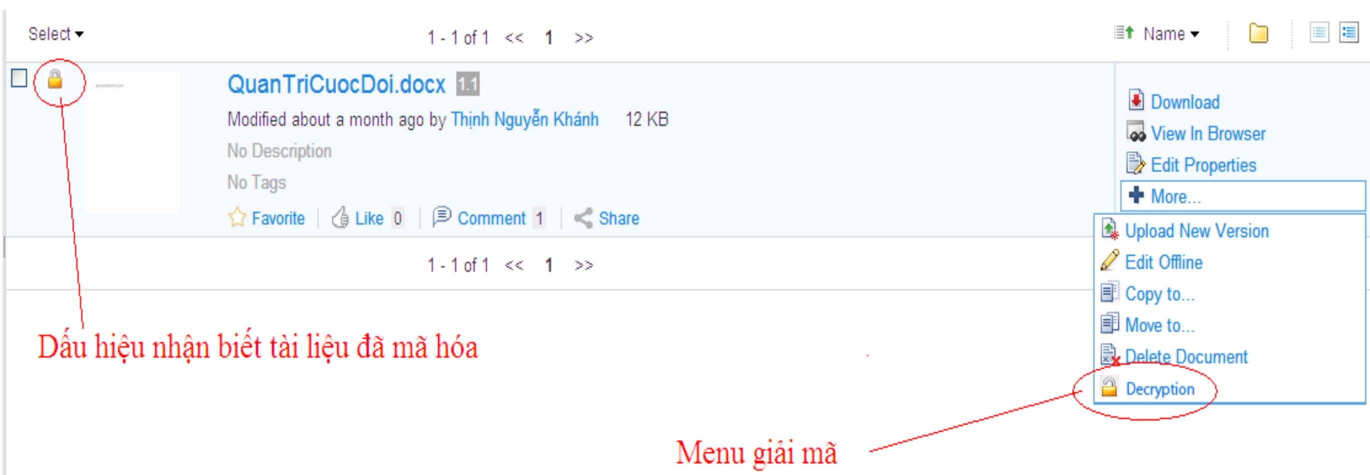
Design Document

Content:

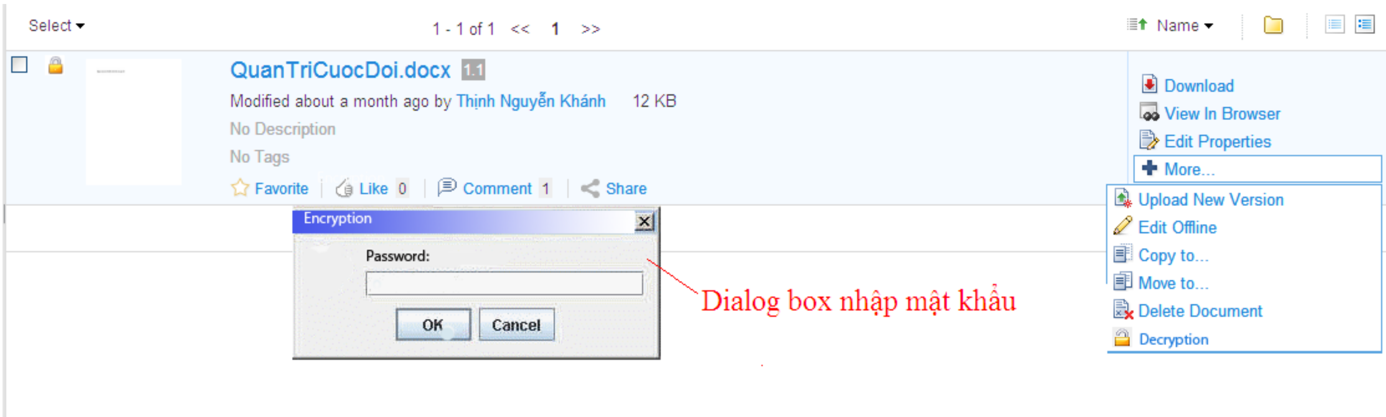
Design the user interface

code chemical

Decoding



Form to enter a password to



General architectural design

Architecture encryption module

in Alfresco, all data are in the form of a "node". The data that is defined as DataType.

Encryption module consists of two main parts Data Processing and Data Encrypting

- First Division Data Processing: Read data from the repository, and write data back to the data property changes.
- The Data Encrypting second part: Encrypt and decrypt data.

Architectural overview

Diagram:.

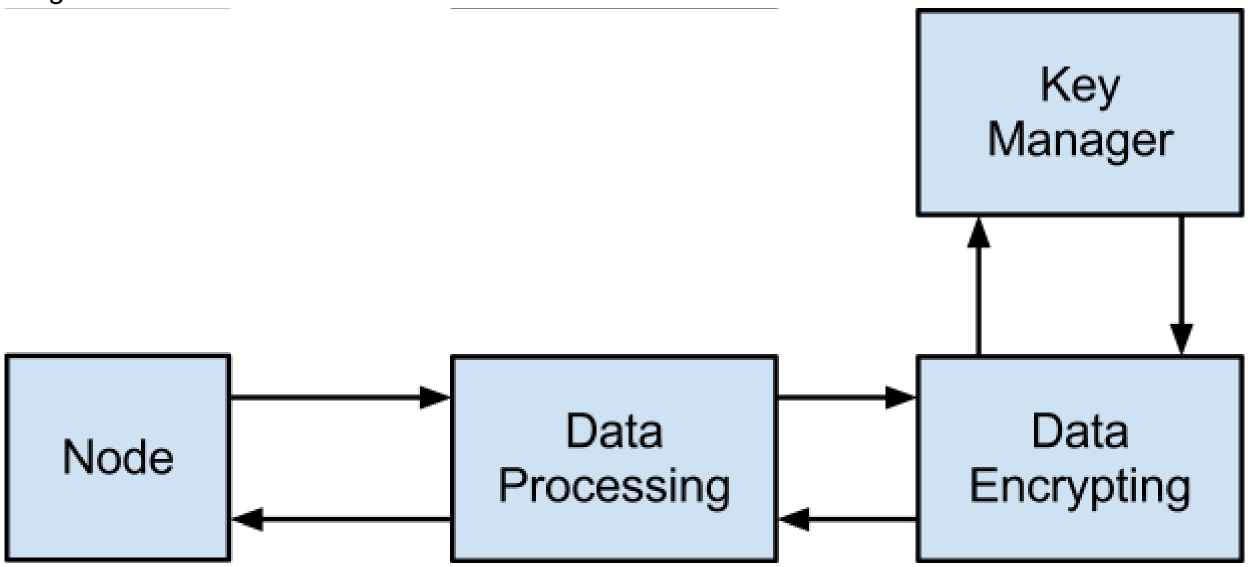
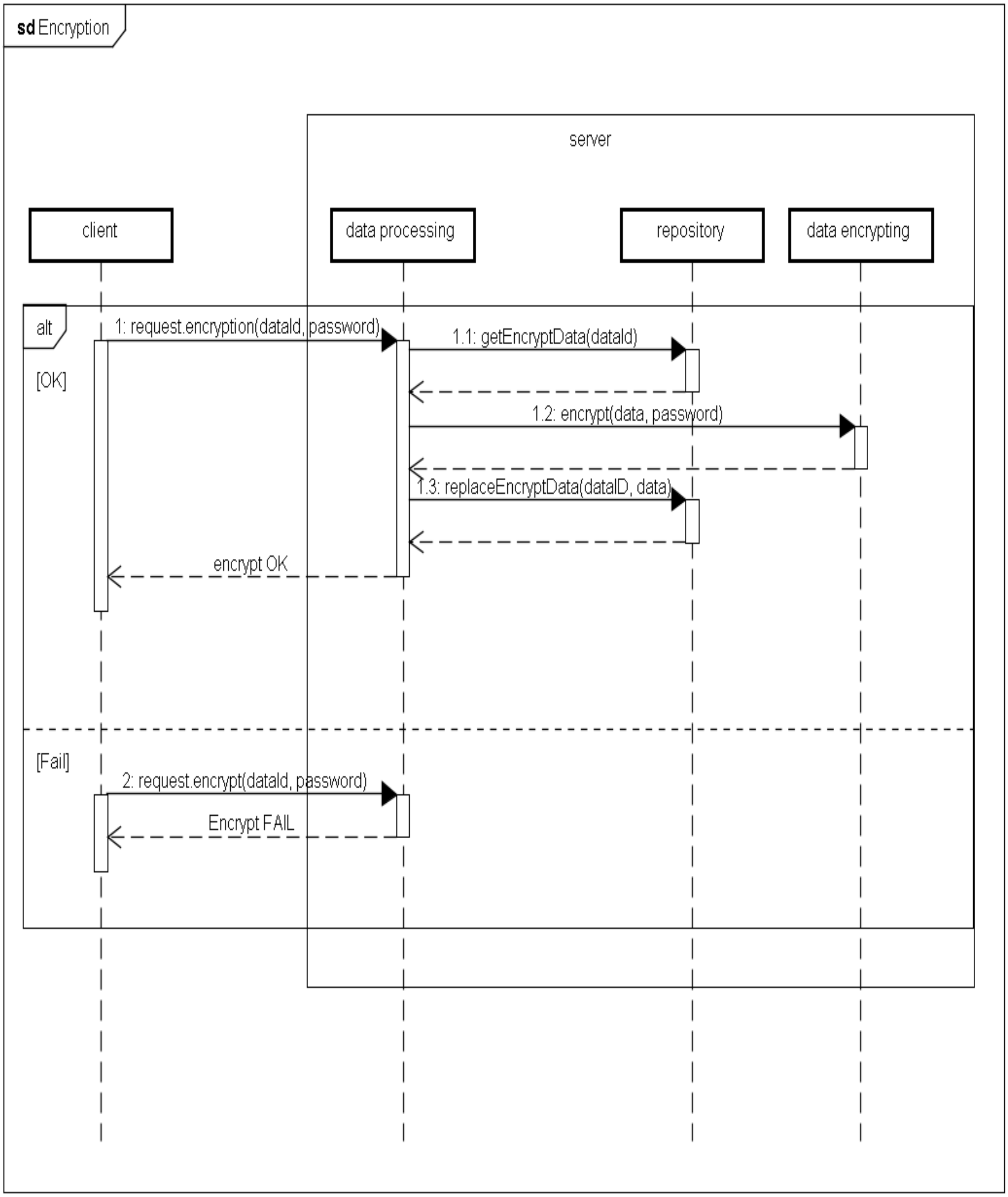
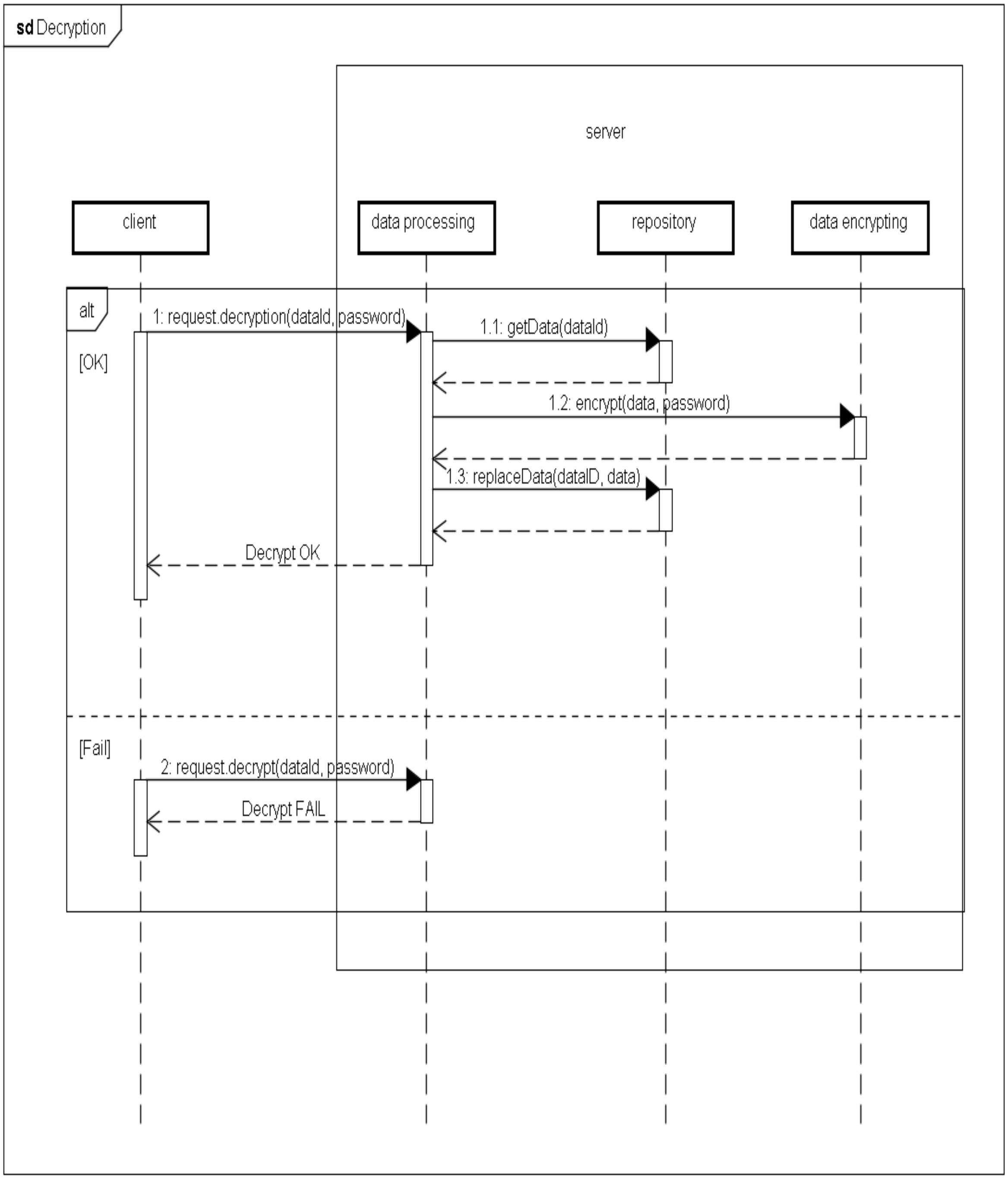


Diagram encoder


sd Encryption
server
client
data processing
repository
data encrypting
alt
[OK]
1: request.encryption(dataId, password)
1.1: getEncryptData(dataId)
1.2: encrypt(data, password)
1.3: replaceEncryptData(dataID, data)
encrypt OK
[Fail]
2: request.encrypt(dataId, password)
Encrypt FAIL

Decoder diagram



AES algorithm

About

AES(Advanced Encryption Standard) is the standard encryption symmetric encryption algorithm Rijndael. Standard is Mi government and NIST (U.S. National Institute of Standard and Technology) workers federal standards. Today the AES encryption standard that is widely used in many areas.
Than meaningful standard, AES are referred to as encryption algorithm tape data blocks of 128 bits keys lengths of 128, 192 or 256 bits corresponds to AES-128, AES-192 and AES-256. algorithmic specification in the design include the following:

1. Definition of concepts, symbols and ham
2. Algorithmic description
3. Problems when installing

Definitions, concepts and symbols of the

term

AES	Advanced Encryption Standard
affine transformation	involves multiplying a transformation matrix and then with one vector
Array	entities numbered lists
Bit	binary value 0 or 1
Block	Chain binary bits of the input, output, State and Round Key. Length of the string is the number of bits it contains. Block can be considered Array of bytes.
Byte	A group of 8 bits as an entity or as Array of 8-bit single
Cipher	plaintext to ciphertext use Cipher Key
Cipher Key	Secret encryption key that is used by a Key Expansion to create a set of Round Keys: can be considered Array of bytes with 4 rows and Nk columns.
Ciiphertext	Data output from the Cipher or Inverse Cipher
Inverse Cipher	Chain variations The record encrypted (ciphertext) report cards (plaintext) of the Cipher Key.
Key Expansion	steps used to create a series of Round Keys from the Cipher Key

plaintext	input data Cipher or output of the Inverse Cipher
Rijndael	algorithm that flowers basis of AES
Round Key	values realized from the Cipher Key by using Key Expansion. Apply to the State in the Cipher and Inverse Cipher
State	coding for intermediate results Array of bytes with 4 rows and Nb columns
S-box	nonlinearity used in turning the byte and Key Expansion to Category 1 - 1 abyte value
Word	Group 32bits as single entities, or Array 4 bytes

Ham, parameter andsymbols

AddRoundKey()	Transformation in the Cipher and Inverse Cipher in Round Key is added to the State content XOR . Round Key length is the size of State.
InvMixColumns ()	Transformation in the Inverse Cipher is inverse MixColumns
InvShiftRows ()	Transformation in the Inverse Cipher is the opposite of ShiftRows ()
InvSubBytes ()	Transformation in the Inverse Cipher is vice SubBytes ()
K	Cipher Key
MixColumns ()	Transformation in the Cipher took all the columns of the State and merge its data independently for a new column
Nb	number of columns in the State, in standards exist Nb = 4
Nk	number of columns in the Cipher Key, in this standard Nk is likely to 4, 6 or 8
Nr	Some of the casualties is a function of Nk and Nb are fixed. In this standard Nr = 10, 12 or 14
Rcon []	A Word Array not wait to go
RotWord ()	Ham Key Expansion receiving 4 bytes and permutations
ShiftRows ()	changes in processing Cipher tape State within three columns Deal

	State with offsets Else
SubBytes ()	transform processing Cipher State using non-linear substitution (S-box) Start up the bytes of independent
SubWord ()	function used in the Key Expansion receive 4 bytes of input and uses S-box for out a word that

Describes the algorithm

In AES algorithm, the length of the input block, output block and State 128 bits. It is expressed through Nb = 4 is the number of 32-bit word in the State.
AES algorithm, the length of the Cipher Key, K, 128, 192, or 256. Key length is represented by Nk = 4, 6, or 8 32-bit words in the Cipher Key.
AES algorithm, the number of rounds (Rounds) was performed during the execution of the algorithm depends on the length key. Number of rounds is represented by Nr. (Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8.)
In the Cipher and Inverse Cipher, the AES algorithm using round function, is created from the transformation bytes:

1. Byte thế using a thế table (S-box)
2. Shifting rows of the State array by different offsets
3. Mixing the data within each column of the State array
4. Adding a Round Key to the State.

Cipher

```
Cipher (byte in [4 * Nb] , byte out [4 * Nb], word w [Nb * (Nr +1)])
begin
     byte state [4, Nb]
     states =print
     AddRoundKey(state, w [0, Nb-1])
     for round = 1 step 1 to Nr-1
          SubBytes (state)
          ShiftRows (state)
          MixColumns (state)
          AddRoundKey (state, w [round * Nb, (round +1) * Nb-1])
     end for
     SubBytes (states)
     ShiftRows (state)
     AddRoundKey ( state, w [Nr * Nb, (Nr +1) * Nb-1])
     out = state
end
```

Inverse Cipher

```
InvCipher (byte in [4 * Nb], byte out [4 * Nb], word w [Nb * (Nr +1)]
)
begin
      byte state [4, Nb]
      states =print
      AddRoundKey(state, w [Nr * Nb, (Nr +1) * Nb-1]) / / See Sec.
      5.1.4
      for round = Nr-1 step -1 downto 1
            InvShiftRows (state)
            InvSubBytes (state)
            AddRoundKey (state, w [round * Nb, (round +1) * Nb-1])
            InvMixColumns (state)
      end for
      InvShiftRows (states)
      InvSubBytes (state)
      AddRoundKey (state, w [0, Nb-1])
      out = state
end
```

Key Expansion

```
KeyExpansion (byte key [4 * Nk], word w [Nb * (Nr +1)], Nk)
begin
      temp word
      i = 0
      while (i <Nk)
            w [i] = word (key [4 * i], key [4 * i +1], key [4 * i +2],
            key [4 * i + 3])
            i = i +1
      end while
      i = Nk
      while (i <Nb * (Nr +1)]
            temp = w [i-1]
            if (i mod Nk = 0)
                  temp = SubWord (RotWord (temp)) xor Rcon [i / Nk]
            else if (Nk> 6 and i mod Nk = 4)
                  temp = SubWord (temp)
            end if
            w [i] = w [i-Nk] xor temp
            i = i + 1
```

```
    end while
end
```