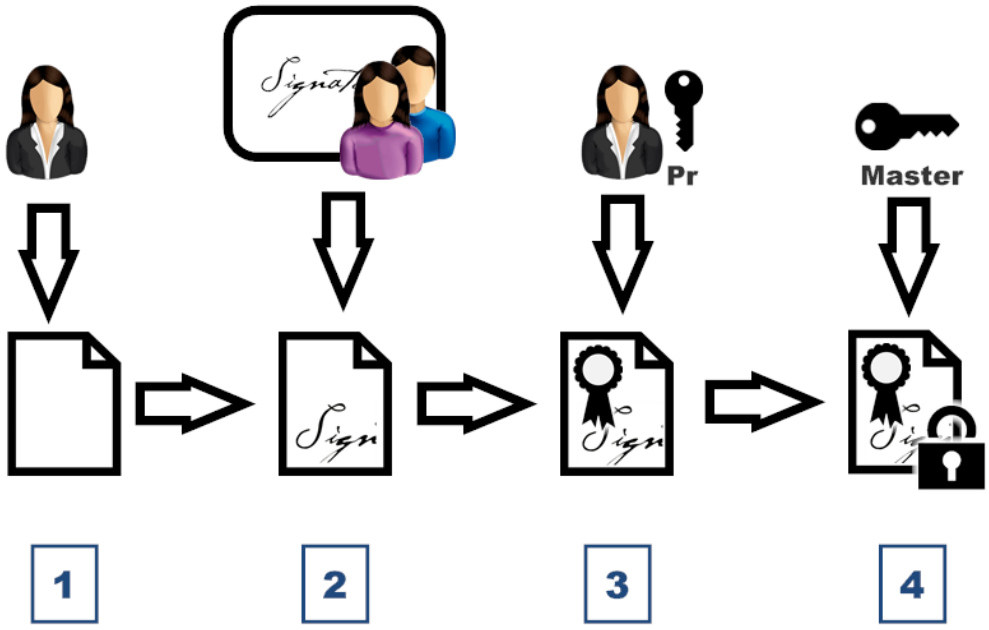
BTOTS Web Signatures Overview

This document provides an overview of how the BTOTS Web system captures digitized signatures, ensures authenticity, securely stores the signed documents, and manages encryption keys.

Signing Process Overview

The following diagram provides an overview of various steps in the signing process. Subsequent sections explore individual steps and supporting requirements in detail.



Step 1: Enter the PDF Data

The user enters the needed information into the BTOTS Web system. At this point a PDF of the information can be viewed.

Step 2: Capture Digitized Signatures

Once the data has been entered the parents and/or user can physically sign online. The digitized signature is added to the PDF. Later, this signature is verified before displaying the PDF to authorized users.

Step 3: Digitally Sign the PDF File

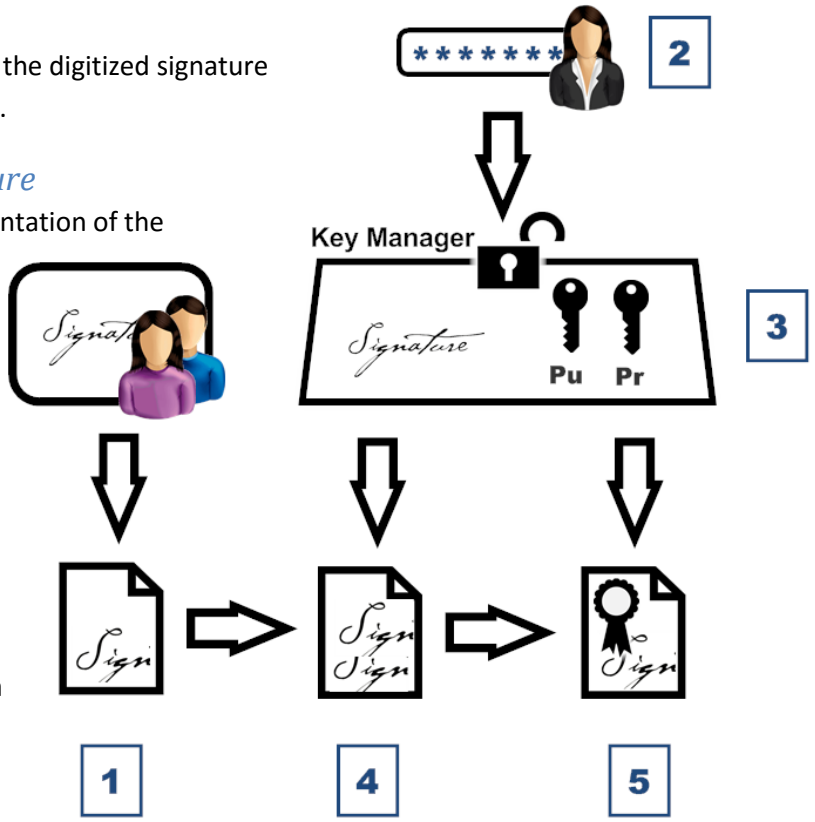
The PDF including the digitized signature is next digitally signed with the user's private RSA key to ensure nonrepudiation, authentication, and file integrity.

Step 4: Encrypt the PDF File

The system then encrypts the PDF which contains the digitized and digital signature using a symmetric 256 bit Rijndael encryption algorithm with CBC (cipher-block-chaining).

Digitized and Digital Signatures

This section provides a more detailed description of the digitized signature capture and the digital signing of the PDF document.



Step 1: Parent (Non-user) Digitized Signature

Digitized signatures (electronically captured representation of the individual's written signature) are to be used for authentication for non-users of the system.

It is unrealistic to expect the parents/guardians of children in the system to remember username and password combinations between signings. The effort on the part of the parent/guardian should be minimal.

These digitized signatures are to be captured via an inexpensive USD drawing tablet (as opposed to a more expensive signature tablet). The system uses a combination of HTML 5 and Javascript to capture the signature. This works well on all modern browsers as well as tablets such as the Apple iPad.

The digitized signature of non-users is stored in a web session (secure web server memory) until the signature is able to be generated in the ultimate encrypted version of the PDF. In this way the digitized signature is only physically stored in an encrypted state.

Steps 2 & 3: Key Manager (User password, digitized signature, and RSA keys)

Users of the system (providers with an account login and password), are required to provide their password each time the signature process is completed. The user's password is needed for gaining access to the user's digitized signature (to add to the PDF file) as well as their private RSA key (for digitally signing the PDF). Both pieces of information are encrypted in the database using a symmetrical encryption key derived from the user's password based on the PBKDF2 algorithm (5000 iterations using SHA256 hash). Using the user's password ensures that the given private RSA key and digitized signature is only usable by the user themselves (password is only known by user).

Step 4: Adding the User's Digitized Signature

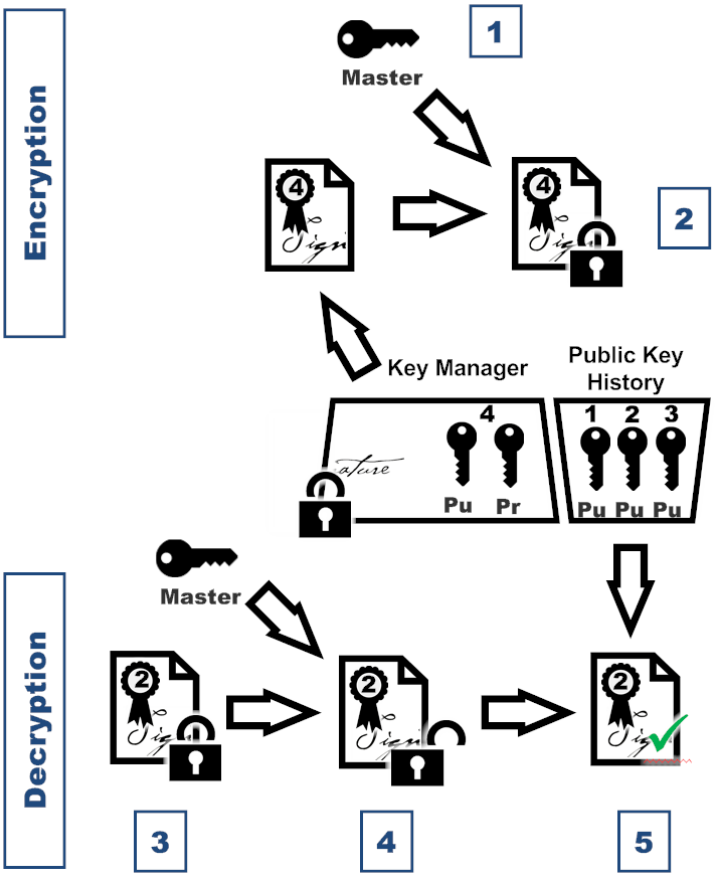
The system user signatures will be encrypted and stored for later use. This allows the user of the system to simply enter their password during the signing process rather than having to enter their password and sign as well. If no signature has been stored for a given user, they will be prompted upon their first signing process and it will be stored encrypted for future use.

Step 5: Adding the Digital Signature

Once all the signatures have been added to the document, the pdf is digitally signed by taking a hash of the document and signing the hash using the user's private RSA key. As mentioned previously, the user's private RSA key is stored in encrypted form in the database based on the user's password. All of the user's public RSA keys are retained in plain text so that the digital signature of the pdf can be used to verify that the document hasn't been tampered with. In addition, the system is able to verify existing signatures before redisplaying the document for the user.

PDF File Encryption & Decryption

This section describes the process of encrypting and decrypting the PDF file.



Steps 1 & 2: Encrypting the PDF File

The PDF file with the digitized and digital signature is encrypted using a symmetric 256 bit Rijndael encryption algorithm with CBC (cipher-block-chaining) with a 256 bit Master Key. The PDF file is only saved to disk in encrypted form (PDF generation and encryption occurs in memory without using temp files).

Steps 3 & 4: Unencrypting the PDF File

When a user requests to view or print the PDF file that was previously encrypted and saved, the Master Key is first used to unencrypt the PDF file in memory. In the example provided, the PDF file was digitally signed with a previous public/private RSA key pair (version #2).

Step 5: Verifying the Digital Signature

The system maintains a Public Key History in plain text so that any previous public keys for the user can still be used for verification purposes. During Step 5, the appropriate public key is loaded and used to verify that the in memory PDF file hasn't been altered since the time of the signing. If verification fails, an invalid signature error is displayed to the user instead of the PDF file.

Additional Points

The following additional discussion points have been included as they are relevant to this document.

Master Key Management

The Master Key used for PDF file encryption is stored in an encrypted configuration file on the webserver. The Microsoft protected configuration mechanism is used to securely encrypt the configuration file (see http://msdn.microsoft.com/en-us/library/53tyfkaw.aspx for more details).

Key Manager and User Password Changes/Resets

Due to the system maintaining the user's public/private RSA key pair in an encrypted state based on the user's password, it is possible for access to the public/private RSA key to be lost (e.g., the user forgets their password and needs it to be reset). This limitation is overcome by (1) always keeping a plain text version of the user's public RSA key in a Public Key History and (2) generating a new public/private RSA key pair whenever the user resets their password.