DID 인증기술_Capstone Design

KCDSA와 TLS를 이용한
전자서명 계약서 웹 개발

• 지도 교수 : 신 승 수

• 학 과 : 정보보호학과

• 팀 명 : 거짓말탐지기

목 차

그림 목차

표 목차

Ⅰ. 서론

프랜차이즈 등 다양한 시장의 발전으로 회사 또는 개인 사업장이 늘어남에 따라 계약서를 작성하는 빈도가 증가하고 있다. 통계청 자료에 의하면 2017년 총사업자는 약 723만 명에서 2021년 약 920만 명으로 4년 사이 27.2% 증가하였다[1].

사업자가 늘어나면서 사업장의 계약 또한 증가하게 되었는데, 대표적으로 R-ONE의 상업용 부동산 임대 동향에 따르면 전국의 상가 공실률은 2018년 2분기부터 2022년 2분기까지 3.2% 줄어들었으며 수치는 매 분기 점차 감소하고 있다[2]. 공실률이 줄어듦으로써 계약서 작성 빈도수도 증가하고 있으며 상가 계약뿐만 아니라 중고차 시장, 근로 계약서 등 다양한 분야에서 계약서 작성이 활발하게 이루어지고 있다.

하지만 개인 사업자나 중소기업의 경우 계약서 작성에 있어 종이 계약서를 이용하는 상황이 많이 발생한다. 이 경우 문서를 물리적으로 보관하여야 하며 훼손될 가능성도 있다[3]. 또한, 법적 문제가 발생하였을 경우 계약서 작성자 또는 서명자가 계약서 자체를 부인할 가능성도 존재한다. 이후에 발생할 법적 문제를 우선으로 방지하기 위하여 물리적인 계약서가 아닌 플랫폼을 통한 전자서명 계약서로 작성 및 보관하는 시스템이 필요하다.

전자서명 계약서 시스템을 제공하는 회사가 존재하지만 주로 대기업, 공공기관 등을 주 고객층으로 비즈니스 모델을 구축하여 사업을 진행하고 있다. 따라서 시스템을 이용하는 비용이 비싸므로 개인 사업자나 중소기업은 금전적인 부담으로 인하여 해당 서비스를 이용하지 못하는 상황이다. 따라서 본 논문에서는 이러한 문제점들을 파악하고 개인 사업자도 쉽고 간편하게 전자서명 계약서를 작성 및 보관할 수 있도록 전자서명 계약서 웹을 개발하고자 한다.

Ⅱ. 동향 분석

1. 기존의 계약서

가. 계약서 종류

국가법령정보센터에서 정의한 계약의 종류로 전형계약과 비전형계약, 쌍무계약과 편무계약, 유상계약과 무상계약, 낙성계약과 요물계약으로 구분된다[4]. 그중 일반적으로 자주 작성되는 계약서로는 근로 계약서, 임대차 계약서, 토지 매매 계약서, 아파트 매매 계약서 등이 있다. 이외에도 B2B, B2C, C2C에서 발생하는 다양한 계약서들이 존재하며, 일반적으로는 양측이 대면으로 종이 계약서를 통해 작성 및 서명하고 있다.

나. 종이 계약서 문제점

기존의 종이 계약서 작성 방식은 양측이 대면으로 만나 계약서 내용을 검토 및 합의하여 날인 또는 서명하는 구조로 이루어진다. 하지만 이러한 방식은 양측이 시간과 장소를 합의하여 오프라인으로 계약서를 작성해야 하므로 시간, 공간 측면에서 비효율적이다. 또한, 계약서 내용이 양측 모두 합의점을 맞추지 못한 경우 새로운 계약서 종이를 출력하여 다시 작성해야 하며, 해당 자료를 온라인으로 보관하기 위해서는 스캔을 추가로 수행하여야 하는 번거로움이 존재한다. 종이 계약서를 작성하는 데 크게 문제가 없다 하더라도 물리적으로 보관해야 하기에 훼손, 분실, 도난, 기밀 정보 유출 등의 문제점이 발생한다. 위의 문제점뿐 아니라 법적 효력에 부합할 수 있도록 서명, 날인 하지 않는 개인 사업자와 중소기업들도 다수 존재하기 때문에 추후 계약서 서명에 대한 법적 다툼까지 이어질 수 있다. 따라서 종이 계약서로 작성하는 기존 방식과 문제점에서 벗어나 온라인 환경에서 간편하게 계약서를 작성 및 서명, 보관할 수 있는 플랫폼이 필요하다.

다. 모두 싸인 시스템

모두 싸인이란 개인 또는 회사를 대상으로 온라인을 통해 계약서를 작성 및 보관하는 전자계약 서비스를 제공하는 회사이다. 해당 서비스를 이용하는 주 고객은 대부분 대기업, 공공기관으로 이루어져 있으며, 국제표준화기구(ISO) 국제전기기술위원회(IEC)에서 ISO27001 보안 인증을 획득하였다.

모두 싸인 시스템은 문서 그대로 업로드하여 계약서를 작성하는 방식이며 서명 도장도 시스템 내에서 제작하여 계약서에 직접 삽입할 수 있다. Client가 계약서를 작성한 후 이용약관 동의 및 서명을 완료하면 서명이 입력된 문서와 감사 추적 인증서가 모든 서명자의 이메일로 전송하게 된다.

전자서명이 이루어진 계약서의 법적 효력은 대한민국 민법에 따라 계약 시 별도의 형식을

요구하지 않고 당사자 간의 합의만으로 계약의 성립을 인정하는 낙성 불요식 계약 원칙을 따르고 있다. 계약 당사자가 계약 내용에 대해서 동의했다는 사실을 증명만 할 수 있다면 그 형태가 무엇이든 법적 효력이 인정된다[5].

2. KCDSA 전자서명 알고리즘

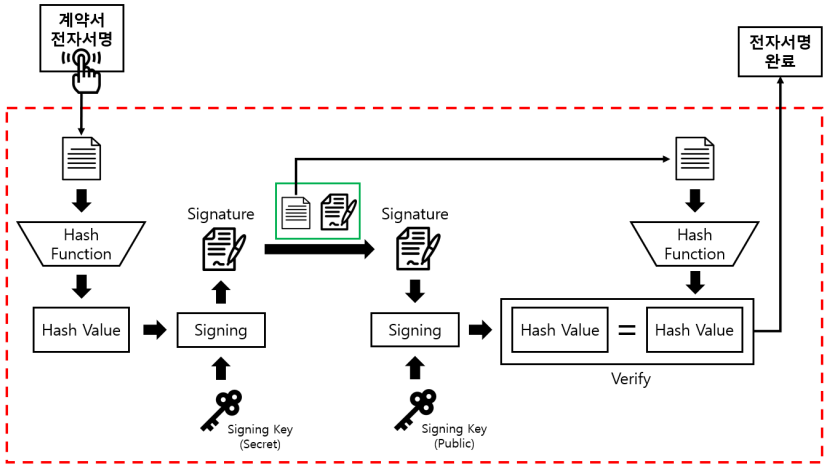
KCDSA 전자서명 알고리즘은 KISA에서 개발한 인증서 기반 부가형 전자서명 알고리즘으로 ElGamal 서명 방식의 변형으로 이산대수 문제에 안전성을 두고 있으며, 1998년 10월 한국정보통신 기술협회에서 표준으로 제정되었다[6]. KCDSA는 가변 크기의 데이터를 고정된 값으로 생성할 수 있는 충돌 방지 암호화 해시 함수가 필요하기에 SHA-256을 적용하여 안전한 알고리즘으로 구현하였다.

가. KCDSA 전자서명 흐름도

Client가 계약서 전자서명 버튼을 클릭할 경우 KCDSA 알고리즘을 통해 계약서에 대한 전자서명이 수행되는 과정은 다음과 같다.

계약서 전자서명 버튼을 클릭하게 되면 SHA-256(해시 함수)을 통해 계약서의 내용에 대한 해시값을 생성하게 되며, 생성된 해시값은 Client의 개인 키로 암호화한다. 암호화된 파일은 서버로 전달하게 되고 서버는 Client의 공개 키를 이용하여 파일을 복호화한다. 또한, 계약서 평문을 Client가 사용한 SHA-256 해시 함수를 이용하여 해시값을 생성한다. 서버가 직접 생성한 계약서 평문의 해시값과 Client로부터 받은 해시값이 일치한 지 비교하여 계약서가 위·변조되지 않았음을 검증한다. 해시값이 일치할 경우 최종적으로 Client의 전자서명이 완료하게 된다.

KCDSA 전자서명 알고리즘이 동작 되는 흐름은 [그림 1]과 같다.



[그림 1] KCDSA 알고리즘 흐름도

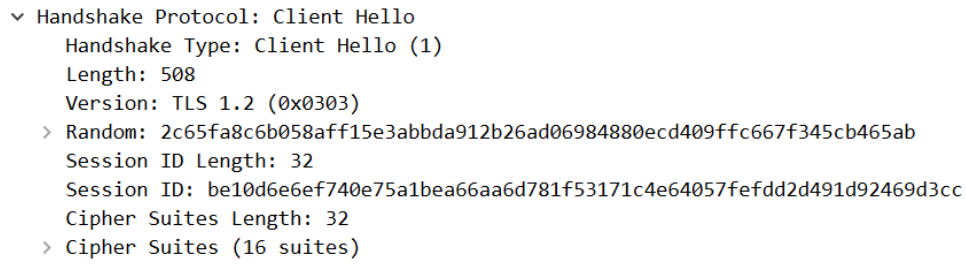
3. SSL/TLS

SSL(Secure Socket Layer) 및 TLS(Transport Layer Security)란 전송 계층 상에서 Client, 서버에 대한 인증 및 데이터 암호화를 수행한다. 클라이언트와 서버 양단간 응용 계층 및 TCP 전송 계층 사이에 안전한 보안 채널을 형성해주는 보안용 프로토콜을 의미한다. HTTPS

로 가장 많이 쓰이고 있으며, 이외에도 FTP, TELNET, SMTP, IMAP 등에서 사용할 수 있다[7].

가. SSL/TLS 동작

SSL과 TLS는 TCP의 일종으로 3-Way Handshake를 통해 통신을 시작한다. 먼저 브라우저는 웹 서버에게 Client Hello 메시지를 보낸다. 이때, 브라우저가 사용하는 SSL/TLS 버전 정보를 확인하고 브라우저가 지원하는 Cipher Suite(암호화 방식), 난수 등을 전송한다. 이후 웹 서버는 클라이언트에게 Server Hello 메시지를 보내며 Cipher Suite(암호화 방식)를 선택하고 웹 서버의 공개키가 담긴 SSL/TLS 인증서와 난수 등을 전송한다. 클라이언트는 브라우저와 서버의 난수를 사용하여 Pre-master Secret 키를 생성하며 서버 17은 Pre-master Secret 값을 복호화한 후 SSL/TLS Handshake는 종료되고 HTTPS 통신이 시작된다[8]. [그림 2]는 TLS v1.2의 브라우저가 웹 서버에게 보낸 Client Hello 메시지 내용이다.



[그림 2] 3-Way Handshake 중 첫 번째 Client Hello 메시지

나. TLS v1.2

TLS v1.0에서는 SSL 3.0의 개선판으로 공개되었으며 SSL 3.0의 취약점을 해결하였다. 이후 공개된 TLS v1.1은 암호 블록체인 공격에 대한 방어와 인터넷 할당 번호 관리 기관(IANA : Internet Assigned Numbers Authority) 등록 Parameter 지원이 추가되었다. 또한, 2021년부터 TLS v1.2 미만에 대한 지원이 중단됨에 따라 SHA-1에서 SHA-2로 Hash 알고리즘을 변경하여 인증 암호화를 강화였으며, TLS 확장 및 AES 암호를 추가하여 2008년 8월, TLS v1.2를 출시하였다. 따라서 본 논문에서는 취약점을 개선하고 보안성을 검증받은 TLS v1.2를 사용한다.

Ⅲ. 전자서명 계약서 시스템

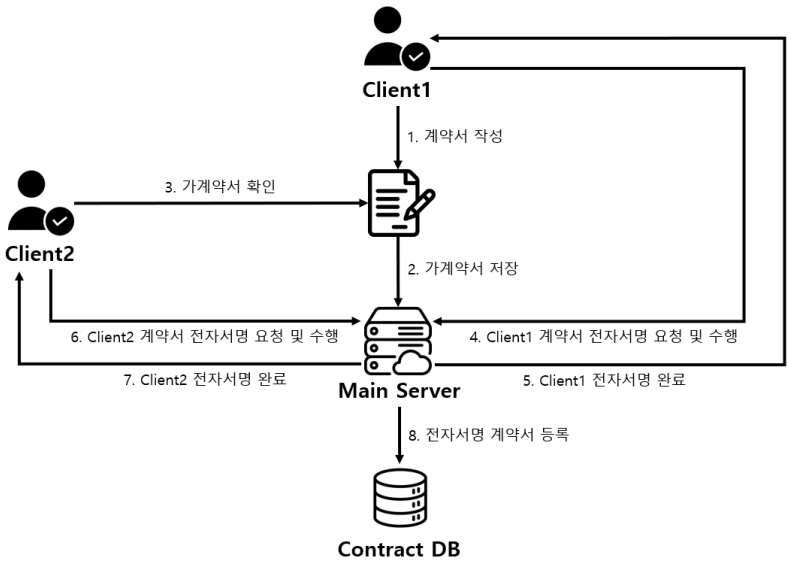
본 장에서는 Client 들의 계약서 작성 및 전자서명 계약서가 최종 등록되는 전체적인 내용을 구성과 흐름으로 나누어 상세히 설명한다. 웹 시스템은 Client1, Client2, 서버, 전자서명 계약서 DB로 분류되며, 전체 시스템은 2계층으로 구성된다.

1. 웹 시스템

본 절에서는 시스템 구성과 시스템이 수행되는 흐름도, 인가된 사용자만 파일에 접근하는 방법에 관해 기술한다.

가. 시스템 구성

시스템 구성은 Client1, Client2, 서버, 계약서 저장 DB로 이루어진다.
Client1은 초기 계약서를 작성한 후 Client2와 함께 가계약서 수정 및 전자서명을 수행한다. Client2는 요청받은 가계약서를 수정 및 검토 후 전자서명을 수행한다. 서버는 가계약서를 저장하고 Client들의 전자서명 요청을 받아 수행한 후 파일을 검증한다. 또한, Client들의 파일 접근 권한을 제어한다. 파일에 인가된 Client들의 전자서명이 완료되면 DB에 최종 전자서명 계약서 정보를 저장한다. 시스템의 전체 구성도는 [그림 3]과 같다.



[그림 3] 시스템 구성도

나. 시스템 흐름도

송・수신자 간의 계약서 작성 및 전자서명 계약서가 최종 등록되는 과정은 다음과 같다.

① 송신자인 Client1은 회원가입 및 로그인을 수행한 후 계약서를 작성한다.

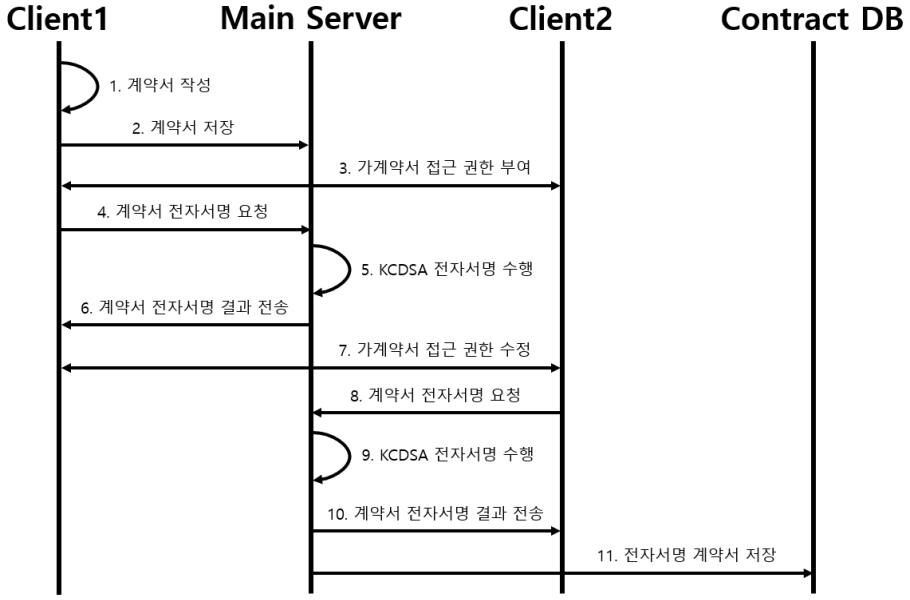
② 계약서 작성을 완료한 후 수신자를 지정하고 가계약서를 저장한다. 서버는 해당 요청을 받은 후 Client1과 Client2에게 가계약서를 열람 및 수정할 수 있도록 권한을 부여한다.

③ 수신자인 Client2는 계약서를 열람한 후 Client1과 의사소통하여 최종적인 계약서를 작성한다.

④ Clien1과 Client2는 서버에게 최종 계약서에 대한 전자서명 요청 및 서명을 수행한다.

⑤ 서버는 송・수신자의 전자서명이 이루어짐을 검증한 후 전자서명 계약서를 저장한다.

시스템 흐름도는 [그림 4]과 같다.



[그림 4] 시스템 흐름도

다. 계약서의 접근 권한 제어

(1) PHP를 이용한 계약서의 접근 권한 제어

본 제안서에는 리눅스의 접근 권한 명령어 ‘chmod’를 이용하여 파일의 접근 권한을 제어한다. 계약서는 작성자인 Client1과 계약 주체에 해당하는 Client2만 열람 및 수정할 수 있도록 하여 기밀성을 제공한다. Client1 또는 Client2 중 하나라도 계약서에 전자서명을 완료한 경우, 관리자를 제외한 Client는 계약서 내용을 수정할 수 없도록 한다. PHP에서 계약서 파일을 불러와 쓰기, 실행 권한을 제외한 읽기 권한을 부여하는 코드는 [그림 5]와 같다.

```php
$fileName = $num['txtName'];

$txtName = "$fileName.txt";

$filepath = "/var/www/html/document/$txtName";

chmod($filepath, 0400);
```

[그림 5] PHP를 이용한 계약서의 접근 권한 제어

(2) 시스템 관리자의 계약서 파일 권한 제어

시스템 관리자는 CentOS 운영체제에서 제공하는 파일 속성의 권한 부분에서 계약서 파일의 권한을 관리할 수 있다. 계약서 파일의 권한을 관리하는 화면은 [그림 6]과 같다.

[그림 6] 시스템 관리자의 계약서 파일 권한 제어

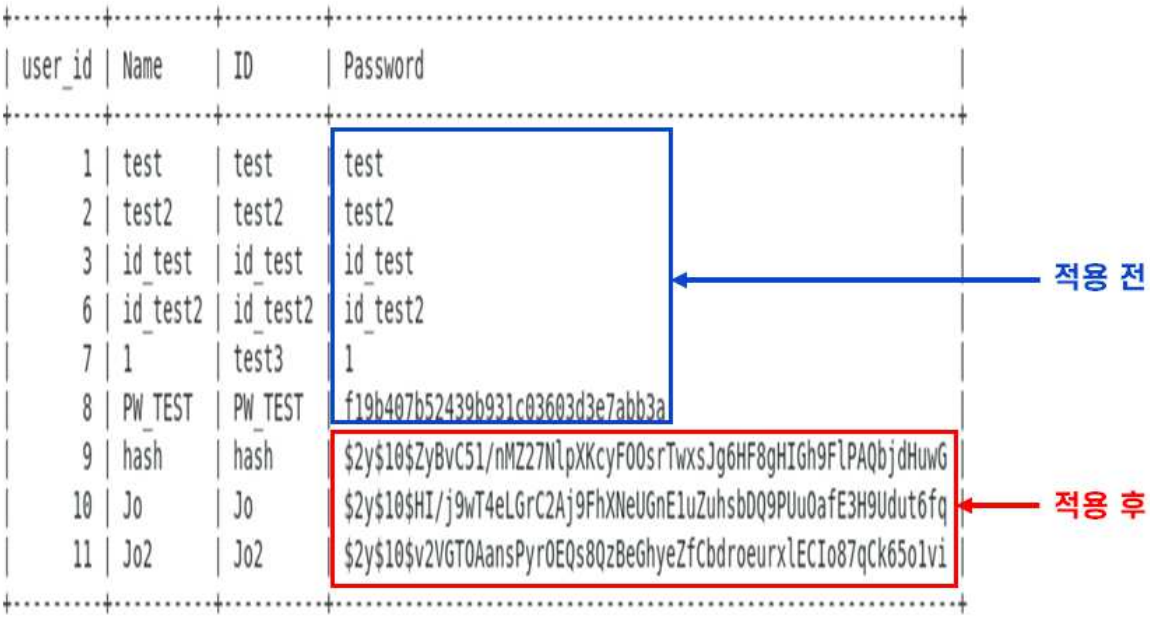
(3) 접근 권한 제어를 통한 계약서 내용의 무결성 제공

전자서명이 완료된 후 Client가 계약서 내용을 수정할 경우, 수정이 불가함을 팝업 창을 통해 안내하여 계약서 내용의 무결성을 제공한다. 전자서명 완료 후 계약서 내용 수정 요청에 관한 결과는 [그림 7]과 같다.

[그림 7] 전자서명 후 계약서의 무결성 제공

라. 해싱과 Salt를 이용한 Password 보안

Client가 회원가입 시 입력하는 Password는 본인의 신원을 증명하기 위한 중요한 요소이기에 적절한 보안을 적용하여야 한다. 따라서 ID, Password를 입력한 후 회원가입을 요청할 경우 해시 함수와 Salt 값을 통해 60bit 길이의 해시값으로 변환하여 DB에 저장하였다. 이때, 해시 함수는 하드웨어 성능에 민감하지 않고 빠르게 해시값으로 변환할 수 있는 SHA-256을 사용하였다. 또한, Salt를 이용하여 16bit 길이의 무작위 값을 추가로 붙여 보안성을 강화하였다. 해싱과 Salt를 통해 해시값으로 변환되어 저장된 Password 정보는 [그림 8]과 같다.



[그림 8] 해시값으로 변환 후 저장된 Password 정보

마. TLS v1.2를 통한 패킷 암호화

해당 시스템에서는 전자서명, 리눅스의 파일 접근 권한 제어를 이용하여 무결성, 인증, 부인 방지 등을 제공하였다. 하지만 Client와 서버 간의 요청과 응답을 하는 과정에서 수많은 패킷이 전달하게 되는데, 보안을 적용하지 않았을 때 어떤 내용을 전달하게 되는지 모두 확인할 수 있다. 이는 데이터의 기밀성을 심각하게 침해하며, Client의 개인 정보와 서버가 구축한 시스템의 구조 등을 파악할 수 있다. 이러한 데이터의 기밀성 문제를 해결하기 위하여 TLS v1.2를 구축하였다.

(1) OpenSSL을 이용한 서버 개인 키 생성

인증서를 생성하기 위하여 서버의 개인 키를 생성한다. 개인 키의 크기는 1024, 2048, 4096 등 관리자가 원하는 크기로 설정하여 생성할 수 있다, 또한, Triple-DES, AES 등 다양한 암호 알고리즘을 관리자가 원하는 암호 알고리즘으로 선택하여 개인 키를 생성할 수 있다. [그림 9]는 생성된 개인 키의 일부 내용이다.

```
[root@localhost tls]# cat /etc/pki/tls/private/example.com.key | head -5
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,B7AB75AE1277204F0D9F59977311CAB4

scTyzgww4PFmMBzVVf/I3obw0oMDNdGEuTTbYO0HPUIxTTk1lWxuUcggRg39SCjv
```

[그림 9] 서버 개인 키 내용 중 일부

(2) 인증 요청서(csr) 파일 생성

인증 요청서 파일을 생성한 후 서버가 생성한 개인 키를 이용하여 암호화한다. 인증 요청서 생성 시 Country Name, State or Province Name, Locality Name, Organization Name, Organizational Unit, Common Name, Email Address로 구성되며, 각 부분에 알맞게 작성하면 정상적으로 인증 요청서 파일이 생성된다. 인증 요청서 파일에 정보를 작성하는 화면은 [그림 10]과 같다.

```
[root@localhost tls]# openssl req -new -key private.key -out ROOTCA.csr
Enter pass phrase for private.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:KR
State or Province Name (full name) []:Busan
Locality Name (eg, city) [Default City]:Centum
Organization Name (eg, company) [Default Company Ltd]:TongMyong
Organizational Unit Name (eg, section) []:Information Security
Common Name (eg, your name or your server's hostname) []:Jaemu
Email Address []:tree4843@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

[그림 10] 인증 요청서 정보 작성

(3) 인증서 생성

OpenSSL 모듈에서 제공하는 옵션을 이용하여 인증서(crt) 파일을 생성한다. 인증서는 X.509 형식을 따르며, 서버에서 생성한 개인 키를 이용하여 인증서에 서명한다. 서명이 정상적으로 완료될 경우, 인증 요청서에 작성한 개인 정보가 인증서의 subject로 추가된다. 서명이 완료된 후 인증서의 일부는 [그림 11]과 같다.

```
[root@localhost tls]# cat /etc/pki/tls/certs/rootCA.crt | head -5
-----BEGIN CERTIFICATE-----
MIIDpjCCAo4CCQDGP5WSUKMhoDANBgkqhkiG9w0BAQsFADCBlDELMAkGA1UEBhMC
S1IxDjAMBgNVBAgMBUJ1c2FuMQ8wDQYDVQQHDAZDZW50dW0xEjAQBgNVBAoMCVRv
bmdteW9uZzEdMBsGA1UECwwUaW5mb3JtYXRpb24gc2VjdXJpdHkxDjAMBgNVBAMM
BWphZW11MSEwHwYJKoZIhvcNAQkBFhJ0cmVlNDg0M0BnbWFpbC5jb20wHhcNMjIx
```

[그림 11] 인증서 내용 중 일부

(4) TLS v1.2 구축

TLS를 구축하기 위한 과정을 거친 후 아파치 서버를 재시작하면 웹에 정상적으로 TLS가 적용된다. 하지만 사이트에 대한 연결이 안전하지 않다고 경고문이 뜨는데, 이는 신뢰하지 않는 기관이 인증서를 생성 및 발급하였기 때문이다. 신뢰할 수 있는 루트 인증 기관은 DigiCert, Certum CA, GlobalSign 등이 있으나 본 시스템에 구축한 인증서는 임의적으로 작성 및 발급되었기에 공격자라고 판단할 수 있다. 하지만 본 논문에서는 데이터의 기밀성 보장을 위한 보안 수단 중 하나로 TLS를 구축하였기 때문에 RSA를 기반으로 패킷을 암호화할 수 있다. 해당 웹 서버에 구축된 인증서 상세 정보는 [그림 12]과 같다.

Certificate

jaemu

Subject Name

Country	KR
State/Province	Busan
Locality	Centum
Organization	Tongmyong
Organizational Unit	information security
Common Name	jaemu
Email Address	tree4843@gmail.com

Issuer Name

Country	KR
State/Province	Busan
Locality	Centum
Organization	Tongmyong
Organizational Unit	information security
Common Name	jaemu
Email Address	tree4843@gmail.com

[그림 12] 인증서 상세 정보 중 일부

2. 설계 및 구현

가. 개발 환경

웹 동작 환경은 인터넷 브라우저를 통해 제공하고 하며 Server는 APM(Apache, PHP, MySQL), Development Tool은 JAVA를 사용하였다. 메인 서버는 리눅스 기반이며 자세한 개발 환경은 <표 2-1>과 같다.

<표 2-1> 웹 개발 환경

구 분	항 목	버 전
Development Tool	Language	JAVA 1.8.352
Server	OS	CentOS Linux release 7.9.2009 (Core)
	MySQL	5.6
	Apache	2.4.6
	PHP	5.4.16

나. 시스템 구현

설계를 바탕으로 웹 시스템의 구현을 서버와 웹으로 나누어 기술한다. 서버는 데이터 저장 및 처리를 위주로 하며, 웹은 기능별 동작 화면과 코드를 위주로 기술한다.

(1) 서버 구현

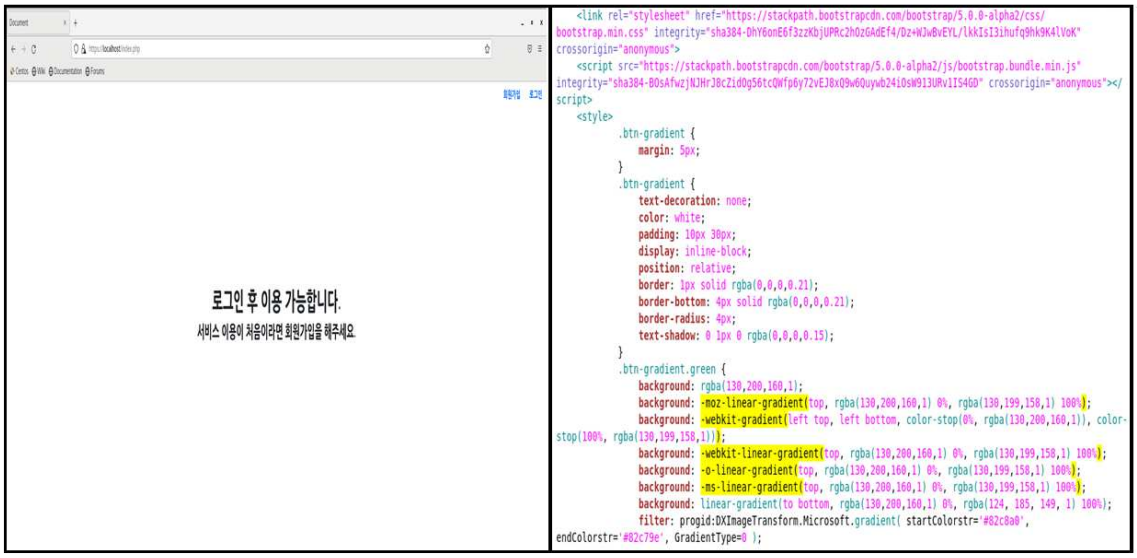
서버는 Client가 웹에서 회원가입과 로그인 요청을 처리한다. 또한, Client가 계약서를 작성할 경우 디렉토리에 저장하며 상황에 따라 Client의 계약서 접근 권한을 수정한다. Client가 계약서에 대한 전자서명을 요청할 경우 KCDSA 알고리즘을 실행하여 정상적으로 전자서명을 처리한다. 계약서 내역 확인 부분에서 계약서의 상세 정보를 확인할 수 있도록 한다.

(2) 웹 구현

전자서명 계약서 웹은 로그인 전 화면, 로그인 화면, 로그인 후 메인 화면, 회원가입 화면, 계약서 작성 화면, 계약서 내역 확인 화면, 저장된 계약서 상세 정보 화면으로 구성된다.

(가) 로그인 전 화면

로그인 전 화면은 해당 웹에 접속하였을 때 최초로 출력되는 화면으로 우측 상단에 회원가입 및 로그인 버튼으로 구성된다. 웹 접속 시 최초로 출력되는 로그인 전 화면과 코드는 [그림 13]과 같다.



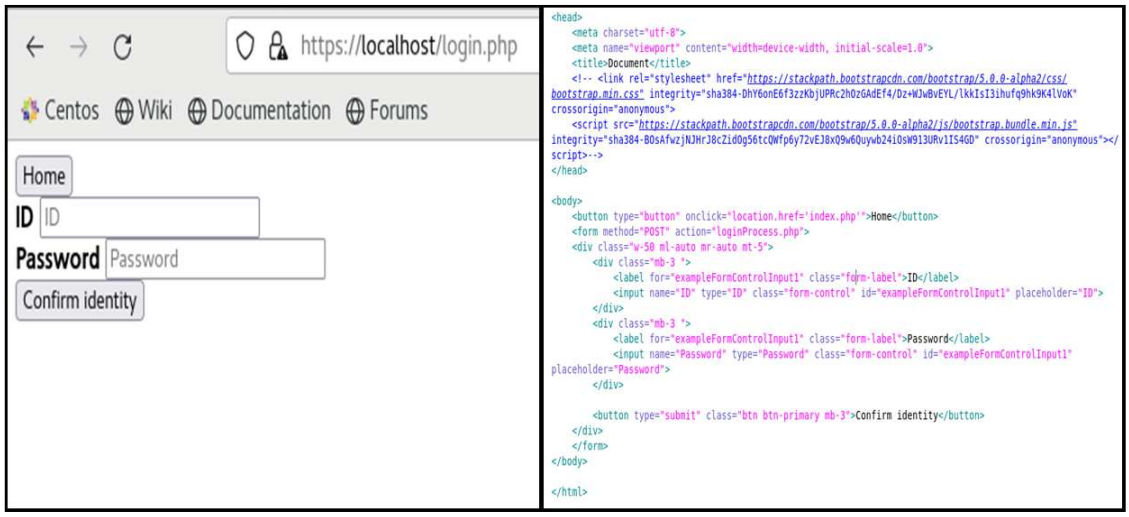
[그림 13] 로그인 전 화면

(나) 로그인 화면

로그인 화면은 최초로 나타나는 로그인 전 화면에서 Client가 로그인 버튼을 클릭하면 출력되는 화면으로 ID와 Password를 입력하는 TextField, Home, Confirm identity 버튼으로 구성된다.

Client가 입력한 ID는 입력값 그대로 화면에 출력되지만, Password는 보안성을 위하여 '•'으로 출력된다.

Client가 로그인 버튼인 Confirm identity를 클릭할 경우 TextField에 입력한 ID와 Password를 서버로 전송하게 되며, Client 정보가 저장된 DB 내에 동일한 값이 존재하면 로그인에 성공하게 된다. 로그인 세션을 생성하여 Client 컴퓨터의 내부 저장소에 저장한다. 로그인 화면과 로그인 기능을 수행하는 코드는 [그림 14]와 같다.



[그림 14] 로그인

(다) 로그인 후 메인 화면

로그인 후 메인 화면은 Client가 로그인 화면에서 ID와 Password 기입 후 Confirm identity 버튼을 클릭하였을 때 입력받은 정보와 DB에 저장된 데이터와 일치할 경우 로그인이 정상적으로 수행되며 출력되는 화면이다. 로그인이 완료되었음을 확인할 수 있도록 우측 상단에 로그인된 Client ID와 로그아웃 버튼이 생성되며, 중앙에는 계약서 작성, 내역 확인 버튼으로 구성된다.

Client가 계약서를 작성을 원할 경우, 좌측의 계약서 작성 버튼인 location1을 클릭하면 Function을 통하여 계약서 작성 화면으로 전환된다.

Client가 저장된 계약서의 내용을 열람하거나 계약서 수정 또는 전자서명 수행을 원할 경우, 좌측의 내역 확인 버튼인 location2를 클릭하면 Function을 통하여 내역 확인 화면으로 전환된다.

또한, 모든 작업을 수행한 후 로그아웃을 원할 경우, 우측 상단의 로그아웃 버튼을 클릭하여 세션을 종료하게 된다. 또한, 팝업 창을 통해 로그아웃이 정상적으로 이루어짐을 확인한 후 로그인 전 화면으로 전환하게 된다. 로그인 후 메인 화면과 계약서 작성, 내역 확인 버튼을 활성화하는 코드는 [그림 15]와 같다.

[그림 15] 로그인 후 메인 화면

(라) 회원가입 화면

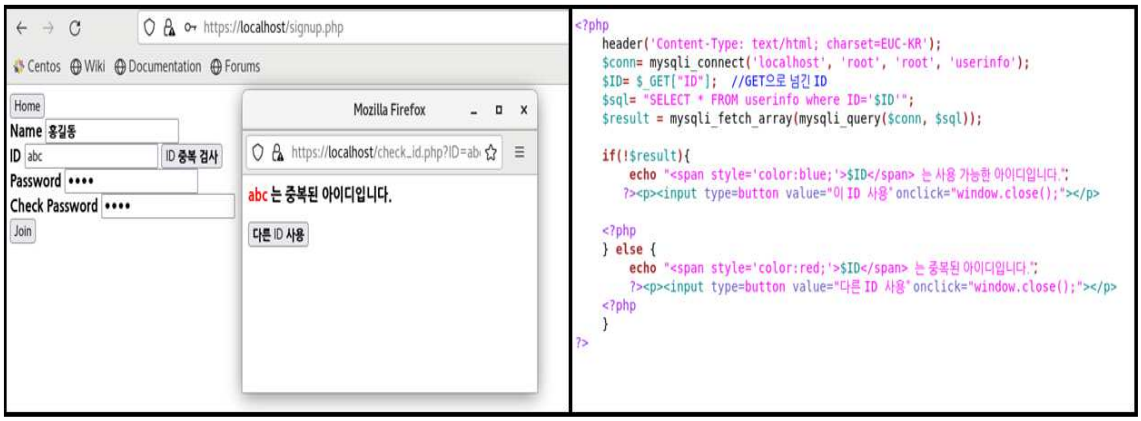
회원가입 화면은 로그인 전 화면에서 회원가입 버튼을 클릭할 경우 출력되는 화면이다. 회원가입 화면임을 나타내는 TextView와 Name, ID, Password, Check Password를 입력하는 TextField와 로그인 전 화면으로 되돌아가는 Home, 기존에 동일한 ID가 존재하는지 확인하는 ID 중복 검사, 회원가입을 완료하는 Join 버튼으로 구성된다.

ID는 입력 후 ID 중복 검사를 완료하여야 회원가입이 가능하다. DB에 저장되어 있지 않은 ID를 입력한다면 사용 가능한 아이디라는 팝업 창이 나타나며, ID 사용 버튼을 클릭 시 ID 등

록이 가능하다.

Password와 Check Password는 로그인 화면과 동일하게 기본값이 '•'이다.

Name, ID, Password, Check Password를 모두 작성한 후 Join 버튼을 클릭하게 되면 입력한 내용이 회원가입이 완료되며 정상적으로 저장된다. 이때, Password는 해시 함수와 Salt 값을 이용하여 길이 60 크기의 해시값으로 암호화되어 안전하게 DB에 저장한다. 회원가입 시 ID 작성 후 중복 ID 검사 버튼을 클릭한 화면과 ID가 중복되었는지 검사하는 코드는 [그림 16]과 같다.



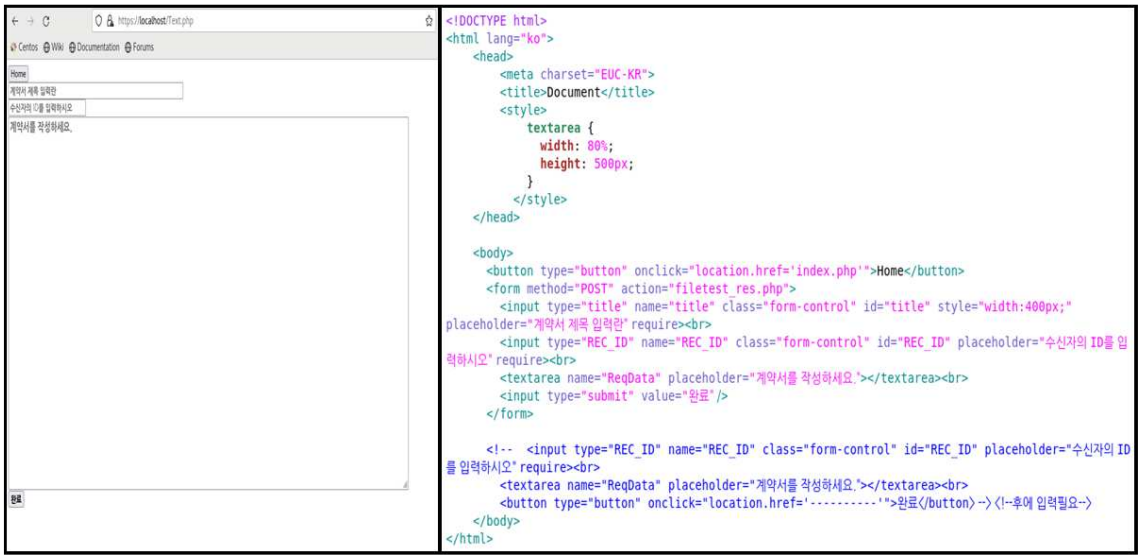
[그림 16] 회원가입

(마) 계약서 작성 화면

계약서 작성 화면은 로그인을 완료한 Client가 로그인 후 메인 화면에서 계약서 작성 버튼을 클릭하였을 때 출력되는 화면으로 메인 화면으로 되돌아가는 Home 버튼과 계약서 제목, 수신자 ID, 계약서 내용을 작성할 수 있는 Text Field와 모든 내용을 작성 후 계약서를 저장할 수 있는 완료 버튼으로 구성된다.

수신자 ID를 입력하는 Text Field에서는 해당 계약서를 확인 및 서명을 수행할 다른 Client의 ID를 기재하여야 한다. 하지만 해당 시스템에서 Client 정보를 저장하고 있는 DB 안에 기입한 ID가 존재하지 않을 경우, 계약서 작성은 완료되지 않는다.

계약서 작성 후 Join 버튼을 클릭하여 계약서를 저장한다. 계약서 저장이 완료되었다는 팝업 창을 출력하여 Client가 정상적으로 계약서가 저장되었음을 확인할 수 있도록 한다. 계약서 파일명은 '계약서 제목.Client1~Client2.txt' 형식으로 저장된다. 저장된 계약서는 서버 내부의 디렉터리에 저장되며 계약서 작성자인 Client1과 계약서에 대한 전자서명을 수행하여야 하는 Client2에게만 계약서 파일의 접근 권한을 부여한다. 또한, 저장된 계약서는 메인 화면의 내역 확인 버튼을 클릭하여 해당 계약서의 상세 정보를 확인할 수 있다. 계약서 작성 화면과 코드는 [그림 17]과 같다.

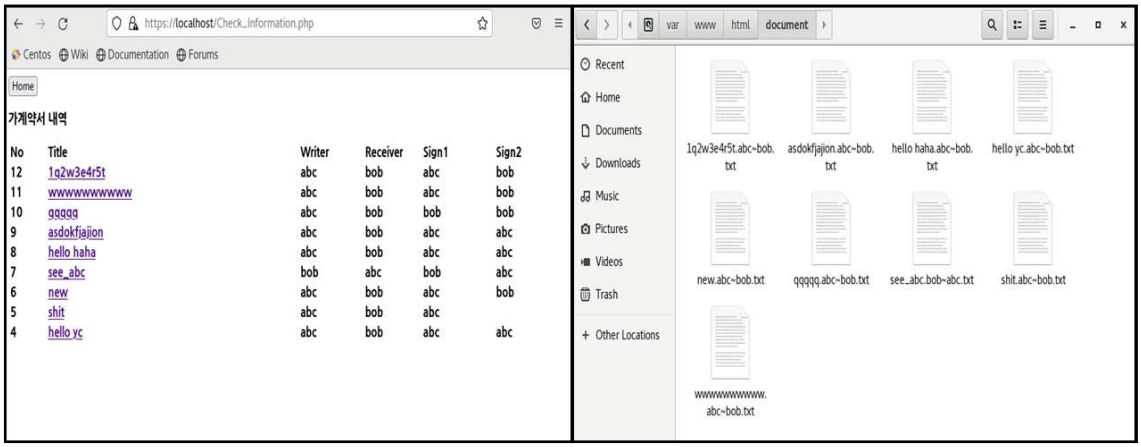


[그림 17] 계약서 작성

(바) 계약서 내역 확인

계약서 내역 확인 화면은 계약서 작성을 완료한 후 로그인 후 메인 화면에서 내역 확인 버튼을 클릭하였을 때 출력되는 화면으로, 해당 계약서의 접근 권한을 부여받은 Client만이 상세 정보를 확인할 수 있다. 계약서 번호에 해당하는 NO, 계약서 제목의 Title, 계약서 작성자인 Writer, 해당 계약서에 함께 참여하게 될 Receiver, 첫 번째로 전자서명을 수행한 Client를 확인할 수 있는 Sign1, 마지막으로 전자서명을 수행한 Client를 확인할 수 있는 Sign2로 구성된다.

계약서 제목을 확인할 수 있는 Title에서 계약서 제목 버튼을 클릭할 경우, 계약서의 세부 내용을 확인 및 수정할 수 있으며 전자서명까지 수행할 수 있다. 계약서 내역 확인 화면과 서버 내부의 디렉터리에 저장된 계약서들은 [그림 18]과 같다.

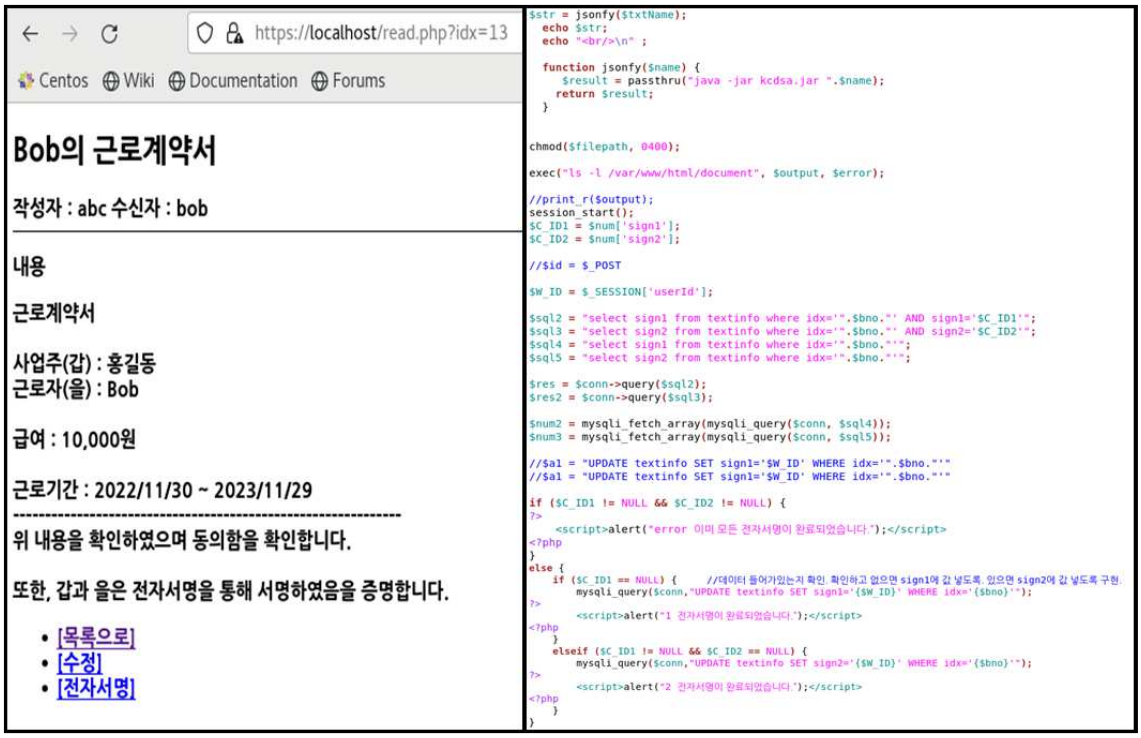


[그림 18] 계약서 내역 확인

(사) 저장된 계약서 상세 정보 화면

저장된 계약서 상세 정보 화면은 계약서 내역 확인에서 Tltie의 계약서 제목 버튼을 클릭하였을 때 출력되는 화면으로, 계약서 제목, 작성자, 수신자, 계약서 내용을 확인할 수 있다. 또한, 계약서 내역 확인 화면으로 전환되는 목록, 계약서 내용을 수정할 수 있는 수정, 계약서를 검토 후 전자서명을 수행할 수 있는 버튼으로 구성된다.

계약서 수정은 Client1과 Client2가 모두 전자서명을 하지 않았을 경우 수행할 수 있으며, 한 명이라도 전자서명을 완료한 경우 계약서를 수정할 수 없도록 한다. 하지만 계약서 내용은 전자서명이 모두 완료된 후에도 열람할 수 있도록 한다. 계약서를 확인 및 수정의 과정을 거친 후 작성자와 수신자 간의 합의가 종료되면 전자서명을 수행한다. 전자서명은 계약서 내용을 KCDSA 알고리즘에 입력 값으로 넣어 계약서 내용의 무결성을 검증하며, Client의 개인키로 암호화된 내용을 Client의 공개 키로 복호화하여 정상적인 Client임을 인증한다. 계약서 상세 정보 화면과 전자서명을 수행하는 코드는 [그림 19]와 같다.



[그림 19] 저장된 계약서 상세 정보

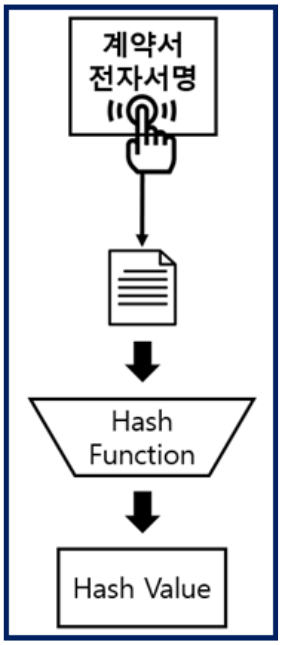
Ⅳ. 분석

1. 전자서명 검증

Client1과 Client2의 계약서 협의를 모두 마친 후 계약서에 대한 전자서명을 수행한다. 이때, 전자서명 버튼을 클릭할 시 어떠한 메커니즘으로 KCDSA 알고리즘이 동작하고 검증 후 최종 완료되는지 구성과 과정으로 나누어 설명한다. 전자서명 검증은 계약서의 해시값 생성, 개인 키를 이용한 해시값 암호화, 복호화 및 검증으로 분류된다.

가. 계약서의 해시값 생성

Client가 계약서에 대한 전자서명을 요청할 경우, 계약서의 내용을 불러와 배열 변수에 삽입한 후 바이트 형태로 변환한다. 변환된 바이트 값을 해시 함수의 입력값으로 넣어 해시값을 생성하게 된다. [그림 20]은 전자서명 클릭 후 해시값을 생성하는 과정이다.

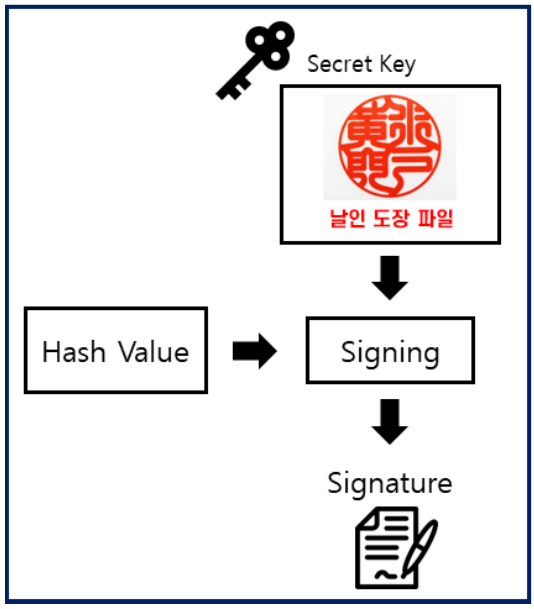


[그림 20] 해시값 생성

나. 개인 키를 이용한 해시값 암호화

해시 함수를 통해 생성된 계약서의 해시값은 Client의 개인 키로 암호화하게 된다. Client의 개인 키는 회원가입 시 등록한 개인 날인 도장 파일을 기반으로 생성한다. 날인 도장 파일은 Client가 직접 생성하여 첨부하게 되며, Main Server는 Client의 날인 파일을 불러와 배열 변수에 삽입한 후 바이트로 변환한다. 변환된 바이트는 시스템 관리자가 임의로 정한 p,

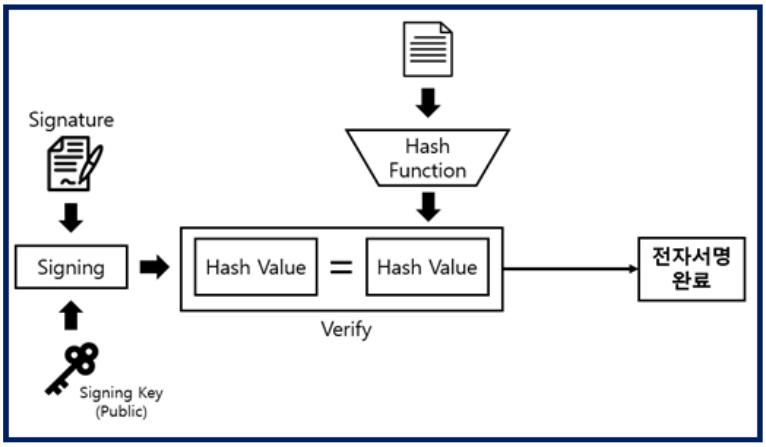
q 값을 기반으로 mod 연산을 거친 후 개인 키와 복호 키가 생성된다. [그림 21]은 계약서의 해시값이 Client의 날인 도장 파일을 기반으로 생성된 개인 키로 암호화하는 과정이다.



[그림 21] 날인 도장을 기반으로 한 암호화

다. 복호화 및 검증

암호화된 Signature 파일은 Main Server로 전달된다. Server는 Client의 복호 키를 이용하여 파일을 복호화한 후 계약서의 해시값과 계약서 평문을 받게 된다. Server는 Client가 보낸 해시값이 일치한 지 검증하기 위하여 계약서 내용을 다시 불러와 바이트 형태로 변환한 후 Client가 사용한 해시 함수를 통해 해시값을 생성한다. Server가 생성한 해시값과 Client가 암호화하여 보낸 해시값의 동일 유무를 판단한 후 일치할 경우 정상적으로 전자서명이 완료된다. 복호화 및 계약서 파일의 해시값 검증 과정은 [그림 22]와 같다.



[그림 22] 복호화 및 해시값 검증

라. KCDSA 전자서명 알고리즘 수행 결과

최종적으로 전자서명이 완료될 경우 Verify 결과를 정수 형태로 반환한다. 정상적으로 전자서명이 완료된 경우 0의 결과가 반환되며, 1~9의 숫자가 반환된 경우 전자서명 과정 중 오류가 발생하였거나 해시값이 일치하지 않다는 것이다. 마지막으로 시스템 관리자는 해당 알고리즘을 수행하며 생성된 28바이트 길이의 개인 키와 해시값 정보를 확인할 수 있다. 알고리즘에서 생성된 개인 키와 해시값 및 검증 결과는 [그림 23]과 같다.

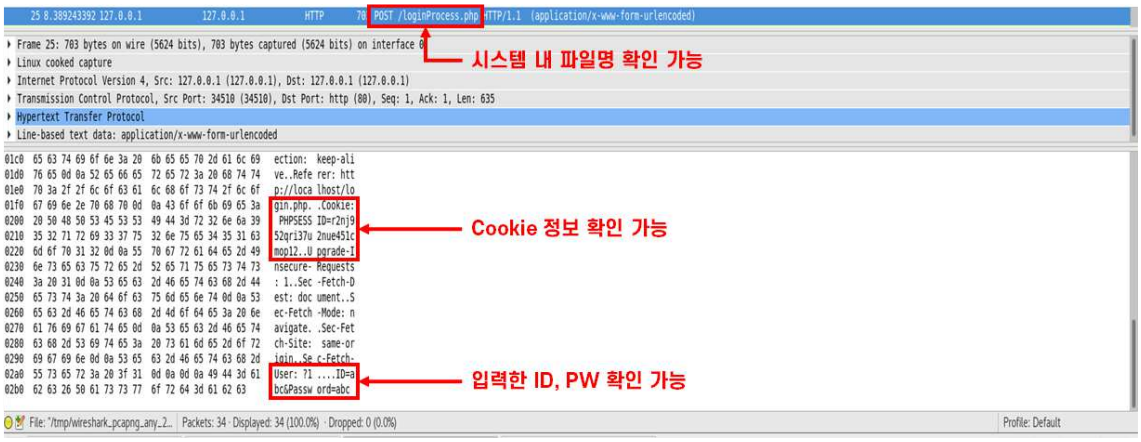
```
ED B7 6A 2D 39 F3 D7 FA 16 D0 82 59 41 18 B0 CF 8B A5 76 92 CF 3B AA EC 6F 6D D9 51
6E E2 B6 FB B5 46 E9 30 1A 22 58 E3 07 82 8F 12 7A 93 DB 87 DE E8 68 DD 76 FA 51 C9
Verify : 0
검증 성공
```

[그림 23] KCDSA 알고리즘 수행 결과

2. TLSv1.2 적용 전후 패킷 비교

가. TLSv1.2 적용 전 패킷

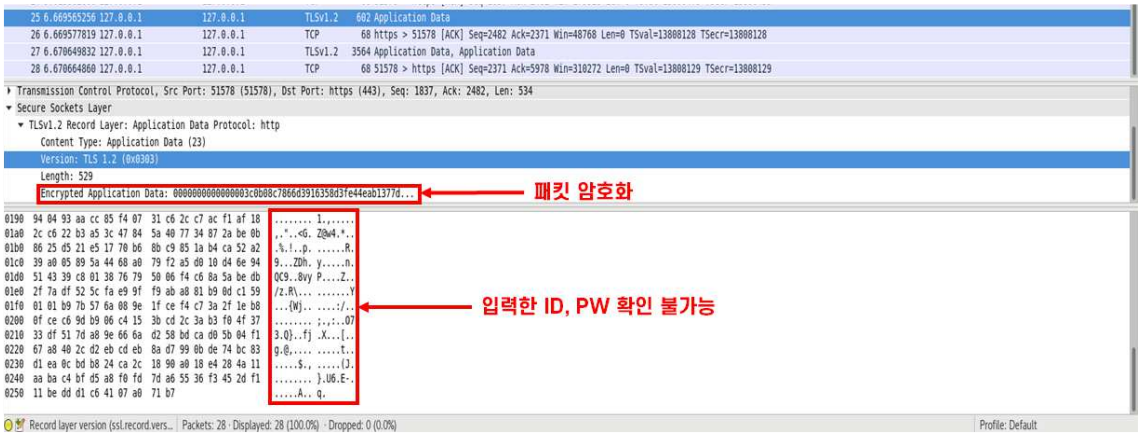
TLS를 적용하기 전에는 전송되는 패킷에 대한 보안이 전혀 적용되지 않기 때문에 Client의 개인 정보 및 계약서의 기밀 정보 등이 유출될 수 있다. TLS 적용 전 Client가 ID, Password를 입력한 후 로그인 요청에 대한 패킷이다. 시스템 내부에서 동작하는 PHP 파일명을 확인할 수 있으며, Client의 쿠키 정보 및 입력한 ID, Password 정보까지 확인할 수 있다. [그림 24]는 TLS 적용 전 Client가 ID, Password를 입력한 후 로그인을 요청한 패킷 정보이다.



[그림 24] TLS 적용 전 전송된 패킷

나. TLSv1.2 적용 후 패킷

TLS가 적용할 경우, Transport Layer에서 Data를 암호화한 후 패킷을 전송하기 때문에 패킷 내부에서 개인 정보 및 기밀 내용을 확인할 수 없다. 이를 통해 기밀성, 데이터의 무결성 및 사용자의 인증 등을 제공한다. [그림 25]는 TLS 적용 후 Client가 ID, Password를 입력한 후 로그인을 요청한 패킷 정보이다.



[그림 25] TLS 적용 후 전송된 패킷

Ⅴ. 결과

본 논문에서는 개인 사업자와 중소기업들을 위하여 온라인 웹상에서 계약서를 작성 및 서명, 보관하는 전자서명 계약서 플랫폼을 제안하였다.

대한민국의 연도별 개인 사업장의 증가와 상가 임대 공실률이 줄어듦을 확인하였으며 자주 사용되는 계약서의 종류, 전자서명의 법적 효력을 살펴보았다. 웹 구축을 위해 APM(Apache, PHP, MySQL)을 사용하였으며, 중복 ID를 확인할 수 있는 회원가입, SHA-256을 통하여 해시값으로 변환된 Password를 검증하는 로그인, 두 Client만을 위한 계약서 작성, 작성된 계약서 확인 및 계약서의 전자서명 유무를 확인할 수 있는 계약서 내역 확인, 계약서의 상세 정보를 확인 및 전자서명을 수행할 수 있는 상세 정보 등을 구현하였다. 또한, 다양한 Client 계정을 생성하여 계약서가 정상적으로 작성되는지 확인하였으며 해당 계약서를 인가된 Client만 열람할 수 있는지 테스트를 통하여 확인하였다.

KCDSA 전자서명 알고리즘을 분석한 후 해당 시스템에 알맞게 수정하여 Client가 정상적으로 계약서에 서명할 수 있도록 하였으며, Database에 칼럼을 추가하여 해당 전자서명 결과를 확인할 수 있도록 하였다. KCDSA 전자서명을 통하여 계약서의 무결성, Client의 인증 및 부인방지를 제공하였고 TLSv1.2를 적용하여 Client와 Main Server 간에 전송되는 패킷을 암호화하여 계약서의 기밀 정보와 Client의 개인 정보를 보호하여 기밀성과 무결성을 보장하였다. 또한, 리눅스 명령어 ‘chmod’를 이용하여 인가된 Client만 계약서를 열람 및 수정할 수 있도록 하였으며, 전자서명 후 계약서 내용이 수정될 수 없도록 접근 권한(읽기)을 제어하였다. 최종적으로 데이터 통신, 서명, Password 해시 변환 후 보관, 접근 권한 제어 등을 적용하여 다양한 보안성을 제공하였다.

추후에는 제안한 시스템을 Application 형태로 구현하여 스마트폰으로도 계약서를 작성, 서명 및 보관할 수 있도록 한다. 또한, Application과 Web을 연동시켜 자유롭게 이용할 수 있는 사용자 친화적인 시스템을 구축할 수 있다. 따라서 다양한 계약서의 법적 효력 문제, 물리적인 종이 계약서 보관 등에서 나타나는 문제점을 해결하며, Web 3.0으로 변화되는 세상에 앞장서는 시스템이 될 수 있도록 한다.

참고문헌

[1] 통계청, “연도별 사업자 현황”
[2] 한국부동산원 부동산통계정보시스템(R-ONE), “상업용 부동산 임대 동향조사 공실률”
[3] 한국정보과학회, 유지현, 임익수, 신용태, “전자문서에 대한 효율적인 장기적 검증 방법”
[4] 법제처 찾기쉬운 생활법령정보, “계약의 성립과 효력” 2022.10.
[5] 전자서명에 관한 법률 제18479호, 일부개정 제3조1항, 제3조2항 2021. 10.
[6] 한국인터넷진흥원, “KCDSA 전자서명 알고리즘에 대한 소스코드 활용 매뉴얼”
[7] 한국정보통신기술협회 “정보통신 용어사전”
[8] 정진희, 조대호, “무선 환경에서 SSL/TLS를 사용하는 IoT의 에너지 효율성 향상을 위한 기법” 2016.6