A real-life project in End-To-End encryption

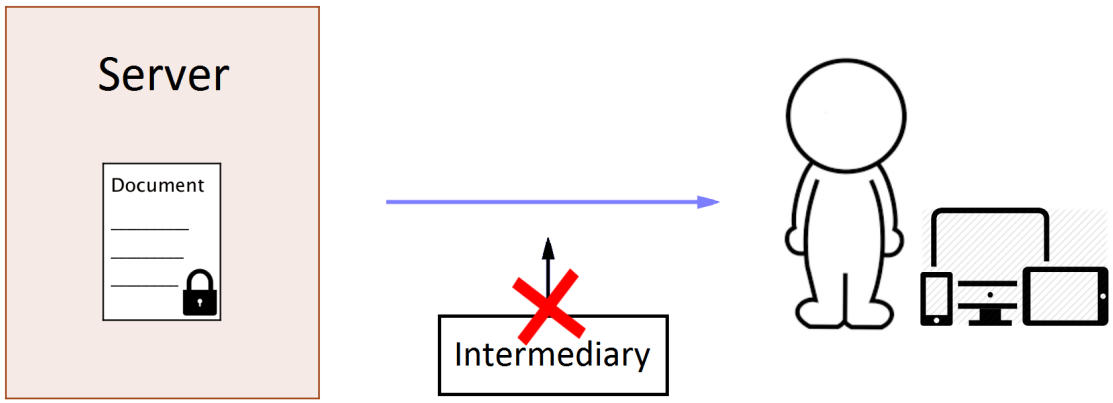
R. Aragona, S. Manni, C. Mascia,
V. Mauri, A. Ottaviano Quintavalle

Laboratorio di Matematica Industriale e Crittografia
Dipartimento di Matematica
Università di Trento

17th December 2015

Introduction

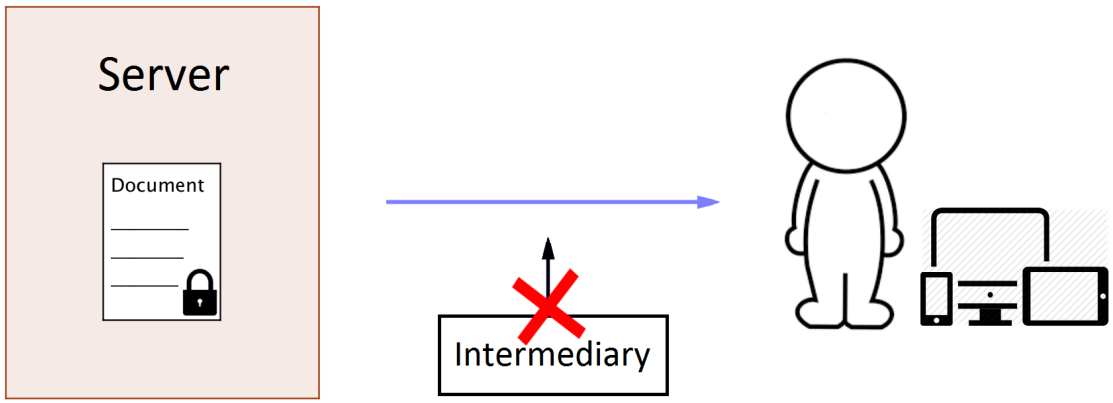
- **A real-life problem:** To send classified documents, in view-only mode, from a server to an authorized device, guaranteeing that nobody can access to the document during transmission.



- **Solution:** Construction of an End-To-End encryption system.

Introduction

- **A real-life problem:** To send classified documents, in view-only mode, from a server to an authorized device, guaranteeing that nobody can access to the document during transmission.



- **Solution:** Construction of an End-To-End encryption system.

End-To-End encryption (EE2E)

What is the End-To-End encryption (EE2E)?

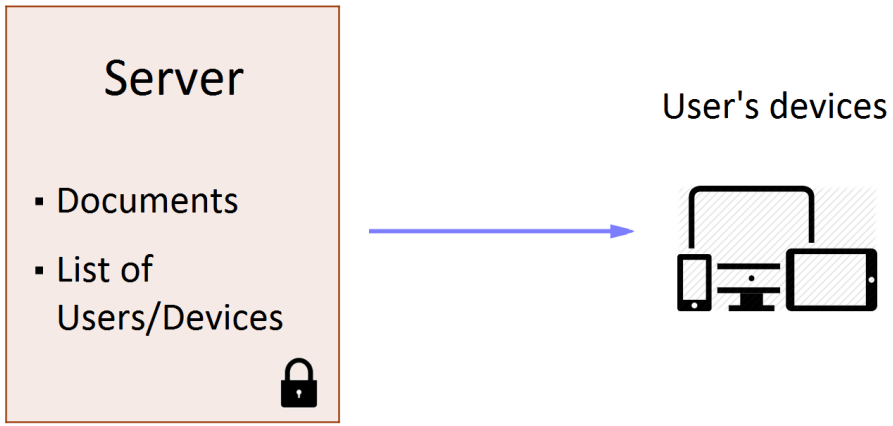
E2EE is a method of **secure communication** that prevents third-parties from accessing data during transmission.

The data is encrypted on the sender's system or device and only the recipient is able to decrypt it. **Nobody** in between, including Internet service provider or, worse, an attacker, can read it or tamper with it.

In general, E2EE uses a **combination** of symmetric and public-key cryptography.

Actors

1) SERVER

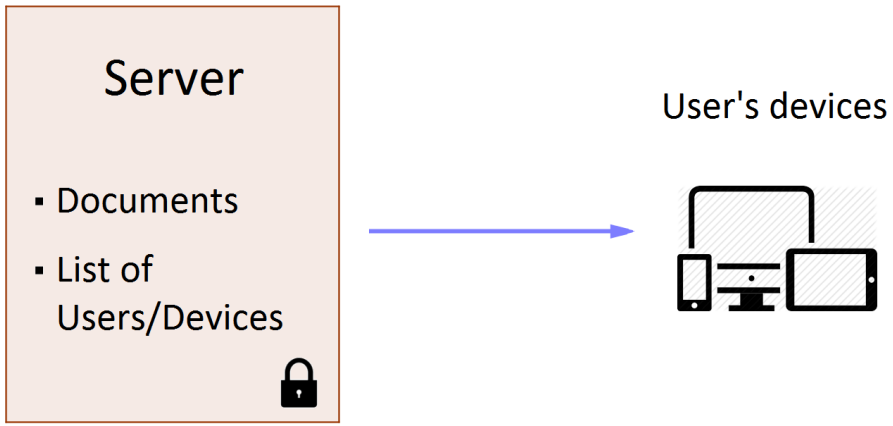


Main activities:

- To encrypt classified documents and storage them.
- To create and manage encryption keys.
- To manage a list of authorized users and devices, called user book.
- To send encrypted documents, in view-only mode, to authorized devices though a secure channel.

Actors

1) SERVER

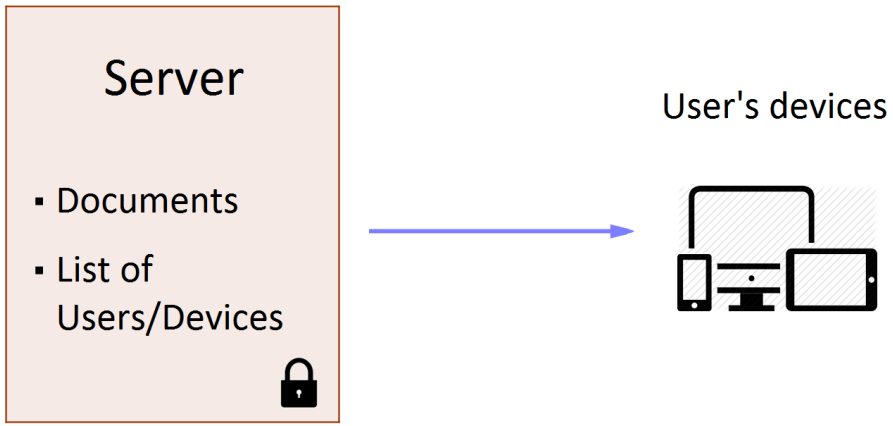


Main activities:

- To encrypt classified documents and storage them.
- To create and manage encryption keys.
- To manage a list of authorized users and devices, called user book.
- To send encrypted documents, in view-only mode, to authorized devices though a secure channel.

Actors

1) SERVER

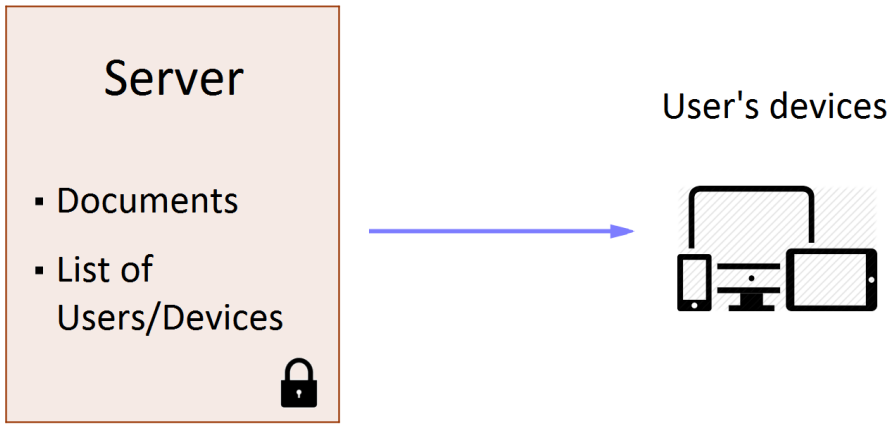


Main activities:

- To encrypt classified documents and storage them.
- To create and manage encryption keys.
- To manage a list of authorized users and devices, called user book.
- To send encrypted documents, in view-only mode, to authorized devices though a secure channel.

Actors

1) SERVER

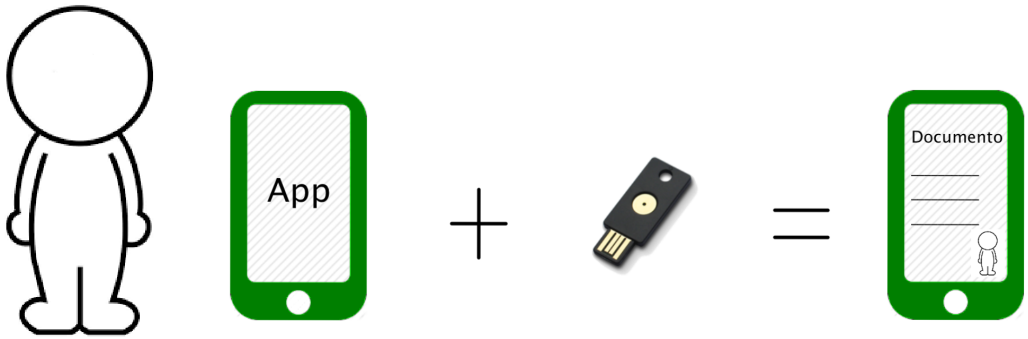


Main activities:

- To encrypt classified documents and storage them.
- To create and manage encryption keys.
- To manage a list of authorized users and devices, called user book.
- To send encrypted documents, in view-only mode, to authorized devices though a secure channel.

Actors

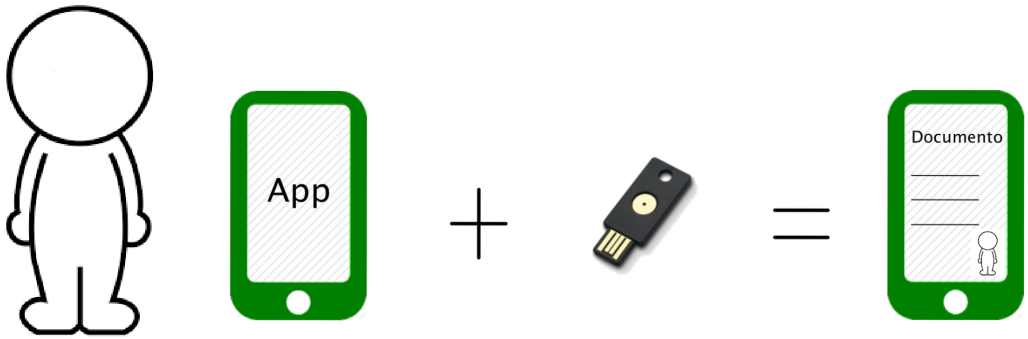
2) USER



- owns authorized devices on which he can install a released application needed for receiving documents
- owns a dongle, a small piece of hardware which is needed for the strong authentication to access to the application, and then to visualize the documents.

Actors

2) USER



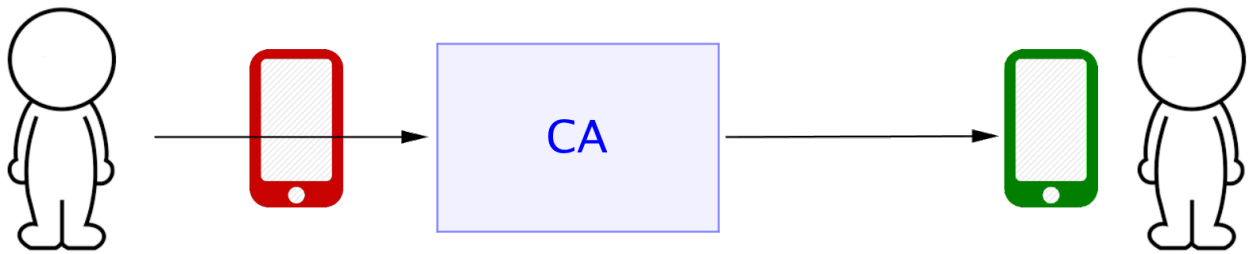
- owns authorized devices on which he can install a released application needed for receiving documents
- owns a dongle, a small piece of hardware which is needed for the strong authentication to access to the application, and then to visualize the documents.

Setup: device authentication

1) The user requests to a Certification Authority (CA) the authorization for one of his devices:
   - The CA installs a certificate on the user's device and publishes an updated list of issued certificates.

Setup: device authentication

1) The user requests to a Certification Authority (CA) the authorization for one of his devices:
   - The CA installs a certificate on the user's device and publishes an updated list of issued certificates.



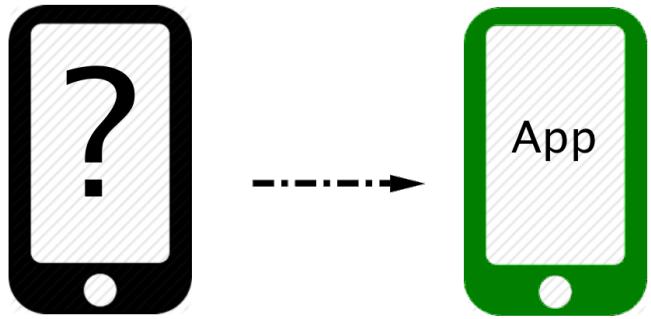
Setup: update user book

2) The server updates the list of authorized users' devices, adding the public key of each device.

USER	**DEVICE**	...
User_1	PBK dev_1^1 PBK dev_1^2	
User_2	PBK dev_2^1	
User_3	PBK dev_3^1 PBK dev_3^2 PBK dev_3^3	
...	...	...

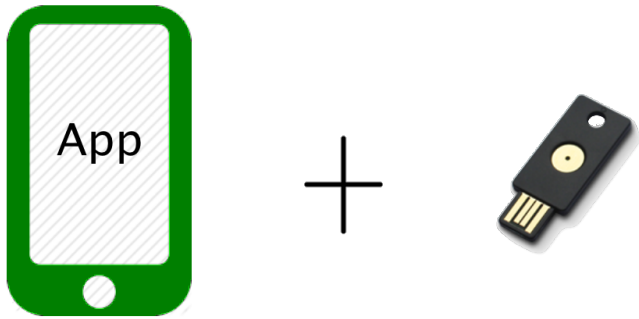
Setup: application

3) The user gets the application that:
   - during the installation checks the validity of the certificate
   - requests the strong authentication to the user
   - creates a pair of keys related to the application itself
   - memorizes on the device the private key of the application encrypted using the credentials of the strong authentication
   - sends to the server the public key signed using the private key of the device.



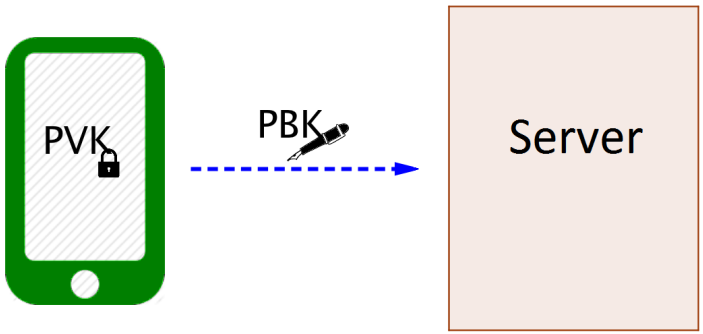
Setup: application

3) The user gets the application that:
   - during the installation checks the validity of the certificate
   - requests the strong authentication to the user
   - creates a pair of keys related to the application itself
   - memorizes on the device the private key of the application encrypted using the credentials of the strong authentication
   - sends to the server the public key signed using the private key of the device.



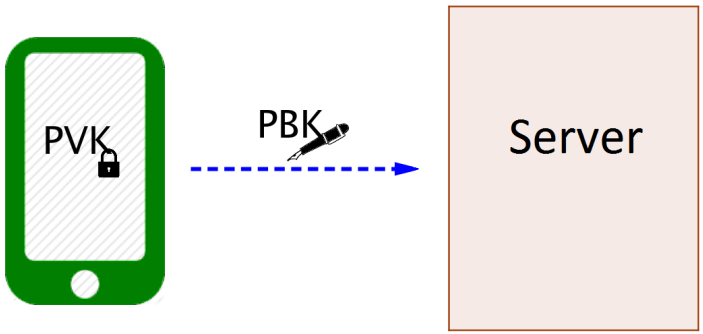
Setup: application

3) The user gets the application that:
   - during the installation checks the validity of the certificate
   - requests the strong authentication to the user
   - creates a pair of keys related to the application itself
   - memorizes on the device the private key of the application encrypted using the credentials of the strong authentication
   - sends to the server the public key signed using the private key of the device.



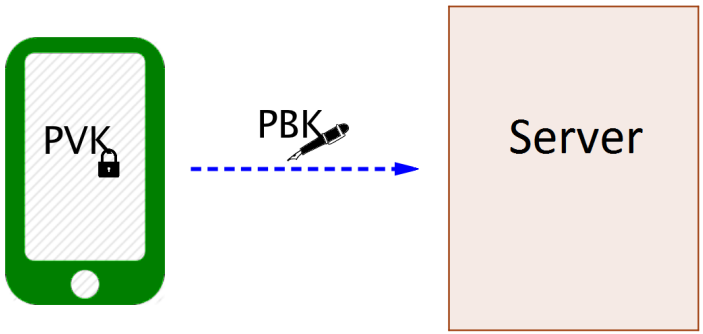
Setup: application

3) The user gets the application that:
   - during the installation checks the validity of the certificate
   - requests the strong authentication to the user
   - creates a pair of keys related to the application itself
   - memorizes on the device the private key of the application encrypted using the credentials of the strong authentication
   - sends to the server the public key signed using the private key of the device.



Setup: application

3) The user gets the application that:
   - during the installation checks the validity of the certificate
   - requests the strong authentication to the user
   - creates a pair of keys related to the application itself
   - memorizes on the device the private key of the application encrypted using the credentials of the strong authentication
   - sends to the server the public key signed using the private key of the device.



Setup: update user book

4) After having got the public key of the application, the server:
   - verifies the signature using the public key of the device
   - updates the user book, adding the public key of the application

USER	DEVICE	APPLICATION
$User_1$	PBK dev_1^1 PBK dev_1^2	PBK app_1^1 PBK app_1^2
$User_2$	PBK dev_2^1	PBK app_2^1
$User_3$	PBK dev_3^1 PBK dev_3^2 PBK dev_3^3	PBK app_3^1 PBK app_3^2 PBK app_3^2
...	...	...

Setup: update user book

4) After having got the public key of the application, the server:
   - verifies the signature using the public key of the device
   - updates the user book, adding the public key of the application

USER	DEVICE	APPLICATION
$User_1$	PBK dev_1^1 PBK dev_1^2	PBK app_1^1 PBK app_1^2
$User_2$	PBK dev_2^1	PBK app_2^1
$User_3$	PBK dev_3^1 PBK dev_3^2 PBK dev_3^3	PBK app_3^1 PBK app_3^2 PBK app_3^2
...	...	...

Setup: update user book

4) After having got the public key of the application, the server:
   - verifies the signature using the public key of the device
   - updates the user book, adding the public key of the application

USER	DEVICE	APPLICATION
$User_1$	PBK dev_1^1 PBK dev_1^2	PBK app_1^1 PBK app_1^2
$User_2$	PBK dev_2^1	PBK app_2^1
$User_3$	PBK dev_3^1 PBK dev_3^2 PBK dev_3^3	PBK app_3^1 PBK app_3^2 PBK app_3^2
...	...	...

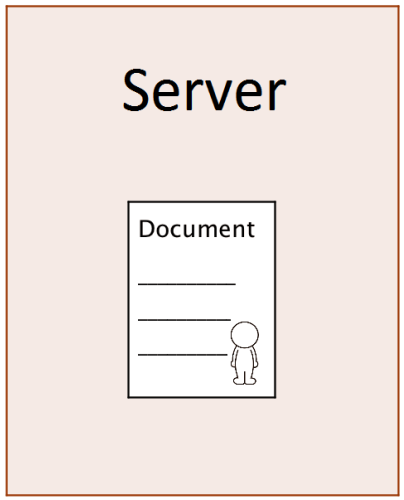
Setup: certificate and public key pinning

The public key of the server is necessary to verify the signature of the messages sent by the server.

It is added to the application at development time.

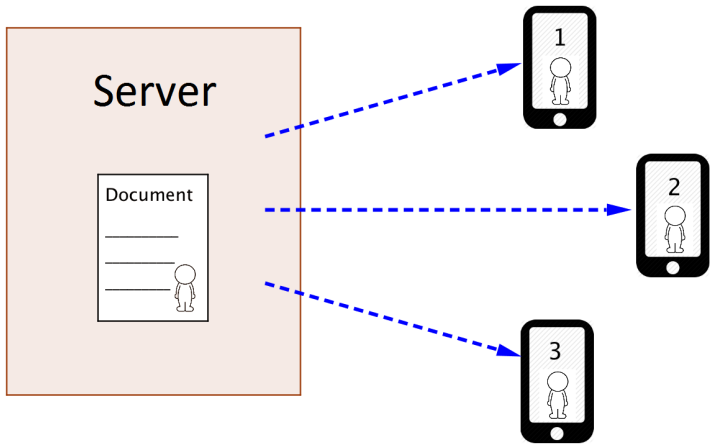
Transmission

1) The server would send a document D to an user, then it:
   - creates a customized copy in view-only mode for the user
   - tries to establish a TLS connection with each user's device
   - creates a session key k to encrypt D
   - encrypts k using each public key of the user's applications
   - signs $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ with own private key and sends it
   - deletes k and the copy of D in a safe way.



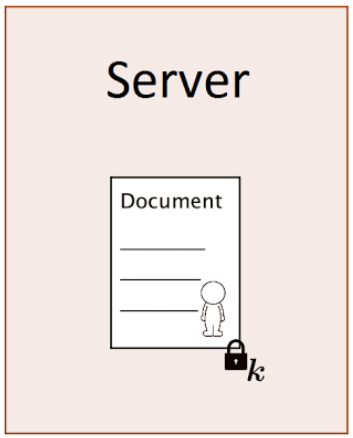
Transmission

1) The server would send a document D to an user, then it:
   - creates a customized copy in view-only mode for the user
   - tries to establish a TLS connection with each user's device
   - creates a session key k to encrypt D
   - encrypts k using each public key of the user's applications
   - signs $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ with own private key and sends it
   - deletes k and the copy of D in a safe way.



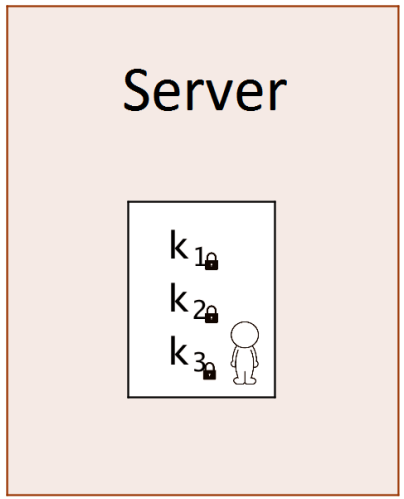
Transmission

1) The server would send a document D to an user, then it:
   - creates a customized copy in view-only mode for the user
   - tries to establish a TLS connection with each user's device
   - creates a session key k to encrypt D
   - encrypts k using each public key of the user's applications
   - signs $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ with own private key and sends it
   - deletes k and the copy of D in a safe way.



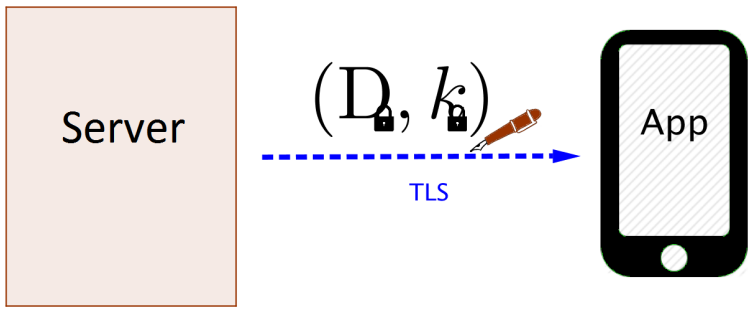
Transmission

1) The server would send a document D to an user, then it:
   - creates a customized copy in view-only mode for the user
   - tries to establish a TLS connection with each user's device
   - creates a session key k to encrypt D
   - encrypts k using each public key of the user's applications
   - signs $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ with own private key and sends it
   - deletes k and the copy of D in a safe way.



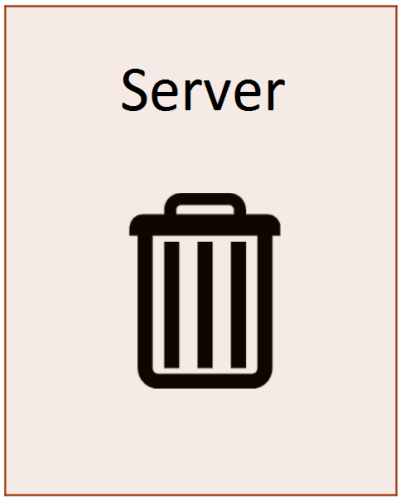
Transmission

1) The server would send a document D to an user, then it:
   - creates a customized copy in view-only mode for the user
   - tries to establish a TLS connection with each user's device
   - creates a session key k to encrypt D
   - encrypts k using each public key of the user's applications
   - signs $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ with own private key and sends it
   - deletes k and the copy of D in a safe way.



Transmission

1) The server would send a document D to an user, then it:
   - creates a customized copy in view-only mode for the user
   - tries to establish a TLS connection with each user's device
   - creates a session key k to encrypt D
   - encrypts k using each public key of the user's applications
   - signs $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ with own private key and sends it
   - deletes k and the copy of D in a safe way.



Reception: first check of the certificate

1) The application verifies the certificate of the device:

   ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.

   ✓ If it is valid, the application:

   - verifies the signature of $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ using the public key of the server
   - saves $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ on the device

Reception: first check of the certificate

1) The application verifies the certificate of the device:

   ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.

   ✓ If it is valid, the application:

   - verifies the signature of $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ using the public key of the server
   - saves $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ on the device.

Reception: first check of the certificate

1) The application verifies the certificate of the device:

   ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.

   ✓ If it is valid, the application:

   - verifies the signature of $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ using the public key of the server
   - saves $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ on the device.

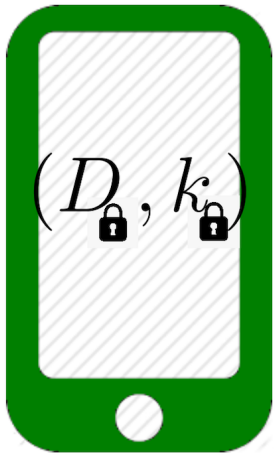
Reception: first check of the certificate

1) The application verifies the certificate of the device:

   ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.

   ✓ If it is valid, the application:

   - verifies the signature of $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ using the public key of the server
   - saves $(\mathrm{Enc}(D), \mathrm{Enc}(k))$ on the device.



Reception: second check of the certificate

2) The user receives a push email on all of his devices and then he starts the application.

3) The application verifies the certificate of the device for the second time:

   - ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.
   - ✓ If it is valid, the application requests the user's credentials of the strong authentication, checks them and goes to the step 4)

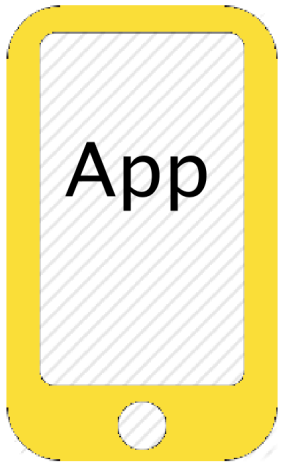
Reception: second check of the certificate

2) The user receives a push email on all of his devices and then he starts the application.

3) The application verifies the certificate of the device for the second time:

   ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.

   ✓ If it is valid, the application requests the user's credentials of the strong authentication, checks them and goes to the step 4).



Reception: second check of the certificate

2) The user receives a push email on all of his devices and then he starts the application.

3) The application verifies the certificate of the device for the second time:

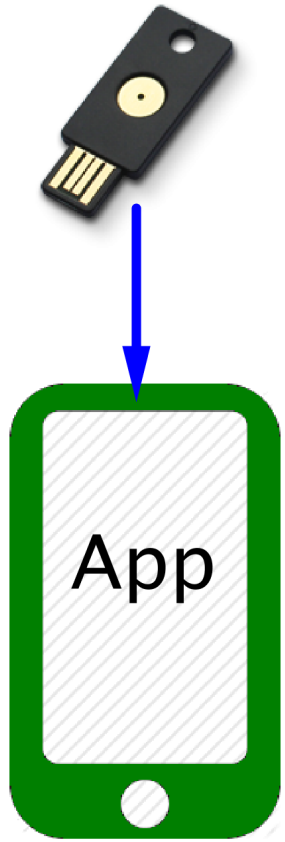
   - ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.
   - ✓ If it is valid, the application requests the user's credentials of the strong authentication, checks them and goes to the step 4).

Reception: second check of the certificate

2) The user receives a push email on all of his devices and then he starts the application.

3) The application verifies the certificate of the device for the second time:

   - ✗ If it is not valid, then the application deletes the public and private keys of the application itself and the encrypted documents stored on the device.
   - ✓ If it is valid, the application requests the user's credentials of the strong authentication, checks them and goes to the step 4).

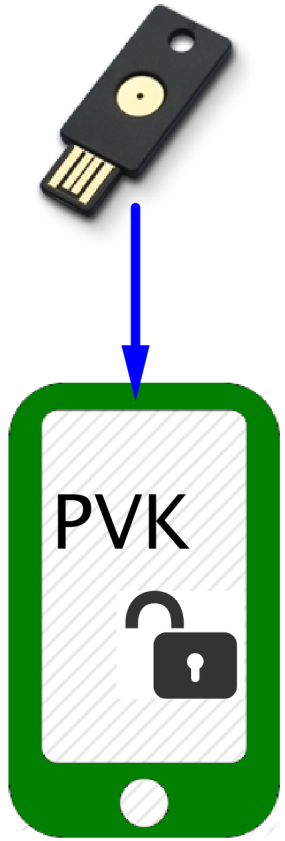


Reception: decryption of the document D

4) The application decrypts:

- the private key of the application using the credentials of the strong authentication
- the session key k using the private key of the application
- the document D using the session key k.

The user can read the document on his device.

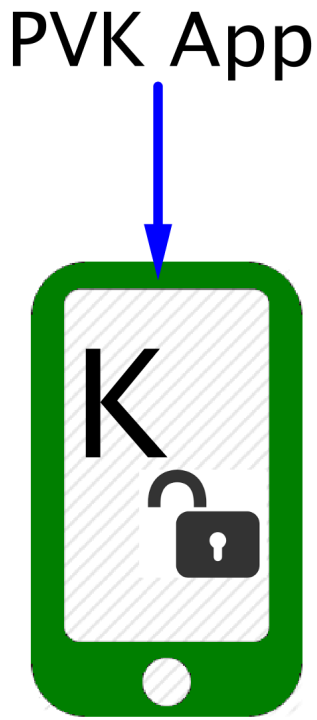


Reception: decryption of the document D

4) The application decrypts:

- the private key of the application using the credentials of the strong authentication
- the session key k using the private key of the application
- the document D using the session key k.

The user can read the document on his device.

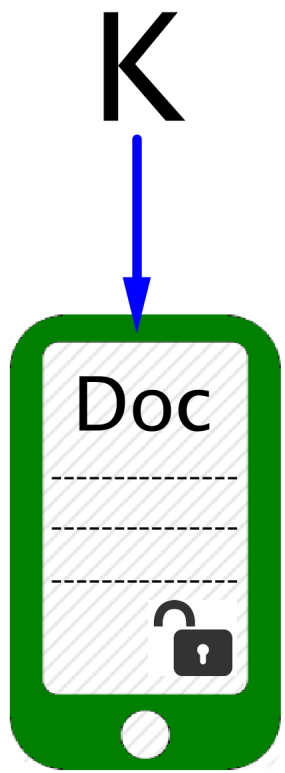


Reception: decryption of the document D

4) The application decrypts:

- the private key of the application using the credentials of the strong authentication
- the session key k using the private key of the application
- the document D using the session key k.

The user can read the document on his device.

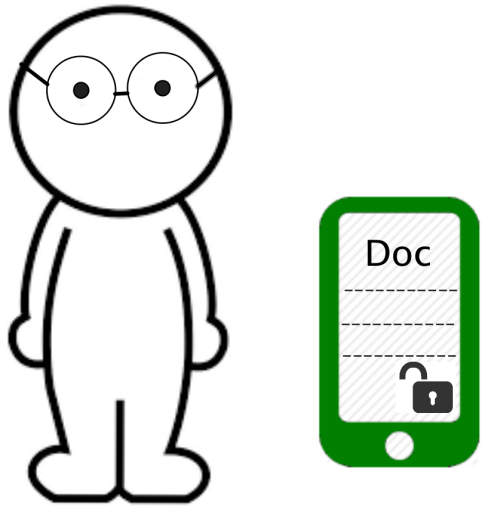


Reception: decryption of the document D

4) The application decrypts:

- the private key of the application using the credentials of the strong authentication
- the session key k using the private key of the application
- the document D using the session key k.

The user can read the document on his device.



Remark

Why does the application check twice the certificate of the device?

- First check: if the certificate of the device is been invalidated after the transmission but before the reception of the document, then the document will never be stored on the device.
- Second check: if the certificate of the device is been invalidated after the reception but before the visualization of the document, the encrypted document is stored on the device but it will be immediately deleted.

Remark

Why does the application check twice the certificate of the device?

- First check: if the certificate of the device is been invalidated after the transmission but before the reception of the document, then the document will never be stored on the device.
- Second check: if the certificate of the device is been invalidated after the reception but before the visualization of the document, the encrypted document is stored on the device but it will be immediately deleted.

Thank you for your attention