Secure and trustworthy file sharing over cloud storage using eID tokens

Eduardo Duarte
Filipe Pinheiro
André Zúquete
Hélder Gomes

universidade de aveiro
deti departamento de electrónica, telecomunicações e informática
ieeta instituto de engenharia electrónica e telemática de aveiro

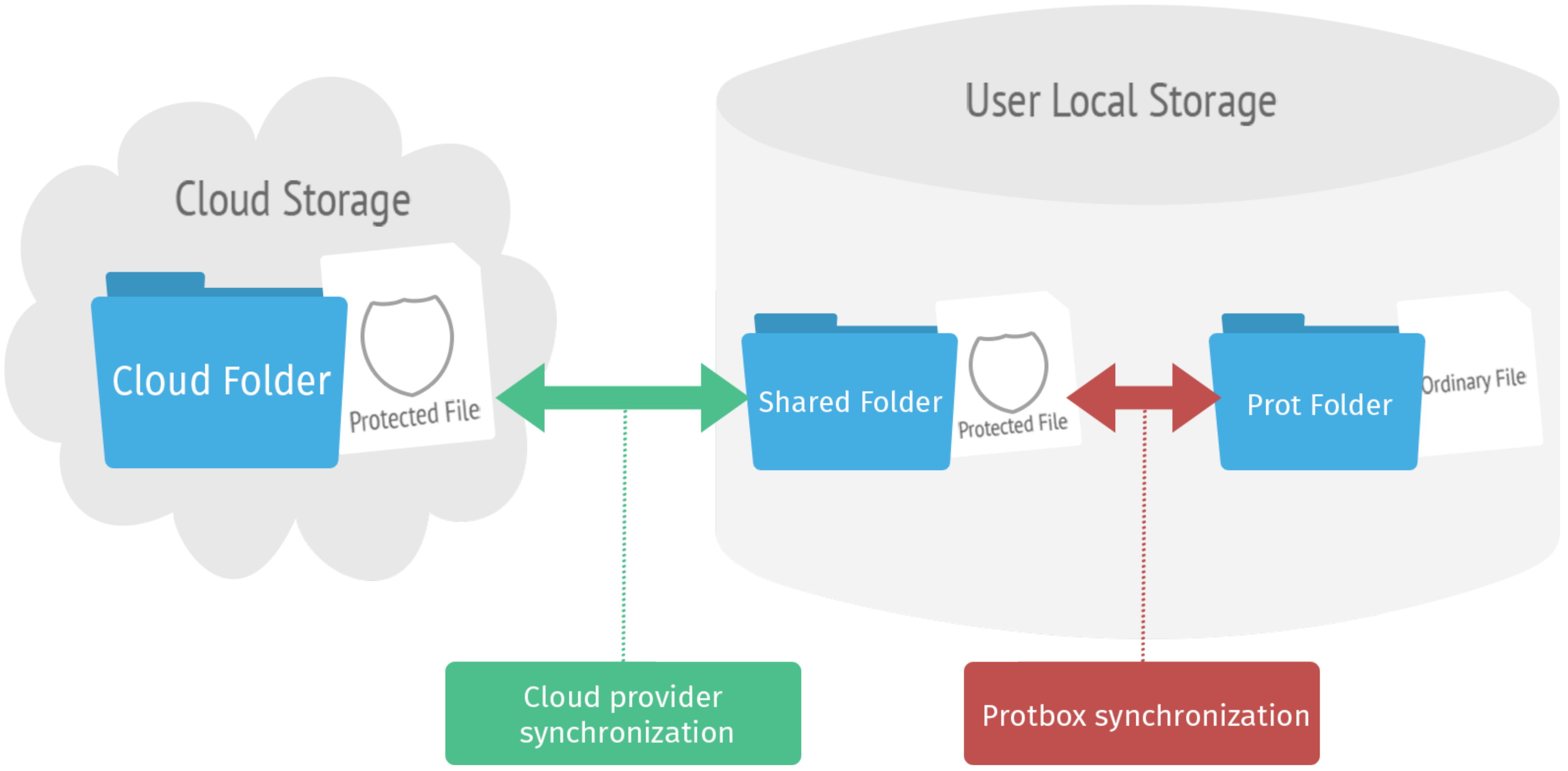
Protbox

- **System conceived for secure file sharing over cloud storage providers**
  - Independent of storage providers
  - Independent of operating systems
    - Implemented in Java
  - Uses eIDs for personal identification
    - Through PKCS #11

Protbox security features

- **Confidentiality**
  - Storage provider has no access to original contents
- **Integrity control**
  - Malicious or involuntary file tampering is detected
- **Content loss**
  - Malicious or involuntary file deletions can be overcome
- **Access control**
  - Personal authorization to access files on shared folders

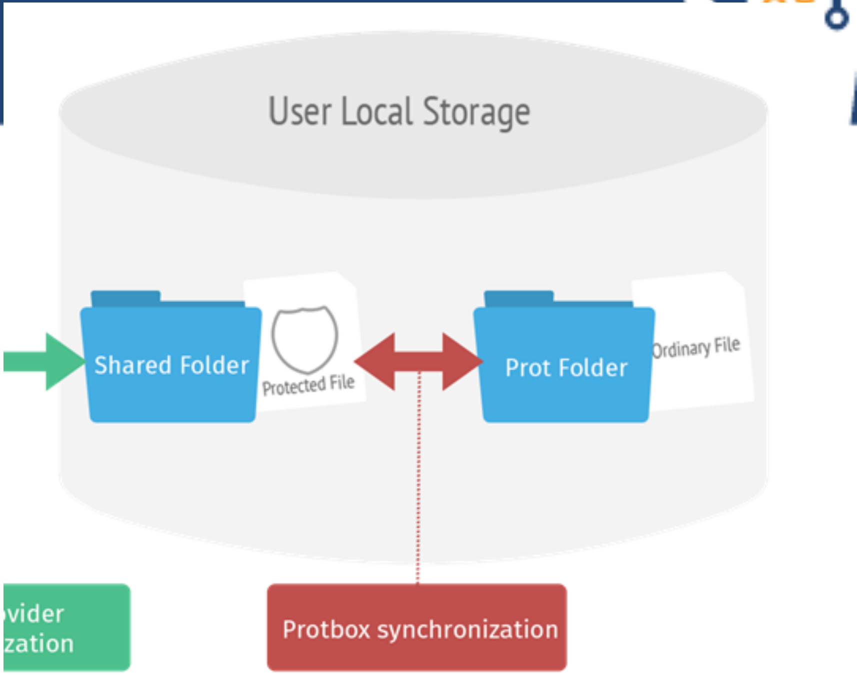
Architecture overview



Architectural requirements

- **Independence from cloud storage solutions**
  - Protbox only uses local folders
  - Shared Folder is a local folder synchronized with a Cloud Folder by software given by the cloud provider
- **eID support**
  - Protbox only requires digital signature support

Terminology



- **Protbox Pair**
  - A pair of directories
    - Shared Folder
    - Prot Folder
  - Both local to the user
- **Pair Key**
  - A symmetric key for encrypting files on a Shared Folder
  - Randomly generated by the first Protbox Pair created upon a Shared Folder

Terminology

- **Key Distribution Key Pair (KDKP)**
  - Asymmetric key pair of a user running Protbox
    - Temporary
    - Created when Protbox starts
  - Public component signed with the user eID
    - Immediately upon creation
  - Usage:
    - Signed requests of Key Pairs
    - Secure communication of Key Pairs

Use case: 1st step

- **Alice and Bob want to share photos**
  - In a private way
- **Alice makes the first move**
  - Creates a Cloud Folder

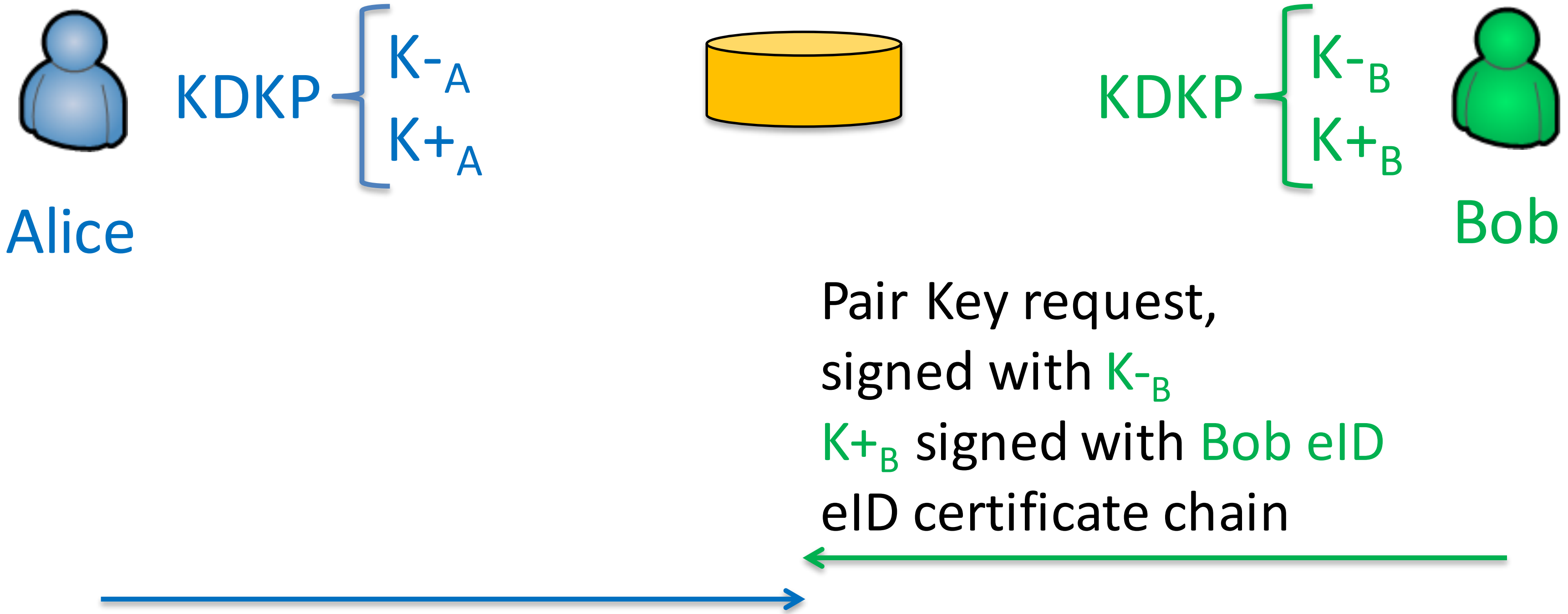
Use case: 2nd step

- **Alice associates the Cloud Folder with a Prot Folder with the photos to share with Bob**
  - – Protbox populates the Cloud Folder with the encrypted versions of Alice’s photos
- **Alice invites Bob to share the Cloud Folder**
  - – Out of the scope of Protbox

Use case: 3rd step

- **Bob associates the Cloud Folder with a Prot Folder**
  - Since the Cloud Folder is not empty, Protbox needs to get its Pair Key
- **Bob's Protbox sends a Pair Key request**
  - Through the Cloud Folder
  - This is a signed request
    - It contains the eID identity of the signer

Pair Key distribution protocol



Pair Key request,
signed with $K-_B$
$K+_B$ signed with Bob eID
eID certificate chain

Pair Key encrypted with $K+_B$
signed with $K-_A$
$K+_A$ signed with Alice eID
eID certificate chain

Use case: 4th step

- **Alice's Protbox pops up Bob's request**
  - Displaying Bob's identity
- **Alice disagrees**
  - The request is overlooked
  - Removed after a timeout
- **Alice agrees**
  - Sends back a confidential reply with the Pair Key
    - Through the Cloud Folder
    - Encrypted with Bob's KDKP public key
  - Signed reply
    - It contains the eID identity of the signer

Use case: 5th step

- **Bob's Protbox gets the reply**
  - And uses the Key Pair to populate his Prot Folder with decrypted versions of Alices' photos
- **Bob adds his photos to the Prot Folder**
  - There encrypted versions will be copied into the Cloud folder
  - Alice can decrypt them into her Prot Folder

Use case: 6th step

- **Alice and Bob can edit the photos**
  - Changes will be propagated as usually
  - But ... Protbox keeps old versions is a log
- **Alice and Bod can delete photos**
  - Changes will be propagated as usually
  - But ... Protbox also keeps a deleted version in the log
- **No file content lost**
  - Unless ... the log limit is exceeded and it gets only populated with gibberish

Protbox control structures

- **Protbox Registry (PReg)**
  - Local data structure
  - Stored in the user home directory
  - Contains all information about the user’s Protbox Pairs
    - Key Pair
    - File’s metadata (name, encrypted name, digests)
    - File’s log

Synchronization issues

- **Alice and Bob simultaneously edit the same photo**
  - And simultaneously save a snapshot of it in their Prot Folder
- **One of them will ‘win’**
  - In terms of Cloud storage
- **But the ‘looser’ does not loose it all**
  - Protbox can detect the conflict and rename files
  - If not, the ‘looser’ version exists in his own log

Privacy issues

- **The identity of Alice and Bob is disclosed to the Cloud provider**
  - It can see that in the signed Pair Key requests and responses

Pair Key distribution issues

- **Anyone with access to the Cloud Folder can provide a signed response**
  - With or without the right Pair Key
  - Responses cannot be reused
    - They are build upon requests
- **Wrong Pair Keys can be a problem**
  - But, at least, attackers are not anonymous

Log management policies

- **On a per file basis**
  - Files may have different relevancy levels
- **On a per user basis**
  - Each Protbox user may have his/her own

Implementation

- **Java application**
  - Publicly available at Github
  - Uses licensed third-party libraries
  - Graphical user interface
- **eIDs are only used when Protbox starts**
  - A new, fresh KDKP is generated
  - Its public key gets signed by the eID owner

Implementation

- **Crypto used**
  - PReg encrypted with a password-derived symmetric key
  - Files encrypted with AES CBC
  - File names encrypted with AES ECB
    - Encoded in a kind of base64 dialect
  - HMAC-SHA1 integrity control
    - Both for files and file names
  - RSA KDKPs
  - eID signatures through PKCS #11 modules

Experience

- **Operating systems**
  - Windows, Linux, MacOS
- **Cloud folders**
  - Dropbox
  - Google Drive
  - Microft OneDrive
  - SugarSync
- **eID solutions**
  - Portuguese eID (Cartão de Cidadão)

Conclusions

- **Protbox enables people to share files through Cloud storage with security**
  - Confidentiality
  - Integrity control
  - Identity assurance
  - Protection against conflicting updates
  - Protection against file deletions

Conclusions

- **Protbox works in different systems and with different Cloud storage providers**
  - No special configurations are required
- **Identity assurance is provided by eID signatures**
  - It should work for many eID solutions
  - Alice and Bob can use different eID solutions