Storgrid encryption

This document combines description of StorgridEncription and also correspondence between Anton and Tatyana. The purpose is to keep all information in one document.

# Storgrid security infrastructure

Storgrid data is divided intonon-intersecting domains. Each domain consists of domain users, groups, and files. Files can be shared between users and users groups inside a domain. Files also can be shared to external users.

When Storgrid is used as a cloud service, installed on provider equipment, then separate domainsare created for each customer.

Users with authorization group “DomainManager” have management access to all domain objects including files of all users. It is important to note that we consider that Storgrid is used as a corporate solution and that a DomainManager needs full control on allcompany data.

# Encryption on server storage

## Problem

Files are saved ona domain storage. Each domain has one or more storages. A storage can be either a local file system ora network storage. The following use cases are possible:

The storage is situated on external provider equipment. In this case, a customer would like to keep files encrypted on storage. (He does not want the provider to view files, or copy the files)

The storage is situated on the company’s own infrastructure and users can access it directly (without Storgrid). In this case it is not desirable to encrypt files on storage. Otherwise, users cannot access their files directly.

## Solution

An administrator can turn on the encryption on storage. In this case, all files on storage will be encrypted by block-cipher with a symmetric-key.

For the beginning, we will use AES encryption with a singlekey for all files.

To extend overall security we will implement the following extensions:

* We will have multiple keys and will automatically choose one of the keys for each file.
* We will re-encrypt files periodically with a new key.
* We will use alternative ciphers like: Blowfish, IDEA, Serpent

## Remarks

* Only files are encrypted never the directory structure!
* All keys are saved on the Storgrid server. Therefore,files are safe on the network storage but if the Storgrid server is hacked then the hacker can steal the keys. We have to save all thekeys on the server because files are belonging to the company not to a single user. If a user is deleted, all files still have to be available for the domain manager.
* If encryption on storage is enabled then direct access to storage (without Storgrid) is useless.

# Encryption for clients

Storgrid clients can synchronize selected files and open/edit them later, even offline. Illegal access to these files is possible if a device is compromised. To solve this particular problem, files can be sent to the device encrypted, and stored locally, also encrypted.

DomainManager defines which folders have to be encrypted.

A file in a Storgrid domain can be shared to individual users or groups of users. Each user/group is associated with a unique attribute and attribute key. An attribute key gives access to files shared with a proper user or group.

A user can belong to several groups. A user key is generated based on the attribute keys of the proper group. Thus, a user has a “key-set” which can open each file shared to this user.

## Authorization

A user is authenticated on the server with a login name and password.

The server defines the set of groups the user belongs to (user attributes).

The server generates a new private key based on the actual user attributes.In addition,theserver will add a timestamp to the key in order limit the time when this key is valid.

Server sends this private key to the client.

The user’s private key can be saved on the client (it is protected with a pin code) and can be used offline. However,the timestamp in the key will protect usinga key longer the determined period.

## File encryption

A unique symmetric key is generated for each file. The file is thenencrypted with an AES-128 cipher and this file key. The encrypted file is sent to the client (I-phone, I-pad, Android). The File key is encrypted with attribute-based encryption according to the file sharing participants list. The encrypted key is also sent to the client and stored locally.

## File decryption

The client checks if the private key of a user is still actual. If yes then the file key is decrypted. If the file key is decrypted successfully then the file is decrypted with the AES-128 cipher.

Note: it can happen that a user does not belong to a certain group any more. In this case his private key will not contain the attribute of this group. As a resultthe user will not have the possibility to decrypt the file that was shared to this group, even if this file was already saved on the device of the user.

## Remarks

Access Based Encryption prevents access to files stored on a device of a user when the user authorization is changed.

## Mobile client workflow

1. User starts-up the Storgrid application, enters the PIN code and opens thedomain.

*Background: login procedure receives actual user private key.*

2. User downloads encrypted file

Background: encrypted file is saved on the device storage

3. User opens the encrypted file in the local Storgrid application

*Background: The application checks the user’s private key validity then decrypts the file and shows the content in the embedded office application (SmartOffice). Non-encrypted files are never available outside the Storgrid application. If a user has upload access level then he can change the file and save it. This modified non-encrypted file is uploaded to the server via a secure channel (HTTPS) and encrypted again on the server*

4. User opens a file with third party application (if allowed).

Background: application checks private key validity then decrypts file and saves non-encrypted file on the device storage. Third party applications can open non-encrypted file. It is the responsibility of the user to save the file back to the Storgrid controlled storage and not to save it anywhere else. When the user saves the file, it is sent to the Storgrid server and encrypted.

# Desktop client

1. The user starts-up the Storgrid application and opens the domain.

*Background: login procedure receives the actual user’sprivate key*

2. User selects the files for synchronization

*Background:encrypted files aresaved on the device storage. A user can see them in his file browser but these files are encrypted and cannot be used directly.*

*For the moment, we want to give these files a special extension. We propose SEF (Storgrid Encrypted File). In reality this means the following: suppose we have a file named “sales\_forecast\_june\_2015.DOCX” When encrypted, the name will be “sales\_forecast\_june\_2015.DOCX.SEF”*

3. User clicks via the context menu “Decrypt”. The file will be decrypted, and directly opened with the application that is associated with the .DOCX extension, normally Microsoft Word.

*Background: the application checks the private key validity, decrypts the file and saves the non-encrypted file in the file system when the file is changed it will be uploaded to the server and will be encrypted again.*

# Web/WebDav client

It is not allowed to download encrypted files with Web/WebDAV clients. (Downloading a file via a browser is not possible)

------------HOW TO SECURE THE KEYS (Tanya’s remarks)------------

There is a concept of encryption keys management (EKM) which is rather well-documented. It includes the questions of storing, renewing, deleting the keys etc. There are many systems of EKM that do most of these operations automatically, but currently we do not need a third-party solution.

http://www.winblog.ru/security/1147764847-28120704.html

https://en.wikipedia.org/wiki/Key\_management

In our case, the server side encryption needs the following simple measures (which can be later changed to more profound ones):

1. Key are kept separately from encrypted files.

2. The keys are kept in a protected key storage.

3. The key storage is protected by the password (minimum level of security) or encrypted with a master key.

4. Master key (MK) is accessible on a server and the reserve copies of MK are kept by the trusted authorities (domain managers) on the special usb devices (tokens) or their mobile devices.

5. MK can also be kept in a shared manner, i.e. a single domain manager cannot get access to MK. Only when several of them get together they get the possibility to restore the MK using the shares and Shamir's SS. This ensures that domain manager can't misuse his copy of MK or even if his device is compromised/lost/stolen the MK is protected.

6. Later on, the server can also participate in the sharing of MK: for example, one part of MK is kept on server and server gets access to encrypted storage once the trusted domain manager plugs his token/usb device into the server machine. Then server gets access to another share of MK and can proceed to work with the encrypted storage.

7. There should be some form of auditing: all activity related to the key storage and encryption must be logged and saved.

30.10:

Защищенное хранение логина-пароля.

1. Для того, чтобы защитить хранение пары логин-пароль необходимо во-первых преобразовать ее к числовому значению - ключу. Это можно сделать тремя способами,
   1. Самое простое – хэшировать эту строку и так хранить или дальше разделять. По хэш значению (например функция SHA2) злоумышленник исходную пару не сможет восстановить.
   2. Можно использовать более надежные методы PBE (password based encryption). В библиотеке Java эти методы реализованы (PBKDF2). Для того чтобы просто защитить неразделенный ключ, полученный из пароля используется многократное хэширование с солью (случайное число):

**PBKDF2** applies a [pseudorandom function](https://en.wikipedia.org/wiki/Pseudorandom_function), such as a [cryptographic hash](https://en.wikipedia.org/wiki/Cryptographic_hash), [cipher](https://en.wikipedia.org/wiki/Cipher), or [HMAC](https://en.wikipedia.org/wiki/HMAC) to the input [password](https://en.wikipedia.org/wiki/Password) or [passphrase](https://en.wikipedia.org/wiki/Passphrase) along with a [salt](https://en.wikipedia.org/wiki/Salt_(cryptography)) value and repeats the process many times to produce a *derived key*, which can then be used as a [cryptographic key](https://en.wikipedia.org/wiki/Key_(cryptography)) in subsequent operations. The added computational work makes [password cracking](https://en.wikipedia.org/wiki/Password_cracking) much more difficult, and is known as [key stretching](https://en.wikipedia.org/wiki/Key_stretching).

* 1. Можно использовать методы восстановления общего пароля (PAKE), когда в протоколе участвуют два участника, знающих общий пароль (в нашем случае клиент и сервер):

**Password authenticated key exchange** (PAKE) is where two or more parties, based only on their knowledge of a password, establish a cryptographic key using an exchange of messages, such that an unauthorized party (one who controls the communication channel but does not possess the password) cannot participate in the method and is constrained as much as possible from brute force guessing the password. (The optimal case yields exactly one guess per run exchange.) Two forms of PAKE are Balanced and Augmented methods.

**Balanced** PAKE allows parties that use the same password to negotiate and authenticate a shared key.

**Augmented** PAKE is a variation applicable to client/server scenarios, in which the server does not store password-equivalent data. This means that an attacker that stole the server data still cannot masquerade as the client unless they first perform a brute force search for the password.

* 1. Можно, ключ полученный в результате операций a или b разделить при помощи СРС. Стандартным методом и быстрым является схема Шамира. Нашим оригинальным методом (сейчас уже стандарт в беларуси) является полиномиальная схема. В самом простом варианте, ключ разделяется и хранится в разных местах устройства (системные папки, память приложения и проч., чтобы злоумышленник не срзу получил к нему доступ).
  2. В более изощренном варианте ключ разделяется между тремя участниками - пользователь, сервер и устройство, у пользователя частичным ключом служит пинкод.

Т.е примерно такая иерархия (от простого к сложному) a, b, a+d, b+d, a+e, b+e (вероятнее всего будете это реализовывать), с (это самое надежное, но я не уверена в существовании стандартной реализации, если заинтересовало, могу разобраться, с этими методами сталкивалась один раз в прошлом году, сейчас в этой области бурное развитие, наверняка уже за год много нового напридумывали).

# Appendix A: Attribute-based encryption algorithm

In order to initialize the system of attribute-based encryption based on the simple selective principle we need the following:

The set of attributes:

The set of attributes corresponds to the set of the access group identifications:

The dataM, or the hash-value of the open text.

The set of user attributes:

The set of attributes of the encrypted text

**The access rule of the selective scheme is as follows:**

**If at least one attribute in the setis equal to the attribute in the set, the corresponding user U can decrypt the text M.**

The encryption system can be implemented as follows:

**Initialization:**

G is the group with the generator g.

The secret master-key which is stored on server and is accessible only for the administrator.

Public keywhich is evaluated according to the master-key and used to access the encrypted data:

where e(g,g) is the bilinear pairing.

**Key generation**

The secret user key D is generation based on the user U attribute set:

This key is sent to the user by the administrator. It depends on the master-key parameters and additional parameter w∈Zq. This parameter serves for the *Di*modification when the public keys need not to be changed.

**Encryption**

The encrypted text E consists of the encrypted message along with the attributes and the public key set{Ei}.

The additional parameter s∈Zqserves for the text re-encryption so that the secret user key set *Di* need not to be changed:

**Decryption**

In order to access the value of M one must evaluate:

The user evaluates using the secret key Di which corresponds to the attribute ti and the public key Ei:

Encryption on server.Development plan.

1. Add "encrypted" flag to storage entity

2. Create structure to save storage encryption keys

New entity StorageKey with the following fields:

intkeynr

String cipher

String key

boolean obsolete

Date generatedWhen

3. Turn on/off encryption on storage from Admin UI

On storage page in admin UI add section "Encryption"

If encryption is disabled show button "Turn on encryption"

Button action:

- set encrypted=true

- generate encryption keys

- start re-encryption process in background

If encryption is enabled show button "Turn off encryption"

Button action:

- set encrypted=false

- mark all encryption keys as obsolete

- start re-encryption process in background

4. Generate keys when turn on encryption

Get the number of needed encryption keys for properties file (by default 100)

Generate AES-128 keys and save them in database.

5. Encrypt output stream when write file on storage

Get one of non-obsolete storage keys (use round-robin).

Write keynr as first 4 bytes of a file. After that write encrypted content.

Add "encrypted" flag in file metadata.

6. Decrypt input stream when read file from storage

Check "encrypted" flag in file metadata.

If file is encrypted then read keynr from the first 4 bytes in input stream.

Find storage key by keynr.

Decrypt file content with the key.

7. Timer for key renew

Create timer which should start once a day.

It should generate N new keys and mark N oldest keys as obsolete.

N is taken from properties file (by default 10)

After that it should start re-encryption process in background.

------>Tanya: The default value can be 100 or even 1000, AES is fast, it won't make a difference

8. Implement re-encryption process

Walk on files tree (including history files)

For each file:

Find file metadata and get the value of "encrypted" flag.

If encryption on storage is disabled and file is encrypted

then decrypt the file and append metadata without encrypted flag.

If encryption on storage is enabled and file is not encrypted

then encrypt the file and append metadata with encrypted flag.

If encryption on storage is enabled and file is encrypted

get encryption keynr, if key is obsolete take another non-obsolete key and decrypt-encrypt the file.

Finally delete obsolete keys which were not used in any file.

Инфраструктура безопасности в защищенном облачном хранилище Storegrid

Базовым сценарием использования Сторгрид является следующий:

Storgrid хостится в датацентре сервиспровайдера. На сервере обслужиаются хранилища различных заказчиков. Новому заказчику выделяется хранилище и он начинает заполнять его файлами. Приэтом

1. Система защиты обладает определенной гибкостью и только часть файлов хранится в зашифрованном виде (защищенное хранилище). Остальные файлы могут шифроваться лишь при передачу наружу (на переносимое устроство)
2. В Сторгриде есть домены – замкнутая система состоящая из пользователей, групп пользователей и файлов доступных пользователям. Есть пользователи с правом DomainManager. Эти пользоваетели имееют право упрвлять всеми объектами внутри домена. В данном документе, говоря Администратор, мы имеем ввиду как раз пользователя DomainManager.

DomainManager может решить, какое хранилилище или часть хранилища должна быть защищенной. После этого DomainManager может дать пользователю доступ к каталогу на этом хранилище (share)cопределенными правами (AccessLevel):

ViewOnly – тоько просмотр

Download – разрешено скачивать файл и сохранять у себя

Download\Upload - разрешено скачивать файл и сохранять у себяи зкачивать измененную версию

Owner – Download\Upload + расшаривать другим пользователям

Таким образом если у меня, обычного пользователя есть Download\Upload доступ к папке ProjectA,и DomainManagerрешил, что содержимое этой папки должно шифроваться, то я все что я загружу в эту папку будет зашифровано.

1. DomainManager(администратор) имеет полный контроль над открытой и защищенной чатью хранилище.
2. Система может разворачиваться рядом с уже существующей инфраструктурой (компания хочет перенести свои данные в облако и при этом защитить)
3. Использование клиента гарантирует защищенность данных.
4. Гарантируется отсутствие возможности для провайдера прочитать защищенные данные в Сторгрид.
5. Гарантируется отсутствие доступа для пользователя в случае, если его ключ не соответсвует актуальным спискам доступа (пользователь удален из группы или из домена).
6. В офф-лайне пользователь имеет доступ к данным со своего устройства лишь в течение ограниченного времени (гарантируется устаревание ключа). По истечении этого времени гарантируется, что пользователь не сможет никаким образом прочитать зашифрованный файл со своего устройства.

В инфраструктуре безопасности Сторгрид участвуют 3 базовых механизма:

1. Блочное шифрование (симметричное). Это шифрование используется 2 раза:
   1. Данные из защищенной части домена (выбранные администратором) хранятся на сервере в зашифрованном виде (постоянное шифрование). При этом администратор устанавливает следующие опции:
      1. Необходимость перешифрования данных
      2. Частота перешифрования данных (раз в сутки, раз в неделю и т.д.)
      3. Наличие одного или нескольких ключей (например, могут быть разные ключи для разных групп пользователей и их папок).
      4. Алгоритмшифрования – AES, Blowfish, IDEA, Serpent. Самый быстрый на 128-битном ключе AES, но уже на 256 битах от теряет в скорости.

Замечание. Общая стратегия здесь такова: чем чаще перешифровываются данные, тем меньше может быть ключ. Скажем, если данные перешифровываются раз в сутки, то можно гарантировать, что при условии хорошей защиты доступа к ключу, достататочно все шифровать одним единственным ключом AES128. А если раз в неделю – то лучше использовать разные ключи для разных папок и ключ длиной 256 бит.

* 1. Файлы при передаче на устройство клиента шифруются и хранятся в зашифрованном виде.
     1. Для каждого файла имеется уникальный сеансовый ключ.
     2. Он хранится на устройстве пользователя в зашифрованном виде. Шифруется ключ при помощи шифрования с атрибутами (ABE).
     3. Этот ключ имеет время жизни, т.е. в нем есть метка времени, и спустя какое то время открыть по этому ключу файл нельзя.
     4. Для клиента важна скорость и простота реализации, поэтому предполагается, что клиент умеет только расшифровывать
     5. Поэтому предлагается использовать AES128 бит для ускорения операции расшифрования, поскольку время жизни и уникальность сеансового ключа гарантируют защищенность данных.

Замечание.Метка времени - это обычно дата, а точнее хэш значение, которое также шифруется ABE и хранится в зашифрованном виде вместе с сеансовым ключом. Второй способ – считать число открытий файла (значение счетчика также шифруется) и если их слишком много – считать, что что-то не так и запрещать доступ.

1. Асимметричное шифрование с атрибутами.

Шифрование с атрибутамииспользуетсядля

* 1. Добавления нового пользователя
     1. Администратор определяет папки, доступные для пользователя и передает на устройство пользователя ключи для этих папок (см. Ниже пользовательские ключи), которые генерируются из мастер-ключа, который есть только у сервера, в защищенном хранилище.
     2. При этом каждая группа пользователей обладает уникальным ключом для доступа ко всем файлам этой группы. Ключ группы не является приватным, а общим для всех членов группы.
     3. Таким образом у каждого пользователя формируется уникальный набор ключей его групп (связка ключей), который соответствует его атрибутам.
  2. Авторизации пользователя при определении того, к каким папкам он имеет доступ:
     1. При авторизации пользователя (на основе пароля, в самом простом случае) сервер определяет атрибуты пользователя,
     2. Сервер обновляети передает клиенту пользовательские ключи и отображает список папок.
  3. Передачи сеансового ключа пользователя.
     1. Пользовательский ключ хранится на устройстве пользователя и становится доступен после ввода им пароля.
     2. Если пользователь прошел аутентификацию на устройстве, то даже в режиме офф-лайн пользовательский ключ позволяет ему читать защищенные файлы.
  4. Определения времени жизни сеансового ключа.
     1. Если метка времени (которая также шифруется пользовательским ключом) истекла, то клент просто меняет или удаляет сеансовый ключ, так что гарантируется отсутствие доступа в офф-лайн к защищенному сеансовым ключом файлу.
     2. Для возобновления доступа пользователю нужно присоединиться к серверу и обновить пользовательские ключи доступа.

Замечание. Пользовательские ключи могут быть постоянными (неизменными), но я сейчас склюняюсь к тому, что их как и серверные ключи нужно периодически менять. Поскольку они служат лишь для шифрования сеансовых ключей и обновленный ключ все равно не влияет в течение какого врмени н доступ в офф-лайн – то их можно обновлять раз в сутки/неделю/при каждом изменении состава группы.

1. Разделенное хранение пользовательских ключей.

Для того чтобы гарантировать отсутствие доступа к сеансовому ключу при помощи невалидного пользовательского ключа (устаревшего). Для этого используется (3, n+2)-схема Шамира, где n – число пользователей в группе.

* 1. Каждому пользователю выдается не сам пользовательский ключ, а его доля (share), аналогичная доля хранится на сервере и в хранилище (s1, s2, s3).
  2. В офф-лайн режиме пользователю доступно значение целиком (s1+s2+s3).
  3. По истечении срока годности пользовательского ключа клиентское приложение изменяет или удаляет доли s2 и s3 и пользователю необходимо Eподключиться к серверу и хранилищу для получения нового ключа.

Замечание. Я бы хотела немного подумать еще на этой частью реализации. Может быть в начальном варианте достаточно первых двух пунктов.

МЕХАНИЗМ ПРОВЕРКИ СВЕЖЕСТИ ПОЛЬЗОВАТЕЛЬСКИХ КЛЮЧЕЙ

Для того, чтобы удостовериться в том, что пользовательские ключи соответствуют схеме доступа в Сторгрид используется три механизма:

1. Обновление ключей.

Все пользовательские ключиDi (а точнее один ключ – соответствующий атрибуту группыAi)меняются при изменении состава группы:

Меняется лишь параметр w. Мастер ключ сервера остается постоянным. Открытые ключи при этомтакже меняются

Алгоритм обновления ключей следующий:

1. Сервер получает информацию об изменении состава группы.
2. Сервер случайно генерирует число w.Wне хранится, а служит лишь для генерации ключа. Это кстати позволяет защититься от атаки на мастер-ключ (в случае утери мастер- ключа злоумышленник все равно не сможет восстановить все пользовательские ключи).
3. Cервер запрашивает доступ к мастер ключу и генерирует значения: Ei, Di
4. Сервер складывает в БД публичный ключ группы и приватный (пользовательский) группы значение Di.
5. При обращении к серверу клиента из группы, сервер проверяет и обновляет его пользовательский ключ (см. ниже).
6. Запрос-ответ. Когда клиент обращается за данными к серверу, серверу необходимо проверить свежесть пользовательского ключа клиента. С этой целью выполняется следующий протокол:
   1. Сервер обрщается к БД атрибутов и ищет соответствующий группе, к которой обращается клиент, открытый/пользовательский ключ. Считается, что клиент уже видит список файлов этой группы но должен подтвердить свежесть ключа.
   2. Сервер отправляет клиенту случайное число.
   3. Клиент шифрует его своим пользовательским ключом и отправляет серверу.
   4. Сервер расшифровывает своим публичным ключом и таким образом, удостоверяется в свежести пользовательского ключа.
   5. Если ключ свежий – то сервер шифрует открытым ключом сеансовый ключ файла и метку времени, а сам файл шифрует сеансовым ключом и отправляет клиенту.
   6. Если ключ устарел, то сервер требует, чтобы клиент прошел авторизацию, при которой ему отправляется связка ключей (весь набор его пользовательских ключей соответствующих атрибутам) и соответственно, обновляется его список видимых (доступных) папок.
7. Метка времени. При доступе к файлу в режиме офф-лайн, клиент хранит вместе с файлом метку времени, зашифрованную пользовательским ключом, с тем чтобы клиент регулярно обновлял свои ключи. При этом предлагается следующий формат проверки:
   1. Пользователь открывает клиент. Клиент автоматически расшифровывает пользовательским ключом (доступным после ввода пароля) метку времени, записанную при отправке файла сервером, и сеансовый ключ.
   2. Если метка времени свежая (например, не больше суток или недели, для разного уровня конфиденциальности), то файл расшифровывается сеансовым ключом и показывается пользователю в рабочем окне клиента.
   3. Если метка времени устарела, то только что расшифрованный сеансовый ключ удаляется/модифицируется клиентом и от пользователя требуется пройти авторизацию на сервере.
   4. После авторизации на сервере клиент получает обновленную связку ключей/список папок.
   5. При доступе к файлам свежая метка времени ~~и сеансовый ключ~~ отправляется клиенту.

Замечание. На самом деле, устаревание метки времени и устаревание ключа – это две разные вещи. Например, при устаревании метки времени, от пользователя можно потребовать лишь слабую авторизацию на сервере (только подключиться и получить новый сеансовый ключ и метку, без обновления связки ключей), а вот при устаревании пользователского ключа (если сервер так считает) уже требовать пройти полную авторизацию с вводом пароля/ссылки на емейл и пр.

---------------------------------------------ВОПРОСЫ ОБНОВЛЕНИЯ КЛЮЧЕЙ----------------------------------

1. Периодическое обновление ключа:
   1. Пользовательский ключ на устройстве должен обновляться с определенно периодичностью. По истечении этого периода клиент запрещает доступ к сеансовому ключу файла и удаляет из памяти публичный ключ, соответствующий устаревшему пользовательскому ключу.
   2. Для того чтобы получить новую пару (пользовательский ключ, публичный ключ) пользователю необходимо обратиться на сервер, так как его публичный ключ теперь не существует и расшифровать сеансовый ключ невозможно даже при условии, что у клиента сохранилась копия старого пользовательского ключа.
   3. Клиенту желательно удалять публичные ключи из памяти еще и по той причине, что в обратном случае, пользователь может воспользоваться своим старым ключом (и старым публичным ключом) для получения доступа к файлам, к которым ему закрыт доступ, поскольку доступ к сеансовому ключу зависит лишь от наличия пары (пользовательский ключ, публичный ключ).
2. Обновление ключа при удалении пользователя из группы.
   1. При удалении пользователя из группы (это может сделать администратор) сервер получает информацию об атрибуте группы из базы данных.
   2. Используя мастер ключ, соответствующий атрибуту группы, и случайное wсервер генерирует новую пару ключей группы.
   3. Сервер складывает эту пару в базу данных, в соответствии с атрибутом.
   4. При периодическом обновлении ключа пользователи группы получат новую пару (ПК, ОК). Перезакачивать файлы им при этом не потребуется.
   5. Удаленный пользователь не сможет сохранить старый ОК и потеряет доступ к файлам группы (не сможет расшифровать сеансовый ключ). При периодическом обновлении ключа, он не сможет видеть папки группы в дереве каталогов и получит новые ключи.
   6. Скорее всего здесь клиент должен предлодить пользователю удалить ненужные файлы (зашифрованные сеансовым ключом, доступ к которому утерян)
3. Обновление ключей при изменении атрибутов файла.
   1. Сервер получает информацию об атрибутах групп, которые потеряли доступ к файлу, из базы данных.

*При этом :*

* *Если для* ***удаленных*** *групп (атрибутов) меняется параметр s (который соответствует группе, атрибуту или пользователю) - пользователям такой группы придется перезакачать все файлы, они потеряют доступ ко всем имеющимся сеансовым ключам. Поскольку открытый ключ их группы теперь поменялся так, что не сможет расшифровать старое значение*
* *Поэтому для каждого соответствия файла (или папки) File\_Aгруппе Group\_Bв БД хранится уникальное значение s\_AB (параметр шаринга), которое определяет доступ к этому файлу:*

*File\_A → Group\_B :s\_AB*

* *Как только у файла File\_A удален атрибут (запрещен доступ группыGroup\_B)генерируется случайное s\_notAB:*

*s\_AB → s\_notAB*

* *После этого перегенерируются пары (ОК, ПК)для вceх пользователей из Group\_B, котораяТЕПЕРЬ НЕ входит в атрибуты файла File\_A с новым s\_notAB:*

*(OK, ПК)=*

* *Значение s\_notABне хранится в базе, запись о шаринге удаляется из базы (?), s\_ABудаляется*
* *После периодического обновления ключей каждый пользователь группыGroup\_Bполучит (ОК, ПК) с s\_notABи не сможет расшифровать файл на своем устройстве:*
* *В ЭТОМ УСЛУЧАЕ НЕОБХОДИМ ОТДЕЛЬНЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ КЛЮЧ ДЛЯ КАЖДОГО ФАЙЛА (ПАПКИ)НА УСТРОЙСТВЕ ПОЛЬЗОВАТЕЛЯ*
* *Как только к файлуFile\_A добавлен атрибут (разрешен доступ группыGroup\_B)генерируется случайное s\_AB (параметр шаринга):*

*s\_AB*

* *После этого перегенерируются пары (ОК, ПК) для вceх пользователей из Group\_B, котораяТЕПЕРЬ входит в атрибуты файла File\_A с новым s\_AB:*

*(OK, ПК)=*

* *Значение s\_ABхранится в базе*
* *После получения файла, зашифрованного сеансового ключа, периодического обновления ключей каждый пользователь группы Group\_B получит (ОК, ПК) с s\_ABи сможет расшифровать файл на своем устройстве:*

Вопросы про устаревание ключей и метки времени:

1) Если метка времени устарела, то только что расшифрованный сеансовый ключ удаляется/модифицируется клиентом и от пользователя требуется пройти авторизацию на сервере.

Не получится ли, что после этого клиент должен перезакачать и сам файл с сервера. Этого хотелось бы избежать. Иначе получится, что клиент должен раз в сутки перескачать все свои файлы.

*Тогда этот сеансовый ключ клиента нужно хранить на сервере. И отправлять ему опять. Или просто запрещать доступ (без изменения сеансового ключа), если считается что клиент достаточно защищен и таким образом злоумышленник не получит возможность получить сеансовый ключ.*

2) Правильно ли я понимаю, что обновленный ключ пользователя позволит расшифровать старые сеансовые ключи? (Конечно при условии что метка времени не устарела и есть пересечение хотя бы по одному атрибуту)

*Сеансовый ключ отправлять и перешифровывать н требуется, т.к.*

*Где K’ – это зашифрованный сеансовый ключ, который обновлять не требуется. Его нужно перешифровать при изменении значения s, которое является параметров глобального устаревания и служит скажем для случая, когда точно известно что что-то взломано в данном домене или группе.*

2) W не хранится, а служит лишь для генерации ключа

Ключи Di генерируются для каждого пользователя отдельно (у каждого свое значение w)?  Если да, то мне кажется перегенерировать ключи надо не при любом изменении группы, а только по факту добавления/удаления человека из группы.

*Под изменением группы именно это и понимается, по другому группу поменять невозможно.*

В формуле шифрования E=Me(g,g)^yws участвует w. где его взять если не хранить?

*Там в открытом ключе wвнизудроби, а в закрытом – вверху. По свойству билинейных отображений – wвообще не останется в степени при дешифровании.*

3) Описанным способом мы обновили ключ если поменялся состав группы. Возможна еще ситуация когда изменился набор атрибутов у файла. Наверное эту ситуацию тоже не плохо бы отработать?

Может при устаревании метки времени клиент пошлет свой старый сеансовый ключ, а сервер перешифрует проверит есть ли все еще доступ к этому файлу и перешифрует его?

*Мне кажется что изменение атрибутов файла – это тоже самое что изменеие состава групп.*

И еще вопрос про алгоритм шифрования:

4) ti - элемент множества. Обязательно ли сразу генерировать ti в достаточном количестве, чтобы назначать их потом группам или достаточно запомнить какие-то параметры множества и генерировать ti по мере появления новых групп?

*Лучше генерировать по мере появления, это безопасно, это ведь на самом деле случайные числа, которые являются частью ключа, всегда безопаснее когда их еще нет (еще не сгенерированы).*

И дополнительный вопрос про шифрование файлов на сервере

5) Ключ от файла хранится на сервере. Если что-то случилось с сервером (поломался,заглючил и т.п.), то расшифровать файлы становится не возможным. Может есть какие-то стандартные практики бэкапа ключей или еще что нибудь подобное?

*Можно хранить ключ этот в разделенном виде у администраторов (например использовать (2,t)-пороговую схему, когда любые два участника схемы могут восстановить ключ. Участниками являются администраторы домена и сервер. При разделенное хранение постоянного ключа я тебе в письме еще писала). Доли постоянного ключа хранятся на устройствах администраторов в защищенном виде, зашифрованы их пользовательскими ключами.*

---------------------------------------------------------------------------------ЭТО ТО ЧТО УЖЕ БЫЛО

Алгоритм шифрования с атрибутами.

На самом деле, как я говорила в чистом виде там шифрование с атрибутами не требуется. Поскольку структура атрибутов довольно прозрачная: Если у пользователя есть атрибуты A1, A2,…,An – то как только один из атрибутов (или соответствующих приватных ключей) подошел – то пользователь может получить доступ к какой-либо группе данных.

Конструкция для этого – парная криптография (бибиотека PBC) это один из видов шифрования на эллиптических кривых Выглядеть математически все должно примерно так:

1. Мастер-ключ (хранится на сервере, доступен только для адиминистратора)

*Эти значения ti – на самом деле являются приватными ключами, соответствующими атрибутам групп. В отличие от приватных ключей пользователей при обычном шифровании с открытым ключом, здесь приватные ключи генерирует и хранит администратор.*

Публичный ключ – зависит от секретных значений мастер-ключа, находится в открытом доступе, служит для доступа к информации:

Здесь e(g,g) – билинейное преобразование.

1. Секретный ключ каждого пользователя формируется в зависимости от его набора атрибутов:
2. Шифрование открытого текста M – это умножение:
3. Расшифрование:

Для этого пользователь использует Diдля атрибута ti который присутствует среди атрибутов текста:

*В данном случае шифруетс и расшифровывается ключ для дступа к зашифрованным данным на устройстве.*

# AZ 28.07.2015Сценарии изменения условий доступа к файлу

Пусть:

* Есть группы пользователей: Студенты (t1), Преподаватели (t2).
* Есть пользователь Вася, член группы Студенты
* Есть файл лекции.txt расшаренный группам Студенты и Преподаватели и для этого файла включено "шифрование наружу".

У Васи есть доступ к файлу лекции.txt, поэтомуStorgrid Client скачал и сохранил файл, зашифрованный сеансовым ключом; получил и сохранил сеансовый ключ, зашифрованный с атрибутами t1 и t2; получил и сохранил приватный ключ с атрибутом t1.

Storgrid Client должен периодически связываться с сервером и обновлять свой приватный ключ, а также информацию о доступе Васи к файлу лекции.txt

Возможные изменения:

A. Файл лекции.txt стал расшарен только группе Преподаватели. Т.е. Вася потерял доступ к этому файлу.

B. Файл лекции.txt стал расшарен только группе Студенты. Т.е. у Васи сохранился доступ к файлу, однако атрибуты доступа изменились.

С. Васю исключили из группы Студенты. Т.е. Вася потерял доступ к файлу лекции.txt.

D. Васю добавили в группу Преподаватели. Т.е. у Васи сохранился доступ к файлу лекции.txt, но у него изменились атрибуты в приватном ключе.

Прошу описать действия клиента для отработки каждого из случаев.

PS: тут надо отметить еще один нюанс. Сейчас система реализована так, что если у пользователя пропал доступ к файлу (случаи A и C), то он этого файла больше не увидит. Т.е. если все нормально, то Storgrid Client обнаружит отсутствие файла и удалит его.

ОТВЕТЫ НА ВОПРОСЫ

1) если мы для каждой пары юзер-шаринг генерируем публичный и приватный ключ и отдаем их клиенту, то для чего здесь пара ключей?

Эта пара нужна помимо того что в шифровании с атрибутами она строится по атрибутам (общий алгоритм как ты помнишь сложнее, и обладает бОльшими возможностями чем то что у нас сейчас), но еще и для того чтобы поддерживать преемственность шифрования при смене ключа. Сеансовый ключ менять нам не надо, перезакачивать файл тоже не надо - потому что он в зашифрованном виде уже есть на клиенте. Если ключ будет только одни то мы этого добиться просто не сможем. Нам придется в таком случае хранить на сервере каждый сеансовый ключ каждого файла каждого клиента и при перешифровании его (сеансового ключа) отправлять клиенту копию этого сеансового ключа, зашифрованную новым (единственным ключом), а старую убивать. В принципе это эквивалентно хранению двух ключей. Кроме того на сервер не нужно в случае шифрования с атрибутами хранить копию пользовательского ключа. А публичный ключ известне всем и защищать его не надо. Т.е. в случае использования связки единственный ключ+сеансовый ключ - на сервере нужно защищать уйму ключей, включая сеансовые. А в случае использования шифрования с атрибутами на сервере достаточно хранить и защищать только мастер ключ шифрования с атрибутами. Кроме того непонятно, как такая схема ключ+сеансовый ключ будет работать с атрибутами файлов и групп.

2) Может достаточно сгенерировать симметричный ключ? А файлы шифровать ключом, который построен на комбинации: ключ шаринга + сессионный ключ. Есть в этом какая-то логика?

Симметричное шифрование повлечет за собой необходимость хранить сеансовый ключ для каждого файла на каждом клиенте. И хранить и защищать каждый ключ шаринга. В случае шифрования с атрибутами публичную часть ключа защищать не надо, она открыта. А пользовательский ключ сервер не хранит и не защищает, просто генерирует и отправляет клиенту. Серверу нужно только защитить мастер-ключ. Кроме того нужно будет вручную связывать (и хранить) симметричные ключи и атрибуты. В шифровании с атрибутами ключи генерируются исходя из атрибутов.  
  
3) дошли до попыток зашифровать что-нибудь на сервере и сразу наткнулись на то, что кроме шифра AES и размера ключа надо выбрать Mode и Padding. Варинтов много. Можешь что-то порекомендавать?

для больших объемов данных обычно используют CBC и paddingmethod 2 по стандарту <http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=50375>.

в России и у нас принят стандарт режима гаммирования CFB. В принцип администритору можно дать возможность выбрать самому. ТОЛьконеставить ECB.

# AZ 04.08.2015 Encryption algorithm description (to replace ABE description in them main part)

### Initial data

*T = {Ti}; i=1..n* – Set of domain members. T includes single domain users and domain groups as well.

Each domain user can be a member of one or more domain groups. Each domain group can be a member of other domain groups i.e. there is a hierarchy of the domain groups.

*S = {Sj}; j=1..k* – Set of shares. Share is a file or folder in the global storage shared to one or more domain members.  
 - Set of the share participants. P(Sj) is a subset of domain members T with the right to access the share *Sj.*

### Encryption initialization

Initialize *G* as the group with the generator *g*.

Generate random and save it on server (secretly, as the part of the master key).

For each domain member *Ti* generate random attribute and save it on server secretly as the exclusive property of domain member.

For each share *Sj* generate random and save it in server as the exclusive property of the share.

Remark: *ti* and *sj* can be generated in a lazy mode, i.e. generated on the fly when needed.

## Authorization

The following procedure takes place after the domain user} authentication i.e. he is already authorized to access server with the accordance to his/her login name and password.

After passing the authentication procedure the client sends request to the server for the actual set of keys in order to receive the data access.

The following procedure is performed on the server side:

* Generate random
* Specify the actual user membership *TU* as a subset of T which includes the user U and all domain groups containing the user U as a member, including recursive membership.
* For each generate the secret user key *Di (this value is not stored on the server side, but immediately sent to user)*
* For each share and each user participating in this share   
  generate (the public key, stored on the server side openly)*,*where *i* is the attribute or domain member index i.e.
* Based on the domain security policy, calculate the key validity period i.e. 24 hours/ When the period is over the key expires and has to be replaced.
* Send Di, and validity period to the client.

The client stores these values in the local memory storage. The user key Di is stored secretly.

### File encryption

When the client requests for the access to a certain file the server performs the following procedure

* Detect the share *Sj* containing the necessary file.
* Securely generate AES-128 key K for the symmetric block encryption.
* Encrypt the file with key K and send the encrypted file to the client
* Encrypt the key K with the ABE algorithm and send the encrypted value to the client
* Send the group identifier ***S****j to the client (a value indicating to the client his mode of access, not equal to sj)*

The client either decrypts and opens the file immediately or saves the encrypted file and the necessary keys for future usage.

### File decryption

In order to decrypt the file the client performs the following procedure

* Check validity period of the stored keys. If validity period is over then delete public keys Ei (look further for the details) and request for the new set of keys.
* Choose the pair of *Di* and corresponding to the group identificator Sj and decrypt the key K
* Decrypt the file with the key K

**По поводу отправки пользовательского ключа:**

ПО-большому счету, да здесь, есть пробел, если мы так часто отправляем пользователю его ключ, который является секретным, то мы должны его отправлять по защищенному каналу, то есть как-то минимально шифровать. Это шифрование мы можем либо привязать к паролю (собирать часть ключа для отправки пользовательского ключа из пароля) либо использовать старый ключ:

1. Клиент на устройстве не убивает старый пользовательский ключ, а только публичный ключ, по истечении срока годности. Без публичного ключа пользователь все равно не получит доступ к файлу.
2. Сервер при отправке нового пользовательского ключа шифрует его старым публичным ключом. (или например он перед убиванием старого публичного ключа шифрует им новый пользовательский ключ).
3. Клиент, получив пользовательский ключ, расшифровывает его и подставляет взамен старого автоматически.

Мне кажется это самый естественный способ. Второй способ – из пароля получать ключ AES (хотя бы просто блоками), шифровать и отправлять клиенту. Тогда при получении ключа пользователю, который правильно ввел пароль будет предоставлен этот самый ключ.

**По поводу пары ключей:**

Пусть сервер сразу отправляет значение e(Ei,Di). Тогда:

1. Это значение – все равно что симметричный ключ. Его нужно отправлять защищенным способом. См. выше
2. Если сервер хочет проверить свежесть ключа клиента, то ему придется хранить это же значение e(Ei,Di) (в защищенном виде) а не только публичный ключ (в открытом виде). Это порождает дополнительные сложности.
3. Каждый раз при смене пары ключей в этом случае клиенту ничего не надо менять. Обновление ключей теряет смысл, нужно придумывать что то другое. Если эты помнишь, при периодической смене ключей у нас это значение не меняется, однако к нему у пользователя нет доступа без его удаленного ключика.
4. Да конечно, злонамеренный пользователь может всегда это значение после получения своей пары ключей и расшифрования где то на крайняк сохранить и потом самолично получать доступ к файлу. Но злонамеренный пользователь помимо этого может еще много чего….. Пока это ситуацию предлагаю отложить. От нее поможет только отказ от концепции хранения ключа к файлу, и принудительное обновление всех ключей, включая сеансовые, перезакачка файлов и проч.
5. В данной версии предлагаю такую активность пользователей пока просто отслеживать (мы как-то обсуждали о том что клиент мониторит активность пользователя и его действия, наверняка запуски функции расшифрования можно отследить при помощи айпи и если число расшифрований файла не соответствует числу расшифрований через клиента – то что-то не так).
6. Хорошо бы просто иметь аварийную функцию удаления всех ключей и сообщения о том, что пользователь должен все перезакачать в принудительном порядке. Как бы комендантский час.