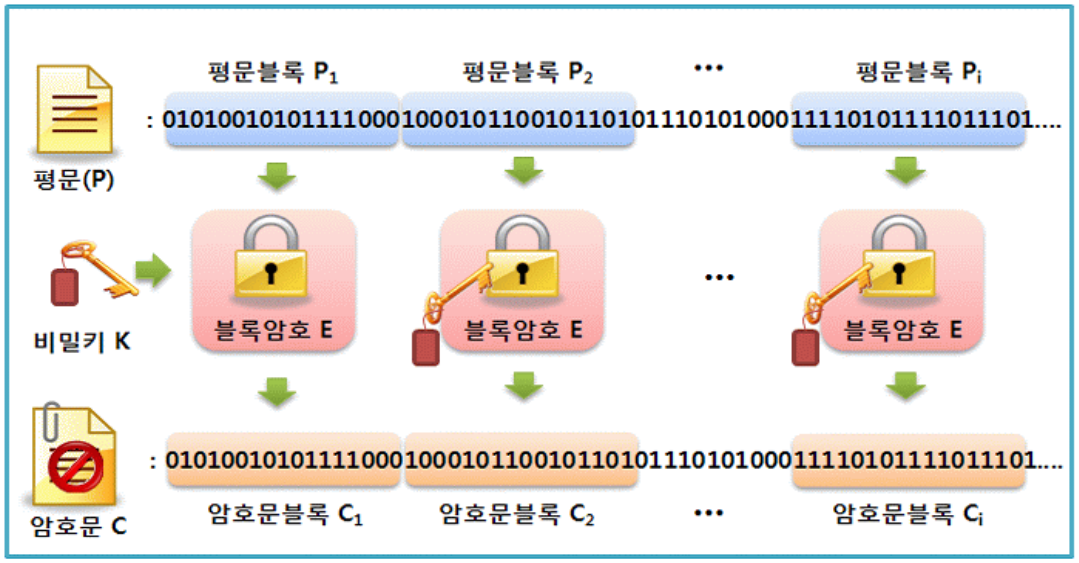
- 비밀번호 등의 단순 암호화용 일방향 해쉬함수 사용 권고 기간

보안강도	일방향 해쉬함수	사용 권고 기간
80비트	HAS-160, SHA-1	~2010년 까지
112비트	SHA-224	~2030년 까지
128비트	SHA-256	2030년 이후
192비트	SHA-384	
256비트	SHA-512	

* SHA-1은 안전성이 80비트 보안강도 이하를 제공하여 새로운 어플리케이션에 적용하는 것을 권장하지 않으나, 현재 광범위하게 사용되므로 해쉬함수 표에 추가

개인정보보호 저장시 활용되는 암호 기술

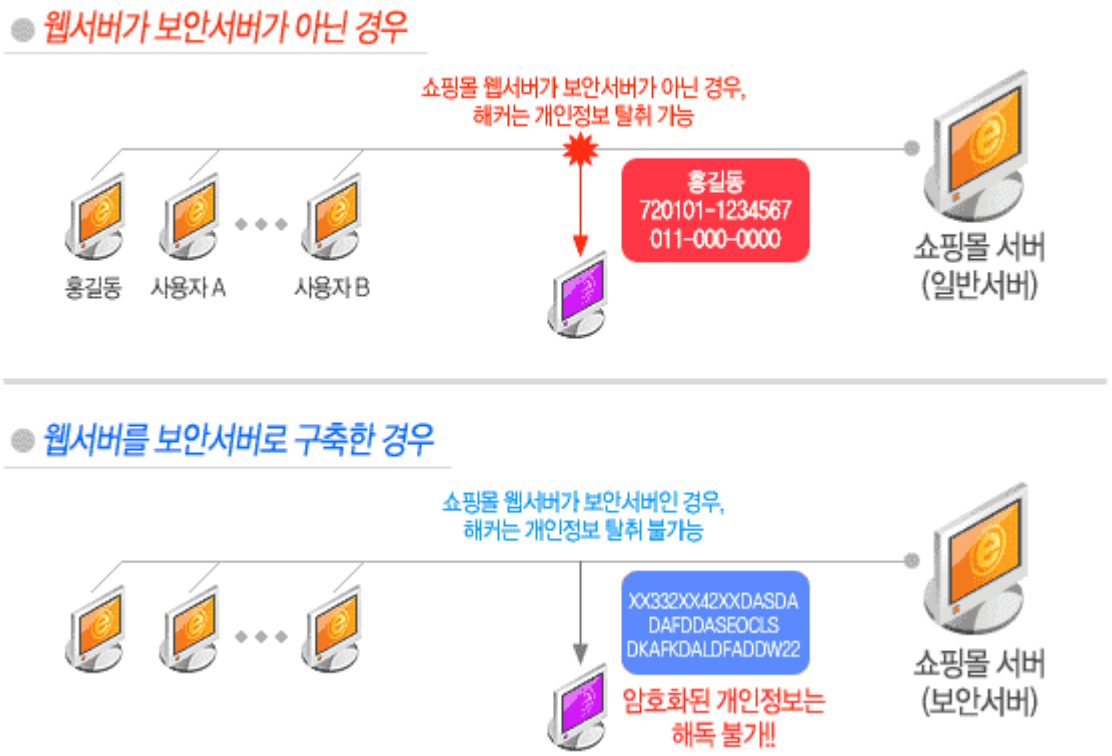
- 블록암호 알고리즘
  - 주민등록번호, 계좌번호 등을 일정한 블록 크기로 나누어, 각 블록을 송수신자 간에 공유한 비밀키를 사용하여 암호화하는 방식



보안강도	해쉬함수	사용 권고 기간(년도)
80 비트	2TDEA	~2010년
112 비트	3TDEA	~2030년
128 비트	SEED,HIGHT,ARIA-128,AES-128	2030년 이후
192 비트	ARIA-192,AES-192	
256 비트	ARIA-256,AES-256	

개인정보 전송시 활용되는 암호 기술

- SSL/TLS
  - 사용자가 입력한 주민등록번호, 계좌번호 등의 중요 정보를 웹서버에 전송하는 경우, 중요 정보가 제3자에게 유출되는 것을 막기 위해 사용

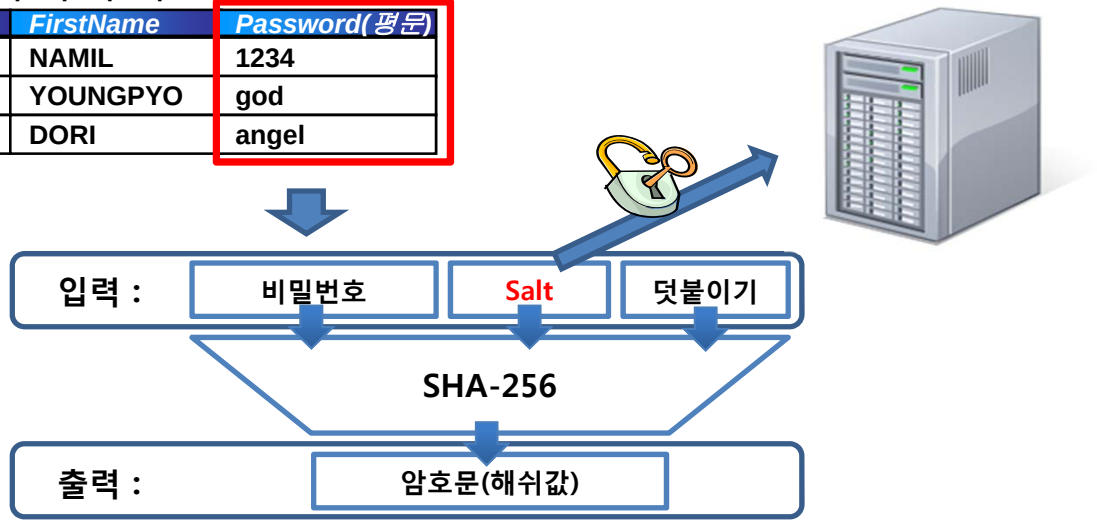


KISA의 보안서버 구축 안내서 참조
(KISA 안내·해설 제 2010-17호)

암호기술 적용 시나리오

- 일방향 해쉬함수 적용 시나리오
  - 비밀번호, 바이오정보 등을 소유자가 아닌 서비스 제공자가 복호화 할 수 없도록 일방향 해쉬하여 저장

EmpNum	LastName	FirstName	Password(평문)
10001	KIM	NAMIL	1234
10002	LEE	YOUNGPYO	god
10003	CHA	DORI	angel



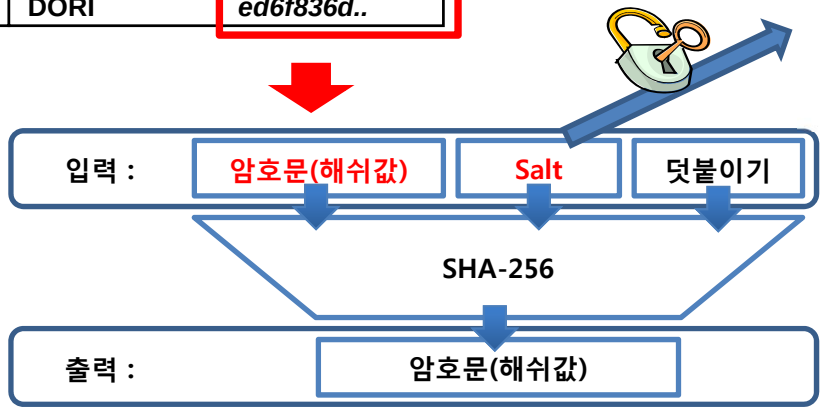
EmpNum	LastName	FirstName	Password(SHA-256)
10001	KIM	NAMIL	03ac674216..
10002	LEE	YOUNGPYO	11043a8795..
10003	CHA	DORI	C9d751a1a..

  - 비밀번호만 해쉬 함수에 적용하는 것은 사전공격에 취약하므로,
    - SALT 값을 비밀번호에 추가하여 일방향 해쉬함수에 적용
  - SALT는 사용자마다 랜덤하게 생성하여 해커가 예측하기 어렵게 만듦

암호기술 적용 시나리오

- 해쉬함수 변경 시나리오
  - 기존 시스템에 보안강도가 낮은 해쉬 함수가 적용되어 있는 경우 보안 강도가 높은 해쉬 함수로 변경하고자 하는 경우
  - 복호화가 불가능한 비밀번호 등을 모두 다시 입력받아 새로운 해쉬 함수로 변경하는 것은 어려움
    - 이미 해쉬된 비밀번호를 다시 새로운 해쉬함수로 해쉬하여 저장(2중해쉬)

EmpNum	LastName	FirstName	Password(MD5)
10001	KIM	NAMIL	*930e1c89..*
10002	LEE	YOUNGPYO	*347ed2f4..*
10003	CHA	DORI	*ed6f836d..*



EmpNum	LastName	FirstName	Password(SHA-256)
10001	KIM	NAMIL	*03ac674216..*
10002	LEE	YOUNGPYO	*11043a8795..*
10003	CHA	DORI	*C9d751a1a..*

암호기술 적용 시나리오

- 새로운 해쉬함수 적용 이후 사용자 로그인시
  - 사용자가 입력한 비밀번호를 임시로 보관
    - 저장된 비밀번호 해쉬값 ≠ Hash(비밀번호) 이면
      저장된 비밀번호 = Hash(Hash(비밀번호))인지 확인
      맞으면 Hash(비밀번호) 저장
  - 사용자 비밀번호의 해쉬 함수 확인 방법

① 새로운 해쉬함수 도입 이후 처음 로그인한 사용자의 경우, Hash(비밀번호) 저장

→ 새로운 해쉬함수 도입 이후 로그인한 적이 있는 사용자의 경우, 이미 비밀번호가 새로운 해쉬함수로 해쉬되어 저장된 사용자

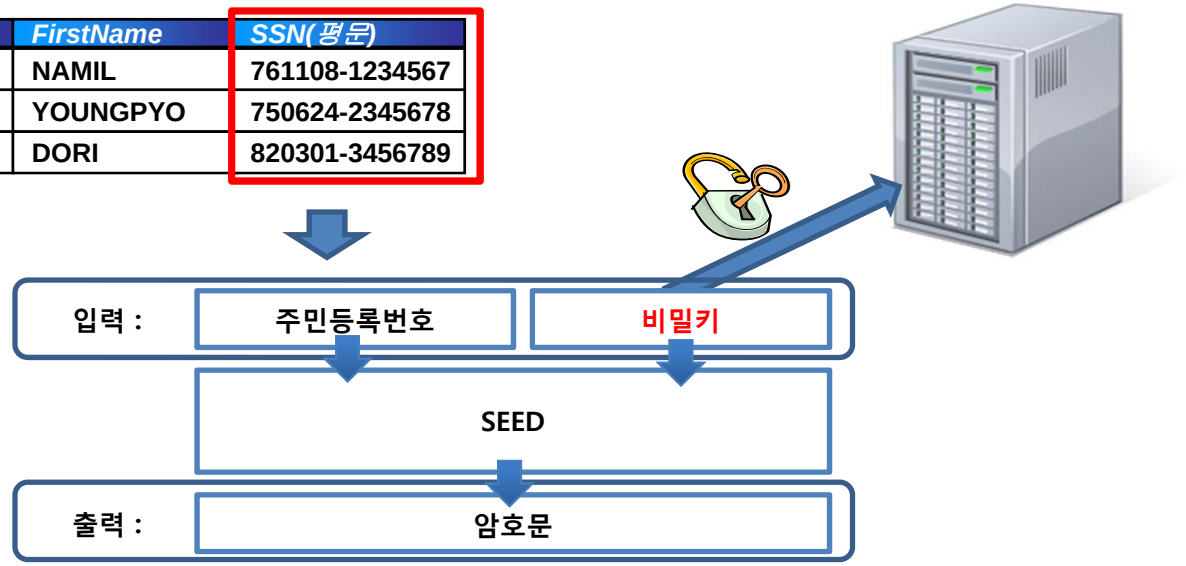
② 새로운 해쉬함수를 사용하여 모든 사용자 비밀번호를 2번 해쉬하여 저장할 때 각 사용자 flag를 0으로 설정

→ flag가 0인 사용자가 로그인 하면 1번 해쉬하여 저장하고 flag를 1로 변경

암호기술 적용 시나리오

- 블록암호 적용 시나리오
  - 암호화가 필요한 데이터는 비밀키를 생성한 후 블록암호알고리즘을 통해 암호화 하여 저장
    - 비밀키는 사용자 DB와 물리적으로 분리된 장소에 별도 보관

EmpNum	LastName	FirstName	SSN(평문)
10001	KIM	NAMIL	761108-1234567
10002	LEE	YOUNGPYO	750624-2345678
10003	CHA	DORI	820301-3456789

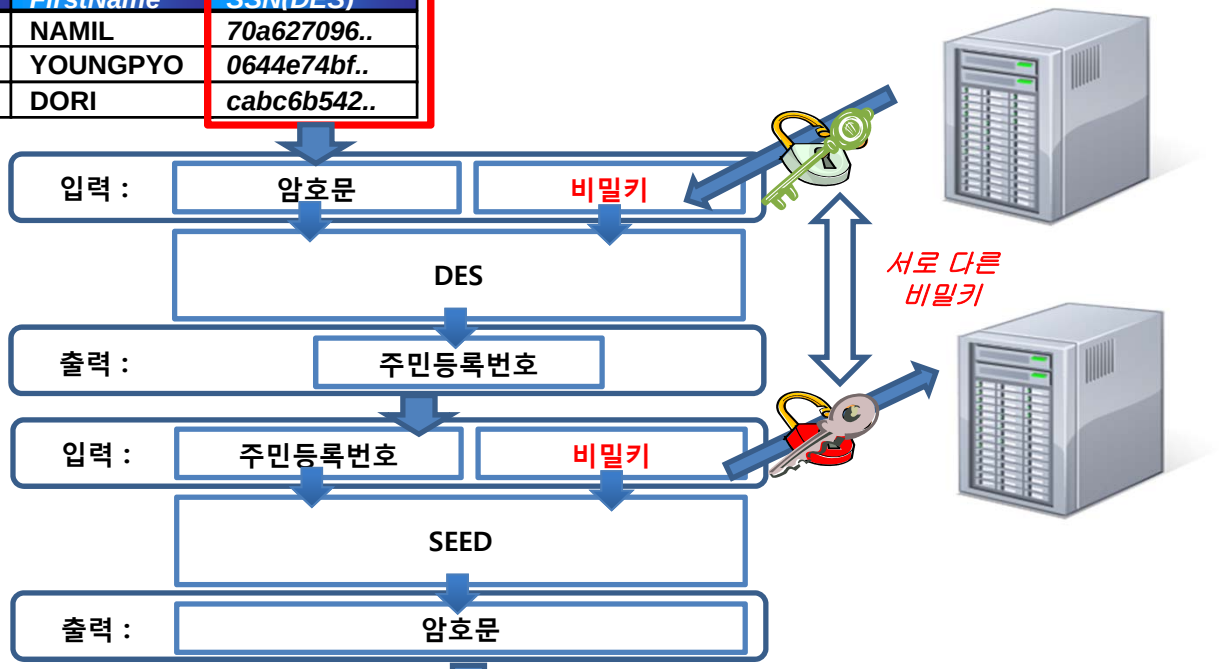


EmpNum	LastName	FirstName	SSN(SEED)
10001	KIM	NAMIL	*7ccbf16566d..*
10002	LEE	YOUNGPYO	*26505d34366..*
10003	CHA	DORI	*e318286c064..*

암호기술 적용 시나리오

- 암호알고리즘 변경 시나리오
  - 보안 강도가 낮은 암호 알고리즘을 높은 알고리즘으로 변경하는 경우
  - 암호화되어 저장된 데이터를 비밀키를 사용하여 복호화한 후,
    - 새로운 비밀키를 생성하여 새로운 암호알고리즘으로 다시 암호화하여 저장
  - 비밀키 노출, 안전성 저하 등의 이유로 비밀키를 변경하는 경우에도 적용

EmpNum	LastName	FirstName	SSN(DES)
10001	KIM	NAMIL	70a627096..
10002	LEE	YOUNGPYO	0644e74bf..
10003	CHA	DORI	cabc6b542..



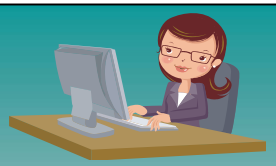
EmpNum	LastName	FirstName	SSN(SEED)
10001	KIM	NAMIL	7ccbf16566d..
10002	LEE	YOUNGPYO	26505d34366..
10003	CHA	DORI	e318286c064..

암호기술 적용 시나리오

사용자

웹 서버

전송계층(SSL)에서의 암호화 방식



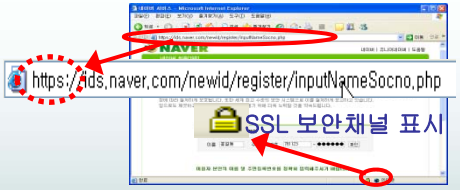
① NAVER 홈페이지 접속 요청

② NAVER 홈페이지 전송

③ 회원가입 요청

④ 회원가입 신청 페이지 전송
(SSL 보안채널 형성)

⑤ 회원가입을 위한 사용자 개인 정보 입력
(주민번호, 아이디, 비밀번호 등)



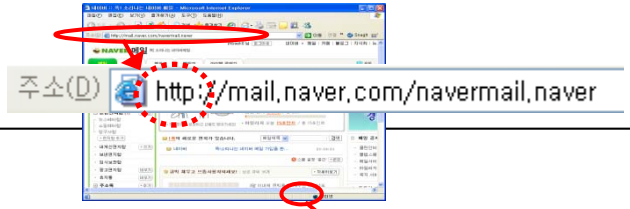
SSL 보안채널

(주민번호, 아이디, 비밀번호 등 사용자 개인정보를 암호화하여 전송)

⑥ 회원정보 확인 및 저장

⑦ 회원가입 완료 통보

⑧ 네이버 메일 서비스 접속 요청



⑨ 네이버 메일 서비스 페이지 전송
(SSL 보안채널 해제)

⑩ 네이버 메일 서비스 이용

SSL 보안 해제

비 보안구간

보안구간

암호기술 적용 시나리오

응용계층에서의 암호화 방식

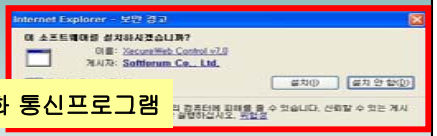
사용자

웹서버

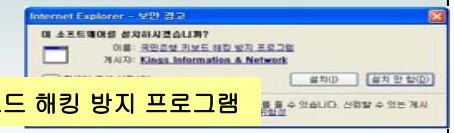
① 인터넷뱅킹 접속

③ 해킹 방지 프로그램 설치

해킹차단기



암호화 통신프로그램



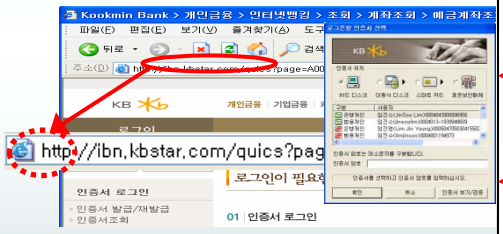
키보드 해킹 방지 프로그램

해킹차단기, 암호화통신 프로그램, 키보드 해킹방지 프로그램

② 해킹 방지 프로그램 전송

④ 해킹방지 프로그램 작동

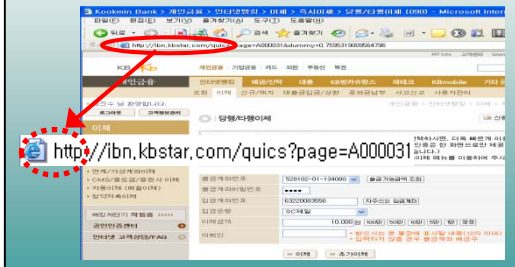
⑤ 인터넷뱅킹 인증서 로그인



⑥ 사용자 공인인증서 검증 (암호화 키공유)

암호화 키 공유

⑦ 계좌이체 정보입력



계좌번호, 계좌이체 금액 등 보안이 필요한 데이터 암호화

⑧ 사용자 정보검증 및 계좌이체

⑨ 계좌이체 결과 전송

⑩ 계좌이체 결과확인

II 안전한 비밀번호 및 키관리 방안

안전한 비밀번호 관리

안전한 비밀번호 기준

안전한 비밀번호란

- 제3자가 쉽게 추측할 수 없는 비밀번호
- 시스템에 저장되어 있는 사용자 정보나 인터넷을 통해 전송되는 정보를 해킹하여 알아낼 수 없는 비밀번호
- 알아내는데 많은 시간이 요구되는 비밀번호

안전한 비밀번호의 문자구성 및 길이 조건

- 세가지 종류 이상의 문자구성으로 8자리 이상의 길이로 구성된 비밀번호
- 두가지 종류 이상의 문자구성으로 10자리 이상의 길이로 구성된 비밀번호
  ※문자종류 : 알파벳 대문자와 소문자, 특수기호, 숫자
- 단, 비밀번호 변경 주기는 6개월 이내로 사용할 것을 권고

취약한 비밀번호

- 7자리 이하 또는 두가지 종류 이하의 문자구성으로 8자리 이하의 비밀번호

  ※ 문자 종류는 알파벳 대문자와 소문자, 특수기호, 숫자 4가지를 의미

- 특정 패턴을 갖는 비밀번호

  - 동일한 문자의 반복(예:aaabbb, 123123)

  - 키보드 상에서 연속한 위치에 존재하는 문자들의 집합(예 : qwerty, asdfgh)

  - 숫자가 제일 앞이나 제일 뒤에 오는 구성(예 : security1)

- 한글, 영어 등을 포함한 사전적 단어로 구성된 비밀번호

- 사용자 ID를 이용한 비밀번호

  예) ID가 'KDHong'인 경우, 비밀번호를'KDHong12'또는'HongKD'로 설정

- 사용자, 유명인, 연예인 등 특정 인물의 이름이나 널리 알려진 단어를 포함한 비밀번호

- 이름, 생일, 전화번호 등 제3자가 쉽게 알 수 있는 개인정보를 바탕으로 구성된 비밀번호

- 숫자와 영문자를 비슷한 문자로 치환한 형태를 포함한 구성의 비밀번호

  예) 영문자 "O"를 숫자 "0"으로, 영문자 "l"을 숫자 "1"로 치환 등의 비밀번호

- 기타

  - 시스템에서 예제로 제시되고 있거나 시스템에서 초기에 설정되어 있는 비밀번호

  - 한글의 발음을 영문으로, 영문단어의 발음을 한글로 변형한 형태의 비밀번호

  예)'사랑'를 "SaRang"으로 표기하는 비밀번호, 영문자 "LOVE"의 발음을 한글 "러브"로 표기하는 비밀번호

안전한 비밀번호 설정 팁

기억하기 쉬운 비밀번호

- 특정명칭을 선택하여 예측이 어렵도록 가공하여 비밀번호 설정
  - 특정명칭의 홀·짝수 번째의 문자를 구분하는 등의 가공방법 사용
  - 국내 사용자는 한글 자판을 기준으로 특정명칭을 선택하고 가공하여 설정

  예) '한국인터넷진흥원'의 경우
  - ☞ 홀수번째는 '한인넷흥'이 'gksdlssptgmd'
  - ☞ 짝수번째는 '국터진원이 'rnrxjwlsdnjs'
- 노래 제목이나 명언, 속담, 가훈 등을 이용, 가공하여 비밀번호 설정

  예) ' This May Be One Way To Remember"를 'TmB1w2R'

  "백설공주와 일곱 난쟁이"를 "백설+7난장"으로 구성 후

  'QorTjf+7SksWkd'으로 활용

예측이 어려운 문자구성의 비밀번호

- 영문자(대·소문자), 숫자, 특수 기호들을 혼합한 구성으로 설정

  예) 10시 20분은 '10H+20Min'으로 변경

  I can Do it을 활용하여 'I!Can&9it'으로 변경
- 비밀번호 길이 증가를 위해 알파벳 문자 앞뒤가 아닌 위치에 특수문자 및 숫자 등을 삽입

  예) 'Security1'이 아니라 'Securi2t&&y'등으로 변경
- 알파벳 대소문자를 구별할 수 있을 경우, 대소문자를 혼합하여 사용

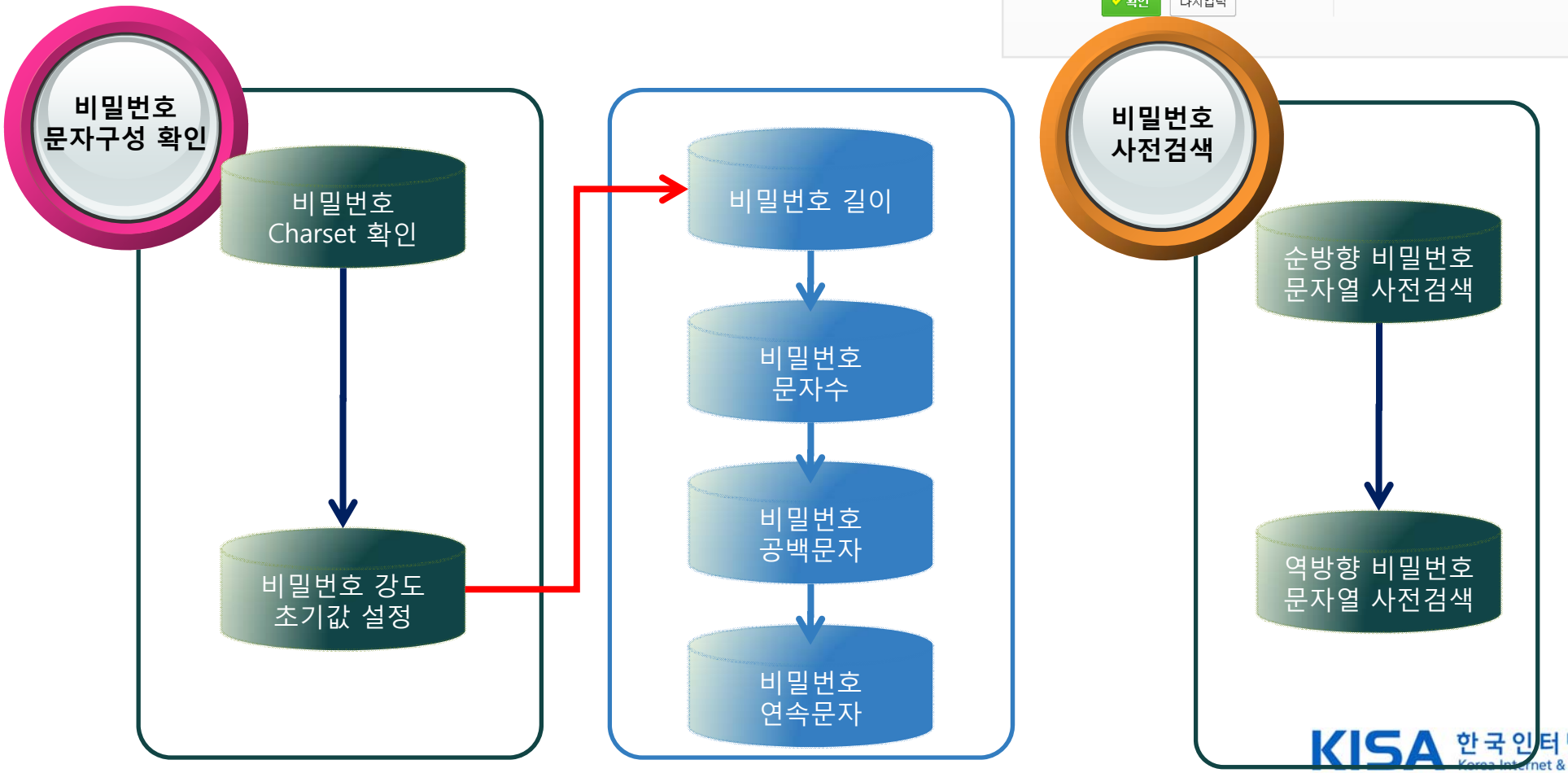
사이트별로 상이한 비밀번호

- 자신의 기본 비밀번호 문자열을 설정하고 사이트별 특정 규칙 적용

  예) 기존 비밀번호 문자열이 "486*+"이고 사이트 이름의 짝수번째 문자 추가
  - ☞ yahoo.com은 '486*+ao.o'
  - ☞ google.co.kr은 '486*+ogec.r'

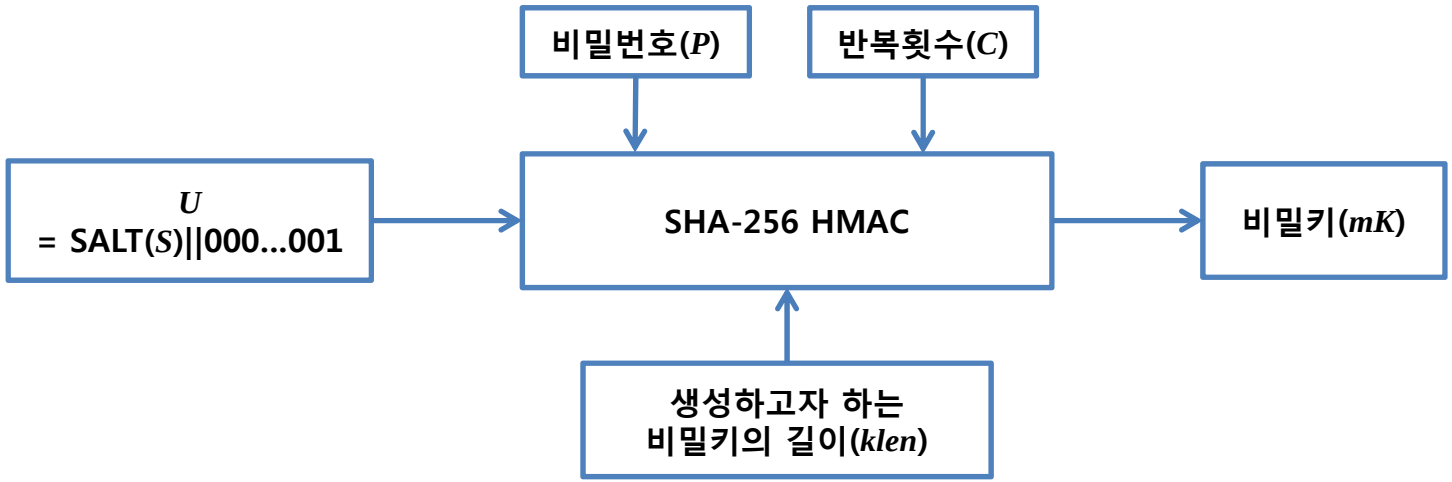
비밀번호 안전성 검증 S/W 구현

웹사이트에 사용자 로그인시, 회원가입을 위한 비밀번호 설정시 사용자 비밀번호의 안전도를 검증하여 사용자에게 안전한 비밀번호 사용을 유도



암호키 생성

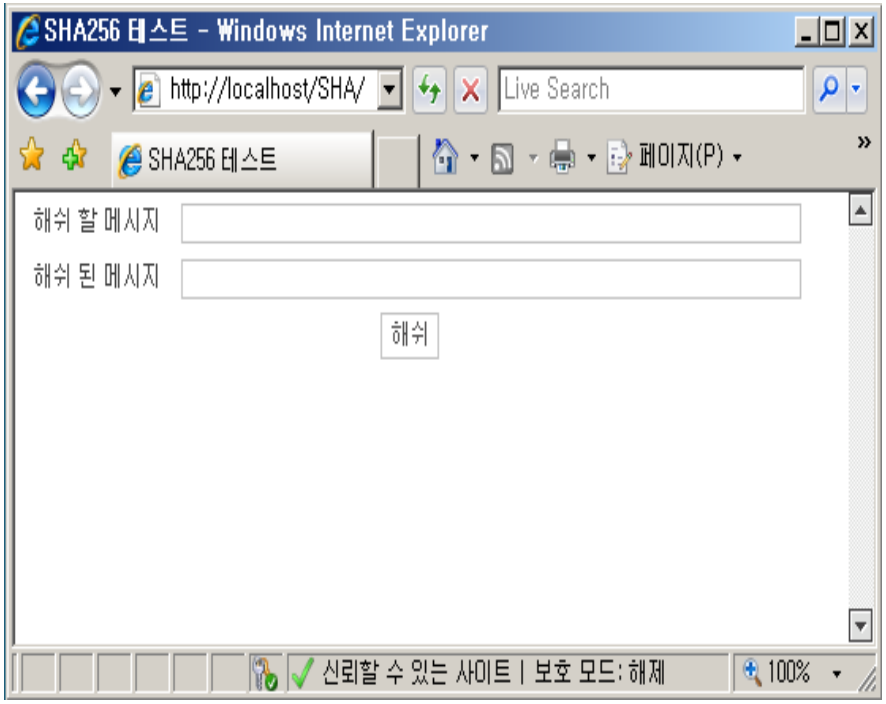
- 관련 표준
  - RSA PKCS#5 v2.1 : Password-Based Cryptography Standard
    - MD5, SHA-1을 사용하고 있어 권고하지 않음
  - NIST SP 800-108 : Recommendation for Key Derivation Using Pseudorandom Functions
  - NIST SP 800-123 : Recommendation for Password-Based Key Derivation Part1 - Storage Applications
- NIST SP 800-132 Password-based Key Derivation Function(PBKDF)
  - SHA-256 HMAC을 이용한 128비트 비밀키 생성 방법



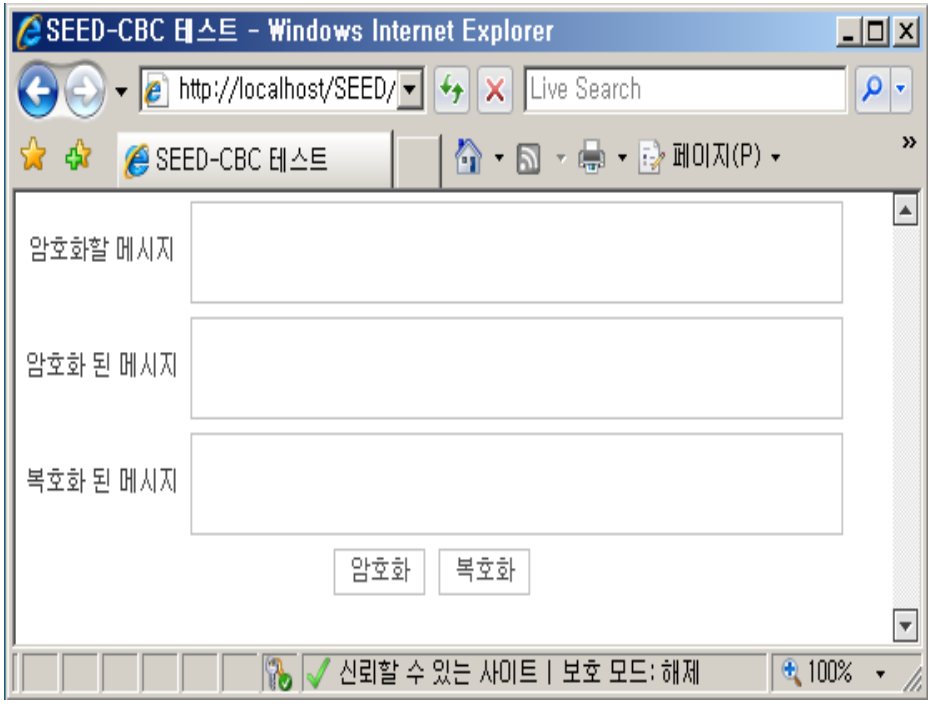
III 암호기술 구현 방법

구현 환경

- 암호 라이브러리
  - KISA에서 배포하는 SEED 및 SHA-256 ASP,JSP 암호라이브러리 활용
- 구현 예제 웹페이지



<SHA-256 이용 예제 웹페이지>



<SEED-CBC 이용 예제 웹페이지>

SHA-256을 이용한 개인정보 해쉬

웹페이지 SHA-256 구현 예(index.html)

```
<html>
<head>
   생략
</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="해쉬" name="enc" onclick=
         "JavaScript:sha256('sha256hash.asp');" /></td>
   ☞ JSP의 경우 sha256hash.asp를 sha256hash.jsp로 변경하여 사용
      </tr>
   </table>
</form>
</body>
</html>
```

SHA-256을 이용한 개인정보 해쉬

- SHA256.dll을 레지스트리에 등록
  - regsvr32.exe %windir%₩system32₩SHA256.dll
- Sha256.asp

```
<%@ 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) )
  ☞ 해쉬 수행 및 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
%>
```

SEED를 이용한 개인정보 암호화

웹페이지 SEED-DBD 암복호화 예(index.html)

```
<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>
```

SEED를 이용한 개인정보 암호화

- SeedCBC.dll 레지스트리 등록
- SeedEncryption.asp

```
<%@ 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) )
   ☞ 암호화 수행 및 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 ☞ 메모리 반환
%>
```

SEED를 이용한 개인정보 암호화

SeedDecryption.asp : SEED-CBC 복호화 수행

```
<%@ 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) )
  ☞ 복호화 수행 및 웹페이지에 평문을 출력하는 명령어
  set oSeed = nothing ☞ 메모리 반환
%>
```

감사합니다!