2020. 12.

개발자 대상

# 개인정보 보호조치 적용 안내서

## PbD 기본원칙을 중심으로

개인정보보호위원회
Personal Information Protection Commission

KISA 한국인터넷진흥원
KOREA INTERNET & SECURITY AGENCY

# CONTENTS

개발자 대상

# 개인정보 보호조치 적용 안내서

PbD 기본원칙을 중심으로

# I 개 요

# 제1절 배경 및 목적

개인정보처리자는 개인정보의 유출을 사전에 차단하기 위해 온라인 서비스 및 개인정보처리시스템에 보호조치를 적용하여야 한다. 통상적으로 기업에서 개인정보시스템을 구축한 이후에 개인정보 보호를 위한 보안 솔루션 등을 적용하는 경우가 있으나, 이러한 방식은 성능 및 비용 측면에서 매우 비효율적이다.

따라서, 개인정보처리자는 개인정보 보호를 위한 요구사항을 개인정보처리시스템 기획 단계부터 파악하고, 구축 단계에서 그 기능을 구현하는 방식을 통해 비용을 절감하고 보안성을 높일 수 있는 Privacy By Design 방식을 적용할 필요가 있다.

본 안내서는 개인정보처리시스템 구축 이후에 프라이버시를 보호하기 위한 단편적인 조치를 적용하는 방법이 아닌, Privacy By Design 개념을 도입하여 개인정보처리시스템 설계 및 구축 단계에서 적용할 수 있는 개인정보 보호조치 방법을 안내하고자 한다.

**Privacy by Design(PbD)**이란?

Privacy by Design이란 「프라이버시를 고려한 설계」를 의미하는 용어로 개인정보 침해 이슈가 발생한 이후에 조치하는 것이 아닌, 서비스 기획·설계 단계부터 개인정보를 보호하기 위한 기술 및 정책을 적용하여 개인정보 유·노출을 예방하는 것을 말한다.

4차 산업혁명 시대가 도래하면서 지능정보기술을 활용한 빅데이터 분석이 가속화됨에 따라 개인정보를 수집하는 다양한 기기에서 사생활 침해문제에 대한 우려가 날로 증가하고 있다. 신기술 발전에 따라 개인정보 침해 이슈가 증가하는 현시점에서 개인정보 보호를 위한 사전적 예방은 필수적이다. 개인정보처리시스템 구축 초기 단계에서 PbD를 고려하여 개인정보 보호조치를 내재화한다면, 개인정보 유출위험을 최소화하고 향후 운영 단계에서 개인정보 보호 활동에 필요한 사회적·경제적 비용을 최소화할 수 있다.

**Privacy by Design 7대 기본원칙**

| 구분 | 원 칙 | 내 용 |
|---|---|---|
| ① | 사후조치가 아닌 사전예방<br>(Proactive not Reactive – Preventative not remedial) | 프라이버시 침해 사고가 발생한 뒤 조치하는 것이 아닌 침해사건을 예상하고 사전에 예방하는 것 |
| ② | 초기설정부터 프라이버시 보호조치<br>(Lead with Privacy as the Default setting) | 프라이버시 보호조치를 기본값으로 설정하여 자동으로 프라이버시가 최대한 보장되도록 하는 것 |
| ③ | 프라이버시 보호를 내재한 설계<br>(Embedded Privacy into Design) | 프라이버시 보호를 설계에 내재화함으로써 프라이버시를 IT시스템 또는 개인정보 처리와 통합·적용하도록 하는 것 |
| ④ | 프라이버시보호와 사업기능의 균형<br>– 제로섬이 아닌 포지티브섬 –<br>(Retain Full Functionality (positive-sum, not zero-sum)) | 사업의 기능성과 프라이버시 보호 두가지 모두 확보하기 위해 노력하는 것 |
| ⑤ | 개인정보 생애주기 전체에 대한 보호<br>(Ensure End-to-End Security) | 개인정보가 수집·이용·저장·제공·파기 전단계에 걸쳐 보호될 수 있도록 안전조치를 적용하는 것 |
| ⑥ | 처리과정에 대한 가시성·투명성 유지<br>(Maintain Visibility and Transparency – keep it open) | 정보주체가 개인정보 처리과정을 완전하고 명확하게 이해하도록 하여 신뢰성을 제고하는 것 |
| ⑦ | 이용자 프라이버시 존중<br>(Respect for User Privacy – keep it user centric) | 프로그램·프로세스 등에서 명시적인 보호체계가 없더라도 사용자의 프라이버시를 보장하기 위한 활동을 수행하는 것 |

## 제2절 적용 범위

본 안내서는 「개인정보 보호법」에 따라 온라인 서비스 및 개인정보처리시스템 구축하려는 개인정보처리자를 대상으로 기획·구축·운영 단계에서 개인정보 관련 법적 요구사항, 개인정보처리시스템 구현 시 고려사항에 대하여 안내한다.

특히, 개발자를 대상으로 Privacy By Design의 7대 주요원칙에 따라 개인정보 처리 전체의 과정에 걸쳐 정보주체의 개인정보를 안전하게 처리하기 위해 시스템 기획·개발 단계에서 고려해야 할 개인정보 보호조치를 중점적으로 다룰 예정이다.

## 제3절 구성형식

본 안내서의 구성은 다음과 같은 형식으로 구성되어있다.

- **제1장 개 요**
  「개발자 대상 개인정보 보호조치 적용 안내서」 발간 배경 및 취지 안내
- **제2장 개인정보처리시스템 기획·구축 단계**
  정보주체가 이용하는 온라인 서비스*와 개인정보취급자가 이용하는 개인정보처리시스템**에서 개인정보 생애주기(수집→저장·관리→이용·제공→파기) 흐름에 따른 기획·구축 방법 안내

  * 개인정보처리자가 서비스 제공을 위해 정보주체로 하여금 개인정보를 입력, 조회, 수정 등을 할 수 있도록 기능을 구현한 웹(PC, 모바일) 서비스 및 어플리케이션

  ** 인터넷 홈페이지 등에서 수집한 정보주체의 개인정보를 개인정보취급자가 검색, 조회, 변경, 출력 등 체계적으로 처리할 수 있도록 화면 및 기능 등을 제공하는 시스템
- **제3장 개인정보처리시스템 운영 단계**
  개인정보처리시스템 운영 단계에서 개인정보 담당자가 알아야 하는 조치사항을 안내

# II 개인정보처리시스템 기획·구축 단계

## 제1절 온라인 서비스 기획·구축

1. 개인정보 수집단계
2. 개인정보 이용 및 전송단계
3. 정보주체 권리 보장 및 파기 단계

## 제2절 개인정보처리시스템 기획·구축

1. 접근 권한 관리
2. 접근통제
3. 개인정보 암호화
4. 접속기록 관리
5. 기타 보호조치

# 제1절 온라인 서비스 기획·구축

## 개요

정보주체가 이용하는 온라인 서비스를 통해 개인정보 유·노출 등 피해를 최소화하기 위해서는 기획·구축 단계에서부터 개인정보 보호조치가 이루어져야 한다.

또한, 개인정보를 안전하게 처리하기 위하여 온라인 서비스를 구축하기 전에 개인정보 처리 단계별(수집-이용·전송-파기) 개인정보 처리 요구사항을 먼저 확인해야 한다.

본 절에서는 정보주체가 이용하는 온라인 서비스 기획 시 준수해야 할 법적 요구사항을 안내하고, 구축 시 적용해야 할 개인정보 보호조치 및 예시 소스코드를 제공함으로써 온라인 서비스 개발자가 쉽게 참조할 수 있도록 하였다.

※ 단, 본 가이드에서 제공하는 소스코드는 개발 시 참고하기 위한 예시 코드이므로, 개발 방법, 환경 등에 따라 적용되지 않을 수 있으며, 개발하려는 환경 등을 고려하여 구현해야 함

## 기본 원칙

온라인 서비스 기획·구축 시 개인정보 보호를 위해 검토하고 확인해야 할 기본 원칙은 아래와 같다.

- 개인정보 수집 시, 개인정보의 유형 및 종류에 따라 동의 절차 마련
- 정보주체가 안전한 비밀번호를 설정할 수 있도록 비밀번호 작성규칙을 수립하고 적용
- 개인정보 전송 시 안전한 알고리즘으로 암호화
- 개인정보 출력 시 마스킹(*) 처리 등 개인정보가 노출되지 않도록 보안조치
- 정보주체가 자신의 개인정보를 언제든지 열람하고 정정·삭제
- 동의 철회 의사를 표시할 수 있도록 회원탈퇴 등 기능마련
- 개인정보가 불필요하게 되었을 때(처리 목적 달성 시 등) 별도보관 및 완전 파기

# 1 개인정보 수집단계

| | |
|---|---|
| 항목명 | 개인정보 수집·이용 시 동의 |
| 요구사항 내용 | 1.1 개인정보 수집·이용 동의<br>- 업무처리에 필요한 개인정보 파악<br>- 개인정보 보유 및 이용 기간 설정<br>- 개인정보 수집·이용을 위한 동의 절차 마련 |
| 관련 근거 | • 개인정보 보호법 제15조(개인정보의 수집·이용)<br>• 개인정보 보호법 제16조(개인정보의 수집 제한)<br>• 개인정보 보호법 제17조(개인정보의 제공)<br>• 개인정보 보호법 제22조(동의를 받는 방법)<br>• 개인정보 보호법 제23조(민감정보의 처리 제한)<br>• 개인정보 보호법 제24조(고유식별정보의 처리 제한)<br>• 개인정보 보호법 제39조의3(개인정보 수집·이용 통의 등에 대한 특례) |

## 1.1 개인정보 수집·이용 동의

### 기획 단계

• 정보주체에게 서비스를 제공하는 데 필요한 **최소한의 개인정보를 파악**

- 수집하는 개인정보가 서비스 제공을 위한 최소한의 개인정보 수집이라는 입증책임은 개인정보처리자가 부담하여야 함

**PbD 적용설계** | (제⑦원칙) 이용자 프라이버시 존중

개인정보를 수집하는 경우 서비스 제공을 위하여 필요 최소한의 정보만을 수집해야 한다.

• 업무의 특성을 고려하여 **보유 및 이용 기간을 설정**

- 관련 법령에 개인정보 보유 기간에 대한 근거가 있는 경우, 해당 법령에 명시된 보유 기간을 준수

• 개인정보 수집·이용을 위해서 정보주체의 동의절차를 마련

- (**법정대리인 동의**) 개인정보를 수집·이용하려는 대상이 **만 14세 미만의 아동**일 경우 법정대리인의 동의를 받아야 함

**14세 미만 아동 회원가입 시 법정대리인 동의 화면(예시)**


**참고** **만 14세 미만의 개인정보 수집 동의 방법**

• 일반적으로, 일반 회원과 만 14세 미만 아동의 가입경로를 구분하여 회원가입을 진행할 수 있다.

• 법정대리인의 동의를 받는 데 필요한 최소한의 정보는 법정대리인의 동의 없이 해당 아동으로부터 직접 수집이 가능하다. 단, 일정기간이 지나도록 법정대리인으로부터 회신이 없거나 법정대리인의 동의 거부로 만 14세 미만 아동의 회원가입이 완료되지 않은 경우, 해당 아동과 법정대리인의 개인정보는 5일 이내에 파기해야 한다.

- (**필수 동의 항목**) 정보주체에게 서비스를 제공하기 위해 필요한 최소한의 개인정보를 '**필수 동의 항목**'을 통하여 동의를 받고 개인정보를 수집해야 함

- (**선택 동의 항목**) 필수 동의 항목 이외에 **선택 동의 항목의 경우**(마케팅, 판매홍보 등), 개인정보 수집에 동의하지 않을 수 있다는 사실을 알리고 개인정보를 수집해야 함

※ 선택 동의 항목이 다수 존재할 경우 각각 동의 여부를 선택할 수 있도록 해야 함

**참고** **필수 동의 항목 vs 선택 동의 항목**

- 필수 동의 항목 : 해당 서비스의 본질적 기능을 수행하기 위해 반드시 필요한 정보
  - 정보주체에게 필수적 동의 요구 가능
  - 정보주체가 필수 동의 항목에 동의하지 않으면 서비스를 제공 받을 수 없음
- 선택 동의 항목 : 개인정보처리자의 필요에 의하거나 추가적인 서비스를 위해 필요한 정보
  - 정보주체가 동의 여부 선택 가능
  - 정보주체가 선택 동의 항목에 동의하지 않는다고 해서 서비스 이용을 거부해서는 안 됨

- (**별도 동의 항목-①**) 수집·이용하려는 정보가 **민감정보***, **고유식별정보****(단, 주민등록번호 제외)일 경우, 정보주체로부터 별도동의를 받고 수집해야 함

* 민감정보 : 사상·신념, 노동조합·정당의 가입·탈퇴, 정치적 견해, 건강, 성생활 등에 관한 정보, 유전정보, 범죄경력자료, 개인의 신체적, 생리적, 행동적 특징에 관한 정보로서 특정 개인을 알아볼 목적으로 일정한 기술적 수단을 통해 생성한 정보, 인종·민족 정보 등

** 고유식별정보 : 여권번호, 운전면허번호, 외국인등록번호, 주민등록번호

**참고** **주민등록번호 수집 제한**

- 주민등록번호 수집은 법률·대통령령에 구체적으로 명시된 경우를 제외하고 금지이며 회원가입 시 주민등록번호 이외의 대체수단을 이용해야 한다.

- (**별도 동의 항목-②**) **개인정보를 제3자에게 제공할 경우**, 개인정보를 수집한 목적범위에서 제공하거나 정보주체로부터 별도 동의를 받아야 함

※ 단, 법률에 특별한 규정이 있는 등 법에 명시된 불가피한 사유가 있는 경우 예외

## 구축 단계

• 개인정보 수집이용 동의절차 구현 시 **쉽고 간결하게 고지**하여, 동의내용을 한눈에 알아볼 수 있도록 구현

**PbD 적용설계** | (제⑥원칙) 처리과정에 대한 가시성·투명성 유지

'법적 고시사항'만 간결하게 고지하고, 일반인이 이해할 수 있도록 이해하기 쉬운 서식과 명확하고 알기 쉬운 언어를 사용해야 한다.

| 구분 | 법적 고지사항 |
|---|---|
| 수집·이용 동의 | 1. 개인정보의 수집·이용 목적<br>2. 수집하려는 개인정보의 항목<br>3. 개인정보의 보유 및 이용 기간<br>4. 동의를 거부할 권리가 있다는 사실 및 동의 거부에 따른 불이익이 있는 경우에는 그 불이익의 내용 (단, **정보통신 서비스제공자 등**의 경우 제외)<br>※ 필수 동의 항목 이외, 선택 동의 항목, 민감정보 및 고유식별정보 수집 시 각각의 동의사항을 구분하여 동의를 받아야 함 |
| 제3자 제공 동의 | 1. 개인정보를 제공받는 자<br>2. 개인정보를 제공받는 자의 개인정보 이용 목적<br>3. 제공하는 개인정보 항목<br>4. 개인정보를 제공받는 자의 개인정보 보유 및 이용기간<br>5. 동의를 거부할 권리가 있다는 사실 및 동의 거부에 따른 불이익이 있는 경우에는 그 불이익의 내용 |
| 국외이전 동의 (정보통신서비스 제공자등에 해당) | 1. 이전되는 개인정보 항목<br>2. 개인정보가 이전되는 국가, 이전일시 및 이전방법<br>3. 개인정보를 이전받는 자의 성명(법인인 경우에는 그 명칭 및 정보관리책임자의 연락처를 말한다)<br>4. 개인정보를 이전받는 자의 개인정보 이용목적 및 보유 · 이용 기간<br>※ '국외 이전'이란 개인정보를 국외에 제공(조회), 처리위탁, 보관하는 경우를 말함(단, 처리위탁 또는 보관에 해당하는 경우, 위의 고지사항을 '개인정보 처리방침'을 통해 알린 경우에는 동의절차 생략 가능) |

**참고** **개인정보 수집 동의 시, 고지사항 안내**

• 정보주체가 동의 항목을 명확하게 인지할 수 있도록 동의 획득 시 고지사항은 가급적 표로 구성하는 것이 바람직하다.

• 개인정보 수집·이용 동의를 위해 고지할 때, 아래 주요 동의내용은 화면에서 글씨크기, 굵기 또는 밑줄 등을 이용하여 알아보기 쉽게 구현

**주요 동의내용**

① 개인정보의 수집·이용 목적 중 재화나 서비스의 홍보 또는 판매 권유 등을 위하여 해당 개인정보를 이용하여 **정보주체에게 연락할 수 있다는 사실**
② **민감정보 및 고유식별정보**(여권번호, 운전면허번호, 외국인등록번호)
③ 개인정보의 **보유 및 이용 기간**(제3자 제공시에는 제공 받는 자의 보유 및 이용 기간)
④ (제3자 제공 시) **개인정보를 제공받는 자, 제공받는 자의 개인정보 이용 목적**

• 개인정보 수집·이용 등에 대한 **정보주체의 동의 사실을 기록**

- 정보주체의 개인정보 수집·이용에 대한 '동의 여부', '동의 일시'를 기록하는 컬럼을 구성하여 기록 및 보관
- 동의를 받아야 하는 사항이 여러 항목인 경우(필수 동의 항목, 선택 동의 항목, 민감정보 및 고유식별정보의 수집·이용 등), 각 항목의 동의 사실을 기록

**TIP 참고** **동의사실 기록에 대한 중요성**

• 동의 사실에 대한 기록은 정보주체의 개인정보 보유 및 이용기간에 따른 파기여부 확인, 개인정보처리자의 법 이행사항 입증 등에 활용될 수 있다.

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

개인정보 동의 항목별로 동의 및 동의철회 이력을 추적·관리할 수 있도록 DB컬럼 및 기능을 설계해야 한다.

※ 특히, 홍보·마케팅 등 선택 동의 사항은 동의를 철회하거나 또는 다시 동의를 하는 것이 반복적으로 발생할 수 있으므로, 그 이력이 관리될 수 있도록 동의 및 동의 철회 기록을 남기는 것이 중요

**PbD 적용설계 | (제②원칙) 초기설정부터 프라이버시 보호조치**

'동의', '동의하지 않음'을 정보주체가 자발적 의사에 따라 선택할 수 있도록, '동의'란에 미리표시(Default 설정)하지 않아야 한다.

**구현 예시**

## 개인정보 수집·이용 동의(예시)

신규 회원가입을 위하여 아래의 개인정보 수집·이용에 대한 내용을 자세히 읽어 보신 후 동의 여부를 결정하여 주시기 바랍니다.

■ [필수] 개인정보 수집·이용 동의

| 이용 목적 | 수집 항목 | 보유기간 | 동의여부 |
|---|---|---|---|
| 회원 식별, 중요 공지 사항의 전달, 고객 문의 응대 | 성명, 생년월일, 성별 이메일 주소, 휴대폰 번호 | **서비스 탈퇴 후 5일까지** | ☐ 동의함<br>☐ 동의안함 |

※ 위와 같이 개인정보를 수집·이용하는데 동의를 거부할 권리가 있습니다. 그러나 동의를 거부할 경우 회원가입이 불가하며 온라인 교육 서비스를 제공 받으실 수 없습니다.

■ [선택] 개인정보 수집·이용 동의

| 이용 목적 | 수집 항목 | 보유기간 | 동의여부 |
|---|---|---|---|
| **신규 상품안내 문자발송** | 휴대전화번호 | **서비스 탈퇴 후 5일까지** | ☐ 동의함<br>☐ 동의안함 |
| 맞춤형 광고 | 쿠키정보, 쿠키를 통해 수집되는 행태정보 | **서비스 탈퇴 후 5일까지** | ☐ 동의함<br>☐ 동의안함 |
| 제휴서비스 이용시 포인트 적립 연계 | 휴대전화번호 | **제휴서비스 종료 후 5일까지** | ☐ 동의함<br>☐ 동의안함 |

※ 동의를 거부하시는 경우에도 온라인 교육 서비스는 이용하실 수 있습니다.

# 2 개인정보 이용 및 전송단계

| | |
|---|---|
| 항목명 | 개인정보 이용 및 전송 |
| 요구사항 내용 | **2.1 비밀번호 작성규칙 적용**<br>- 안전한 비밀번호 작성규칙 수립·적용<br>- 비밀번호 입력 시 표시제한(마스킹 조치)<br>**2.2 개인정보 전송 시 암호화 적용**<br>- 암호화 대상 개인정보 확인<br>- 개인정보 전송구간 암호화 적용<br>**2.3 기타 개인정보 안전조치 적용**<br>- URL/소스코드를 통한 개인정보 유·노출 예방<br>- 임시저장 페이지 삭제조치 |
| 관련 근거 | • 개인정보 보호법 제29조(안전조치의무)<br>• 개인정보의 안전성 확보조치 기준 제5조(접근 권한의 관리)<br>• 개인정보의 안전성 확보조치 기준 제6조(접근통제)<br>• 개인정보의 안전성 확보조치 기준 제7조(개인정보의 암호화)<br>• 개인정보의 기술적·관리적 보호조치 기준 제5조(접근통제)<br>• 개인정보의 기술적·관리적 보호조치 기준 제6조(개인정보의 암호화) |

## 2.1 비밀번호 작성규칙 적용

**기획 단계**

• 정보주체가 안전한 비밀번호를 설정하여 이용할 수 있도록, **비밀번호 작성규칙을 수립하여 적용**

**PbD 적용설계 | (제①원칙) 사후조치가 아닌 사전예방**

정당한 접속 권한을 가지지 않는 자가 정보주체의 비밀번호를 추측해서 불법적으로 접속하는 것을 방지하기 위해, 정보주체에게 안전한 비밀번호 작성규칙을 적용할 수 있도록 사전에 설계해야 한다.

**비밀번호 작성규칙 설정 예시**

| 구분 | 법적 고지사항 |
|---|---|
| 최소 길이 | • 최소 8자리 이상 : 두 종류 이상의 문자를 이용하여 구성한 경우<br>※ 문자 종류 : 알파벳 대문자와 소문자, 특수문자, 숫자<br>• 최소 10자리 이상 : 하나의 문자종류로 구성한 경우<br>(단, 숫자로만 구성할 경우 취약할 수 있음) |
| 추측하기 어려운 비밀번호 | • 동일한 문자 반복(aaabbb, 123123 등)이 포함되지 않아야 함<br>• 키보드 상에서 나란히 있는 문자열(qwer 등)이 포함되지 않아야 함<br>• 일련번호(12345678 등), 생일, 전화번호 등이 포함되지 않아야 함<br>• 잘 알려진 단어(love, happy 등) 또는 키보드 상에서 나란히 있는 문자열도 포함되지 않아야 함<br>• 이용자의 ID가 'KDHong'인 경우, 패스워드를 'KDHong12' 또는 'HongKD'가 포함되지 않아야 함 |
| 동일한 비밀번호 사용 제한 | • 비밀번호 변경시 이전 비밀번호를 교대로 사용하지 못하도록 제한하여야 함 |
| 비밀번호 노출 시 즉시변경 | • 비밀번호가 제3자에게 노출되었을 경우 즉시 새로운 비밀번호로 변경 |

**TIP**

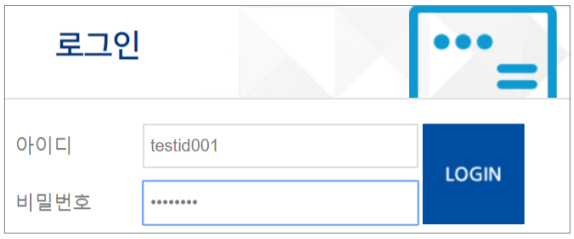
• 안전한 비밀번호 설정을 위해 한국인터넷진흥원(KISA)의 암호이용활성화 홈페이지(https://seed.kisa.or.kr)에서 제공하는 "패스워드 선택 및 이용 안내서"를 활용할 수 있다.

• 비밀번호 입력 시, 마스킹 조치를 적용

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

정보주체가 화면에서 비밀번호 입력 시, 마스킹 조치를 통해 불필요한 개인정보 노출을 방지할 수 있다.

**로그인 화면에서 비밀번호 표시제한 화면 예시**



## 구축 단계

• 정보주체의 비밀번호 설정 시, 아래의 내용을 고려하여 구현

**비밀번호 설정 기능 구현 시 고려사항**

① '비밀번호 작성규칙'을 안내하고 설정한 비밀번호가 올바르게 설정되었는지 확인

② 비밀번호 입력 시, 화면에 마스킹(*) 처리하여 표기

③ 비밀번호 변경기간 경과여부 확인을 위해 '비밀번호 설정 일시'를 저장 및 관리

※ 비밀번호 변경기간 경과여부 확인을 위한 목적이므로 필요 시 적용

※ 비밀번호 저장 시 안전한 알고리즘을 이용하여 저장하여야 함

• 정보주체의 비밀번호 변경 시, 아래의 내용을 고려하여 구현

**비밀번호 변경 기능 구현 시 고려사항**

① 정보주체가 로그인할 때, 비밀번호 설정일시와 현재시간을 비교하여 비밀번호 변경기간이 도래한 경우 비밀번호를 변경할 수 있도록 안내

※ 온라인 서비스에서 정보주체의 비밀번호 주기적 변경은 법적인 준수사항은 아니므로, 주기적인 변경 알림 기능은 제공하되 강제화할 필요는 없음

② 이전과 동일한 비밀번호의 사용제한을 위해 기존에 사용한 비밀번호를 일정개수 이상 저장하여, 과거에 사용했던 비밀번호와 동일할 경우 변경을 제한

• 정보주체의 비밀번호 초기화 시, 아래의 내용을 고려하여 구현

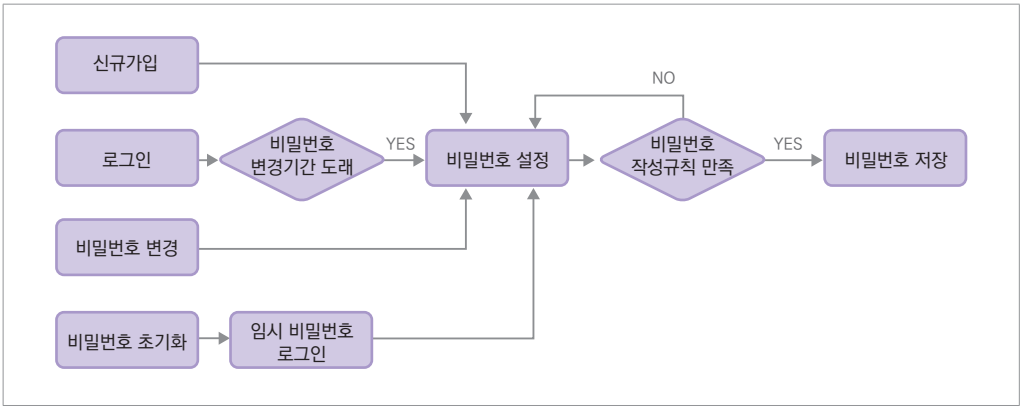
**비밀번호 초기화 기능 구현 시 고려사항**

① 아이핀, 휴대폰 본인인증 등 인증수단을 통해 비밀번호를 변경하려는 정보주체 본인임을 인증하는 절차를 수행

② 임시 비밀번호를 정보주체에게 발급할 경우, 난수값을 사용하고 정보주체가 회원가입 시 등록한 E-mail, 휴대폰 문자 등으로 전달

③ 임시 비밀번호를 이용하여 정보주체가 로그인 할 경우, 비밀번호를 변경한 후 서비스를 이용할 수 있도록 안내

비밀번호 작성규칙 적용 흐름도 예시




## 구현 예시

비밀번호 표시제한 로그인 화면

```
<body>
  <div class="login-form">
    <form action="${ctxPath }/user/login" method="post">
      <h2 class="text-center">로그인</h2>
      <div class="form-group">
        <input type="text" name="userId" class="form-control" placeholder="정보주체ID"
required="required">
      </div>
      <div class="form-group">
      // 비밀번호 입력 시 마스킹 처리될 수 있도록, HTML의 <input>태그 type속성을 “password”로 설정
        <input type="password" name="password" class="form-control" placeholder="비밀번호"
required="required">
      </div>
      <div class="form-group">
        <button type="submit" class="btn btn-primary btn-block">로그인</button>
      </div>
    </form>
  </div>
</body>
```

## 비밀번호의 작성 규칙을 점검하기 위한 메소드

### 1. 대문자 포함 여부 확인 메소드

```
/**
 * 주어진 문자열이 영어 대문자(A~Z)를 포함하고 있는지 여부를 돌려 준다.
 *
 * @param password 점검할 문자열.
 * @return 영어 대문자가 한 글자라도 포함되어 있으면 true, 그렇지 않으면 false.
 * password 변수의 값이 null거나 빈 문자열이면 false.
 */
public static boolean hasUpperCase(String password) {
  if ( password == null || password.length() == 0 ) return false;
  boolean result = false;
  int length = password.length();
  for (int i = 0; i < length; i++) {
    char ch = password.charAt(i);
    if ( 'A' >= ch && ch <= 'Z' ) {
      result = true;
      break;
    }
  }

  return result;
}}
```

### 2. 소문자 포함 여부 확인 메소드

```
/** 영어 소문자 정규식 패턴. */
private final static Pattern PATTERN_LOWER_CASE = Pattern.compile("[a-z]");
/**
 * 주어진 문자열이 영어 소문자(a~z)를 포함하고 있는지 여부를 돌려 준다.
 *
 * @param password 점검할 문자열.
 * @return 영어 소문자가 한 글자라도 포함되어 있으면 true, 그렇지 않으면 false.
 * password 변수의 값이 null거나 빈문자열이면 false.
 */
public static boolean hasLowerCase(String password) {
  if ( password == null || password.length() == 0 ) return false;
  Matcher matcher = PATTERN_LOWER_CASE.matcher(password);
  return matcher.find();
}
```

### 3. 숫자를 포함 여부 확인 메소드

```
/** 숫자 정규식 패턴. */
private final static Pattern PATTERN_DIGIT = Pattern.compile("[\\d]");
/**
 * 주어진 문자열이 숫자(0~9)를 포함하고 있는지 여부를 돌려 준다.
 *
 * @param password 점검할 문자열.
 * @return 숫자가 한 글자라도 포함되어 있으면 true, 그렇지 않으면 false.
 * password 변수의 값이 null거나 빈문자열이면 false.
 */
public static boolean hasDigit(String password) {
  if ( password == null || password.length() == 0 ) return false;
  Matcher matcher = PATTERN_DIGIT.matcher(password);
  return matcher.find();
}
```

### 4. 특수문자 포함 여부 확인 메소드

```
/** 특수 문자 정규식 패턴. */
private final static Pattern PATTERN_SPECIAL_CHAR = Pattern.compile("[\\p{Punct}]");
/**
 * 주어진 문자열이 특수문자(!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~)를 포함하고 있는지 여부를 돌려 준다.
 *
 * @param password 점검할 문자열.
 * @return 특수문자가 한 글자라도 포함되어 있으면 true, 그렇지 않으면 false.
 * password 변수의 값이 null거나 빈문자열이면 false.
 */
public static boolean hasSpecialChar(String password) {
  if ( password == null || password.length() == 0 ) return false;
  Matcher matcher = PATTERN_SPECIAL_CHAR.matcher(password);
  return matcher.find();
}
```

### 5. 비밀번호 구성 문자 집합 점검 메소드

```
/**
 * 주어진 문자열을 구성하고 있는 문자 영역({@link CharacterArea}) 목록을 돌려 준다.
 *
 * @param password 점검할 문자열.
 * @return 문자열의 구성 문자 영역에 해당하는 {@link CharacterArea}들의 목록.
 * @exception IllegalArgumentException 점검할 문자열이 null이거나 빈 문자열인 경우.
 */
public static List<CharacterArea> examineCharacterArea(String password) {
```

```
  if ( password == null || password.length() == 0 ) {
    throw new IllegalArgumentException("점검할 문자열이 넘어오지 않았습니다.");
  }
  ArrayList<CharacterArea> charAreaList = new ArrayList<CharacterArea>();
  if ( hasUpperCase(password) ) charAreaList.add(CharacterArea.UpperCaseLetters);
  if ( hasLowerCase(password) ) charAreaList.add(CharacterArea.LowerCaseLetters);
  if ( hasDigit(password) ) charAreaList.add(CharacterArea.Digits);
  if ( hasSpecialChar(password) ) charAreaList.add(CharacterArea.SpecialCharacters);

  return charAreaList;
 }
```

## ‘비밀번호 작성규칙’을 준수하였는지 점검

```
/**
 * 비밀번호가 '비밀번호 작성규칙'에 맞게 설정되었는지 점검하는 클래스.
 */
public class PasswordChecker {
  private ISecurityPolicy securityPolicy;
  private Encrypter passwordEncrypter;

  public PasswordChecker(ISecurityPolicy securityPolicy, Encrypter passwordEncrypter) {
    this.securityPolicy = securityPolicy;
    this.passwordEncrypter = passwordEncrypter;
  }
  /**
   * 주어진 비밀번호가 어떤 비밀번호 정책을 위반했는지를 점검한다.
   *
   * @param password 점검할 비밀번호.
   * @param account 점검할 비밀번호 사용 계정.
   *
   * @return 위반한 비밀번호 정책({@link PasswordViolation}) 목록.
   * @exception IllegalArgumentException 점검할 비밀번호 값이 null이 거나 빈 문자열일 때.
   */
  public List<PasswordViolation> checkPasswordPolicy(String password, IAccount account) {
    return checkPasswordPolicy(password, account, null);
  }

  /**
   * 주어진 비밀번호가 어떤 비밀번호 정책을 위반했는지를 점검한다.
```

```
   *
   * @param password 점검할 비밀번호.
   * @param account 점검할 비밀번호 사용 계정.
   *
   * @return 위반한 비밀번호 정책({@link PasswordViolation}) 목록.
   * @exception IllegalArgumentException 점검할 비밀번호 값이 null이 거나 빈 문자열일 때.
   */
  public List<PasswordViolation> checkPasswordPolicy(String password, IAccount account,
List<String> prevPasswordHashList) {
    if ( password == null || "".equals(password) ) {
      throw new IllegalArgumentException("점검할 비밀번호 값이 넘어오지 않았습니다.");
    }

    ArrayList<PasswordViolation> violationList = new ArrayList<PasswordViolation>();
    // 비밀번호에 사용할 수 없는 문자 점검
    if ( ! PasswordUtil.isAllPasswordCharacters(password) ) {
      violationList.add(PasswordViolation.UNACCEPTABLE_CHARACTER);
    }
    // 최소 길이 점검
    if ( password.length() < this.securityPolicy.getPasswordMinLength() ) {
      violationList.add(PasswordViolation.MINIMUM_LENGTH);
    }
    // 최소 문자 구성 점검
    if ( PasswordUtil.examineCharacterArea(password).size() < this.securityPolicy.
getPasswordMinConfigCount() ) {
      violationList.add(PasswordViolation.MINIMUM_CONFIGURATION);
    }

    return violationList;
  }
}
```

## 2.2 개인정보 전송 시 암호화 적용

### 기획 단계

• 개인정보를 전송할 경우 암호화를 적용하여야 함

**처리자 유형에 따른 개인정보 암호화 대상**

| 구 분 | 개인정보처리자 | 정보통신서비스 제공자등 |
|---|---|---|
| 비밀번호 | ○ | ○ |
| 고유식별정보 | ○ | ○ |
| 바이오정보 | ○ | ○ |
| 개인정보 (성명, 연락처 등) | - | ○ |
| 인증정보 | - | ○ |

※ 개인정보처리자의 경우 고유식별정보, 비밀번호, 바이오정보를 제외한 개인정보(성명, 연락처 등)은 전송구간 암호화 조치가 필수가 아니나, 개인정보의 위·변조 및 유·노출 등을 고려하여 암호화 조치를 권장

• 개인정보 전송구간(웹서버와 클라이언트 간 통신 등)에서의 암호화 방식으로는 TLS방식, 응용프로그램 방식 등을 고려할 수 있음

**웹 서버와 클라이언트 간 암호화 방식 비교**

| 방식 | 데이터 부분 암호화 | 개발 비용 |
|---|---|---|
| TLS 방식 | 지원하지 않음 | 낮음 |
| 응용프로그램 방식 | 지원함 | 높음 |

- **TLS 방식**: 웹페이지 전체를 암호화하는 방식이며, 웹브라우저에 기본적으로 내장된 TLS프로토콜을 이용

TLS 방식에서 나타나는 웹브라우저 자물쇠 표시


Internet Explorer 브라우저 Chrome 브라우저

- 응용프로그램 방식: 특정 데이터만을 선택적으로 암호화하여 전송할 수 있지만, 웹 브라우저 등에서 부가적인 프로그램 설치해야 함

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

개인정보를 네트워크를 통해 전송할 때에는 불법적인 노출 또는 위·변조 방지를 위해, 전송구간에 암호화를 적용하여 개인정보가 안전하게 전송될 수 있도록 지원해야 한다.

## 구축 단계

- 전체 웹페이지를 암호화하는 방법(https 적용) 및 암호화 대상 웹페이지를 구별하여 부분적으로 전송 시 암호화 가능

※ 부분적으로 암호화할 경우, 아래 예시 표와 같이 송·수신하는 페이지에 대하여 개인정보 전송 시 암호화를 적용할 수 있음

**암호화 적용 대상 페이지 선별 예시**

| 구분 | 암호화 적용 대상 페이지 |
|---|---|
| 입력 창 | 회원가입 창, 비밀번호 변경 창, 로그인 창 등 |
| 출력 창 | 회원 정보 조회 창(서버→정보주체), 회원 정보 변경 창(정보주체→서버) 등 |
| 전송 기능 | 쿠키 정보 조회 후 전송, 쿠키 설정 값 전송 |

TIP 참고

**쿠키 값 전송 시 암호화 적용**

- 쿠키에 개인정보를 기록 시, 개인정보를 암호화하여 쿠키에 기록
- 쿠키에 SECURE 속성을 설정하여 암호화가 적용된 상황에서만 전송되도록 조치

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
 {
   String command = request.getParameter("command");
   if (command.equals(LOGIN_CMD))
   {
      String userId = request.getParameter(USER_ID_PARM);
      String password = request.getParameter(PASSWORD_PARM);
       ...
      Cookie idCookie = new Cookie("id", userId);
      Cookie passwordCookie = new Cookie("password", password);

      idCookie.setSecure(true);
      passwordCookie.setSecure(true);

      //전송시 쿠키에 SECURE 옵션을 준후에 값을 저장
      //SSL 통신 - HTTPS 사용시에만 적용 가능함

      response.addCookie(idCookie);
      response.addCookie(passwordCookie);
   }
      ...
 }
```

- 개인정보 전송구간 암호화 적용 시, 안전한 암호화 알고리즘 및 암호 키를 선택해야 함

TIP

- 안전한 암호화 알고리즘 및 암호 키 설정을 위해 한국인터넷진흥원(KISA)의 암호이용활성화 홈페이지(https://seed.kisa.or.kr)에서 제공하는 “암호 알고리즘 및 암호 키 길이 이용 안내서”를 참고할 수 있다.

**구현 예시**

## 응용 프로그램 암호화 방식

### 1. 암호화 로그인 페이지 호출시 비대칭 암호키 생성

```
/**
 * 개인정보처리시스템의 로그인 요청을 담당하는 Controller.
 * 공개키 정보는 로그인 화면에서 사용하고, 개인키는 Session에 저장 후 복호화 할 때 사용한다.
 */
@Controller
@RequestMapping(value="/admin")
public class AdminSignInController {
  /**
   * 암호화 로그인 페이지 요청 처리.
   * @throws NoSuchAlgorithmException
   * @throws InvalidKeySpecException
   */
  @RequestMapping(value="/login-secured",method=RequestMethod.GET)
  public String showSecuredLoginPage(
      HttpServletRequest request
  ) throws NoSuchAlgorithmException, InvalidKeySpecException {
    KeyPairGenerator generator = KeyPairGenerator.getInstance("RSA");
    generator.initialize(2048);

    KeyPair keyPair = generator.genKeyPair();
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");

    PublicKey publicKey = keyPair.getPublic();
    PrivateKey privateKey = keyPair.getPrivate();
    HttpSession session = request.getSession();

    // 세션에 공개키의 문자열을 키로 하여 개인키를 저장한다.
    session.setAttribute("__rsaPrivateKey__", privateKey);

    // 공개키를 문자열로 변환하여 JavaScript RSA 라이브러리 넘겨준다.
    RSAPublicKeySpec publicSpec = (RSAPublicKeySpec) keyFactory
        .getKeySpec(publicKey, RSAPublicKeySpec.class);
    String publicKeyModulus = publicSpec.getModulus().toString(16);
    String publicKeyExponent = publicSpec.getPublicExponent().toString(16);
```

```
    request.setAttribute("publicKeyModulus", publicKeyModulus);
    request.setAttribute("publicKeyExponent", publicKeyExponent);

    return VIEW_LOGIN_SECURED;
  }
}
```

2. 브라우저 로그인 화면 및 JavaScript

```
<body>
  <div class="login-form">
    <form action="${ctxPath }/admin/secured-login"method="post">
      <h2 class="text-center">로그인</h2>
      <div class="form-group">
        <input type="text" name="userId" class="form-control" placeholder="정보주체ID"
required="required">
      </div>
      <div class="form-group">
        <input type="password" name="password" class="form-control" placeholder="비밀번호"
required="required">
      </div>
      <div class="form-group">
        <button type="submit"class="btn btn-primary btn-block">로그인</button>
      </div>
    </form>
  </div>
  <script>
  var rsaPublicKeyModulus ='${rsaPublicKeyModulus }';
  var rsaPublicKeyExponent ='${rsaPublicKeyExponent }';
  </script>
</body>


;(function(window, document, $) {
  var form = document.loginForm;
  $(form).submit(function(event) {
    event.preventDefault();

    var userId = form.userId.value;
    if (userId =='') {
      alert('아이디를 입력해 주세요');
      form.userId.focus();
```

```
    form.userId.select();
    return false;
  }

  var password = form.password.value;
  if ( password =='' ) {
    alert('비밀번호를 입력해 주세요');
    form.password.focus();
    form.password.select();
    return false;
  }

  var encryptedUserId = rsa.encrypt(userId);
  var encryptedPassword = rsa.encrypt(password);
  try {
    var rsa =new RSAKey();
    rsa.setPublic(rsaPublicKeyModulus, rsaPublicKeyExponent);

    // 정보주체ID와 비밀번호를 RSA로 암호화한다.
    encryptedUserId = rsa.encrypt(userId);
    encryptedPassword = rsa.encrypt(password);
  }
  catch(e) {
    alert(e);
    throw e;
  }

  form.userId.value = encryptedUserId;
  form.password.value = encryptedPassword;

  form.submit();

  form.userId.value = userId;
  form.password.value = password;

  return false;
 });
})(window, document, jQuery);
```

### 3. 서버측 로그인 요청 처리

```
/**
 * 개인정보처리시스템의 로그인 요청을 담당하는 Controller.
 * 암호화된 정보주체 ID, 비밀번호를 복호화하고, 로그인 모듈을 호출한다.
 */
@Controller
@RequestMapping(value="/admin")
public class AdminSignInController {
 private String decryptRsa(PrivateKey privateKey, String securedValue) throws Exception{
  Cipher cipher = Cipher.getInstance("RSA");
  byte[] encryptedBytes = hexToByteArray(securedValue);

  cipher.init(Cipher.DECRYPT_MODE, privateKey);
  byte[] decryptedBytes = cipher.doFinal(encryptedBytes);

  String decryptedValue = new String(decryptedBytes,"utf-8");// 문자 인코딩 주의.

  return decryptedValue;
 }

 /**
  * 암호화 로그인 요청 처리.
  * @throws InvalidKeySpecException
  * @throws NoSuchAlgorithmException
  */
 @RequestMapping(value="/login-secured",method=RequestMethod.POST)
 public String doSecuredLogin(
  @RequestParam("userId") String encryptedUserId,
  @RequestParam("password") String encryptedPassword,
  HttpServletRequest request
 ) throws NoSuchAlgorithmException, InvalidKeySpecException {
  HttpSession session = request.getSession();

  PrivateKey privateKey = (PrivateKey) session.getAttribute("__rsaPrivateKey__");

  if ( privateKey == null ) {
   request.setAttribute("message","암호화 비밀키 정보를 찾을 수 없습니다.");
   return showSecuredLoginPage(request);
  }

  String view = VIEW_LOGIN_SECURED;
```

```
    String userId = null;
    String password = null;
    try {
     // 정보주체ID, 비밀번호 복호화
     userId = decryptRsa(privateKey, encryptedUserId);
     password = decryptRsa(privateKey, encryptedPassword);

     // 로그인 로직을 이용한다.(저장된 비밀번호는 해시값이므로 비교 시 해시로 비교)
     // 단, 로그인 페이지로 돌아갈 때 암호화 로그인 페이지로 돌아간다.
     view = doLogin(userId, password, request);
     if ( VIEW_LOGIN.equals(view) ) {
       view = showSecuredLoginPage(request);
     }
    }
    catch (Exception e) {
     request.setAttribute("message", e.getLocalizedMessage());
     view = showSecuredLoginPage(request);
    }

    return view;
   }
  }
```

## 2.3 기타 개인정보 안전조치 적용

### 기획 단계

• URL 내 파라미터에 개인정보 포함되지 않도록 해야 함

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

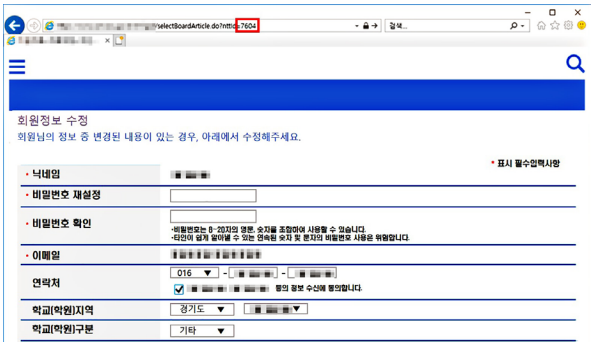
정보주체가 온라인 서비스 이용 시, URL을 통해 개인정보가 불필요하게 노출되지 않도록 URL 파라미터에 개인정보(ID, PW, 연락처 등)를 사용하지 않아야 한다.

• URL 내 포함된 파라미터 값 조작을 통해 타인의 개인정보를 조회하는 것을 방지해야 함

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

URL 파라미터에 포함된 개인정보 식별 값을 변경하여, 타인의 개인정보를 조회·변경할 수 없도록 비인가자에 대한 접근 통제 조치를 해야 한다.

**온라인 서비스를 통한 개인정보 유출사례**

| 사례 | 개인정보 유·노출 온라인 서비스 화면(예시) |
|---|---|
| URL 파라미터로 주민등록번호 노출 (GET방식 사용) |  |
| URL변경을 통한 타인의 개인정보 수정페이지 접근 |  |

• 불필요한 개인정보가 파일명, 소스코드에 포함되지 않도록 주의해야 함

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

파일명, 소스코드 등 개인정보처리시스템 구현 시 개인을 식별할 수 있는 정보를 포함하지 않도록 주의하여 향후 불필요하게 개인정보가 노출되지 않도록 해야 한다.

• 게시글 등의 임시저장 페이지를 통해 개인정보가 노출되지 않도록 접근통제, 주기적 자동 삭제 등의 조치를 적용해야 함

## 구축 단계

• 웹브라우저 표시줄에 개인정보가 포함된 파라미터 값이 보이지 않도록, **GET방식 보다는 POST방식으로 구현**

- 개인정보가 불필요하게 노출되는 것을 방지하기 위해, 회원을 구분하기 위한 URL 파라미터 값을 개인정보로(연락처, 생년월일 등) 설정하지 않도록 주의

※ 정보주체를 식별하는 값으로 ID, 학번 등 일련번호를 사용하여 구현

- 회원정보 조회/변경 페이지에 타인이 접근하지 못하도록 구현

※ URL 파라미터 변조를 통해 다른 사람의 개인정보를 조회·변경하지 못하도록 인증정보 확인 등 비인가자에 대한 접근 통제 수행

• 파일명, **소스코드(주석문 포함) 등 민감한 개인정보가 포함되지 않도록 구현**

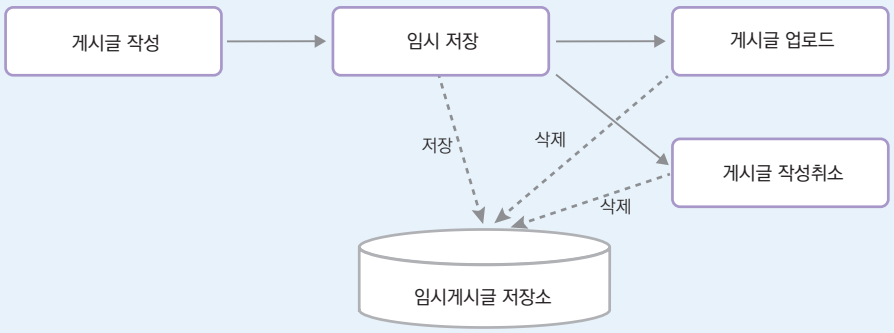
**소스코드에 개인정보가 포함되는 주요사례**

| 주요 사례 | 설 명 |
|---|---|
| 식별 값으로 개인정보 사용 | 화면에는 표시되지 않으나 소스코드 상에 개인정보(연락처, 여권번호 등)가 hidden 값으로 노출된 경우 |
| 파일명으로 개인정보 사용 | 정보주체의 회원정보 조회 창에서 사진을 표시하면서 사진 파일명이 주민번호로 되어 있어 소스코드 상에 노출 |
| 표시 창 설정으로 개인정보 표시제한 | 소스코드 상에는 전체 주민등록번호가 표시되나 표시 창(edit box 등) 설정으로 표시제한 되어 보이는 경우 |

• 게시글 작성 완료 및 작성 취소 시, 저장된 임시 저장 페이지가 바로 삭제되도록 구현해야 하며 일정 기간이 지난 임시 저장 페이지는 자동으로 삭제되도록 구현

TIP

• 게시판 임시저장 페이지 삭제조치 구현 흐름도 예시




| 게시글 기능 | 기능 설계(예시) |
|---|---|
| 임시저장 | • 임시 게시글을 저장하고 저장 시간을 기록<br>• 임시 게시글 저장 기간 경과 시 데이터 삭제 기능 구현<br>• 임시 게시글도 작성자만 조회할 수 있도록 접근통제 기능 구현 |
| 작성완료 | • 게시글 저장하고, 임시 저장 게시글을 삭제 |
| 작성취소/삭제 | • 게시글 및 임시 저장 게시글을 모두 삭제 |

## 구현 예시

### 회원정보 조회 및 수정

```
/**
 * 파라미터가 아닌 Session에 저장된 회원ID로 회원 정보를 조회한다.
 */
@Controller
@RequestMapping("/personal")
public class PersonalController {
  @Resource(name="userManager")
  private IUserManager userManager;

  /** 회원정보 변경 화면. */
  private final static String VIEW_MY_INFO_CHANGE_PAGE = "/personal/my-info";
  /**
```

```
  * 회원정보 변경 화면을 요청한다.
  *
  * @return 회원정보 변경 화면 View 경로.
  */
 @RequestMapping(value="/my-info", method=RequestMethod.GET)
 public String showPersonalInfoPage(
    HttpServletRequest request
 ) {
  User user = (User) request.getSession().getAttribute(User.KEY);

  User myinfo = this.userManager.getById(user.getId()); // 최신 정보를 조회

  request.setAttribute("myinfo", myinfo);

  return VIEW_MY_INFO_CHANGE_PAGE;
 }
}
```

## 게시글 임시 저장 시 일정 기간(24시간) 경과 후 자동 삭제

```
/**
 * 게시글 정보를 취급하기 위한 페이지 요청을 처리하는 Controller 클래스
 */
@Controller
@RequestMapping("/board-doc")
public class BoardDocHtmlController {
 @Resource(name="boardDocService")
 private IBoardDocService boardDocService;

 /** 임시로 저장된 게시글 삭제 실행 Timer. */
 @Resource(name="temporarySavedDocRemoveTimer")
 private Timer temporarySavedDocRemoveTimer;

 /**
  * 새 게시글 정보를 임시로 저장한다.
  *
  * @return 정보를 등록할 뷰 경로 "/board-doc/board-doc-new".
  */
 @RequestMapping(value="/temporarily",method=RequestMethod.POST)
```

```
public String saveBoardDocTemporarily(
  IBoardDoc boardDoc,
  HttpServletResponse response,
  Model model
) {
  String viewPage = null;
  try {
    boardDoc.setTemporarily(true);
    final String temporarilySavedBoordDocId = this.boardDocService.addBoardDoc(boardDoc);

    // 임시 저장이기 때문에 일정기간(24시간), 여전히 임시로 남아 있을 경우 삭제한다.
    Calendar cal = Calendar.getInstance();
    cal.add(Calendar.HOUR, 24); // 24시간 뒤
    Date executionTime = cal.getTime();

    this.temporarySavedDocRemoveTimer.schedule(new TimerTask() {
      @Override
      public void run() {
        IBoardDoc temporarilySavedBoardDoc =
            boardDocService.getBoardDoc(temporarilySavedBoardDocId);
        if ( temporarilySavedBoardDoc != null
            && temporarilySavedBoardDoc.isTemporarily() ) {
          boardDocService.removeBoardDoc(temporarilySavedBoardDocId);
        }

        // 1회만 실행하기 위해 취소
        cancel();
      }
    }, executionTime);
    viewPage = showViewPage(temporarilySavedBoardDocId, response, model);
  }
  catch (Exception e) {
    model.addAttribute("error", e);
    model.addAttribute("messageLevel","warning");
    model.addAttribute("message", e.getLocalizedMessage());
    model.addAttribute("boardDoc", boardDoc);
    viewPage = showNewPage();
  }

  return viewPage;
 }

}
```

# 3 정보주체 권리 보장 및 파기단계

| | |
|---|---|
| 항목명 | **정보주체 권리 보장 및 파기 절차 마련** |
| 요구사항 내용 | **3.1 개인정보 열람 및 정정**<br>- 정보주체의 개인정보 열람 및 정보수정<br>- 개인정보의 이용내역 통지<br>**3.2 개인정보 수집 등 동의 철회**<br>- 개인정보 수집·이용 및 제3자 제공 동의 철회 절차 구현<br>**3.3 개인정보 파기 및 분리보관**<br>- 개인정보 파기<br>- 개인정보의 분리보관 |
| 관련 근거 | • 개인정보 보호법 제21조(개인정보의 파기)<br>• 개인정보 보호법 제35조(개인정보의 열람)<br>• 개인정보 보호법 제36조(개인정보의 정정·삭제)<br>• 개인정보 보호법 제39조의6(개인정보의 파기에 대한 특례)<br>• 개인정보 보호법 제39조의7(이용자의 권리 등에 대한 특례)<br>• 개인정보 보호법 제39조의8(개인정보 이용내역의 통지)<br>• 개인정보 보호법 시행령 제48조의4(개인정보의 파기 등) |

## 3.1 개인정보 열람 및 정정

### 기획 단계

• 정보주체가 개인정보를 언제든지 열람·정정할 수 있도록, 회원가입 절차만큼 쉬운 방법으로 **개인정보 열람·정정 절차를 마련**

**PbD 적용설계** | (제⑥원칙) 처리과정에 대한 가시성·투명성 유지

개인정보처리자는 개인정보의 정확성, 안전성 및 최신성을 확보하기 위하여 정보주체가 스스로 개인정보를 확인(열람)하고 정확한 정보를 입력할 수 있는 절차 및 방법(정정)을 마련해야 한다.

**회원정보 수정페이지 예시**

## 마이페이지

**기본 입력사항 (* 필수입력사항)**

| | |
|---|---|
| 이름 * | 홍길동 |
| 생년월일 * | 1975년 04월 12일 |
| 성별 * | 남성 |
| 아이디 * | privacy_support |
| 비밀번호 * | 8~16자 영 대소문자, 숫자, 특수문자 이용 |
| 비밀번호 확인 * | |

**연락처 입력 (* 필수입력사항)**

| | |
|---|---|
| 이메일 * | privacy_support @ korea.kr 직접입력 중복확인 |
| 집전화 | - - |
| 직장전화 | - - |
| 휴대전화 * | 010 - 0000 - 0000 |
| 우편번호 * | 우편번호 주소검색 |
| 상세주소 * | |

수정하기

## 구축 단계

- 일반적으로, 정보주체가 웹사이트를 통해 회원가입을 한 경우에는 웹사이트 내에 '회원정보 수정' 메뉴를 통해 개인정보 열람 및 정정 절차를 구현할 수 있음
- 회원 정보 수정페이지에서는 필수적으로 수집하는 개인정보에 별표 표시(*)하여 필수·선택 정보를 구분하여 표시하는 것이 바람직 함

**TIP**

- **개인정보 정정 시, 오기입 방지를 위한 유효성 검사**

  개인정보 정정 시, 개인정보를 잘못 입력하는 경우(오타방지 등)를 위해 개인정보 유형에 따른 정규식 패턴을 반영하여 유효성 검사를 수행할 수 있다.

| 개인정보 유형 | 개인정보 패턴 예시 |
|---|---|
| 이메일 주소 | /^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA-Z]{2,3}$/i; |
| 핸드폰번호 | 01[016789][-~.[:space:]][0-9]{3,4}[-~.[:space:]][0-9]{4} |
| 주민등록번호 | ([01][0-9]{5}[[:space:],~-]+[1-4][0-9]{6}\|[2-9][0-9]{5}[[:space:],~-]+[1-2][0-9]{6}) |
| 신용카드번호 | [34569][0-9]{3}[-~.[:space:]][0-9]{4}[-~.[:space:]][0-9]{4}[-~.[:space:]][0-9]{4} |
| 계좌번호 | ([0-9]{2}[-~.[:space:]][0-9]{2}[-~.[:space:]][0-9]{6}\|[0-9]{3}[-~.[:space:]]([0-9]{5,6}[-~.[:space:]][0-9]{3}\|[0-9]{6}[-~.[:space:]][0-9]{5}\|[0-9]{2,3}[-~.[:space:]][0-9]{6}\|[0-9]{2}[-~.[:space:]][0-9]{7}\|[0-9]{2}[-~.[:space:]][0-9]{4,6}[-~.[:space:]][0-9]\|[0-9]{5}[-~.[:space:]][0-9]{3}[-~.[:space:]][0-9]{2}\|[0-9]{2}[-~.[:space:]][0-9]{5}[-~.[:space:]][0-9]{3}\|[0-9]{4}[-~.[:space:]][0-9]{4}[-~.[:space:]][0-9]{3}\|[0-9]{6}[-~.[:space:]][0-9]{2}[-~.[:space:]][0-9]{3}\|[0-9]{2}[-~.[:space:]][0-9]{2}[-~.[:space:]][0-9]{7})\|[0-9]{4}[-~.[:space:]]([0-9]{3}[-~.[:space:]][0-9]{6}\|[0-9]{2}[-~.[:space:]][0-9]{6}[-~.[:space:]][0-9])\|[0-9]{5}[-~.[:space:]][0-9]{2}[-~.[:space:]][0-9]{6}\|[0-9]{6}[-~.[:space:]][0-9]{2}[-~.[:space:]][0-9]{5,6}) |

## 구현 예시

### 개인정보 입력값 유효성 점검 함수구현

```
/**
 * 입력된 개인정보의 유효성을 점검한다.
 *
 * @param window
 * @param document
```

```
 * @returns
 */
;(function(window, document) {
  var Validator =function() {};
  Validator =new Validator();
  var validateFunctions = {};
  /**
   * 주어진 값의 유효성을 점검한다.
   *
   * @param value 유효성 점검을 위한 값 또는 Form Element.
   * @param type 유효성 점검할 종류. 예) email, url 등
   * value 변수의 값이 Form Element인 경우 type 변수의 값이 없으면,
   * value 요소의 data-type 속성 값, type 속성값 순으로 얻는다.
   * 점검하지 않고 무조건 true를 돌려준다.
   * @param 유효성 점검 결과 유효하면 true, 그렇지 않으면 false.
   */
  Validator.validate =function(value, type) {
    var isFormElement =!!value.form;
    var val =isFormElement ?value.value :value;
    if ( !type &&isFormElement ) {
      type =value.getAttribute('data-type')
          ||value.getAttribute('type');
    }

    // 일단 검증할 종류 값이 없다면 유효한 것으로 판단한다.
    if ( !type ) return true;

    var validateFunc =validateFunctions[type];

    // 유효성 점검 함수가 있으면 함수 호출한 결과를 돌려주고,
    // 점검 함수가 없으면 true를 돌려준다.
    return validateFunc ?validateFunc(val) :true;
  };

  /**
   * 주어진 유형에 대한 유효성 점검 함수를 등록한다.
   *
   * @param type 유효성 점검할 타입값
   * @param func 유효성 점검할 함수
   */
  Validator.setValidator(type, func) {
```

```
  if ( typeof(type) !='string' ) {
    alert('type 파라미터의 값이 string 객체가 아닙니다.');
    return;
  }
  if ( typeof(func) !='function' ) {
    alert('func 파라미터의 값이 function 객체가 아닙니다.');
    return;
  }

  validateFunctions[type] =func;
};

// 휴대전화번호 함수 등록.
var cellphoneNoValidator =function(phoneNo) {
  if ( phoneNo ==undefined ||phoneNo ==null ||phoneNo =='' ) return true;
  return phoneNo.match(cellphoneNoValidator.regexp);
};
cellphoneNoValidator.regexp = /01[016789][-~\\s][0-9]{3,4}[-~\\s][0-9]{4}/g;
Validator.setValidator('cellphone', cellphoneNoValidator);

// 이메일 주소 점검 함수 등록.
var emailValidator =function(email) {
  if ( email ==undefined ||email ==null ||email =='' ) return true;
  return email.match(emailValidator.regexp);
};
emailValidator.regexp = /^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA
-Z]{2,3}$/i;
Validator.setValidator('email', emailValidator);

// URL 점검 함수 등록.
var urlValidator =function(url) {
  if ( url ==undefined ||url ==null ||url =='' ) return true;
  return url.match(urlValidator.regexp);
};
urlValidator.regexp =/^(?:(http|https|ftp|mailto)?:\/\/)?[\w.-]+(?:\.[\w\.-]+)+[\w\-\._~:/?#[\]@!
\$&'\(\)\*\+,;=.]+$/i;
Validator.setValidator('url', urlValidator);

// 신용카드번호 점검 함수 등록.
var creditCardNoValidator =function(cardNo) {
  if ( cardNo ==undefined ||cardNo ==null ||cardNo =='' ) return true;
```

```
  return cardNo.match(creditCardNoValidator.regexp);
 };
 creditCardNoValidator.regexp = /[34569][0-9]{3}[-~\\s][0-9]{4}[-~\\s][0-9]{4}[-~\\s][0-9]
{4}/g;
 Validator.setValidator('credit-card', creditCardNoValidator);

 // 계좌번호 점검 함수 등록
 var accountNoValidator =function(cardNo) {
  if ( cardNo ==undefined ||cardNo ==null ||cardNo =='' ) return true;
  return cardNo.match(accountNoValidator.regexp);
 }
 accountNoValidator.regexp =new RegExp('('
   +'[0-9]{2}[-~\\s][0-9]{2}[-~\\s][0-9]{6}'
   +'|[0-9]{3}[-~\\s]([0-9]{5,6}[-~\\s][0-9]{3}'
   +'|[0-9]{6}[-~\\s][0-9]{5}'
   +'|[0-9]{2,3}[-~\\s][0-9]{6}'
   +'|[0-9]{2}[-~\\s][0-9]{7}'
   +'|[0-9]{2}[-~\\s][0-9]{4,6}[-~\\s][0-9]'
   +'|[0-9]{5}[-~\\s][0-9]{3}[-~\\s][0-9]{2}'
   +'|[0-9]{2}[-~\\s][0-9]{5}[-~\\s][0-9]{3}'
   +'|[0-9]{4}[-~\\s][0-9]{4}[-~\\s][0-9]{3}'
   +'|[0-9]{6}[-~\\s][0-9]{2}[-~\\s][0-9]{3}'
   +'|[0-9]{2}[-~\\s][0-9]{2}[-~\\s][0-9]{7}'
   +'|[0-9]{4}[-~\\s]([0-9]{3}[-~\\s][0-9]{6}'
   +'|[0-9]{2}[-~\\s][0-9]{6}[-~\\s][0-9])'
   +'|[0-9]{5}[-~\\s][0-9]{2}[-~\\s][0-9]{6}'
   +'|[0-9]{6}[-~\\s][0-9]{2}[-~\\s][0-9]{5,6}'
   +')', 'g');
 Validator.setValidator('back-account', accountNoValidator);

 window.Validator =Validator;
})(window, document);
```

## 개인정보 유효성 검사기능 구현

```
/**
 * 정보주체 자신의 정보를 수정하기 위한 웹페이지
 */
;(function(window, document, $) {
  var form =document.form;

  $(form).submit(function(event) {
    event.preventDefault();

    var isAllValid =true;

    // 이메일 주소 점검
    if ( !Validator.validate(form.email.value, 'email') ) {
      showInvalidMessage(form.email, '이메일 주소가 형식에 맞지 않습니다.');
      isAllValid =false;
    }

    // 휴대전화번호 점검
    if ( !Validator.validate(form.cellPhoneNo.value, 'cellphone') ) {
      showInvalidMessage(form.email, '핸드폰 번호가 형식에 맞지 않습니다.');
      isAllValid =false;
    }

})(window, document, jQuery);
```

## 3.2 개인정보 제공 동의 철회

### 기획 단계

- 개인정보 수집·이용 항목에 대하여, **정보주체가 제공한 개인정보를 언제든지 동의 철회할 수 있는 절차 마련**
  - 회원탈퇴, 수집·이용 동의 철회절차는 회원가입 절차만큼 쉬운 방법으로 정보주체에게 제공해야 함

**PbD 적용설계 | (제⑦원칙) 이용자 프라이버시 존중**

정보주체가 개인정보 수집·이용 동의한 내역에 대해 철회할 수 있는 절차를 마련하여, 정보주체의 권리를 행사할 수 있도록 해야 한다.

### 구축 단계

- 선택 동의 항목(마케팅 정보 활용, 광고성 정보수신 등) 및 제3자 제공에 대한 동의의 경우, 회원 탈퇴와 무관하게 동의철회가 가능하므로 정보주체가 수시로 동의 상태를 변경할 수 있도록 구현해야 함
  - 온라인 서비스 내 '마이페이지', '이용동의 현황' 등의 메뉴구성을 통해 정보주체에게 동의 철회기능을 제공

    ※ 이 경우, 동의사실 및 동의철회 등 변경 이력에 대하여 '동의사실 여부', '동의변경 일시'를 기록

- '회원 탈퇴' 메뉴를 통해 정보주체가 회원가입 시 제공했던 개인정보 수집·이용에 대한 동의 철회 절차를 구현할 수 있음

**참고 회원탈퇴 시 고려사항**

① 회원탈퇴 절차 : 웹사이트 회원가입 절차 보다 간소하게 구성

② 회원탈퇴 시, 정보주체 인증 수행 : 패스워드 등 최소 1회 이상 수행

③ 회원탈퇴에 대한 주의사항 고지

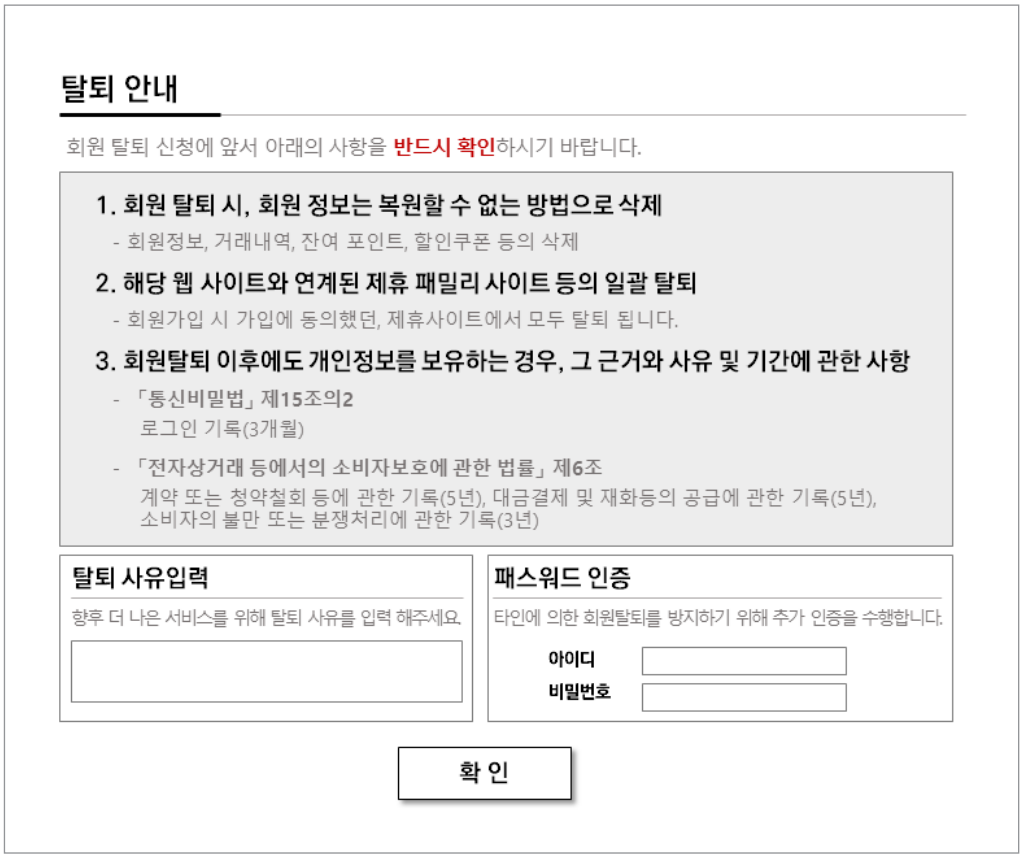
※ 회원탈퇴로 인해 발생할 수 있는 모든 직·간접적 영향 및 피해에 대한 주의사항 고지

※ 탈퇴 이후 법률 근거 등에 의해 개인정보를 보유하는 경우, 보유 기간과 근거를 안내

④ 회원탈퇴 사유 입력 : 웹사이트 자체 규정에 따라 입력 여부 결정

## 구현 예시

회원탈퇴 화면 예시




## 3.3 개인정보 파기 및 분리보관

### 기획 단계

- 개인정보가 불필요하게 된 경우, 정당한 사유가 없는 한 그로부터 **5일 이내**에 개인정보를 파기하여야 함
  - 개인정보는 다시 복원하거나 재생할 수 없는 형태로 완벽히 파기해야 함

**개인정보가 불필요하게 된 경우**

1) 개인정보처리자가 당초 고지하고 동의를 받았던 보유 기간의 경과
2) 동의를 받거나 법령 등에서 인정된 수집·이용·제공 목적의 달성
3) 회원탈퇴, 제명, 계약관계 종료, 동의철회 등에 따른 개인정보처리의 법적 근거 소멸
4) 개인정보처리자의 폐업·청산
5) 대금 완제일이나 채권소멸시효기간의 만료

**PbD 적용설계 | (제⑤원칙) 개인정보 생애주기 전체에 대한 보호**

불필요한 개인정보를 계속해서 보유할 경우 개인정보의 유출과 오용 가능성이 높아지므로, 개인정보가 불필요하게 되었을 때 지체없이 파기하여 개인정보를 안전하게 보호해야 한다.

- 회원탈퇴 이후에도 다른 법령에 따라 보존해야 하는 경우, 해당 개인정보를 다른 개인정보와 분리하여서 저장·관리

**개인정보 보존의무를 규정하고 있는 입법례**

| 관련법률 | 법적 의무 보관 기간 |
|---|---|
| 「전자상거래 등에서의 소비자보호에 관한 법률」제6조 | ① 표시·광고에 관한 기록 : 6월<br>② 계약 또는 청약철회 등에 관한 기록 : 5년<br>③ 대금결제 및 재화등의 공급에 관한 기록 : 5년<br>④ 소비자의 불만 또는 분쟁처리에 관한 기록 : 3년 |
| 「통신비밀보호법」 제15조의2 | ① 가입자 전기통신일시, 전기통신개시·종료시간, 상대방 가입자번호, 사용도수, 발신기지국 위치추적자료 : 12개월<br>② 위의 자료 중 시외·시내전화역무와 관련된 통신사실확인자료 : 6개월<br>③ 컴퓨터통신 또는 인터넷의 로그기록자료, 정보통신기기의 위치를 확인할 수 있는 접속지 추적자료: 3개월 |
| 「의료법」 시행규칙 제15조* | ① 환자 명부 : 5년<br>② 진료기록부 : 10년<br>③ 처방전 : 2년<br>④ 수술기록 : 10년<br>⑤ 검사소견기록 : 5년<br>⑥ 방사선사진 및 그 소견서 : 5년<br>⑦ 간호기록부 : 5년<br>⑧ 조산기록부: 5년<br>⑨ 진단서 등의 부본 (진단서·사망진단서 및 시체검안서 등을 따로 구분하여 보존할 것) : 3년 |

* 다만, 계속적인 진료를 위하여 필요한 경우에는 1회에 한정하여 위의 각 항목별 법적 의무 보관 기간의 범위에서 그 기간을 연장하여 보존할 수 있음

• 특히, 정보통신서비스 제공자등은 **1년 동안 미이용하는 이용자의 개인정보에 대하여 즉시 파기**하거나 다른 이용자의 개인정보와 분리하여 별도로 저장·관리해야 함

- 이 경우, 기간 만료 30일 전까지 이용자에게 이메일 등 방법으로 고지해야 함

**1년간 미이용자 파기(분리 보관)을 위해 고지해야 할 사항**

1) 개인정보가 파기(혹은, 분리 보관)되는 사실,
2) 기간 만료일
3) 파기(혹은, 분리 보관)되는 개인정보 항목

## 구축 단계

• 정보주체의 개인정보 파기 시, 아래의 내용을 고려하여 구현

**개인정보 파기 기능 구현 시 고려사항**

① 회원탈퇴 시, 탈퇴한 회원의 개인정보가 개인정보처리시스템에서 자동으로 삭제될 수 있도록 구현
※ 데이터가 복구되지 않도록 데이터베이스 내에 탈퇴한 회원의 원본(Raw) 데이터를 삭제(Delete)하거나 Null 값으로 덮어쓰기를 수행하도록 구현

② 보유기간 도래, 목적달성 등 파기 절차를 자동화하여 불필요한 개인정보는 자동으로 삭제될 수 있도록 구현

• 정보주체의 개인정보 분리보관 시, 아래의 내용을 고려하여 구현

**개인정보 분리 보관 시 고려사항**

① 별도로 보관하는 DB(테이블)의 경우 일반 개인정보 DB의 접근 권한과 다르게 설정하여 불필요한 접근·조회·유출을 방지

② 개인정보를 별도 보관 시, 개인정보 항목별로 법령 기간에 따른 보유 기간을 데이터베이스에 함께 저장하여 보유 기간이 경과한 개인정보는 자동 삭제하도록 구현

# 제2절 개인정보처리시스템 기획·구축

## 개요

개인정보처리시스템을 통해 개인정보 유·노출 등 피해를 최소화하기 위해서는 시스템 개발 단계에서부터 개인정보보호 조치가 이루어져야 한다. 개인정보처리시스템을 기획하는 단계에서 접근 권한 관리, 접근통제, 개인정보 암호화, 접속기록 관리 등 개인정보의 안전성을 확보하기 위한 조치사항을 먼저 확인해야 한다.

본 절에서는 개인정보취급자가 이용하는 개인정보처리시스템 기획 시 준수해야 할 법적 요구사항을 안내하고, 구축 시 적용해야 할 개인정보 보호조치 및 예시 소스코드를 제공함으로써 개인정보처리시스템 개발자가 쉽게 참조할 수 있도록 하였다.

## 기본 원칙

개인정보처리시스템 구축 시 개인정보보호를 위해 검토하고 확인해야 할 기본 원칙은 아래와 같다.

- 개인정보처리시스템에 대한 접근 권한 설정 및 내역 보관 기능을 개발해야 함
- 개인정보취급자가 안전한 비밀번호를 설정할 수 있도록 비밀번호 작성규칙을 수립하고 적용해야 함
- 개인정보처리시스템에 대한 비인가 접근을 방지하기 위해 접근통제 및 접근제한 조치를 반영하여 개발해야 함
- 개인정보 저장 시 안전한 알고리즘으로 암호화되도록 개발해야 함
- 개인정보취급자가 시스템에 접속한 이력을 남기고 별도로 저장·관리하도록 개발해야 함
- 개인정보를 조회, 출력 등 기능 구현 시 불필요하거나 과도한 개인정보가 검색되지 않도록 개발해야 함
- 마스킹처리 등 개인정보 표시제한 조치 시 일관성을 고려하여 개발해야 함

# 1 접근 권한 관리

| | |
|---|---|
| 항목명 | 개인정보처리시스템 접근 권한 관리 |
| 요구사항 내용 | **1.1 접근 권한 설정**<br>- 개인정보취급자 계정 발급<br>- 접근 권한의 차등 부여<br>- 접근 권한의 변경/말소<br>- 접근 권한 내역(부여/변경/말소) 기록 및 보관<br>**1.2 개인정보취급자 계정의 로그인 실패 및 장기 미접속에 따른 접근제한**<br>- 계정정보 및 비밀번호 입력 오류 시 접근제한<br>- 장기 미접속자 계정 잠금<br>**1.3 개인정보취급자의 비밀번호 작성규칙 적용**<br>- 개인정보취급자의 안전한 비밀번호 작성규칙 수립·적용<br>- 개인정보 수집·이용을 위한 동의 절차 마련 |
| 관련 근거 | • 개인정보 보호법 제29조(안전조치의무)<br>• 개인정보 보호법 시행령 제30조(개인정보의 안전성 확보 조치)<br>• 개인정보 보호법 시행령 제48조의2(개인정보의 안전성 확보 조치에 대한 특례)<br>• 개인정보의 안전성 확보조치 기준 제5조(접근 권한의 관리)<br>• 개인정보의 기술적·관리적 보호조치 기준 제4조(접근통제) |

## 1.1 접근 권한 설정

### 기획 단계

• 개인정보처리시스템에 접속할 수 있는 계정은 개인정보취급자 별로 발급하고, 다른 개인정보취급자와 공유되지 않도록 해야 함

- 개인정보취급자별 계정(ID 등)을 발급하여, 개인정보 처리에 따른 문제 발생 시 계정을 기반으로 처리 내역에 대한 책임 소재를 파악할 수 있어야 함

• 개인정보처리시스템에 대한 접근 권한을 업무 수행에 필요한 최소한의 범위로 업무 담당자에게 차등적으로 부여

- 개인정보처리시스템의 데이터베이스(DB)에 직접 접근하는 운영·관리자에게 한정하는 등의 접근통제를 위한 안전조치를 취하여야 함

• 인사이동, 퇴직 등으로 인해 개인정보취급자가 변경되었을 경우, 지체없이* 개인정보처리시스템의 접근 권한을 변경·말소

* '지체없이'란, 정당한 사유가 없는 한 5일 이내를 의미함

**TIP**

• 인사시스템과 연동을 통해 자동으로 반영하는 것을 고려할 수 있으며, 자동화가 어려울 경우 운영 측면에서 관리적인 절차를 마련할 수 있다.
(예, 퇴직 점검표에 개인정보취급자 계정의 말소항목 반영)

**PbD 적용설계 | (제①원칙) 사후조치가 아닌 사전예방**

인사이동, 퇴직 등으로 개인정보취급자가 변경되었을 경우, 인가되지 않은 자가 개인정보처리시스템에 접근하지 못하도록 이전 개인정보취급자의 계정을 변경·말소 조치해야 한다.

• 개인정보취급자별 **접근 권한의 부여/변경/말소 내역을 3년간 기록 및 보관** (단, 정보통신서비스 제공자등의 경우 최소 5년간 보관)

**접근 권한 내역 기록·보관 항목 예시**

| 구분 | 설명 |
|---|---|
| ID | 접근 권한이 부여·변경·말소된 대상 ID |
| 대상자 식별정보 | 사후에 해당 개인정보취급자를 확인할 수 있는 정보(이름, 사번 등)<br>※ ID 등을 통해 다른 정보와 결합하여 추적이 가능하다면 별도로 저장하지 않을 수 있음 |
| 접근 권한 정보 | 부여·변경·말소된 접근 권한에 관한 정보(접근 권한명 등)<br>※ 조직별, 그룹별, 역할별, 개인정보취급자별 등 권한부여 방식에 따라 다양하게 표현할 수 있지만, 해당 접근 권한의 상세내용(대상 메뉴/화면 및 입력/조회/변경/삭제/출력/다운로드 등 세부 권한)을 확인할 수 있어야 함 |
| 유형 | 접근 권한 부여/변경/말소 |
| 신청사유 | 접근 권한 부여·변경 또는 말소 사유(예 : 신규입사, 조직변경, 퇴사 등) |
| 신청자 정보 | 해당 접근 권한의 신청자<br>※ 외부자 등에 따라 대리 신청이 가능한 경우 신청자 정보를 기록 |
| 신청일시 | 접근 권한의 부여·변경·말소를 신청한 일시(YYYYMMDD HH:MM:SS) |
| 승인자 정보 | 해당 접근 권한 신청에 대한 승인자<br>※ 내부적으로 접근 권한 승인 절차가 존재하는 경우 승인자 정보를 기록 |
| 승인일시(적용일시) | 접근 권한의 부여·변경·말소를 승인한 일시(YYYYMMDD HH:MM:SS) |

## 구축 단계

- 개인정보취급자별 계정을 각각 부여하고 계정 발급내역 및 상태 관리

※ 계정관리 내역으로는 '개인정보취급자 정보', '계정상태(활성·비활성 여부 등)', '계정 생성·삭제·비활성화된 일시(YYYYMMDD HH:MM:SS)' 등을 기록할 수 있음

### 계정 관리내역 예시

| 아이디 | 이름 | 부서명 | ... | 계정상태 | 계정생성 일시 | 생성자 | 비활성화 일시 | 작업자 |
|---|---|---|---|---|---|---|---|---|
| gdhong | 홍길동 | IT팀 | | 활성(Y) | 20190112 14:30:20 | SYSTEM | - | - |
| cskim | 김철수 | IT팀 | | 활성(Y) | 20190220 10:22:01 | gdhong | - | - |
| yhkim | 김영희 | 고객팀 | | 비활성(N) | 20190228 17:33:50 | gdhong | 20190420 13:55:20 | gdhong |
| ... | | | | | | | | |

### 접근 권한 설정 시 고려사항

① 개인정보처리시스템 기능(메뉴)별 개인정보 처리 내용을 식별
※ 조회, 쓰기, 수정, 삭제, 출력, 다운로드 등

② 업무에 따라 최소한의 권한만 부여될 수 있도록 접근 권한 그룹 정의한다. (예: 최고관리자, 회원관리자, 게시판 관리자 등)
※ 접근 권한 형태는 개인정보취급자별, 조직별, 그룹별, 역할별 등 조직의 특성에 따라 설계 가능

③ 특히, 대량의 개인정보 유출 위험이 있는 '다운로드' 권한에 대해서는 업무상 필요한 경우로 한정하여 권한을 정의하는 것이 바람직 함

### 접근 권한 규칙설정(예시)

| 메뉴명 | | 개인정보 처리 여부 | 권한 그룹 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 최고 관리자 | | | | | 회원 관리자 | | | | | 게시판 관리자 | | | | | ... |
| 1단계 | 2단계 | | 조회 | 쓰기 | 수정 | 삭제 | 다운로드 | 조회 | 쓰기 | 수정 | 삭제 | 다운로드 | 조회 | 쓰기 | 수정 | 삭제 | 다운로드 | ... |
| 회원관리 | 회원정보 관리 | O | V | V | V | V | V | V | V | V | V | - | - | - | - | - | - | |
| | 문의/상담 관리 | O | V | V | V | V | V | V | V | V | V | - | - | - | - | - | - | |
| | 회원 통계 | O | V | V | V | V | V | V | V | V | V | - | - | - | - | - | - | |
| 게시판 관리 | 공지사항 | - | V | V | V | V | V | V | - | - | - | - | V | V | V | V | - | |
| | 자유게시판 | O | V | V | V | V | V | V | - | - | - | - | V | V | V | V | - | |
| ... | ... | | | | | | | | | | | | | | | | | |

• 침해사고 발생 시 원인 분석 등을 위하여 접근 권한 관련 내역을 사후에 추적할 수 있도록 **접근 권한 부여/변경/말소 내역을 기록**

**TIP**

• 향후, 접근 권한 변동 내역을 확인하기 위해서는 기존의 접근 권한 내용을 변동된 사항으로 수정(UPDATE)하는 방식이 아닌, 신규 기록(INSERT)하는 방식으로 저장해야 한다.

**접근 권한 내역·보관 기록 예시**

| 번호 | 사용자ID | 사용자명 | 권한명 | 권한ID | 유형 | 일자 | 작업자 | 사유 | ... |
|---|---|---|---|---|---|---|---|---|---|
| ... | | | | | | | | | |
| 51503 | cskim | 김철수 | 회원관리자 | S0002 | 부여 | 20190220 10:22:01 | gdhong | 담당업무 변경 | ... |
| 51504 | yhkim | 김영희 | 상담관리자 | C0005 | 부여 | 20181210 09:50:33 | gdhong | 상담팀 입사 | ... |
| 51505 | yhkim | 김영희 | 상담관지라 | C0005 | 말소 | 20190420 13:55:20 | gdhong | 퇴사 | ... |
| ... | | | | | | | | | |

**구현 예시**

### 개인정보취급자 접근 권한 부여/말소기능 구현

1. 접근권한 부여

```
/**
 * 개인정보취급자에게 자원에 대한 권한을 부여한다.
 *
 * @param roleId 권한을 부여할 개인정보취급자 ID.
 * @param resourceId 권한 부여 대상 자원의 ID.
 * @param actions 권한 부여 대상 행위 목록.
 */
@RequestMapping(value="/user/{userId}/resource/{resourceId}", method=RequestMethod.POST)
public String addUserAuthority(
  @PathVariable("userId") String userId,
  @PathVariable("resourceId") String resourceId,
  @RequestParam("reason") String reason,
  @RequestParam("action") ServiceType [] actions,
  HttpServletRequest request
) {
  User sessionUser = (User) request.getSession().getAttribute(User.KEY);
```

```
  if ( ! sessionUser.isAdmin() ) {
    throw new HasNoAuthorityException("권한을 부여할 수 있는 관리자가 아닙니다.");
  }

  ArrayList<IAuthority> authorityList = new ArrayList<IAuthority>();
  for (ServiceType service : actions) {
    authorityList.add(new Authority(resourceId, service));
  }
  User user = this.userManager.getById(userId);
  this.authorityManager.addAuthority(sessionUser.getId(), user, reason, authorityList);

  return showUserAuthorityManagePage();
 }
```

2. 접근권한 말소

```
 /**
  * 개인정보취급자에게 부여했던 특정 자원에 대한 권한을 말소한다.
  *
  * @param roleId 권한을 부여할 역할의 ID.
  * @param resourceId 권한의 대상이 되는 자원의 ID.
  * @param operators
  */
 @RequestMapping(value="/user/{groupId}/resource/{resourceId}", method=RequestMethod.DELETE)
 public String removeUserAuthority(
    @PathVariable("userId") String userId,
    @PathVariable("resourceId") String resourceId,
    @RequestParam("reason") String reason,
    @RequestParam("action") ServiceType [] actions,
    HttpServletRequest request
 ) {
  User sessionUser = (User) request.getSession().getAttribute(User.KEY);

  if ( ! sessionUser.isAdmin() ) {
    throw new HasNoAuthorityException("권한을 말소할 수 있는 관리자가 아닙니다.");
  }

  ArrayList<IAuthority> authorityList = new ArrayList<IAuthority>();
  for (ServiceType service : actions) {
    authorityList.add(new Authority(resourceId, service));
  }
  User user = this.userManager.getById(userId);
  this.authorityManager.removeAuthority(sessionUser.getId(), user, reason, authorityList);

  return showUserAuthorityManagePage();
 }
```

## 개인정보취급자 권한 부여/변경/말소 이력 관리기능 구현

### 1. 접근권한 부여/변경/말소 이력 정보 클래스 선언

```
/**
 * 권한 부여/변경/말소에 대한 로그 정보를 담는 클래스.
 */
public class AuthorityLog implements IAuthorityLog {

  private AuthorityChangeType authorityLogType;      //권한로그 유형
  private String resourceId;                         //대상 자원 ID
  private ServiceType service;                       //권한 종류
  private AuthorityTargetType authorityTargetType;   //권한 부여 대상 유형
  private String reason;                             //사유
  private Date loggingDate;                          //로그 기록 일시
  private String targetUserId;                       //신청자 ID
  private String adminId;                            //승인자 ID

}
```

### 2. 접근 권한 클래스 선언

```
/**
 * 접근권한 정보를 담는 클래스.
 */
public class Authority implements IAuthority {

  private String resourceId;                 //자원 ID
  private ServiceType service;               //접근권한 행위

}
```

### 3. 개인정보취급자 권한 부여/변경/말소 이력 관리

```
/**
 * 개인정보취급자에 권한을 부여/변경/말소
 */
public class AuthorityManager implements IAuthorityManager {
  /**
   * 개인정보취급자의 특정 자원에 대한 권한을 변경한다.
   *
   * @param authorityChangeType
```

```
 * @param hasIdentifier
 * @param adminId
 * @param reason
 * @param authorityList
 */
private void changeUserAuthority(
    AuthorityChangeType authorityChangeType,
    User user,
    String reason,
    String adminId,
    List<IAuthority> authorityList
) {
  // 접근권한 부여/변경/말소 이력(로그)를 기록한다.
  AuthorityLog authorityLog = new AuthorityLog();
  authorityLog.setLoggingDate(new Date());
  authorityLog.setAuthorityLogType(authorityChangeType);
   authorityLog.setAuthorityTargetType(AuthorityTargetType.USER);
  authorityLog.setTargetId(user.getId());
  authorityLog.setReason(reason);
  authorityLog.setAdminId(adminId);
  for (IAuthority authority : authorityList) {
    authorityLog.setResourceId(authority.getResourceId());
    authorityLog.setService(authority.getService());
    this.authorityLogStore.insertAuthorityLog(authorityLog);
  }

  if ( AuthorityChangeType.ADD == authorityChangeType ) {
    this.userAuthorityStore.insertAuthority(user.getId(), authorityList);
  }
  else if ( AuthorityChangeType.MODIFY == authorityChangeType ) {
    this.userAuthorityStore.modifyAuthority(user.getId(), authorityList);
  }
  else if ( AuthorityChangeType.REMOVE == authorityChangeType ) {
    this.userAuthorityStore.deleteAuthority(user.getId(), authorityList);
  }
}
```

## 1.2 비정상 접근 및 장기 미접속시 계정 잠금

### 기획 단계

- 계정정보 및 비밀번호를 **일정 횟수 이상 잘못 입력한 경우, 개인정보취급자의 계정 비활성화**
  - 개인정보처리시스템에 대한 개인정보취급자 계정 잠금 해제 시, 개인정보취급자 여부를 확인 후 계정 잠금 해제 등의 조치가 필요
    ※ ID/PW 외 추가적인 인증수단을 적용하여 정당한 접근 권한자임을 확인하도록 하거나, 내부 승인 후 관리자가 해제 처리(관리자 화면 구현 필요)

- 개인정보취급자가 개인정보처리시스템에 **일정기간 장기 미접속한 경우, 계정 잠금 등 접속을 차단**

※ 장기 미접속에 따른 계정 잠금 적용이 필요한 기간은 최소한의 기간으로 설정해야 함(일반적으로 30일~90일 이내)

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

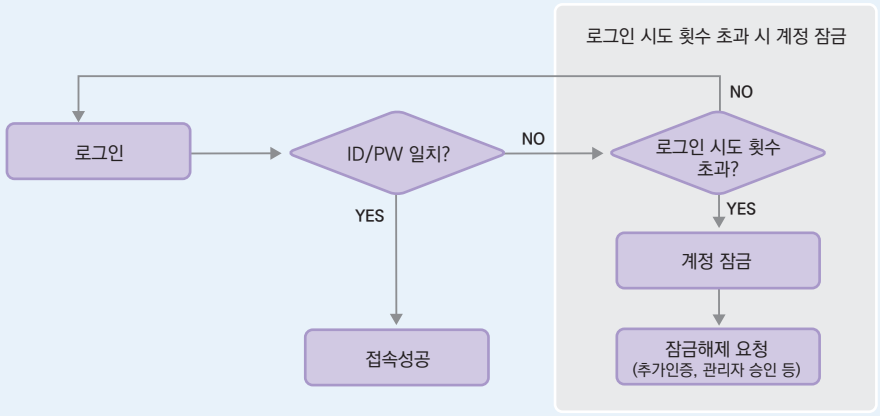
개인정보처리시스템에 권한 없는 자의 비정상적인 접근을 방지하기 위해, 개인정보취급자의 인증정보 입력 오류 및 미사용 계정에 대한 제한 조치를 마련해야 한다.

### 구축 단계

- 개인정보취급자가 일정 횟수 이상 로그인에 실패하는 경우, 계정이 비활성화 되도록 조치
  - 개인정보취급자 계정별 로그인 횟수를 추적하기 위해 관련정보 기록
    ※ '로그인 실패횟수', '계정잠금여부', '마지막으로 성공·실패한 로그인 시간정보', '로그아웃한 시간정보' 등
  - 계정이 비활성화 된 이후에는 추가적 인증수단(인증서, OTP 등)으로 정당한 접근 권한자임을 인증한 후에 로그인 되도록 설계

**TIP**

• 로그인 시도 횟수 초과시 접근제한 흐름도 예시




| 기능 | 기능설계(예시) |
|---|---|
| 로그인 시도 횟수 | 로그인 시도 마다 1씩 증가시켜, 시도 횟수가 일정 횟수보다 큰 경우 일정시간 동안 로그인 기능이 잠금 상태가 되도록 설계(단, 로그인 성공시 0으로 초기화) |
| 마지막으로 실패한 로그인 시간 | 로그인 잠금 상태가 유지되어야 하는 시간을 설정할 수 있음 |
| 마지막으로 성공한 로그인 시간 | 마지막으로 로그인한 시간 정보를 개인정보취급자에게 알려줌으로써, 자기 계정이 다른 사람에 의해 사용되었는지 여부를 개인정보취급자가 점검할 수 있도록 설계 |

- 실패한 인증시도에 대한 정보 기록을 통해 인증시도 실패를 추적할 수 있도록 구현

※ 반복된 로그인 실패에 대한 로깅 정책을 설정하고 로그 저장을 통해 허용되지 않은 로그인 시도를 분석

• 장기 미접속자 계정 보호를 위한 개인정보취급자의 계정 잠금 조치

- 장기 미접속자의 계정을 비활성화 하기 위한 관련 정보 기록

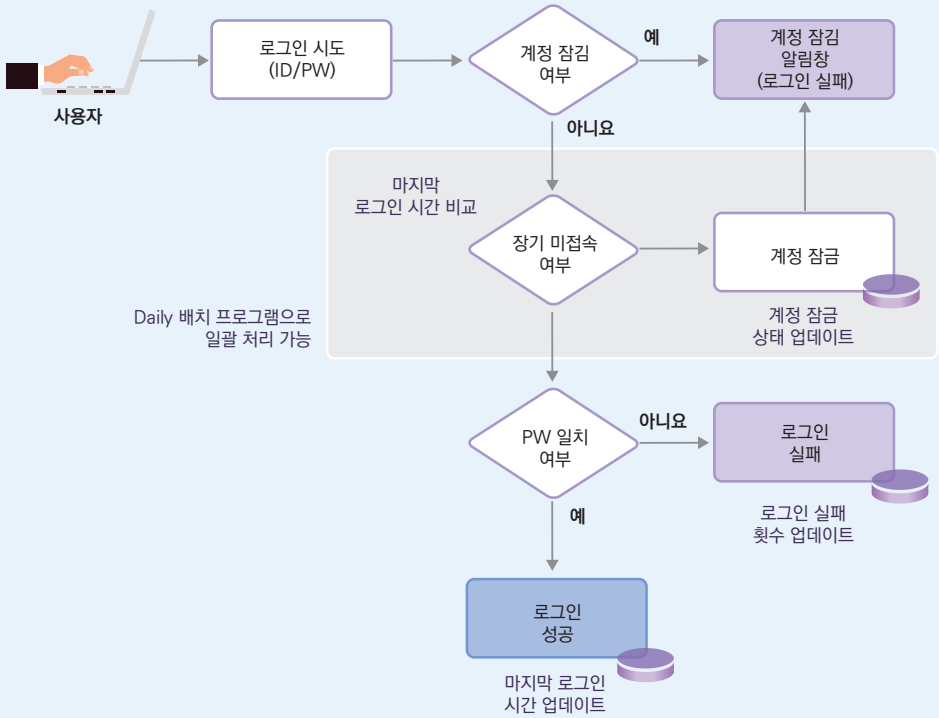
※ '마지막 로그인 시간'과, 계정 잠금 여부를 확인할 수 있는 '계정상태'(활성/비활성 등) 등

- 비활성화 계정에 대해 로그인 시도 시, 정상적인 사용을 위한 계정 잠금 해제 절차 및 기능을 마련

※ 예 : 내부 승인 후 관리자가 계정 잠금 해제 처리 등

TIP 참고

### 장기 미접속자 계정 잠금 조치 예시




| 기능 | 기능설계(예시) |
|---|---|
| 계정 잠금 시 안내 | • 개인정보취급자 로그인 처리 로직에서 사용자 테이블의 '계정 상태' 컬럼을 확인하여 계정 잠금(또는 비활성화) 상태인 경우, 계정 잠김 관련 안내창 표출 |
| 일정시간 미접속 시 계정 잠금 조치 | • 계정 잠금 기준일과 개인정보취급자 테이블의 '최종 로그인 시간'을 비교하여 기준일을 초과한 경우 개인정보취급자 테이블의 '계정 상태' 컬럼을 계정 잠김(또는 비활성화) 상태로 변경<br>※ 배치프로그램으로 등록하여 매일 야간에 실행 가능 |
| 마지막으로 성공한 로그인 시간 | • 계정 잠금 상태가 아닌 경우 패스워드 유효성 검증 및 '최종 로그인 시간' 컬럼을 현재 시간으로 업데이트하고 로그인 성공 처리<br>※ 추가적으로 계정 잠금 해제 처리를 위한 로직 구현 필요(관리자 기능) |

## 구현 예시

### 일정 횟수 이상 로그인에 실패 시, 계정 비활성화 기능 구현

```
/**
 * 개인정보처리시스템의 로그인 요청을 담당하는 Controller
 */
@Controller
@RequestMapping(value="/admin")
public class AdminSignInController {


 /**
  * 로그인 요청 처리
  */
 @RequestMapping(value="/login",method=RequestMethod.POST)
 public String doLogin(
  @RequestParam("userId") String userId,
  @RequestParam("password") String password,
  HttpServletRequest request
 ) {
  User user = this.userManager.getById(userId);

  // 비밀번호가 일치하지 않을 때
  String encryptedPw = PasswordHashMaker.makeHash(this.passwordEncrypter, user, password);

  if ( ! encryptedPw.equals(user.getEncryptedPassword()) ) {
   int wrongPwCount = user.getWrongPasswordCount();
   wrongPwCount++;
   if ( wrongPwCount < this.securityPolicy.getMaxWrongPasswordCount() ) {
    // 로그인 실패 횟수 등록
    this.userManager.updateLoginState(LoginResult.FAIL, wrongPwCount, clientIpAddr);
   }
   else {
    // 최대 비밀번호 오류 허용 횟수에 도달 시, 계정 상태 잠금
    this.userManager.updateLoginState(LoginResult.FAIL, UserUsageState.BLOCKED,
     wrongPwCount, clientIpAddr);
   }

   // 로그인 시도 로그 기록
   loginLogManager.add(user.getId(), clientIpAddr, LoginResult.FAIL, UserUsageState.BLOCKED);
```

```
    request.setAttribute("message","ID와 비밀번호가 일치하지 않습니다.");

    return showLoginPage(request);
   }

   // 로그인 성공 개인정보취급자 상태 갱신
   this.userManager.updateLoginState(LoginResult.SUCCESS, UserUsageState.NORMAL,
    0, clientIpAddr);

   // 로그인 시도 로그 기록.
   loginLogManager.add(user.getId(), clientIpAddr, LoginResult.SUCCESS, UserUsageState.NORMAL);
  }
```

## 장기 미접속자 계정 잠금조치 기능 구현

```
/**
 * 개인정보처리시스템의 로그인 요청을 담당하는 Controller
 */
@Controller
@RequestMapping(value="/admin")
public class AdminSignInController {

 /**
  * 로그인 요청 처리
  */
 @RequestMapping(value="/login", method=RequestMethod.POST)
 public String doLogin(
   @RequestParam("userId") String userId,
   @RequestParam("password") String password,
   HttpServletRequest request
 ) {
   User user = this.userManager.getById(userId);

   // 등록된 개인정보취급자가 아닐 경우 경고창을 띄운다.
   if ( user == null ) {
     request.setAttribute("message","ID와 비밀번호가 일치 하지 않습니다.");
     return showLoginPage(request);
   }
```

```
    // 장기간 미사용 등으로 잠긴 계정으로 로그인 시 경고창을 띄운다.
    if ( user.getAccountState() == UserAccountState.LOCKED ) {
      request.setAttribute("message","장기간 미사용 등의 이유로 잠긴 계정입니다.");
      request.setAttribute("help","관리자에게 문의해 주십시오.");
      return showLoginPage(request);
    }

    Calendar cal = Calendar.getInstance();

    // 장기간 사용이 없던 계정은 잠금 상태로 전환한다.
    cal.setTime(user.getLastSuccessLoginDate());
    cal.add(Calendar.DATE,this.securityPolicy.getSystemUnusedAllowableTerm());
    Date unusedAllowableLimitDate = cal.getTime();
    if ( unusedAllowableLimitDate.after(new Date()) ) {
      this.userManager.updateAccountState(userId, UserAccountState.LOCKED);
      request.setAttribute("message",
        "오랫동안 사용하지 않아 계정이 잠겼습니다. 관리자에게 문의해 주세요.");
      return showLoginPage(request);
    }
  }
}
```

## 1.3 개인정보취급자의 비밀번호 작성규칙 적용

### 기획 단계

- 개인정보취급자가 안전한 비밀번호를 설정하여 이용할 수 있도록, 비밀번호 작성규칙을 수립하여 적용
  - 단, 정보통신서비스 제공자등의 경우 개인정보취급자를 대상으로 다음에 해당하는 비밀번호 작성규칙을 적용하여야 함

**개인정보취급자의 비밀번호 작성규칙(정보통신서비스 제공자등)**

① 영문, 숫자, 특수문자 중 2종류 이상을 조합하여 최소 10자리 이상 또는 3종류 이상을 조합하여 최소 8자리 이상의 길이로 구성

② 연속적인 숫자나 생일, 전화번호 등 추측하기 쉬운 개인정보 및 아이디와 비슷한 비밀번호는 사용하지 않는 것을 권고

③ 비밀번호에 유효기간을 설정하여 반기별 1회 이상 변경

**PbD 적용설계 | (제①원칙) 사후조치가 아닌 사전예방**

정당한 접속 권한을 가지지 않는 자가 추측하여 접속을 시도하기 어렵도록 개인정보취급자에게 안전한 비밀번호 작성규칙을 적용해야 한다.

### 구축 단계

- 개인정보취급자의 비밀번호 설정 시, 아래의 내용을 고려하여 구현

**비밀번호 설정 기능 구현 시 고려사항**

① '비밀번호 작성규칙'을 안내하고 설정한 비밀번호가 올바르게 설정되었는지 확인

② 비밀번호 입력 시, 화면에 마스킹(*) 처리하여 표기

③ 비밀번호 변경기간 경과여부 확인을 위해 '비밀번호 설정 일시'를 저장 및 관리

※ 비밀번호 변경기간 경과여부 확인을 위한 목적이므로 필요 시 적용

※ 비밀번호 저장 시 안전한 알고리즘을 이용하여 저장하여야 함

• 개인정보취급자의 비밀번호 변경 시, 아래의 내용을 고려하여 구현

**비밀번호 설정 기능 구현 시 고려사항**

① 개인정보취급자가 로그인할 때, 비밀번호 설정일시와 현재시간을 비교하여 비밀번호 변경기간이 도래한 경우 비밀번호를 변경할 수 있도록 안내

※ 정보통신서비스 제공자등의 경우 개인정보취급자의 비밀번호를 반기별 1회 이상 변경해야 함(단, 정보주체의 경우 비밀번호의 주기적 변경이 법적인 준수사항은 아니므로, 주기적인 변경 알림 기능은 제공하되 강제화할 필요는 없음)

② 이전과 동일한 비밀번호의 사용제한을 위해 기존에 사용한 비밀번호를 일정개수 이상 저장하여, 과거에 사용했던 비밀번호와 동일할 경우 변경을 제한

• 개인정보취급자의 비밀번호 초기화 시, 아래의 내용을 고려하여 구현

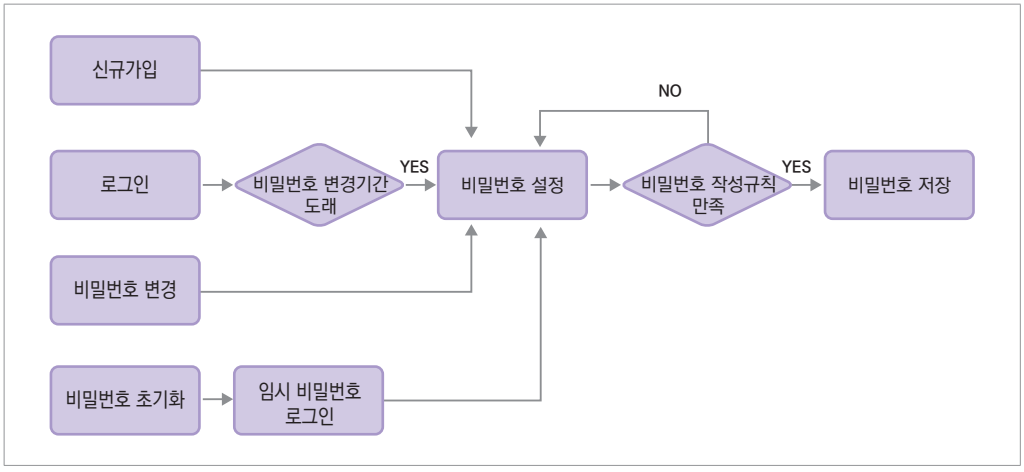
**비밀번호 초기화 기능 구현 시 고려사항**

① 아이핀, 휴대폰 본인인증 등 인증수단을 통해 비밀번호를 변경하려는 개인정보취급자 본인임을 인증하는 절차를 수행

② 임시 비밀번호를 정보주체에게 발급할 경우, 난수값을 사용하고 개인정보취급자가 회원가입 시 등록한 E-mail, 휴대폰 문자 등으로 전달

③ 임시 비밀번호를 이용하여 개인정보취급자가 로그인 할 경우, 비밀번호를 변경한 후 서비스를 이용할 수 있도록 안내

**비밀번호 초기화 기능 구현 시 고려사항**



**구현 예시**

### 개인정보취급자의 비밀번호 변경기능 구현

```
/**
 * 개인정보처리시스템의 개인정보취급자 정보를 처리하기 위한 Controller.
 */
@Controller
@RequestMapping("/personal")
public class PersonalController {
 /** 비밀번호 변경 화면. */
 private final static String CHANGE_PW_PAGE_VIEW = "/personal/password";

 /**
  * 비밀번호 변경 화면을 요청한다.
  * @return 비밀번호 변경 화면 View 경로.
  */
 @RequestMapping(value="/password", method=RequestMethod.GET)
 public String showPasswordChangePage() {
  return CHANGE_PW_PAGE_VIEW;
 }
 /** 비밀번호 변경 성공 화면. */
 private final static String CHANGE_PW_SUCCESS_VIEW = "/personal/password-success";
 /**
  * 비밀번호를 변경한다.
  *
  * @param password 현재 비밀번호.
  * @param newPassword 새로운 비밀번호.
  * @param newPassword2 새로운 비밀번호 확인.
  * @param model
  * @return
  */
 @RequestMapping(value="/password", method=RequestMethod.POST)
 public String changePassword(
    @RequestParam("password") String password,
    @RequestParam("newPassword") String newPassword,
    @RequestParam("newPassword2") String newPassword2,
    HttpServletRequest request
 ) {
  // 새 비밀번호와 비밀번호 확인 값이 같은지 점검한다.
  if ( ! newPassword.equals(newPassword2) ) {
    request.setAttribute("message","비밀번호와 확인 값이 일치하지 않습니다.");
```

```
    return CHANGE_PW_PAGE_VIEW;
   }

   User user = (User) request.getSession().getAttribute(User.KEY);

   // 현재 비밀번호가 맞는지 확인한다.
   String encryptedPw = PasswordHashMaker.makeHash(this.passwordEncrypter, user, password);
   if ( ! encryptedPw.equals(user.getEncryptedPassword()) ) {
    request.setAttribute("message","현재 비밀번호와 일치하지 않습니다.");
    return CHANGE_PW_PAGE_VIEW;
   }

   // 개인정보취급자 비밀번호를 변경한다.
   try {
    this.userManager.updatePassword(user.getId(), newPassword);
   }
   catch (PasswordPolicyViolatationException e) {
    request.setAttribute("message","비밀번호 생성 정책에 어긋납니다.");
    request.setAttribute("violations", e.getViolationList());
    return CHANGE_PW_PAGE_VIEW;
   }

   return CHANGE_PW_SUCCESS_VIEW;
  }
}
```

## 비밀번호 일방향 암호화 기능 구현

1. SHA256을 이용한 비밀번호 암호화

```
**
 * SHA256 해시 생성 클래스
 */
public class Sha256HashEncrypter implements Encrypter {
  /**
   * "SHA-256" 해시 알고리즘을 이용하여 주어진 문자열의 해시 값을 생성한다.
   *
   * @param salt 해시에 사용될 SALT 값
   * @param plainText 해시값을 구할 문자열(비밀번호)
   */
```

```
  @Override
  public String encrypt(String salt, String plainText) {
    MessageDigest mdSHA256 = null;
    try {
      mdSHA256 = MessageDigest.getInstance("SHA-256");
      String saltedText = salt + plainText;
      mdSHA256.update(saltedText.getBytes("UTF-8"));
    }
    catch (Exception e) {
      throw new RuntimeException(e);
    }
    // 해시 계산 반환값은 바이트 배열
    byte[] sha256Hash = mdSHA256.digest();

    return Base64.encodeBase64String(sha256Hash);
  }
}
```

2. 사용자 정보를 솔트 값으로 적용하여 비밀번호 암호화

```
/**
 * 비밀번호의 해시값을 만드는 클래스.
 */
public class PasswordHashMaker {
  /**
   * <p>주어진 Encrypter를 이용하여 사용자 정보와 비밀번호를 활용한
   * 해시 값을 만들어 돌려 준다.</p>
   *
   * @param encryter 해시 생성기.
   * @param account 계정 객체.
   * @param password 비밀번호.
   * @return
   */
  public static String makeHash(Encrypter encryter, IAccount account, String password) {
    // 여기서는 사용자 ID 뒤 4자리를 Salt 값으로 활용하여 해시를 만든다.
    // ID 뒤 4자리를 Salt 값으로
    String salt = account.getId().substring(account.getId().length() - 4);

    String encryptedPw = encryter.encrypt(salt, password);

    return encryptedPw;
  }
}
```

# 2 접근통제

| | |
|---|---|
| 항목명 | 개인정보처리시스템 접근통제 적용 |
| 요구사항 내용 | 2.1 일정시간 미입력시 자동 접속차단 기능 적용<br>– 일정시간 업무를 처리하지 않는 경우 시스템 자동 접속 차단 |
| 관련 근거 | • 개인정보 보호법 제29조(안전조치의무)<br>• 개인정보 보호법 시행령 제30조(개인정보의 안전성 확보 조치)<br>• 개인정보 보호법 시행령 제48조의2(개인정보의 안전성 확보 조치에 대한 특례)<br>• 개인정보의 안전성 확보조치 기준 제6조(접근통제)<br>• 개인정보의 기술적·관리적 보호조치 기준 제4조(접근통제) |

## 2.1 일정시간 미입력시 자동 접속차단 기능 적용

### 기획 단계

• **개인정보취급자가 일정시간 업무를 처리하지 않는 경우**, 업무용 컴퓨터 등에서 해당 **개인정보처리시스템에 대한 접속을 차단**

※ 단, 업무용 컴퓨터의 화면잠금, 화면보호기 등은 접속차단에 해당하지 않음

- 개인정보를 처리하는 방법 및 환경, 업무특성 등을 고려하여 적정한 시스템 접속차단 시간을 결정

※ 시스템 접속 차단 시간은 최소한(일반적으로 10~30분 이내)으로 정하는 것을 권장

- 개인정보처리시스템에 대한 접속이 차단된 이후 다시 접속하는 경우, **최초 로그인과 동일한 방식으로 로그인을 수행하도록 절차 마련**

- 단, 정보통신서비스 제공자등은 접속이 필요한 최대시간을 설정하여 최대 접속시간이 경과하면 개인정보처리시스템과 연결이 완전히 차단될 수 있도록 설계하는 등의 조치를 해야 함

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

비인가자가 개인정보처리시스템의 정보를 탈취할 목적으로 불법 접근하는 것을 방지하기 위하여, 개인정보취급자가 일정시간 업무를 하지 않는 경우 개인정보처리시스템에서 자동으로 접속을 차단하는 조치가 필요하다.

## 구축 단계

• 개인정보처리시스템의 자동 접속 차단은 웹서버 설정 및 직접 구현을 통해 조치할 수 있음

**웹서버에서 자동 접속차단 기능 설정 방법**

| 웹서버 | 설정 방법 |
|---|---|
| IIS | 응용프로그램 툴 → 고급설정 → 프로세스 모델 내 "유효시간 제한(분)"을 통한 설정 |
| Apache | Apache 환경설정 파일 중 httpd-default.conf 파일 내 "Timeout"을 통한 설정 |
| Tomcat | Tomcat 환경설정 내에 있는 "web.xml"을 통한 설정<br>"web.xml" 설정(10분으로 설정) 예시는 아래와 같음<br><web-app><br><session-config><br><session-timeout>10</session-timeout><br></session-config><br></web-app> |

# 3 개인정보의 암호화

| | |
|---|---|
| 항목명 | 개인정보 저장 시 암호화 적용 |
| 요구사항 내용 | 3.1 개인정보 저장 시 암호화 적용<br>– 비밀번호 저장 시 일방향 암호화하여 저장<br>– 개인정보 저장시 안전한 암호 알고리즘으로 암호화<br>– 개인정보 암·복호화를 위한 암호키의 안전한 보관·관리 |
| 관련 근거 | • 개인정보 보호법 제24조(고유식별정보의 처리 제한)<br>• 개인정보 보호법 제24조의2(주민등록번호 처리의 제한)<br>• 개인정보 보호법 제29조(안전조치의무)<br>• 개인정보 보호법 시행령 제30조(개인정보의 안전성 확보 조치)<br>• 개인정보 보호법 시행령 제48조의2(개인정보의 안전성 확보 조치에 대한 특례)<br>• 개인정보의 안전성 확보조치 기준 제7조(개인정보의 암호화)<br>• 개인정보의 기술적·관리적 보호조치 기준 제6조(개인정보의 암호화) |

## 3.1 개인정보 저장 시 암호화 적용

### 기획 단계

• 비밀번호의 경우 복호화되지 않도록, **일방향 암호화하여 저장**

- 비밀번호 암호화 저장 시, 국내·외 암호연구 관련 기관에서 사용 권고하는 안전한 암호 알고리즘으로 암호화하여 저장해야 함
- 일방향 암호 알고리즘에 솔트(Salt) 값을 적용하여 암호화 보안 강도를 높일 수 있음

TIP 참고 **비밀번호 암호화 시 솔트값 적용**

• 레인보우테이블* 등을 통해 해시값을 미리 계산하여 비밀번호를 찾을 수 있는 위험성을 고려하여 암호화 저장할 때 솔트값**을 함께 적용하여 암호화 권고

* 레인보우테이블 : 해시 함수를 사용하여 변환 가능한 모든 해시 값을 저장시켜 놓은 표를 말하며, 일반적으로 해시 함수를 이용하여 저장된 비밀번호로부터 원래의 비밀번호를 추출해 내는데 사용될 수 있음

** 솔트값: 동일한 비밀번호에도 다른 해시값이 나오도록 하기 위하여 해시함수를 적용하기 전에 비밀번호에 덧붙이는 임의의 값을 말함

• **개인정보** 저장 시, 안전한 알고리즘으로 **암호화하여 저장**

**개인정보 저장 시 암호화 적용 항목**

| 구 분 | 개인정보처리자 | 정보통신서비스 제공자등 |
|---|---|---|
| 바이오정보 | ○ | ○ |
| 주민등록번호 | ○ | ○ |
| 고유식별정보 | ○* | ○ |
| 계좌번호 | - | ○ |
| 신용카드정보 | - | ○ |

※ 개인정보처리자의 경우 내부망에 주민등록번호를 제외한 고유식별정보를 저장할 때, 위험도 분석 결과 및 개인정보 영향평가 결과에 따라 적용여부 등을 결정할 수 있음

- 개인정보 저장 시, 안전한 암호화 알고리즘으로 암호화하여야 함

**개인정보 저장 시 암호화 적용 항목**

| 분류 | 미국(NIST) | 일본(CRYPTREC) | 유럽(ECRYPT) | 국내 |
|---|---|---|---|---|
| 대칭키 암호 알고리즘 | AES-128/192/256 3TDEA | AES-128/192/256 Camellia-128/192/256 | AES-128/192/256 Camellia-128/192/256Serpent-128/192/256 | SEED HIGHT ARIA-128/192/256 LEA-128/192/256 |
| 공개키 암호 알고리즘 (메시지 암·복호화) | RSA (사용 권고하는 키길이 확인 필요) | RSAS-OAEP | RSA-OAEP | RSAES |

• 개인정보 암호화에 사용된 암호 키는 안전하게 저장·관리

- 암호 키를 통해 암호화된 개인정보를 복호화할 수 있으므로, 암호키를 안전하게 보관해야 함

※ 암호화 키는 암호화 키 관리 시스템(Key Management System)을 통해 암호화 키에 대한 접근, 이용, 보관, 폐기 절차를 자동화하는 것을 권고

- 개인정보 암호화에 사용된 암호 키는 운영 데이터와 분리하여 보관하여야 함

**TIP**

- 개인정보 암호화에 대한 자세한 사항은 개인정보보호 포털(https://www.privacy.go.kr)에서 제공하는 "개인정보의 암호화 조치 안내서"를 참고할 수 있다.
- 암호화에 대한 자세한 사항은 암호이용 활성화(https://seed.kisa.or.kr)에서 제공하는 "암호 알고리즘 및 키 길이 이용 안내서", "암호 키 관리 안내서" 등을 참고할 수 있다.

**PbD 적용설계** | (제③원칙) 프라이버시 보호를 내재한 설계

개인정보가 비인가자에게 유·노출 되더라도, 내용을 알아보지 못하도록 암호화하여 저장해야 한다.

## 구축 단계

- 개인정보를 암호화할 수 있는 방식은 암·복호화 모듈의 위치와 암·복호화 모듈의 요청 위치의 조합에 따라 다음과 같은 암호화 방식을 적용할 수 있음

**모듈 위치별 암호화 방식**

| 암호화 방식 | 암·복호화 모듈 위치 | 암·복호화 요청 위치 | 설 명 |
|---|---|---|---|
| 응용 프로그램 자체 암호화 | 어플리케이션 서버 | 응용 프로그램 | • 암·복호화 모듈이 API 라이브러리 형태로 각 어플리케이션 서버에 설치되고, 응용프로그램에서 해당 암·복호화 모듈을 호출하는 방식<br>• DB 서버에 영향을 주지 않아 DB 서버의 성능 저하가 적은 편이지만 구축 시 응용프로그램 전체 또는 일부 수정 필요<br>• 기존 API 방식과 유사 |
| DB 서버 암호화 | DB 서버 | 응용 프로그램 | • 암·복호화 모듈이 DB서버에 설치되고 DB 서버에서 암·복호화 모듈을 호출하는 방식<br>• 구축 시 응용프로그램의 수정을 최소화 할 수 있으나 DB 서버에 부하가 발생하며 DB 스키마의 추가 필요<br>• 기존 Plug-In 방식과 유사 |
| DBMS 자체 암호화 | DB 서버 | DB 서버 | • DB 서버의 DBMS 커널이 자체적으로 암·복호화 기능을 수행하는 방식<br>• 구축 시 응용프로그램 수정이 거의 없으나, DBMS에서 DB 스키마의 지정 필요<br>• 기존 커널 방식(TDE)과 유사 |
| DBMS 암호화 기능 호출 | DB 서버 | 응용 프로그램 | • 응용프로그램에서 DB 서버의 DBMS 커널이 제공하는 암·복호화 API를 호출하는 방식<br>• 구축 시 암·복호화 API를 사용하는 응용프로그램의 수정이 필요<br>• 기존 커널 방식(DBMS 함수 호출)과 유사 |
| 운영체제 암호화 | 파일서버 | 운영체제 (OS) | • OS에서 발생하는 물리적인 입출력(I/O)을 이용한 암·복호화 방식으로 DBMS의 데이터파일 암호화<br>• DB 서버의 성능 저하가 상대적으로 적으나 OS, DBMS, 저장장치와의 호환성 검토 필요<br>• 기존 DB 파일암호화 방식과 유사 |

**구현 예시**

## 개인정보 양방향 암호화 기능 구현

### 1. 암·복호화 클래스 정의

```
/**
 * "AES256" 암·복호화 알고리듬을 이용한 암호화, 복호화 클래스
 */
public class Aes256Cipher implements Encrypter, Decrypter {
  /**
   * @throws NoSuchPaddingException
   * @throws NoSuchAlgorithmException
   * @see Encrypter#encrypt(String, String)
   */
  public String decrypt(String key, String encryptedText) {
    byte[] decryptedTextBytes = null;
    try {
      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
      ByteBuffer buffer = ByteBuffer.wrap(Base64.decodeBase64(encryptedText));

      byte[] saltBytes = new byte[20];
      buffer.get(saltBytes,0, saltBytes.length);
      byte[] ivBytes = new byte[cipher.getBlockSize()];
      buffer.get(ivBytes,0, ivBytes.length);
      byte[] encryoptedTextBytes = new byte[buffer.capacity() - saltBytes.length - ivBytes.length];
      buffer.get(encryoptedTextBytes);

      SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
      PBEKeySpec spec = new PBEKeySpec(key.toCharArray(), saltBytes, 70000, 256);

      SecretKey secretKey = factory.generateSecret(spec);
      SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(),"AES");

      cipher.init(Cipher.DECRYPT_MODE, secret,new IvParameterSpec(ivBytes));
      decryptedTextBytes = cipher.doFinal(encryoptedTextBytes);
    }
    catch (Exception e) {
      throw new RuntimeException(e);
    }

    return new String(decryptedTextBytes);
```

```
}

/**
 * @see Decrypter#decrypt(String, String)
 */
public String encrypt(String key, String plainText) {
  SecureRandom random = new SecureRandom();
  byte saltBytes[] = new byte[20];
  random.nextBytes(saltBytes);

  // Initial Vector(1단계 암호화 블록용)
  byte[] ivBytes;
  byte[] encryptedTextBytes;
  try {
    // Password-Based Key Derivation function 2
    SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");

    // 10000번 해시하여 256 bit 길이의 키를 만든다.
    PBEKeySpec spec = new PBEKeySpec(key.toCharArray(), saltBytes, 10000, 256);
    SecretKey secretKey = factory.generateSecret(spec);
    SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(),"AES");

    // 알고리즘/모드/패딩
    // CBC : Cipher Block Chaining Mode
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, secret);
    AlgorithmParameters params = cipher.getParameters();
    ivBytes = params.getParameterSpec(IvParameterSpec.class).getIV();
    encryptedTextBytes = cipher.doFinal(plainText.getBytes("UTF-8"));
  }
  catch (Exception e) {
    throw new RuntimeException(e);
  }


  byte[] buffer = new byte[saltBytes.length + ivBytes.length + encryptedTextBytes.length];
  System.arraycopy(saltBytes,0, buffer,0, saltBytes.length);
  System.arraycopy(ivBytes,0, buffer, saltBytes.length, ivBytes.length);
  System.arraycopy(encryptedTextBytes,0, buffer, saltBytes.length + ivBytes.length,
encryptedTextBytes.length);


  return Base64.encodeBase64String(buffer);
```

```
 }
}
```

2. 개인정보의 암·복호기능 구현

```
/**
 * 개인정보의 암·복호화 기능 구현
 */
public class UserInfoCipher {
 private Encrypter encrypter;
 private Decrypter decrypter;

 /**
  * 개인정보취급자 정보 암·복호화 객체를 생성한다.
  *
  * @param encrypter 암호화에 사용할 Encrypter.
  * @param decrypter 복호화에 사용할 Decrypter.
  */
 public UserInfoCipher(Encrypter encrypter, Decrypter decrypter) {
  this.encrypter = encrypter;
  this.decrypter = decrypter;
 }

 /**
  * 평문을 암호화한다.
  *
  * @param user
  * @param plainText 암호화할 평문 문자열.
  * @return
  */
 public String encrypt(User user, String plainText) {
  // 암호화 Key를 생성한다. (안전한 랜덤 값 사용 권장한다.)
  String key = user.getId().substring(user.getId().length() - 4);

  return this.encrypter.encrypt(key, plainText);
 }


 /**
  * 암호화된 문자열을 복호화한다.
  *
  * @param user
  * @param decryptedText 복호화할 암호화된 문자열.
```

```
 * @return
 */
public String decrypt(User user, String decryptedText) {
 // 복호화 Key를 생성한다. (안전한 랜덤 값 사용 권장한다.)
 String key = user.getId().substring(user.getId().length() - 4);
 return this.decrypter.decrypt(key, decryptedText);
 }
}
```

### 3. 개인정보 입력 시 암호화

```
@Resource(name="userInfoCipher")
private UserInfoCipher userInfoCipher;

/**
 * 개인정보를 암호화 한다.
 *
 * @param user 등록할 개인정보취급자 정보를 담고 있는 User 객체.
 * @throws PasswordPolicyViolatationException
 */
public int addUser(User user) throws PasswordPolicyViolatationException {
 // 비밀번호 유효성 검사
 String password = user.getPassword();
 PasswordChecker passwordChecker = new PasswordChecker(securityPolicy, passwordEncrypter);
 List<PasswordViolation> pwViolationList = passwordChecker.checkPasswordPolicy(password, user);
 if (pwViolationList != null && !pwViolationList.isEmpty()) {
  throw new PasswordPolicyViolatationException(pwViolationList);
 }
 String passwordHash = PasswordHashMaker.makeHash(this.passwordEncrypter, user, password);
 user.setEncryptedPassword(passwordHash);


 // 평문화되어 있는 개인정보를 암호화한다.
 // 고유식별번호: 주민등록번호
 String encryptedSsn = this.userInfoCipher.encrypt(user, user.getSsn());
 user.setEmailAddr(encryptedSsn);

 // 이메일 주소
 String encryptedEmailAddr = this.userInfoCipher.encrypt(user, user.getEmailAddr());
 user.setEmailAddr(encryptedEmailAddr);

 // 핸드폰번호
 String encryptedCellPhoneNo = this.userInfoCipher.encrypt(user, user.getCellPhoneNo());
```

```
    user.setCellPhoneNo(encryptedCellPhoneNo);

    // 전화번호
    String encryptedPhoneNo = this.userInfoCipher.encrypt(user, user.getPhoneNo());
    user.setPhoneNo(encryptedPhoneNo);
    return this.userStore.insertUser(user);
  }
```

4. 암호회된 개인정보 복호화

```
  @Resource(name="userInfoCipher")
  private UserInfoCipher userInfoCipher;

  @Override
  public User getById(String userId) {
    User user = this.userStore.selectById(userId);
    // 암호화한 값을 복호화 한다.
    // 고유식별정보
    String decryptedSsn = this.userInfoCipher.decrypt(user, user.getSsn());
    user.setEmailAddr(decryptedSsn);

    // 이메일 주소
    String decryptedEmailAddr = this.userInfoCipher.decrypt(user, user.getEmailAddr());
    user.setEmailAddr(decryptedEmailAddr);

    // 휴대전화번호
    String decryptedCellPhoneNo = this.userInfoCipher.decrypt(user, user.getCellPhoneNo());
    user.setCellPhoneNo(decryptedCellPhoneNo);

    return user;
```

# 4 접속기록 관리

| | |
|---|---|
| 항목명 | 개인정보처리시스템 접속기록 보관·점검 적용 |
| 요구사항 내용 | 4.1 개인정보취급자 이용 시스템 접속기록 보관·점검<br>- 개인정보취급자 이용 시스템의 접속기록 보관<br>- 개인정보취급자 이용 시스템의 접속기록 점검 |
| 관련 근거 | • 개인정보 보호법 제29조(안전조치의무)<br>• 개인정보 보호법 시행령 제30조(개인정보의 안전성 확보 조치)<br>• 개인정보 보호법 시행령 제48조의2(개인정보의 안전성 확보 조치에 대한 특례)<br>• 개인정보의 안전성 확보조치 기준 제8조(접속기록의 보관 및 점검)<br>• 개인정보의 기술적·관리적 보호조치 기준 제5조(접속기록의 위·변조방지) |

## 4.1 접속기록 보관 및 점검

### 기획 단계

• 개인정보취급자가 개인정보처리시스템에 접속하여 처리한 업무내역을 로그파일 또는 로그관리시스템 등에 보관·관리하여야 함

**접속기록 보관·관리 기준**

| 구 분 | 개인정보처리자 | 정보통신서비스 제공자등 |
|---|---|---|
| 보관기간 | 최소 1년 이상 | 최소 1년 이상 |
| | ※ 단, 5만명이상의 개인정보를 처리하거나 고유식별정보 또는 민감정보를 처리하는 경우 최소 2년 이상 | ※ 단,「전기통신사업법」제5조에 따른 기간통신사업자의 경우 최소 2년 이상 |
| 점검주기 | 월 1회이상 | |
| 기록항목 | 계정 | 계정(식별자) |
| | 접속일시 | 접속일시 |
| | 처리한 정보주체 정보 | - |
| | 접속지 정보 | 접속지를 알 수 있는 정보 |
| | 수행업무 | 수행업무 |
| 다운로드 사유확인 | 내부 관리계획 등으로 정하는 바에 따라, 다운로드 사유확인 | - |

**접속기록 작성 예시**

1) 계정 : 개인정보처리시스템에 접속한 자(개인정보취급자 등)의 계정정보

2) 접속일시 : 접속한 시점 또는 업무를 수행한 시점(년-월-일, 시:분:초)

3) 접속지 정보: 개인정보처리시스템에 접속한 자의 컴퓨터 또는 서버 IP

4) 처리한 정보주체 정보 : 누구의 개인정보를 처리하였는지 알 수 있는 정보

※ 과도한 개인정보가 저장되지 않도록 개인의 식별정보(ID, 학번, 사번 등)를 활용하여 기록

※ 대량의 개인정보를 처리하는 경우 검색조건문(쿼리)으로 대체 가능

5) 수행업무 : 개인정보취급자가 개인정보처리시스템을 이용하여 개인정보를 처리한 내용

※ 검색, 열람, 조회, 입력, 수정, 삭제, 출력, 다운로드 등

**접속기록 항목 작성 예시**

| 개인정보 취급자 계정 | 접속일시 | 접속지 정보 | 처리한 정보주체의 정보 | 수행업무 |
|---|---|---|---|---|
| A0001 | 2020-02-25, 17:00:00. | 192.168.100.1 | kdhong | 개인정보 수정 |

※ 위 항목은 반드시 기록하여야 하며, 처리하는 업무환경에 따라 책임추적성 확보에 필요한 항목은 추가로 기록하여야 함

- 개인정보 다운로드 시, 다운로드 사유 확인이 필요한 기준(내부 관리계획에 포함)을 수립하고, 수립한 기준에 따라 사유를 남기도록 개인정보처리시스템 구축 시 반영

**다운로드 사유 확인 관련 내부관리계획 수립 예시**

제○조(접속기록의 보관 및 점검) ① ○○○○○(개인정보처리자명)는 개인정보취급자가 개인정보처리 시스템에 접속한 기록을 1년 이상 보관·관리하여야 한다.

② ○○○○○(개인정보처리자명)는 개인정보의 분실·도난·유출·위조·변조 또는 훼손 등에 대응하기 위하여 개인정보처리시스템의 접속기록 등을 월 1회 이상 점검하여야 한다. 특히, 개인정보처리시스템에서 다음 각 호중 어느 하나에 해당하는 개인정보 다운로드가 발견되었을 경우에는 그 사유를 반드시 확인하여야 한다.

1. 개인정보취급자가 100명 이상의 정보주체에 대한 개인정보를 다운로드 한 경우
2. 개인정보취급자가 1시간 내 다운로드한 횟수가 10건 이상인 경우
3. 개인정보취급자가 업무시간 외(휴무일 등) 개인정보를 다운로드한 경우

③ ○○○○○(개인정보처리자명)는 제2항에 따라 개인정보취급자가 정당한 사유 없이 다운로드한 것이 확인된 경우 지체 없이 개인정보취급자 다운로드 한 개인정보를 회수하여 파기할 수 있다.

- 개인정보가 위·변조, 도난, 분실되지 않도록 접속기록을 안전하게 보관
  - 접속기록 보관 시스템에 불법적으로 접근하여 임의적인 수정·삭제 등이 불가능하도록 접근권한을 제한하는 등 조치를 해야 함

**접속기록 위·변조 방지를 위한 보호조치 예시**

① 정기적으로 접속기록에 대한 백업을 수행하여, 별도의 물리적인 저장장치에 보관

② CD-ROM, DVD-R, WORM(Write Once Read Many) 등 덮어쓰기 방지매체 이용

③ 접속기록의 무결성 보장을 위한 위·변조 방지기술 적용 등

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

개인정보 접속기록을 안전하게 보관·관리함으로써, 개인정보처리시스템의 불법적인 접근과 내부자에 의한 개인정보 유출 및 오·남용 등에 효과적으로 대응할 수 있다.

## 구축 단계

- 개인정보취급자의 접속기록 저장 기능 구현 시 아래사항을 고려하여 구현
  - 계정, 접속일시, 접속지 정보, 처리한 정보주체의 정보, 수행업무 등의 정보를 기록 저장하도록 설계·구현해야 함

- 관리자 접속기록, 인증 성공/실패 기록, 계정/권한의 등록/변경/삭제 등의 기록을 저장하도록 설계·구현할 수 있음
- 대량의 개인정보를 처리하는 경우, 접속기록 중 '처리한 정보주체의 정보'로 처리된 개인정보 전체가 아닌 해당 검색조건문(쿼리)으로 대체할 수 있음

**'처리한 정보주체의 정보' 항목으로 검색조건문을 남기는 예시**

- '김'씨 성을 가진 회원을 조회하는 경우
  - 정보주체의 정보 : SELECT * FROM student WHERE name LIKE '김%';
    ※ name: 학생이름(컬럼), student: 학생정보(테이블)
- 영화를 연간 50회 이상 관람한 고객에게 VIP 등급부여
  - 정보주체의 정보 : UPDATE member SET membership='VIP' WHERE movie_count_per_year〉=50;
    ※ member: 회원정보(테이블), membership: 고객정보(컬럼), movie_count_per_year: 연간 영화관람 건수(컬럼)

**TIP**

- 변경이 빈번하게 발생하거나 임시적으로 활용하는 테이블에 저장된 개인정보를 처리하는 경우, 검색조건문을 정보주체의 정보로 기록하면 책임추적성 확보가 어려울 수 있으므로 해당시점의 DB를 백업하는 등 필요한 조치를 하여야 한다.

- **접속기록 점검(월 1회)의 편의성을 위한 기능 설계·구현할 수 있음**
  - 접속기록을 모니터링 할 수 있는 기능 설계·구현(조회, 분류, 통계 등)
  - 비인가된 개인정보 처리, 대량의 개인정보의 조회, 정정, 다운로드, 삭제 등의 비정상 행위 탐지 기능을 설계·구현
- **개인정보취급자의 접속기록 자동백업 기능 설계·구현할 수 있음**
  - 상시적으로 접속기록을 백업을 수행하도록 설계(개인정보처리시스템 이외의 별도의 보조저장매체나 별도의 저장장치에 보관되도록 설계)

**TIP**

- 스토리지 용량 및 백업 고려
  - 접속기록 보관 시 최소 1년 이상 보관해야 하므로 서버의 용량 또는 백업을 고려하여 구현해야 함
- 별도 로그관리 시스템 고려
  - 로그 기록이 안전하게 보관될 수 있도록 별도의 보관 시스템 구현도 고려하여 적용할 수 있음

**구현 예시**

## 개인정보취급자 접속기록 보관 기능 구현

### 1. 접속기록 항목에 대한 클래스 선언

```
/**
 * 개인정보취급자의 접속기록 항목 정의
 */
public class ActionLog {

  private Date actingDate;                    // 접속 일시
  private String clientIpAddr;                // 접속지 정보
  private String userId;                      // 개인정보취급자의 계정
  private String target;                      // 처리한 정보주체의 정보
  private ServiceType serviceType;            // 수행업무
  private String remark;                      // 부가설명

}
```

### 2. 개인정보취급자의 접속기록 생성

```
/**
 * 개인정보취급자의 접속기록(로그) 생성
 */
public class ActionLogManager implements IActionLogManager {
  @Resource(name="actionLogStore")
  private IActionLogStore actionLogStore;

  @Override
  public void add(String userId, String remoteAddr, String target, ServiceType serviceType, String
remark) {
    ActionLog actionLog = new ActionLog();

    actionLog.setUserId(userId);
    actionLog.setClientIpAddr(remoteAddr);
    actionLog.setTarget(target);
    actionLog.setServiceType(serviceType);
    actionLog.setRemark(remark);

    this.actionLogStore.insert(actionLog);
  }
}
```

# 5 기타 보호조치

| 항목명 | 개인정보처리시스템 접속 및 개인정보 조회·출력시 보호조치 적용 |
|---|---|
| 요구사항 내용 | **5.1 LIKE검색 제한**<br>– 개인정보 조회 시, 불필요하거나 과도한 정보가 조회되지 않도록 조치<br>**5.2 동시접속 제한**<br>– 개인정보취급자 계정 동시 접속 제한<br>**5.3 개인정보 표시 제한**<br>– 개인정보취급자 이용 시스템의 조회·출력 기능 구현 시 일괄적 개인정보 표시 제한 기준 수립·적용<br>– 개인정보 조회·출력 시 필요한 정보만 표시하여 출력항목 최소화 |

## 5.1 LIKE 검색 제한

### 기획 단계

- 개인정보 조회 시, 불필요하거나 과도한 정보가 조회되지 않도록 조치
  - 개인정보 조회 초기 화면에서, 개인정보 목록이 나타나지 않아야 함
  - 검색조건 없이 '검색' 버튼만 누를 경우, 전체결과가 나타나지 않아야 함
    ※ 한 개 이상의 검색조건을 입력해야 검색이 되도록 설계
  - 검색조건은 가급적 '일치조건'으로만 조회하도록 설계해야 함
    ※ 검색조건의 입력값에 조회되는 조건의 일부(이름/전화번호/ID 등 일부)만 입력하여 검색할 수 없도록 설계
  - 개인정보 업무 필요성에 따라 최소한의 개인정보만 취급해야 함
    ※ (예시) 고객상담 시스템에서 각 상담사가 자신이 상담한 고객의 상담내역만 조회할 수 있도록 하고 모든 상담사가 처리한 고객의 내역은 팀장(부서장)만 조회 수 있도록 함

**PbD 적용설계** | (제③원칙) 프라이버시 보호를 내재한 설계

LIKE 검색 제한을 통해, '일치검색' 또는 '두 가지 항목 이상의 검색' 조건만을 허용하여 개인정보 취급자의 과도한 개인정보 조회를 방지할 수 있다.

## 구현 예시

### 개인정보 조회 시 전체검색 제한기능 구현

```
/**
 * 개인정보처리시스템에서 개인정보를 처리하는 Controller
 */
@Controller
@RequestMapping(value="/user")
public class UserManageController {
 @Resource(name="userManager")
 private IUserManager userManager;

 private final static String LIST_URL = "/user/user-list";


 /**
 * 정보주체 목록 조회 화면을 호출
 *
* @return 결과를 보여줄 뷰 경로 {@link #LIST_URL}.
 */
 @RequestMapping(method=RequestMethod.GET,headers={"accept=text/html"})
 public String showListPage() {
  return LIST_URL;
 }


 /**
 * 정보주체 목록 조회 화면을 호출.
 *
 * @param pageNo 보려는 페이지 번호.
 * @param pageSize 한 페이지 당 항목 수.
 * @param request 요청 정보를 담고 있는 HttpServletRequest 객체.
 */
```

```
 @RequestMapping(method=RequestMethod.GET,headers={"accept=applicaton/json"})
 @ResponseBody
 public IQueryResult<IPageList<User>> getPageList(
  @RequestParam(value="pageNo",required=true,defaultValue="1") Integer pageNo ,
  @RequestParam(value="pageSize",required=true,defaultValue="10") Integer pageSize,
  HttpServletRequest request
 ) {
  pageNo = pageNo == null ?1 : Math.max(pageNo ,1);
  pageSize = pageSize == null ?10 : Math.max(pageSize,1);

  // 검색 파라미터
  String userName = request.getParameter("userName");
  String phoneNo = request.getParameter("phoneNo");
  String ssn = request.getParameter("ssn");
  String address = request.getParameter("addr");

  HashMap<String, String> queryParam = new HashMap<String, String>();

  if ( userName != null && userName.trim().length() > 0 ) queryParam.put("userName", userName);
  if ( phoneNo != null && phoneNo .trim().length() > 0 ) queryParam.put("phoneNo", phoneNo);
  if ( ssn != null && ssn .trim().length() > 0 ) queryParam.put("ssn", ssn);
  if ( address != null && address .trim().length() > 0 ) queryParam.put("address", address);

  QueryResult<IPageList<User>> queryResult = new QueryResult<IPageList<User>>();

  // 조회 조건이 없으면, 조회할 수 없도록 한다.
  if ( queryParam.isEmpty() ) {
   queryResult.setResult(IQueryResult.FAIL);
   queryResult.setMessage("검색 조건 없이 검색할 수 없습니다.");

   return queryResult;
  }

  IPageList<User> pageList = this.userManager.queryByPage(queryParam, pageNo, pageSize);
  queryResult.setData(pageList);

  return queryResult;
 }
}
```

## 5.2 동시접속 제한

### 기획 단계

- 개인정보취급자 계정의 동시접속 제한
  - 동일한 계정을 이용하여 동시접속을 수행하는 경우, 한 개의 접속만을 허용하도록 해야 함

※ 동시 접속이 허용되는 경우 신규 기기(IP, MAC 등)에서 동시 접속이 이루어지면 해당 사용자에게 알리는(SMS, 이메일 등) 기능을 설계할 수 있음

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

동시접속 제한 조치를 통해 개인정보취급자 계정을 여러 사람이 공유하는 것을 방지하고, 정당한 개인정보취급자가 아닌 비인가자의 접속을 방지할 수 있다.

### 구현 예시

**개인정보처리시스템 동시접속 방지기능 구현**

```
/**
 * 개인정보처리시스템의 로그인 요청을 담당하는 Controller.
 * 로그인 시 ServletContext에도 로그인 사용자 정보 등록
 * 로그아웃 시 ServletContext의 로그인 사용자 정보도 삭제
 */
@Controller
@RequestMapping(value="/admin")
public class AdminSignInController {

  /**
   * 로그인 요청 처리.
   */
  @RequestMapping(value="/login",method=RequestMethod.POST)
  public String doLogin(

    // 동시 접속 여부 확인한다.
    HttpSession session = request.getSession();
```

```
    ServletContext context = session.getServletContext();
    Map<String, User> sessionUserMap = (Map<String, User>) context.getAttribute(User.KEY);
    if ( sessionUserMap == null ) {
      sessionUserMap = new HashMap<String, User>();
      context.setAttribute(User.KEY, sessionUserMap);
    }

    // 이미 접속한 정보가 있을 때, 기존 연결을 로그아웃 시킨다.
    if ( sessionUserMap.get(userId) != null ) {
      sessionUserMap.remove(userId);
      context.setAttribute(User.KEY, sessionUserMap);
    }


    // 동시접속을 체크할 수 있도록 현재 개인정보취급자 정보를 기록한다.
    sessionUserMap.put(userId, user);
    context.setAttribute(User.KEY, sessionUserMap);

  }

  /**
   * 개인정보취급자가 로그아웃 시, 컨텍스트에서 취급자 정보를 삭제한다.
   */
  @RequestMapping(value="/loout",method=RequestMethod.POST)
  public String doLogout(
    HttpServletRequest request
  ) {

    // 컨텍스트의 사용자 정보 삭제
    ServletContext context = session.getServletContext();
    Map<String, User> sessionUserMap = (Map<String, User>) context.getAttribute(User.KEY);
    if ( sessionUserMap != null ) {
      sessionUserMap.remove(user.getId());
    }
    context.setAttribute(User.KEY, sessionUserMap);
    return showLoginPage(request);
  }
}
```

## 5.3 표시 제한조치

### 기획 단계

• 개인정보 표시제한(마스킹) 기준 수립 및 적용

- 조회·출력 기능 설계 시 표시제한이 필요한 개인정보(고유식별정보, 이름, 비밀번호, 주소, 전화번호, 계좌번호, 신용카드번호, 민감정보 등) 항목을 지정하고 지정된 개인정보 항목의 일부를 마스킹 해야 함

**개인정보 마스킹 기준 예시**

| 개인정보 | 설명 | 예시 |
|---|---|---|
| 성명 | 성명 중 이름의 첫 번째 글자 이상 | 홍*동 |
| 주민번호 | 뒤에서부터 6자리 | 711231-1****** |
| 여권번호 | 뒤에서부터 4자리 | 12345**** |
| 연락처 | 전화번호 또는 휴대폰 뒤 4자리 | 010-1234-**** |
| 카드번호 | 7번째에서 12번째 자리 | 9430-82**-****-2393 |
| 계좌번호 | 뒤에서부터 5자리 | 430-20-1***** |

**TIP**

• 개인정보를 다수의 개인정보처리시스템 등에서 각기 다른 방식으로 마스킹 할 경우, 다수의 개인정보처리시스템을 이용하여 개인정보취급자가 정보주체의 개인정보 집합을 구성할 수 있으므로 동일한 방식의 표시제한 조치가 필요함

(예시) A시스템에 저장된 "홍*동"의 정보와 B시스템에 저장된 "홍길*"의 정보를 조합하여 "홍길동"이라는 개인을 알아볼 수 있으므로, 적절한 마스킹 예시라고 볼 수 없음

**PbD 적용설계 | (제③원칙) 프라이버시 보호를 내재한 설계**

개인정보취급자가 정보주체의 정보를 조회할 때, 개인정보에 마스킹 조치를 통해 개인정보가 과도하게 노출되지 않도록 조치한다.

# III 개인정보처리시스템 운영 단계

## 제1절 개인정보처리시스템의 운영

1. 개인정보처리 위탁 시 준수사항
2. 개인정보처리시스템 원격접속 시 보안 조치
3. 개인정보처리시스템 취약점 진단
4. 개인정보처리시스템 접속기록 점검
5. 개인정보 유출 시 통지 및 신고
6. 개인정보 이용내역 통지

# 제1절 개인정보처리시스템의 운영

## 개요

본 절에서는 개인정보처리시스템 운영 단계에서 개인정보 담당자가 알아야 하는 조치사항을 기술하였다.

## 적용범위

개인정보처리시스템을 운영(운영 및 폐기)하는 단계에서 개인정보 담당자가 확인하고 적용해야 할 사항을 안내한다.

## 기본원칙

개인정보처리시스템을 운영하는 단계에서 개인정보보호를 위해 검토하고 확인하여야 할 기본원칙은 아래와 같다.

- 개인정보처리 업무 위탁 운영 시 적절한 관리 방안 마련 및 이행
- 개인정보처리시스템에 원격접속 시 보호조치 적용
- 개인정보처리시스템에 대해 주기적으로 취약점 진단 수행
- 개인정보처리시스템에 대한 접속기록 및 접근권한 주기적 검토
- 개인정보 처리 과정에서 개인정보 유출 시 정보주체 통지 및 전문기관 신고
- 수집한 개인정보에 대해서는 그 이용내역의 주기적인 통지

# 1 개인정보 처리 위탁 시 준수사항

**관련 근거**

- 개인정보 보호법 제26조(업무위탁에 따른 개인정보의 처리 제한)
- 개인정보 보호법 시행령 제28조(개인정보의 처리 업무 위탁 시 조치)

- 개인정보처리자가 업무의 효율을 위해 개인정보처리 업무를 위탁할 때아래의 사항을 포함하여 문서(계약서 등)로 하고 주기적으로 이행 점검을 하도록 한다.

① 위탁 업무의 목적 및 범위
② 위탁업무 수행 목적 외 개인정보의 처리 금지에 관한 사항
③ 재위탁 제한에 관한 사항
④ 개인정보에 대한 접근 제한 등 안전성 확보 조치에 관한 사항
⑤ 개인정보 관리 현황 점검 등 관리감독에 관한 사항
⑥ 수탁자의 손해배상 등 책임에 관한 사항

- 개인정보처리자는 개인정보 처리업무 위탁에 관한 사항을 온라인 서비스 등에 정보주체가 알기 쉽도록 공개하여야 한다. 개인정보처리방침에 이를 포함하여 공개하는 것도 가능하다.
- 업무를 위탁받아 처리하는 자(수탁자)가 개인정보 보호법을 위반하여 발생한 손해배상책임에 대하여, 수탁자를 개인정보처리자의 소속 직원으로 본다. 따라서, 개인정보 처리를 위탁하는 경우, 수탁자가 개인정보를 안전하게 처리하는지를 감독하여야 한다.

**주요 위반사례**

① 위탁 시 필수사항을 문서(계약서)에 미반영
- 위탁 시 필수사항을 문서에 일부 또는 전부 누락
  ➡ 제26조제1항 위반: 1천만원 이하의 과태료

② 개인정보 처리 위탁내용 및 수탁사 미공개
- 수탁자를 홈페이지에 공개하지 않음
- 수탁자 00개소 중 0개소 공개 누락
  ➡ 제26조제2항 위반: 1천만원 이하의 과태료

③ 수탁사 교육 및 실태점검 등 관리·감독 소홀
- 수탁사에 대한 개인정보 교육 미실시
  ➡ 제26조제4항 위반: 위반행위와 관련한 매출액의 100분의 3이하에 해당하는 과징금(정보통신서비스 제공자등에 해당)

## 2 개인정보처리시템 원격접속 시 보안조치

**관련 근거**

- 개인정보 보호법 제29조(안전조치의무)
- 개인정보의 안전성 확보조치 기준 제6조(접근통제)
- 개인정보의 기술적·관리적 보호조치 기준 제4조(접근통제)

- 외부에서 개인정보처리시스템 접속 시, 안전한 접속수단 또는 안전한 인증수단을 적용하여야 한다. 단, 정보통신서비스 제공자등의 경우 안전한 인증수단을 반드시 적용하여야 한다.

  - 안전한 인증수단의 적용: 개인정보취급자 이용 시스템에 사용자계정과 비밀번호를 입력하여 정당한 개인정보취급자 여부를 식별·인증하는 절차 이외에 추가적인 인증 수단의 적용을 말함 (예: 인증서, 보안토큰, 일회용 비밀번호 등)

**안전한 인증수단 예시**

| 인증수단 | 설명 |
|---|---|
| 인증서 (PKI) | 전자상거래 등에서 상대방과의 신원확인, 거래사실 증명, 문서의 위·변조 여부 검증 등을 위해 사용하는 전자서명으로서 해당 전자서명을 생성한 자의 신원을 확인하는 수단 |
| 보안토큰 | 암호 연산장치 등으로 내부에서 저장된 정보가 외부로 복사, 재생성 되지 않도록 공인인증서 등을 안전하게 보호할 수 있는 수단으로 스마트 카드, USB 토큰 등이 해당 |
| 일회용 비밀번호 (OTP) | 무작위 생성되는 난수를 일회용 비밀번호로 한 번 생성하고, 그 인증 값이 한 번만 사용 가능하도록 하는 방식 |

**TIP**

- 안전한 인증수단만을 적용하는 경우에 통신 보안을 위한 암호화 기술의 추가 적용이 필요할 수 있으므로, 보안성 강화를 위하여 안전한 접속수단을 권고한다.

  - 안전한 접속수단의 적용: 보안프로토콜을 이용하여 암호화 통신을 적용하거나, 물리적으로 독립된 회선을 사용하여 통신하는 등 정보통신망을 통해 개인정보처리시스템에 안전하게 접속하는 수단의 적용을 말함(예: VPN, 전용선 등)

**안전한 접속수단 예시**

| 인증수단 | 설명 |
|---|---|
| 가상사설망 (VPN) | 개인정보취급자가 사업장 내의 개인정보처리시스템에 대해 원격으로 접속할 때 IPsec이나 SSL 기반의 암호 프로토콜을 사용한 터널링 기술을 통해 안전한 암호통신을 할 수 있도록 해주는 보안 시스템을 의미 |
| 전용선 | 물리적으로 독립된 회선으로서 두 지점간에 독점적으로 사용하는 회선으로 개인정보처리자와 개인정보취급자, 또는 본점과 지점간 직통으로 연결하는 회선 등을 의미 |

# 3 개인정보처리시스템 취약점 진단

**관련 근거**

- 개인정보 보호법 제29조(안전조치의무)
- 개인정보의 안전성 확보조치 기준 제5조(접근통제)

- 인터넷 홈페이지를 통해 고유식별정보를 처리하는 경우, 고유식별정보가 유출·변조·훼손되지 않도록 연 1회 이상 웹 취약점을 점검해야 한다.

**TIP**

- 중소기업의 경우, 한국인터넷진흥원에서 제공하는 무료 원격 웹 취약점 점검 서비스를 이용해 웹사이트의 취약점 진단을 수행할 수 있다.
  ※ 관련URL: https://www.boho.or.kr/webprotect/webVulnerability.do

- 인터넷 홈페이지의 취약점 점검은 개인정보처리자의 자체인력, 보안업체 등을 활용할 수 있으며, 취약점 점검은 상용 도구, 공개용 도구, 자체 제작 도구 등을 사용할 수 있다.

**웹 취약점 표준 점검항목('19년)**

| No. | 코드 | 점검항목 | No. | 코드 | 점검항목 | No. | 코드 | 점검항목 |
|---|---|---|---|---|---|---|---|---|
| 1 | OC | 운영체제 명령 실행 | 8 | BF | 약한 문자열 강도(브루트포스) | 15 | FD | 경로추적 및 파일다운로드 |
| 2 | SI | SQL 인젝션 | 9 | IN | 불충분한 인증 및 인가 | 16 | AE | 관리자페이지 노출 |
| 3 | XI | XPath 인젝션 | 10 | PR | 취약한 패스워드 복구 | 17 | PL | 위치공개 |
| 4 | DI | 디렉토리 인덱싱 | 11 | SM | 불충분한 세션 관리 | 18 | SN | 데이터 평문전송 |
| 5 | IL | 정보누출 | 12 | CF | 크로스사이트 리퀘스트 변조 (CSRF) | 19 | CC | 쿠키 변조 |
| 6 | CS | 악성콘텐츠 | 13 | AU | 자동화공격 | 20 | MS | 웹 서비스 메소드 설정 공격 |
| 7 | XS | 크로스사이트 스크립트(XSS) | 14 | FU | 파일업로드 | 21 | UP | URL/파라미터 변조 |

※ 출처: 전자정부 서비스 웹 취약점 표준 점검항목(행정안전부, 2019.8.)

**TIP**

- 취약점 점검 및 조치에 활용할 수 있는 기술문서는 아래와 같이 다양한 자료가 있다.
  - 공개SW를 활용한 소프트웨어 개발보안 점검 가이드(행정안전부, 2019.6.)
  - 소프트웨어 보안약점 진단가이드(행정안전부, 2019.6.)
  - 소프트웨어 개발보안 가이드(행정안전부, 2019.11.)
  - 시큐어코딩가이드(C, Java)(행정안전부, 2012.9.)
  - 기술적 취약점 분석·평가 방법 상세가이드(과학기술정보통신부, 2017.12.)

# 4 개인정보처리시스템 접속기록 점검

**관련 근거**

- 개인정보 보호법 제29조(안전조치의무)
- 개인정보의 안전성 확보조치 기준 제8조(접속기록의 보관 및 점검)
- 개인정보의 기술적·관리적 보호조치 기준 제5조(접속기록의 위·변조방지)

- 개인정보취급자가 개인정보처리시스템에 접속하여 개인정보를 열람·수정·삭제·출력 등의 작업을 한 경우 시스템 이상 유무의 확인 등을 위해 접속기록을 1년 이상 접속기록을 보관·관리해야 한다.

  - 다만, 다음에 해당하는 경우는 접속기록을 2년이상 보관·관리해야 한다.

    1) 5만명 이상의 정보주체를 처리하는 경우
    2) 고유식별정보 또는 민감정보를 처리하는 경우
    3) 「전기통신사업법」제5조에 따른 기간통신사업자의 경우

- 접속기록은 월 1회 이상 정기적으로 확인 및 감독해야 하며, 개인정보 처리를 위한 업무수행과 관련이 없거나 과도한 개인정보의 조회, 정정, 다운로드, 삭제 등 비정상적인 행위를 탐지하고 적절한 대응조치를 하여야 한다.

**접속기록 내 비정상 행위 예시**

- 계정 : 접근권한이 부여되지 않은 계정으로 접속한 행위 등
- 접속일시 : 새벽시간, 휴무일 등 업무시간 외에 접속한 행위 등
- 접속지 정보 : 인가되지 않은 단말기 또는 지역(IP)에서 접속한 행위 등
- 정보주체 정보 : 특정 정보주체에 대하여 과도하게 조회, 다운로드 등의 행위 등
- 수행업무 : 대량의 개인정보에 대한 조회, 정정, 다운로드, 삭제 등의 행위 등
- 그 밖에 짧은 시간에 하나의 계정으로 여러 지역(IP)에서 접속한 행위 등

  - 체계적인 점검을 수행하고 점검의 효율성을 확보하기 위해서, 아래와 같은 사항 등을 사전에 계획하고 시행할 수 있다.

① 점검주체 : 개인정보보호 부서, 감사부서, 개인정보보호 전문업체 등
② 점검시기 : 월 1회 이상
③ 점검 항목 및 내용 : 비인가된 개인정보 처리 및 대량의 개인정보 다운로드 등 비정상 행위탐지, 접속기록의 위·변조여부 등
④ 점검 후속조치 : 개선조치, 결과보고 등

# 5 개인정보 유출시 통지 및 신고

**관련 근거**

- 개인정보 보호법 제34조(개인정보 유출통지 등)
- 개인정보 보호법 제39조의4(개인정보 유출등의 통지·신고에 대한 특례)
- 개인정보 보호법 시행령 제39조(개인정보 유출 신고의 범위 및 기간)
- 개인정보 보호법 시행령 제40조(개인정보 유출 통지의 방법 및 절차)
- 개인정보 보호법 시행령 제48조의4(개인정보 유출 등의 통지·신고에 관한 특례)

• 개인정보가 유출되었음을 알게 되었을 때에는 지체 없이 해당 정보주체(이용자)에게 서면, 전자우편 등을 통해 유출 사실을 알려야 한다.

- 만약, 구체적인 유출 내용확인을 하지 못한 경우에(개인정보 유출시 통지 항목 1, 2번) 개인정보가 유출된 사실과 유출이 확인된 사항만을 먼저 알리고, 나중에 확인되는 사항을 추가로 알릴 수 있다.

**개인정보 유출 시 통지 항목**

| 대상 | 개인정보처리자 | 정보통신서비스 제공자등 |
|---|---|---|
| 정보주체 통지 항목 | 1. 유출된 개인정보의 항목 | 1. 유출등이 된 개인정보 항목 |
| | 2. 유출된 시점과 그 경위 | 2. 유출등이 발생한 시점 |
| | 3. 유출로 인하여 발생할 수 있는 피해를 최소화하기 위하여 정보주체가 할 수 있는 방법 등에 관한 정보 | 3. 이용자가 취할 수 있는 조치 |
| | 4. 개인정보처리자의 대응조치 및 피해 구제절차 | 4. 정보통신서비스 제공자등의 대응 조치 |
| | 5. 정보주체에게 피해가 발생한 경우 신고 등을 접수할 수 있는 담당부서 및 연락처 | 5. 이용자가 상담 등을 접수할 수 있는 부서 및 연락처 |

- (개인정보처리자의 경우) 1천명 이상 정보주체의 개인정보 유출시, 정보주체 개별 통지와 동시에 홈페이지에 '정보주체 통지항목'을 홈페이지에 7일 이상 게재하여야 한다.

  ※ 홈페이지가 없을 경우, 서면·사업장 등 보기 쉬운 곳에 7일간 게시해야 함

- (정보통신서비스 제공자등의 경우) 이용자의 연락처를 알 수 없는 등 정당한 사유가 있는 경우, 자신의 인터넷 홈페이지에 30일 이상 게시하여 접속하는 이용자가 알 수 있도록 유지해야 한다.

• 피해확산 방지를 위해 아래 '개인정보 유출 신고 기준'에 따라 개인정보보호위원회 또는 한국인터넷진흥원에 신고하여야 한다.

- 개인정보보호 포털(https://privacy.go.kr) 〉 민원마당 〉 개인정보 유출·침해신고

**개인정보 유출 신고기준**

| 대상 | 개인정보처리자 | 정보통신서비스 제공자등 |
|---|---|---|
| 신고기준 | 1천명 이상 정보주체의 개인정보 유출 시 | 1명 이상 이용자의 개인정보 유출 시 |
| 신고기한 | 지체 없이(5일 이내) | 지체 없이(24시간 이내) |

**개인정보 유출 표준 통지문안**

| 표준 통지문안 예시 | 부가 설명 |
|---|---|
| 개인정보 유출 사실을 통지해 드리며, 깊이 사과드립니다. | 〈제목〉<br>- '유출 통지' 문구 포함 |
| 고객님의 개인정보 보호를 위해 최우선으로 노력하여 왔으나, 불의의 사고로 고객님의 소중한 개인정보가 유출되었음을 알려 드리며, 이에 대하여 진심으로 사과를 드립니다. | (사과문)<br>- 유출 통지 사실 알림<br>- 사과문을 먼저 표현 |
| 고객님의 개인정보는 2020년 3월 5일 회원관리시스템 장애 처리를 위한 데이터 분석 과정에서 유지보수업체로 전달되었고, 유지보수업체는 자체 서버에 저장·보관하다가 안전한 조치를 다하지 못해 2020년 4월경 해커에 의한 해킹으로 유출되었습니다.<br>유출된 정확한 일시는 ○○경찰청에서 현재 수사가 진행 중이며, 확인되면 추가로 알려 드리도록 하겠습니다. | 〈유출된 시점과 그 경위〉<br>- 유출된 시점과 경위를 누구나 이해할 수 있게 상세하게 설명<br>- '귀하', '고객님' 등으로 유출된 정보주체 명시<br>※ 부적합한 표현 : 일부 고객, 회원정보의 일부<br>- 추가 확인된 사항은 반드시 추가로 통지 |
| 유출된 개인정보 항목은 이름, 아이디(ID), 비밀번호, 주민등록번호, 이메일, 연락처 등 총 6개입니다. | 〈유출된 개인정보의 항목〉<br>- 유출된 항목을 누락 없이 모두 나열<br>※ '등'으로 생략하거나, '회사전화번호' 및 '집전화번호'를 합쳐서 '전화번호'로 표시 안됨 |
| 유출 사실을 인지한 후 즉시 해당 IP와 불법접속 경로를 차단하고, 취약점 점검과 보완 조치를 하였습니다. 또한, 유지보수업체 서버에 있던 귀하의 개인정보는 즉시 삭제 조치하였습니다. | 〈개인정보처리자의 대응조치〉<br>- 접속경로 차단 등 예시된 항목 외에도 망 분리, 방화벽 설치, 개인정보 암호화, 인증 등 접근 통제, 시스템 모니터링 강화 등 조치한 내용 설명 |
| ○○경찰청이 발표한 수사 결과에 따르면 현재 해커는 검거되었고, 해커가 불법 수집한 개인정보는 2차 유출하거나 판매하지는 않은 것으로 확인되었습니다.따라서 현재로서는 이번 사고로 인한 2차 피해가 발생할 가능성이 높지 않아 보이나, 혹시 모를 피해를 최소화하기 위하여 귀하의 비밀번호를 변경하여 주시기 바랍니다.<br>그리고 개인정보 악용으로 의심되는 전화, 메일 등을 받으시거나 기타 궁금하신 사항은 연락주시면 친절하게 안내해 드리고, 신속하게 대응하도록 하겠습니다. | 〈피해 최소화를 위한 정보주체의 조치방법〉<br>- 유출 경위에 따라 정보주체가 할 수 있는 방법을 안내<br>- 사건에 따라 다양한 피해를 추정하여 예방 가능한 방법을 모두 안내(보이스 피싱, 피싱 메일, 불법 TM, 스팸문자 등) |
| 아울러, 피해가 발생하였거나 예상되는 경우에는 아래 담당부서에 신고하시면 성실하게 안내와 상담을 해 드리고, 필요한 조사를 거쳐 손실보상이나 손해배상 등의 구제절차를 진행하도록 하겠습니다.<br>한국인터넷진흥원의 개인정보 분쟁 조정이나 민사 상 손해배상 청구, 감독기관인 0000부 민원신고센터 등을 통해 피해를 구제받고자 하실 경우에도 연락주시면 그 절차를 안내하고 필요한 제반 지원을 아끼지 않도록 하겠습니다. | 〈개인정보처리자의 피해 구제절차〉<br>- 보상이나 배상이 결정된 경우에는 그 내용을 상세히 기재<br>- 보상이나 배상이 결정되지 않은 경우 계획과 절차를 안내<br>- 감독기관 등을 통한 구제절차도 안내 |
| 앞으로 장애처리 과정에 대한 개인정보 보호 조치 강화 등 내부 개인정보 보호 관리체계를 개선하고, 관계 직원 교육을 통해 인식을 제고하여, 향후 다시는 이와 유사한 사례가 발생하지 않도록 최선의 노력을 다하겠습니다. | (개인정보처리자의 향후 대응계획)<br>- 추가적인 향후 대응계획을 포함 |
| 항상 믿고 사랑해 주시는 고객님께 심려를 끼쳐 드리게 되어 거듭 진심으로 사과드립니다. | (사과문) |
| • 피해 등 접수 담당부서 : 고객지원과<br>• 피해 등 접수 전화번호 : 02-○○○-○○○○<br>• 피해 등 접수 e-메일주소 : ○○○@○○○.co.kr | 〈피해 등 신고 접수 담당부서 및 연락처〉<br>- 전담처리부서 안내를 원칙으로 하되, 대량 유출로 일시적으로 콜센타 등 다른 부서를 지정한 경우 해당 부서를 안내 |
| (주)OOO 임직원 일동 | (발신명의) |

※ 부가설명 란에 필수사항은 〈 〉, 참고사항은 ( )로 표기하였음
※ 필수사항이 확인되지 않아 통지문에 포함하지 않은 경우 추후 확인되면 반드시 추가 통지
※ 예시를 참고하여 유출 상황에 적합하게 내용을 변경하여 활용

# 6 개인정보 이용내역 통지

**관련 근거**

- 개인정보 보호법 제39조의8(개인정보 이용내역의 통지)
- 개인정보 보호법 시행령 제48조의6(개인정보 이용내역의 통지)

• 정보통신서비스 제공자등은 이용자가 자신의 개인정보 이용내역을 정확히 알고 자기정보를 통제할 수 있도록 하기 위해, 개인정보 이용내역을 연 1회 이상 이용자에게 주기적으로 통지하여야 한다.

**개인정보 이용내역 통지 대상자 및 항목**

| 구분 | 세부내용 |
|---|---|
| 통지의무 대상자 | 아래 두가지 요건 중 하나라도 해당하는 정보통신서비스 제공자<br>– 정보통신서비스 제공자로서 전년도 말 기준 직전 3개월간 개인정보가 저장·관리되고 있는 이용자 수가 일일평균 100만명 이상<br>– 정보통신 서비스 부문 전년도 매출액이 100억원 이상인 정보통신서비스 제공자 |
| 통지 내용 | 1. 개인정보의 수집 · 이용 목적 및 수집한 개인정보의 항목<br>2. 개인정보를 제공받은 자와 그 제공 목적 및 제공한 개인정보의 항목<br>※「통신비밀보호법」 제13조, 제13조의2, 제13조의4 및 「전기통신사업법」 제83조 제3항에 따른 예외 존재 |
| 통지 방법 | 전자우편, 서면, 전화 또는 이와 유사한 방법 중 어느 하나 선택 |
| 통지 주기 | 연 1회 이상 |

- 만약, 연락처 등 이용자에게 통지할 수 있는 개인정보를 수집하지 않은 경우 이용내역을 통지하지 않아도 된다.

**개인정보 이용내역 통지 예시**

• 개인정보의 수집·이용 목적 및 수집한 개인정보의 항목
  – 회원가입 : 성명, 휴대전화번호, 이메일 주소
  – 서비스 이용 및 상담 : 성명, 휴대전화번호, 이메일주소
  – 결제서비스 제공 : 신용카드사명, 카드번호, 유효기간
  – 취소·환불 : 은행명, 계좌번호
  – 배송 : 수취인 성명, 휴대전화 번호, 주소

• 개인정보를 제공받은 자와 그 제공목적 및 제공한 개인정보의 항목

| 연번 | 서비스명 | 제공받는 자 | 목적 | 항목 |
|---|---|---|---|---|
| 1 | ○○페이 이용회원 | (계좌결제) ○○은행,<br>(신용카드 결제) ○○카드,<br>(휴대폰결제) ○○텔레콤,<br>※ 제공한 업체 목록 모두 기재 | 상품 구매 및 배송 서비스 제공 | 아이디, 성명, 이메일 주소, 휴대전화번호, 상품구매정보, 수취인정보(성명, 주소, 전화번호) |
| 2 | ○○예약 | ○사, ○사, ○사<br>※ 제공한 업체 목록 모두 기재 | 예약, 본인의사의 확인 고객상담 및 불만처리 | 아이디, 성명, 전화번호, 예약내역 |

## 참고문헌


[1] 개인정보영향평가 수행안내서, 행정안전부/한국인터넷진흥원, 2018.4.

[2] 개인정보의 기술적 관리적 보호조치 기준 해설서, 행정안전부/한국인터넷진흥원, 2017.12.

[3] 소프트웨어 개발보안 가이드, 행정자치부/한국인터넷진흥원, 2017.01.

[4] 암호기술 구현 안내서, 미래창조과학부, 2013.12.

[5] 암호이용안내서, 방송통신위원회, 2010.01.

[6] 암호정책 수립기준 안내서, 방송통신위원회, 2010.01

[7] 웹사이트 회원탈퇴 기능 구현 안내서, 한국인터넷진흥원, 2009.10.

[8] 정보보호 사전점검 해설서, 방송통신위원회/한국인터넷진흥원, 2018.6.

[9] 정보보호시스템 구축을 위한 실무가이드, 과학기술정보통신부/한국인터넷진흥원, 2018.6.

[10] 정보시스템 개발·운영자를 위한 홈페이지 취약점 진단·제거 가이드, 한국인터넷진흥원, 2013.12.

[11] 패스워드 선택 및 이용안내서, 과학기술정보통신부/한국인터넷진흥원, 2019.06.

[12] 홈페이지 개발 보안안내서, 한국인터넷진흥원, 2010.01.

**유의사항**

본 가이드는 중·소사업자 등이 개인정보를 처리하기 위한 온라인 서비스 및 개인정보처리시스템을 구축·운영할 때, 「개인정보 보호법」에 따른 법적 요구사항 준수를 돕기 위해 발간된 가이드입니다. 각 장에서 제시하고 있는 시스템의 설계·구현 및 운영에 관한 예시는 개발자 및 운영자가 개인정보를 안전하게 처리하기 위한 개인정보처리시스템의 기획, 개발, 운영 시 활용할 수 있는 참고자료일 뿐 절대적이지 않습니다.

특히, 본 가이드에 수록된 내용(향후 법제도 현황에 따라 본 가이드의 요구사항은 변경될 수 있음) 이외에도 개인정보가 유출·변조·훼손되지 않도록 추가적으로 시큐어코딩 적용, 보안약점 제거 등 보안활동을 통해 그 결과에 따른 개선 조치를 해야 합니다. 취약점 점검 및 조치에 활용할 수 있는 기술문서는 아래와 같이 다양한 자료를 활용할 수 있습니다.

- 개인정보 수집제공 동의서 작성 가이드라인(행정안전부, 2018.3.)
- 개인정보의 암호화 조치 안내서(행정안전부, 2017.1.)
- 개인정보 처리 위 · 수탁 안내서(행정안전부, 2018.6.)
- 공개소프트웨어를 활용한 소프트웨어 개발보안 진단가이드(행정안전부, 2019.6.)
- 소프트웨어 보안약점 진단가이드(행정안전부, 2019.6.)
- 소프트웨어 개발보안 가이드(행정안전부, 2019.11)
- 시큐어코딩가이드(C, JAVA)(행정안전부, 2016.3.)
- Web 2.0 정보보호 실무가이드(행정안전부, 2010.5.)
- 홈페이지 개인정보 노출방지 안내서(행정안전부, 2018.11.)
- 패스워드 선택 및 이용 안내서(과학기술정보통신부, KISA, 2019) 등