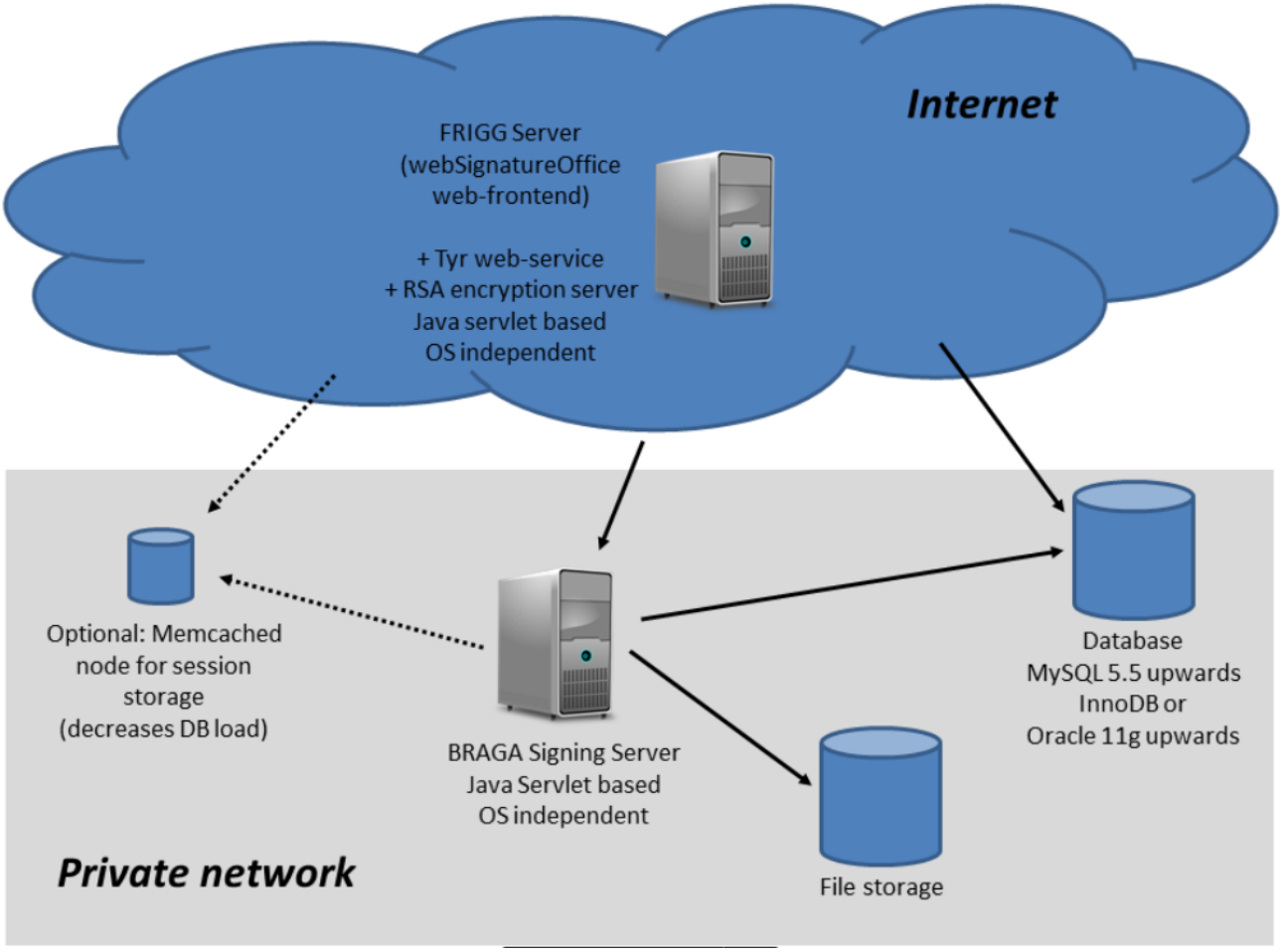
webSignatureOffice security



Introduction

During the development of webSignatureOffice (short webSO) one of the focus points was security. The used Frameworks (Spark framework, Spring framework) have a lot of built in tools to create highly secure web applications-e.g. SafeHtml to sort out XSS weak points.
Also, an additional security layer was added, which will be explained in detail in the following chapters.

Components

- Minimum Browser Requirements: Edge 40, Chrome 49, Firefox 50, Safari 10
  The web application is a rich client (Javascript), which communicates via REST with the server.
- Server: Java Servlets, which can be scaled horizontally.
  All servers share the database and the same file storage, the remaining parts follow the share nothing concet. Due to the split of frontend server and signing server, the signing server can be placed in a highly secured network.
- Minimum Requirements Mobile Devices: iOS Version 12, Android Version 7.

Datatransfer

Browser <-> Server

The data transfer between the server and the browser is based on a REST API (Spark framework). Every request is signed with a key (hash value) to make “Request Forging” more difficult. Therefore the Request XML + Key are used to create an SHA1 hash, which will be checked on the server. The key for the viewer (component for integration into customer’s website) is saved obfuscated in the JavaScript code. It will be reset every time the user logs in to webSO. The sessions are managed by cookies.

Every request also has a Nonce (a number which can be used only once) to prevent attacks via „Request Replays“. Additionally a timestamp is sent with every request to nullify requests, which maybe intercepted after a given amount of time.

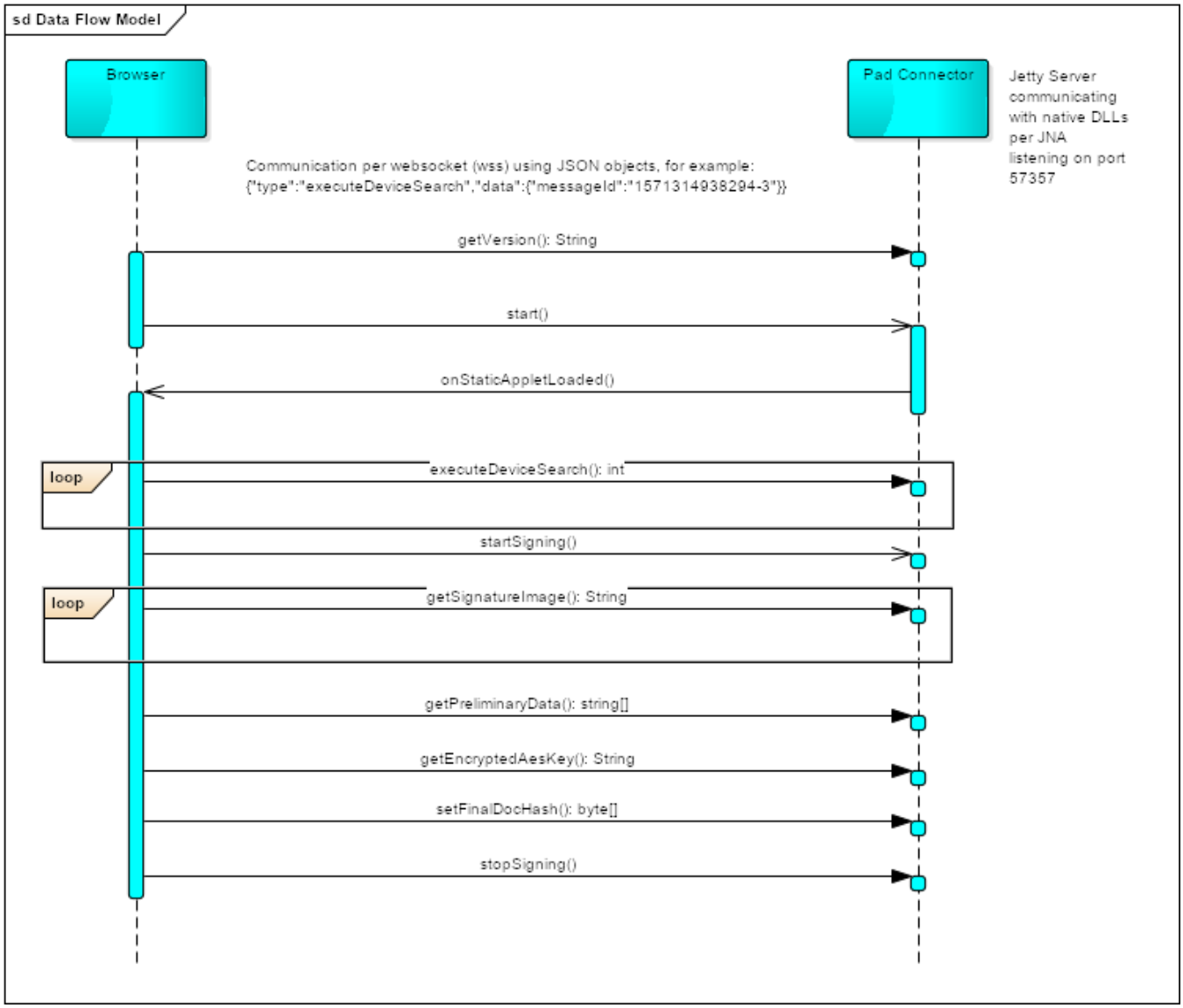
In order to sign with biometric data, two requests which build on one another are needed. The data transfer is SSL encrypted and the JavaScript Code is also obfuscated.

Pad Connector <-> Browser

The communication between Browser and Pad Connector is done by using secure web sockets (wss). The SSL certificate and the keystore are stored locally in the installation directory and are used to overcome the browsers mixed content barrier. It has no security relevance as the data does not leave the local realm of the users computer. The SSL certificate is issued for a host name that has a localhost DNS adress (signsocket.stepover.com). The Pad Connector web socket only excepts connections from localhost.

The Pad Connector is a Java applet, which communicates with the signature pads via USB HID Library per JNA.

All biometric data is transfered encrypted via a Jetty based server. The encryption is happening on the StepOver signature pad (hybride AES256Bit /RSA2048Bit / https://en.wikipedia.org/wiki/Hybrid_cryptosystem)



Browser <-> RSA Encryption Server

After signing a field with the HTML5-Signer, the signature data has to be encrypted by RSA cryptosystem. So the Browser sends data to the RSA Encryption Server (part of the backend system infrastructure) via HTTPS. A serverside Java-Servlet encrypts the given data before sending them back to the Browser.

Mobile Devices <-> Server

The communication between the mobile devices and the server is SSL encrypted. The used protocol is standard XML RPC (http://en.wikipedia.org/wiki/XML-RPC). The security mechanisms are comparable to the ones between browser and server. Every request is signed with a key (the key is a SHA1 hash of a random value). The request includes a Nonce, timestamp and session ID. The key is saved obfuscated in the App.

The Random Key, which is needed for the encryption of the biometric data, is stored in the keychain for iOS devices (secure Memory, which can only be read by the App). Android devices encrypt it with AES and save it in a place, where only the App can read it. All biometric data is transferred encrypted and is done in the mobile device (Tablet or phone, hybrid AES256Bit/RSA2048Bit encryption / http://en.wikipedia.org/wiki/Hybrid_cryptosystem).

Both Apps are digitally signed to prevent later manipulation.

Server <-> Server

The communication between the servers is also done by the same Interface, which is used by the mobile devices (Tyr-Service). They key, which is used to sign the requests, is safely stored on each server. While using the Tyr-Service also an additional Application Key is used to activate or deactivate certain features.

General

Passwords

Passwords are generally not saved, instead a password hash (salted) is saved. By using this method, no passwords can be found, if an attack should be successful.

Authorizer

All service calls traverse an Authorizer, which checks, if the given action is allowed for the user. Not only roles are checked with this method, but also the provided data.

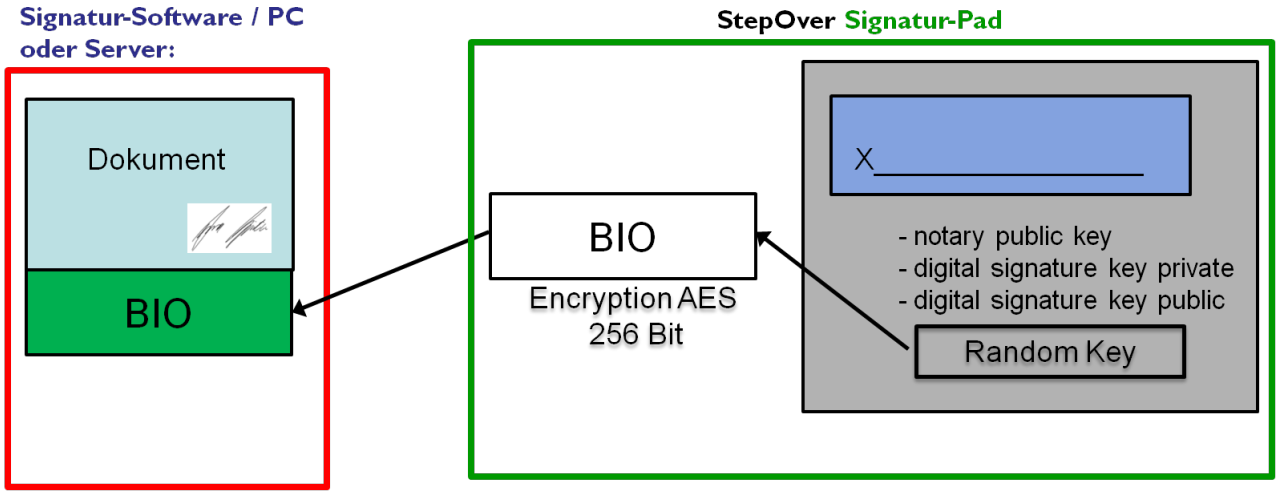
Applying the Signature

The signature is applied to the document on the sign server. All biometric data is sent to the server encrypted.

The certificates for the digital signatures are managed in encrypted PKCS12 containers on the server. The server itself does not save any passwords, which means the use of the certificates is secured by a renewed input of the password.

Encryption, transmission and embedding of the biometric data

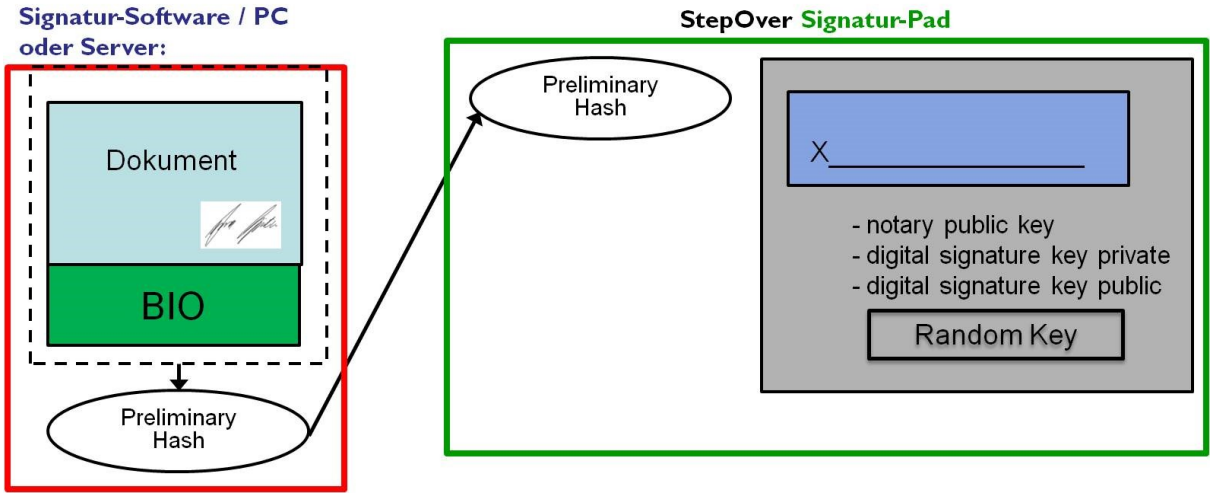
The biometric data of the signature is stored in the Signature-Pad or iOS/Android App (only for the time of the signature). It is send finally encrypted (encrypted and linked to the document) to the signature server. As key for the encryption, a value is used, which is specifically generated for each signature. The so called “Random-Key”. This “Random-Key” is the password for the encryption of the biometric data (AES256), which then is send to the signature server. The encrypted biometric data is embedded (invisible) into the document.



At this point in time the encrypted biometric data has no link to the document. Without the “Random-Key”, which was used for the encryption, the biometric data is inaccessible. This key only exists during the encryption process in the processor of the signature pad or the secure memory of the App.

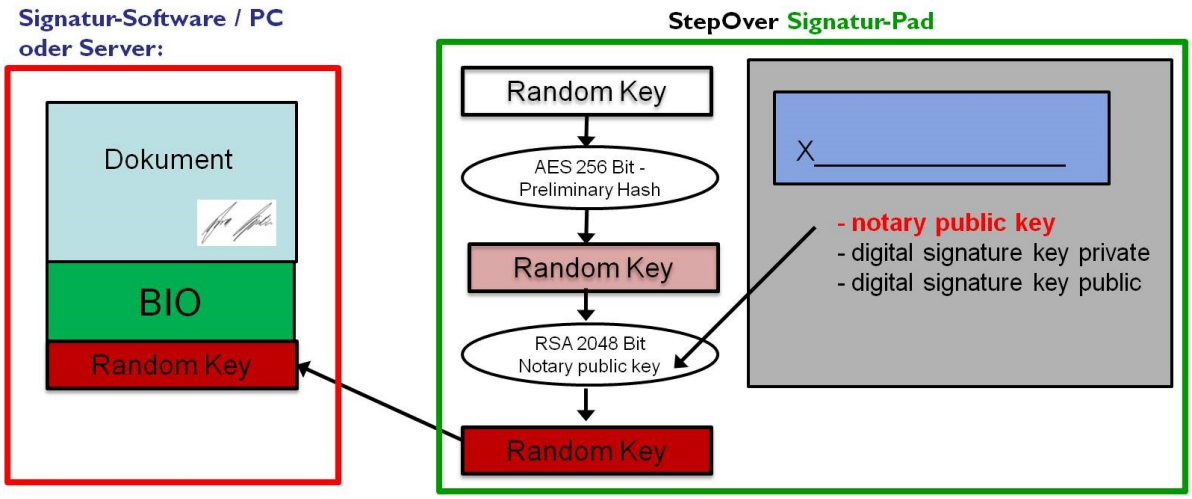
Connection to the document and RSA encryption

The document to be signed now stores the image of the signature and the (AES256) encrypted biometric data of the signature. At this point the signature server creates a checksum (SHA-256) of the document incl. image of the signature and encrypted biometric data, the so called “Preliminary Hash” and sends it to the Signature-Pad or the APP.



The “Random Key” will be encrypted with the “Preliminary Hash” inside the Signature-Pad or App (AES256). Doing this, an irrevocable connection between the document and the biometric data is created.

Afterwards the already encrypted “Random Key” is encrypted a second time with the “Notary Public Key” using RSA 2048. The Notary Public Key is a 2048 Bit RSA Public Key, which is created by an trusted authorization, e.g. a notary. The corresponding “Private Key”, which is needed for decryption, can only be accessed by the trusted authority).



The double encrypted “Random Key” is transferred from the Pad/App to the signature server, where it is embedded into the PDF. The biometric signature is finished now.

Additionally the whole document is digitally signed with a second key pair named “Digital Signature Key Pair“. The second signature can be used for in-house verification of the document integrity.

Conclusion of the process and advantages of the procedure:

- Unencrypted biometric data is never transferred, nor in the memory of a computer or server provider at any point in time (which could be attacked by hackers or special employees of the provider).
- Also no unencrypted biometric data can be found on the server, which could be used for signing different documents.
- You can only decrypt the biometric data with the “Notary Private Key”. But it is stored at a trusted authority (e.g. notary). This means the operator can proof that he has no way to access the data.
- If the document is changed, the biometric data is lost. Because the creation of the value, which is needed to decrypt the random key, is impossible without the correct “Preliminary Hash”. This can be used to proof the connection between a document and a signature.
- For the susceptibility of proof ONLY the notary key pair is used. A notary is a trusted authority for courts.
- The susceptibility of proof is completely independent to the second key pair („Digital Signature Key Pair“). This means, the susceptibility of proof is also independent to the operator of the signature solution (and the trustworthiness of its employees).