Cryptographic RBAC Compiler

Second Sprint

29/10/18 - 11/11/18

FONDAZIONE BRUNO KESSLER

Layout of the slides by Coman Catalin Andrei

Agenda

- Recap
- Sprint Backlog
- User Stories
- Design
- Implementation
- Testing
- Review
- Questions

Sprint Backlog: start by adding **User** and **Role** classes to the system. Generate and distribute **Keys**. No DB (integration with SecurePG will come later). **Tuple** creation and signing process.

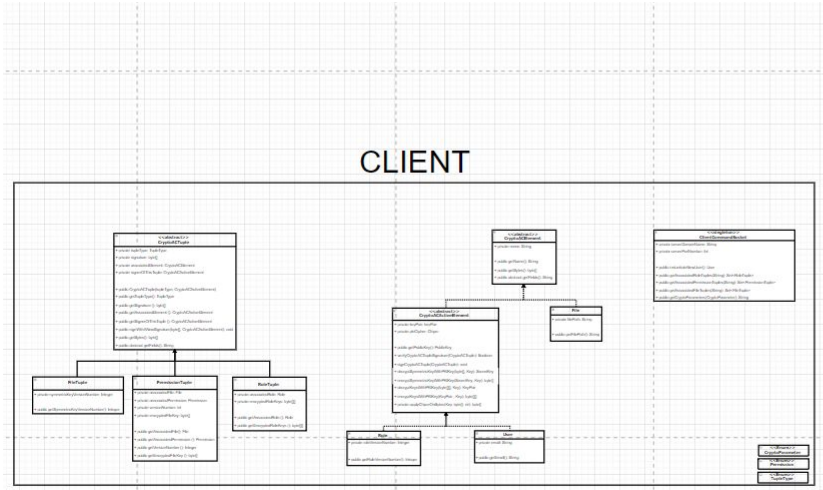
Related components:

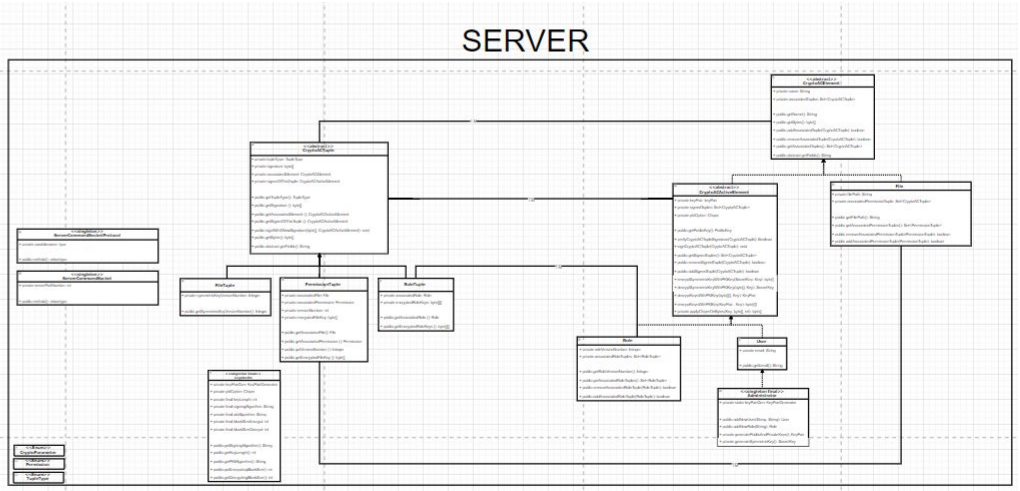
- CryptoUtil (generatePublicAndPrivateKeys, encrypt/decryptKeysWithPKIKey, ...)
- User (getUsername, getEmail, getPublicKey)
- CommandInterpreter (addNewUser, addNewRole)
- Administrator (addNewUser, addNewRole)
- Role (getRoleName, getPublicKey, getRoleVersioNumber)
- TupleAssociableElement
- ShareUtil (shareKeysWithUser)
- Tuple (and subclasses)

Sprint Backlog

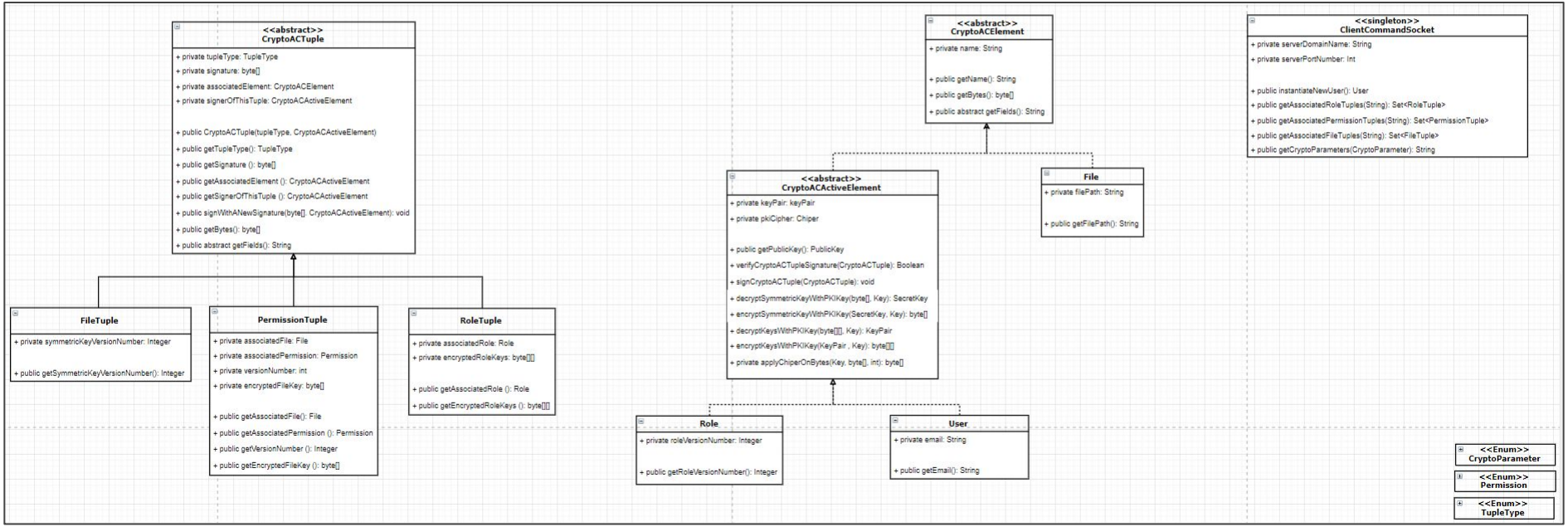
- Refactoring of the system architecture in **two subsystems**: client and server (class diagram, user stories, test-cases, UML-sequence diagrams, packages, modifiers of variables and methods, ...)
- Design and implementation of Java classes for **communication** between these two parts (through **sockets**)
- Creation of a **Keystore** (server side for the admin, client side for the users) for managing the keys
- Add **symmetric keys** to the environment (generation,encrypt, decrypt...)

Created two subsystems - Client and Server





Client


<<abstract>>
CryptoACTuple
+ private tupleType: TupleType
+ private signature: byte[]
+ private associatedElement: CryptoACElement
+ private signerOfThisTuple: CryptoACActiveElement
+ public CryptoACTuple(tupleType, CryptoACActiveElement)
+ public getTupleType(): TupleType
+ public getSignature (): byte[]
+ public getAssociatedElement (): CryptoACActiveElement
+ public getSignerOfThisTuple (): CryptoACActiveElement
+ public signWithANewSignature(byte[], CryptoACActiveElement): void
+ public getBytes(): byte[]
+ public abstract getFields(): String
FileTuple
+ private symmetricKeyVersionNumber: Integer
+ public getSymmetricKeyVersionNumber(): Integer
PermissionTuple
+ private associatedFile: File
+ private associatedPermission: Permission
+ private versionNumber: int
+ private encryptedFileKey: byte[]
+ public getAssociatedFile(): File
+ public getAssociatedPermission (): Permission
+ public getVersionNumber (): Integer
+ public getEncryptedFileKey (): byte[]
RoleTuple
+ private associatedRole: Role
+ private encryptedRoleKeys: byte[][]
+ public getAssociatedRole (): Role
+ public getEncryptedRoleKeys (): byte[][]
<<abstract>>
CryptoACElement
+ private name: String
+ public getName(): String
+ public getBytes(): byte[]
+ public abstract getFields(): String
<<abstract>>
CryptoACActiveElement
+ private keyPair: keyPair
+ private pkiCipher: Chiper
+ public getPublicKey(): PublicKey
+ verifyCryptoACTupleSignature(CryptoACTuple): Boolean
+ signCryptoACTuple(CryptoACTuple): void
+ decryptSymmetricKeyWithPKIKey(byte[], Key): SecretKey
+ encryptSymmetricKeyWithPKIKey(SecretKey, Key): byte[]
+ decryptKeysWithPKIKey(byte[][], Key): KeyPair
+ encryptKeysWithPKIKey(KeyPair , Key): byte[][]
+ private applyChiperOnBytes(Key, byte[], int): byte[]
File
+ private filePath: String
+ public getFilePath(): String
Role
+ private roleVersionNumber: Integer
+ public getRoleVersionNumber(): Integer
User
+ private email: String
+ public getEmail(): String
<<singleton>>
ClientCommandSocket
+ private serverDomainName: String
+ private serverPortNumber: Int
+ public instantiateNewUser(): User
+ public getAssociatedRoleTuples(String): Set<RoleTuple>
+ public getAssociatedPermissionTuples(String): Set<PermissionTuple>
+ public getAssociatedFileTuples(String): Set<FileTuple>
+ public getCryptoParameters(CryptoParameter): String
<<Enum>>
CryptoParameter
<<Enum>>
Permission
<<Enum>>
TupleType

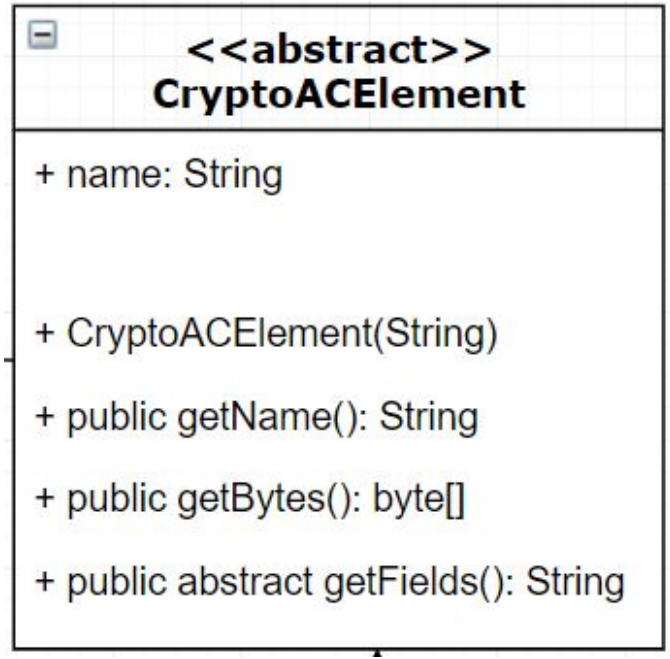
From Javadoc:

"Class that defines an element in the system, thus involved in the Access Control process (i.e., assignable to a crypto AC tuple).
More concrete examples are users, files and roles."

`GetFields` returns all the fields of the object

`GetBytes` returns the fields as a byte array

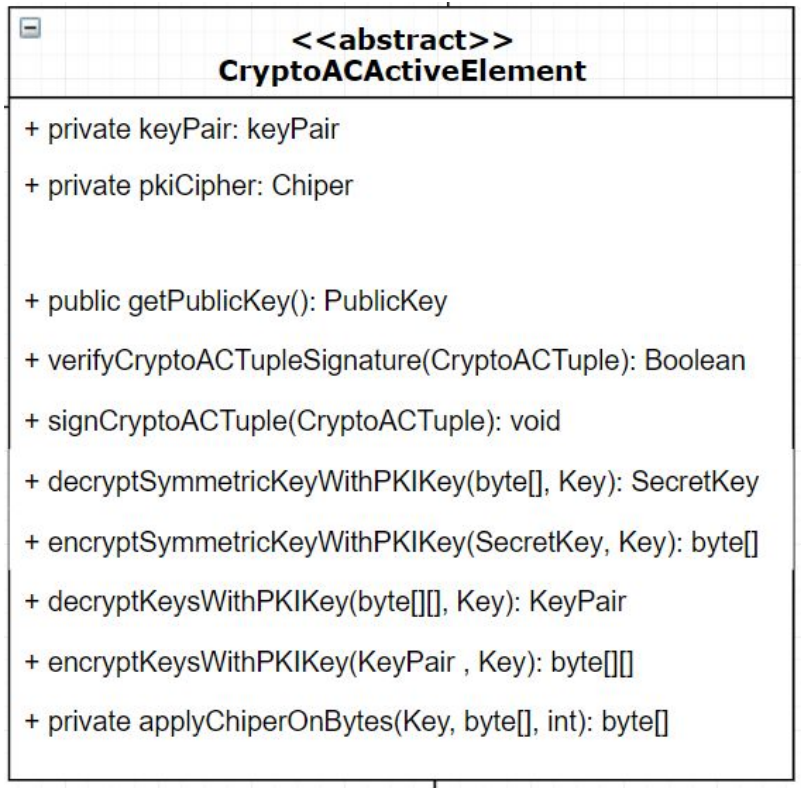
Rename in RBACACElement?



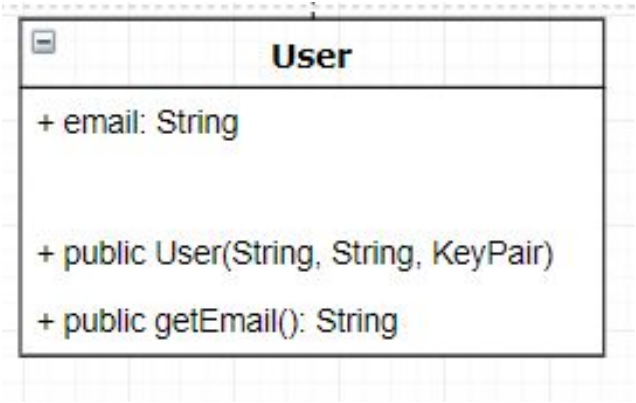
From Javadoc:

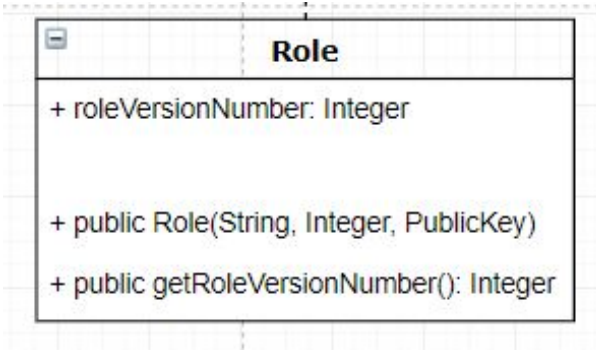
“Class that defines an active element in the Access Control system. An active element has cryptographic keys, thus the ability to encrypt, decrypt, sign.... ”

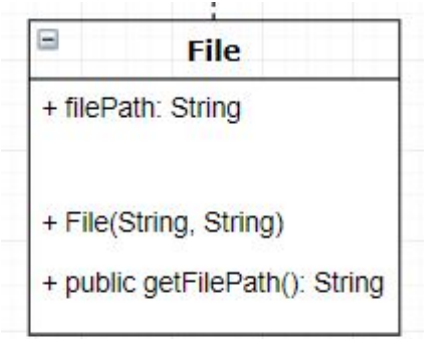
Superclass is CryptoACElement.
Subclasses are User, Role and Administrator



Client-side, these classes are just something more than wrappers for data

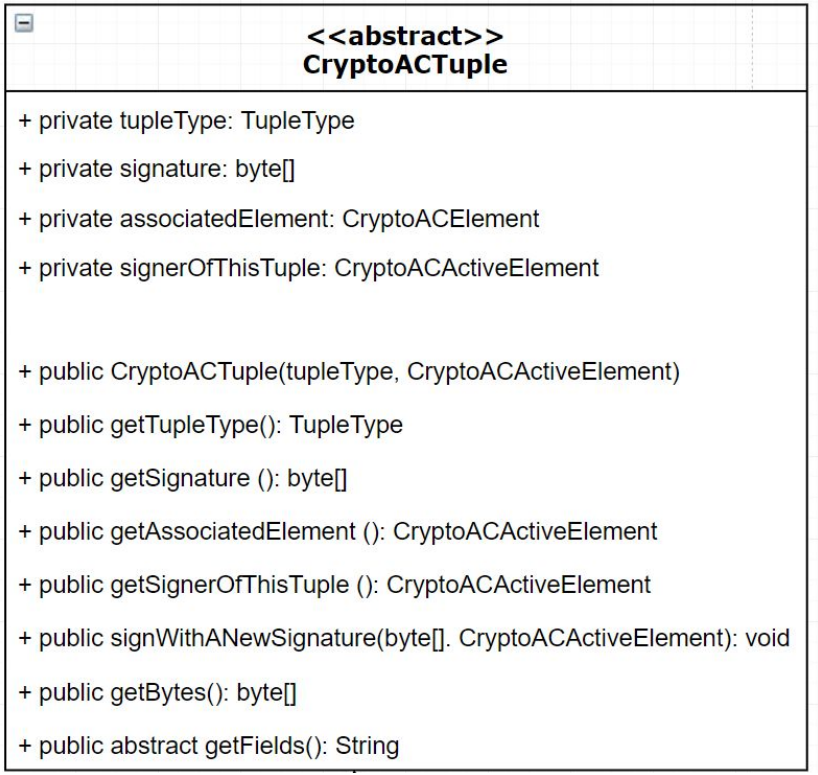




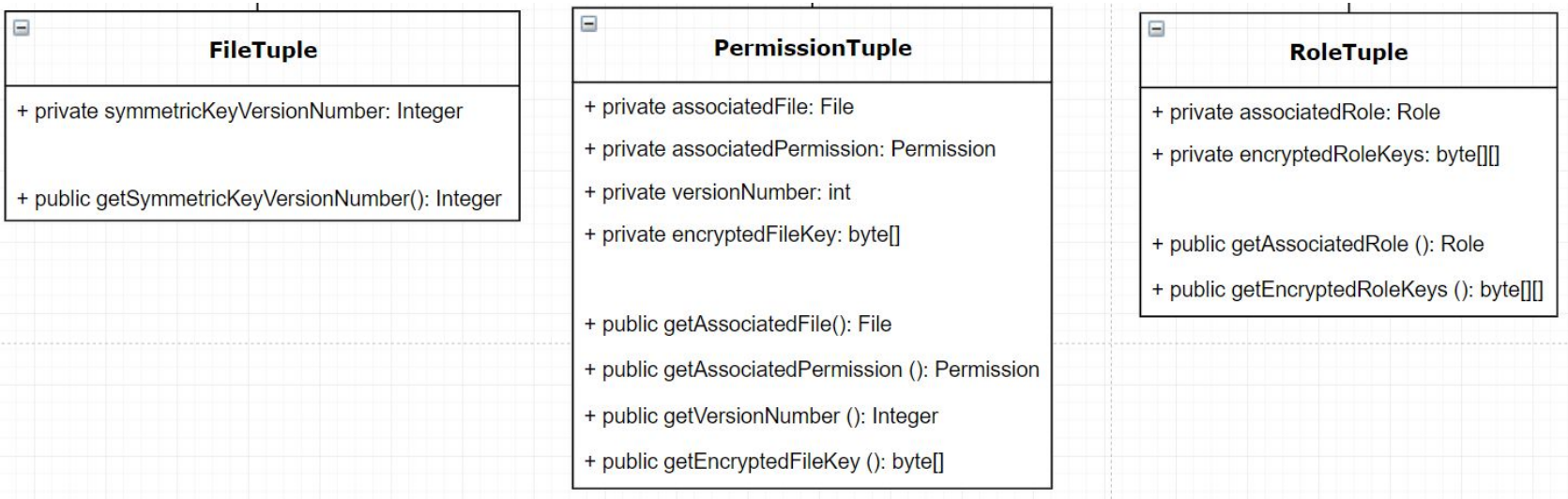


From Javadoc:

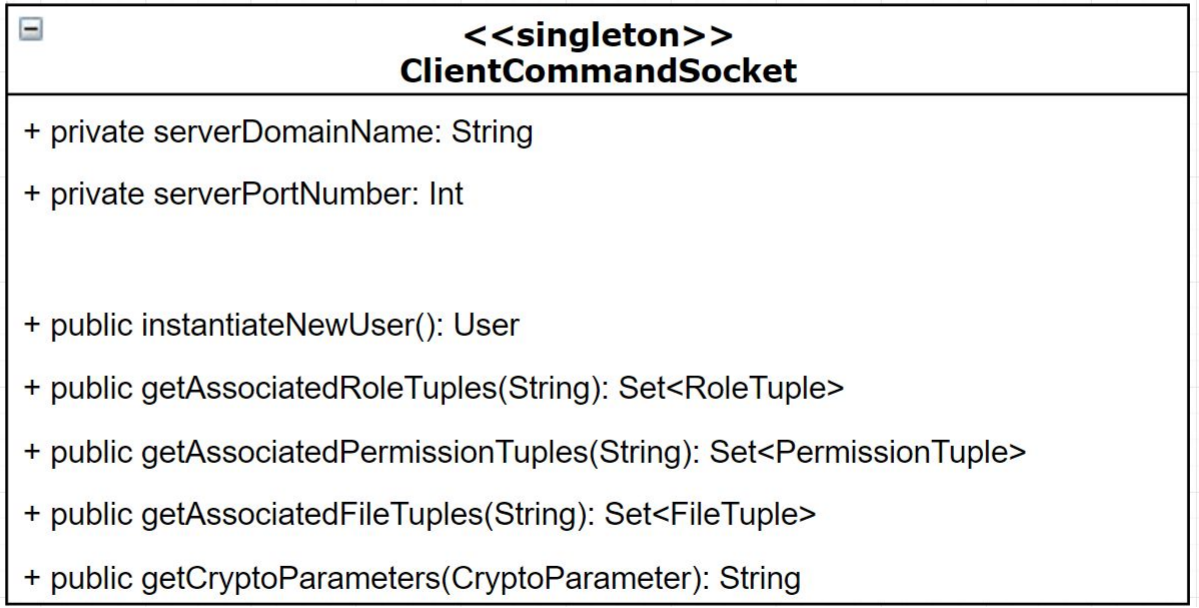
"General class for tuples in the crypto AC system. A tuple usually **binds two elements** (thus a couple of elements), the associatedElement (CryptoACElement, that may be a User, a Role or a File) and a tuple-specific element. A tuple is supposed to always have a signer (CryptoACActiveElement) and a signature"



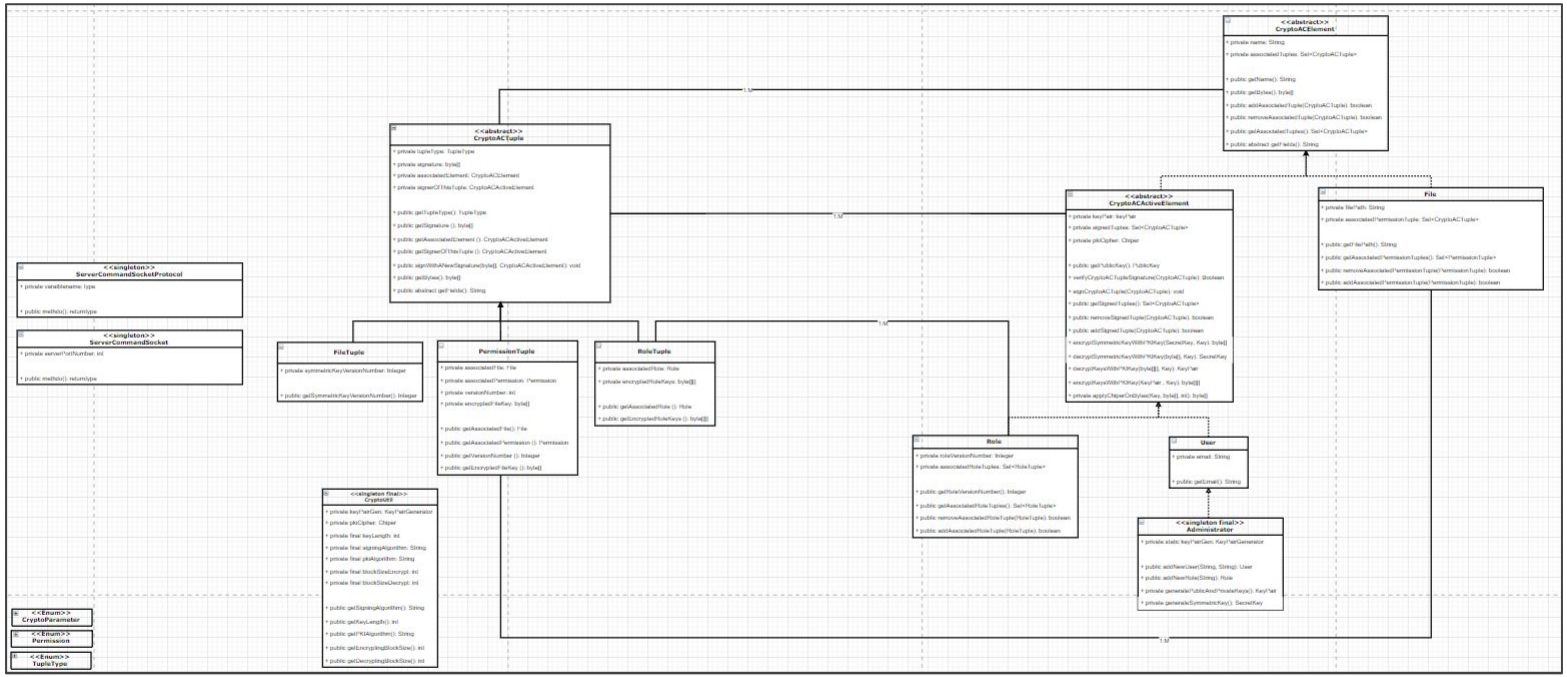
Specification of the general CryptoACTuple object



Java class handling communication to the server



Server



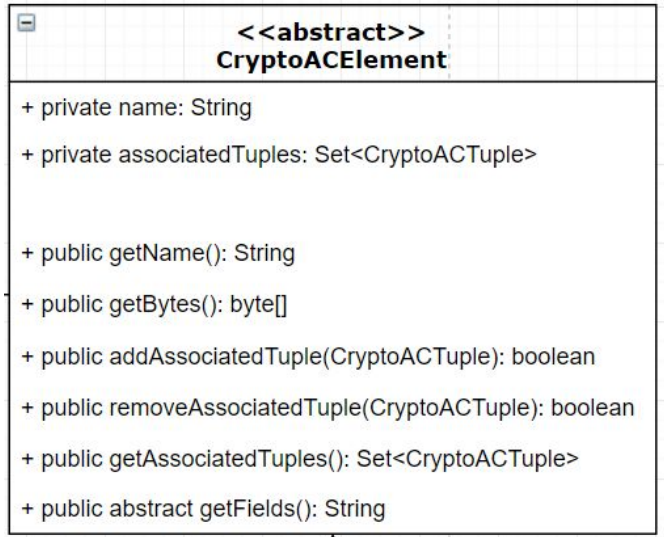
Server-Side we have almost a specular structure, **but** with some differences

What differs?

Here we have a list of associated tuples that acts as a sort of a cache (and related methods)

Actually the list is derived from the UML diagram (so it should be implemented also in the client), but it was not the case to have it in the client (complexity, updating issues, no real utility, ...). Just keep it in the server.

Same for CryptoACActiveElement, Role, File.

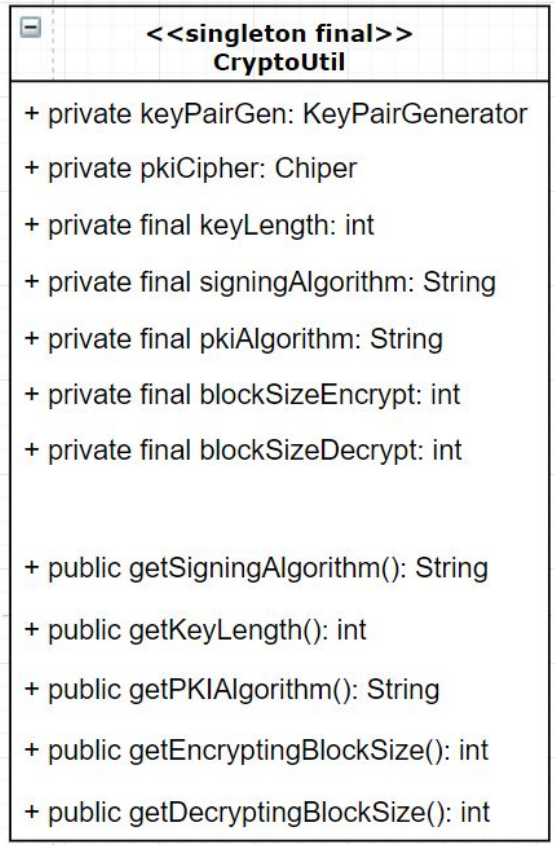


From Javadoc:

“This class is used to group all cryptographic related parameters (pki algorithm, key length, signing algorithm, ...)”

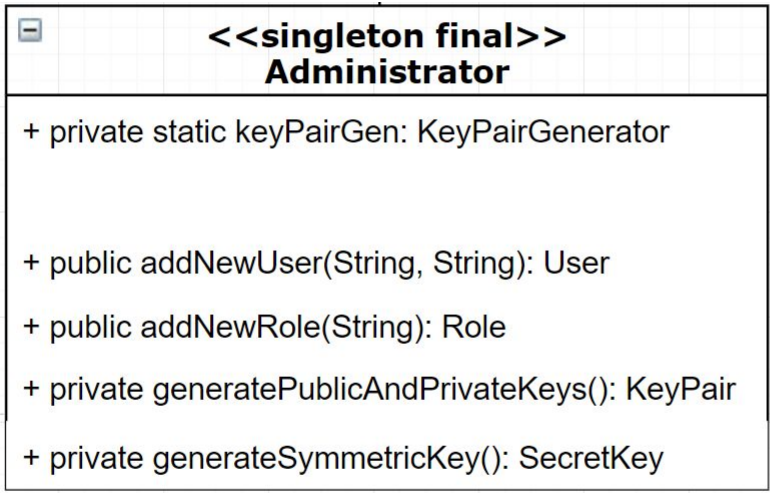
Thus it is a collection of the parameters used by crypto. This way there is a unique point for updating these values

In the future it may be the case to create a config file (?)



Subclass of Users. From Javadoc:

"This class takes the role of the admin of the system (it is of course a singleton). The admin can create and add new users and roles to the system, and generate PKI (?) and symmetric keys."



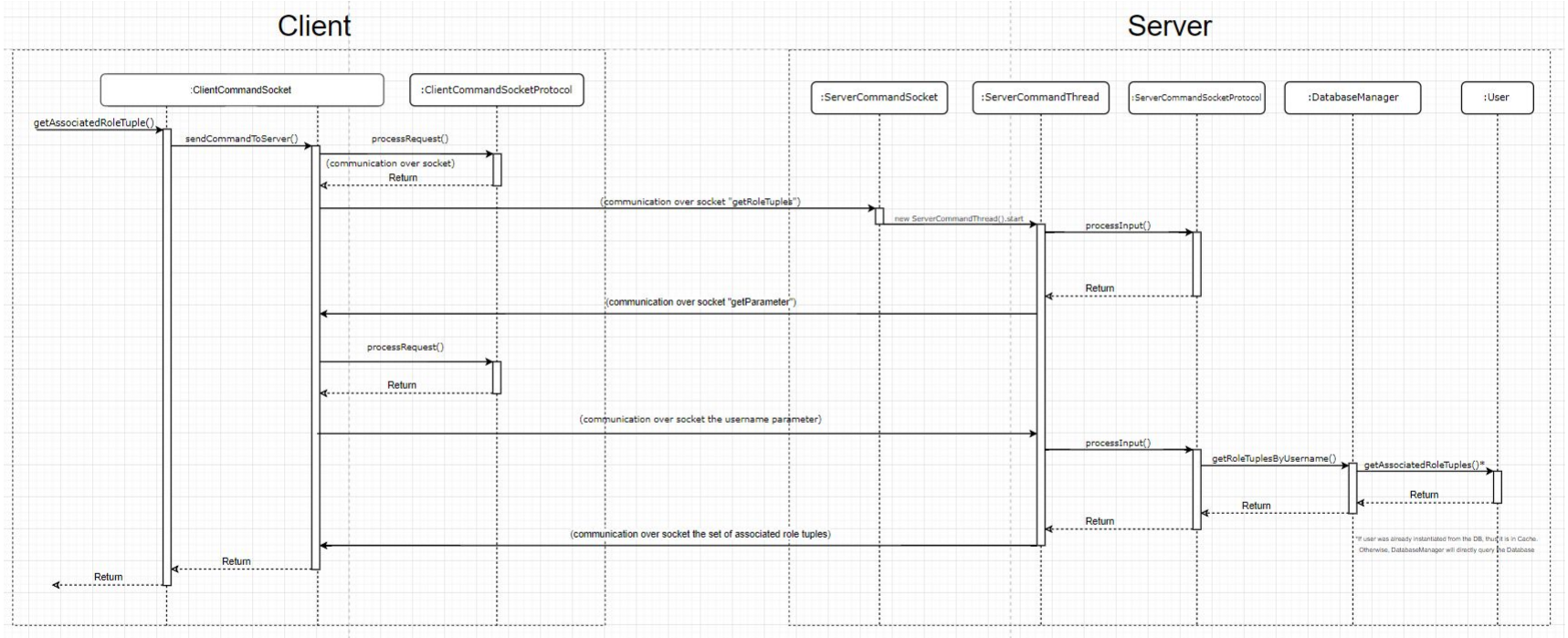
Of course, server-side we also have classes for managing the DB (but must wait for the integration with Secure-PG)

4. As a **User**, I want to get the tuples or the roles associated with me to decrypt the roles' keys.

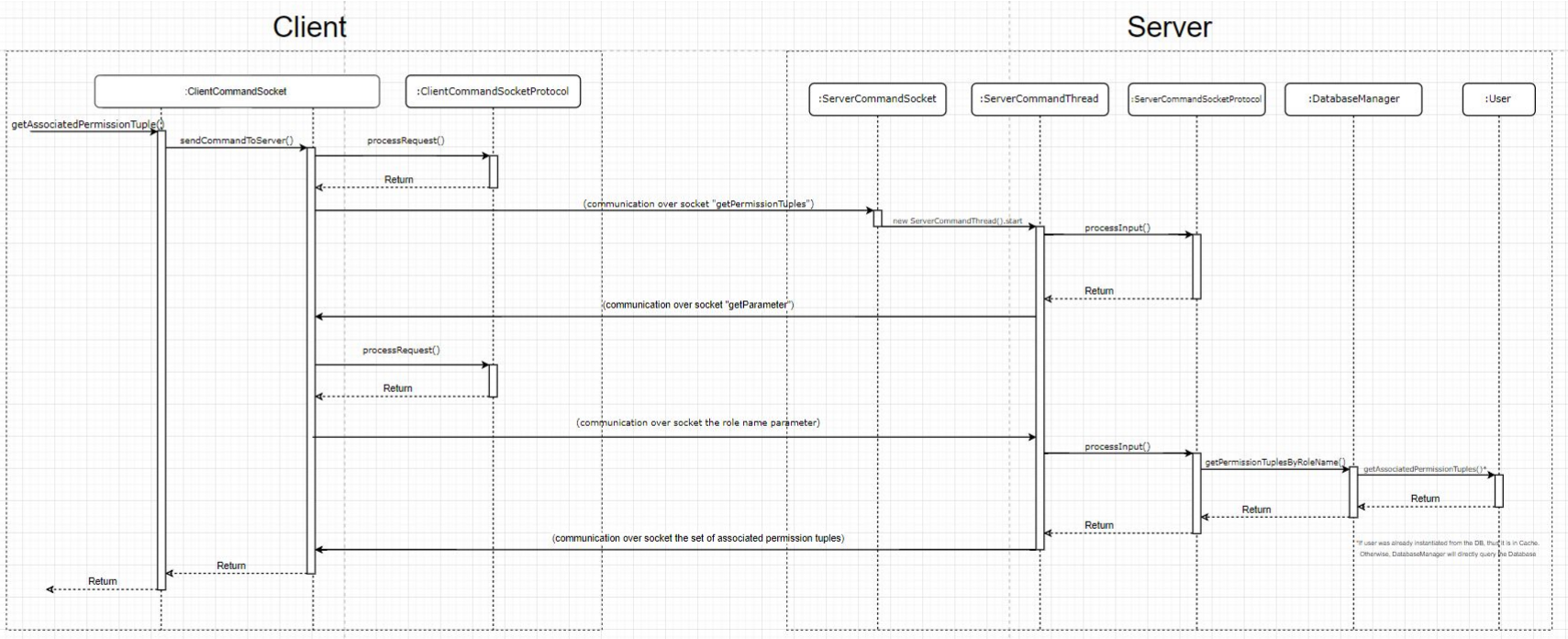
5. As a **Role**, I want to get the tuples or the permissions associated with me in order to later decrypt the symmetric key.

6. As a **User**, I want to get the cryptographic parameters (e.g. the Key Length) from the Server in order to later use them.

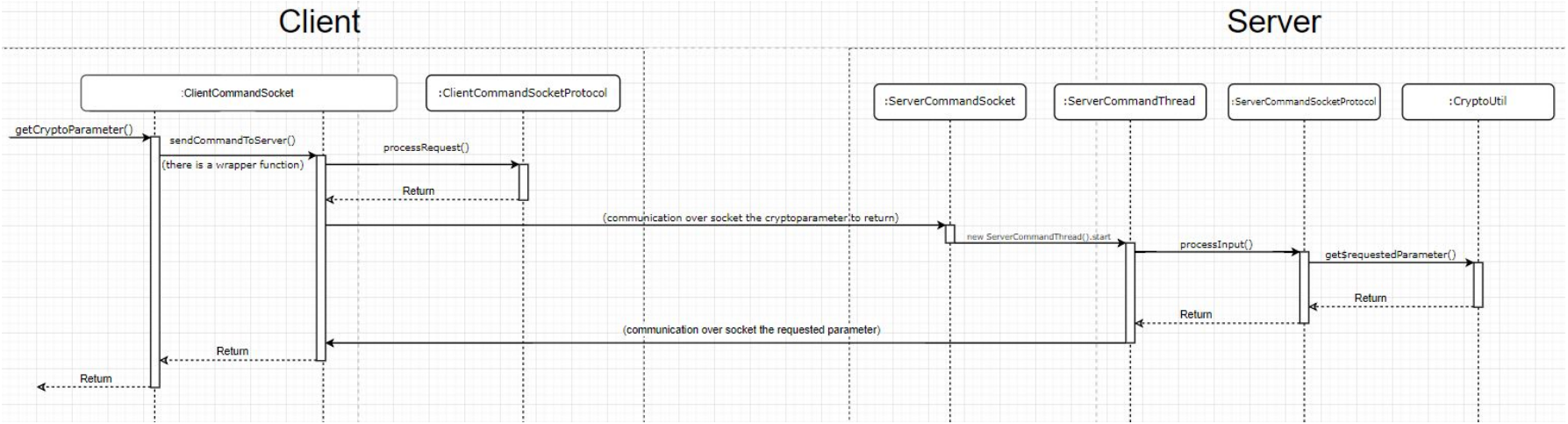
As a **User**, I want to get the tuples or the roles associated with me to decrypt the roles' keys



As a Role, I want to get the tuples or the permissions associated with me in order to later decrypt the symmetric key (structure is same UML as before).



As a User, I want to get the cryptographic parameters (e.g. the Key Length) from the Server in order to later use them.



Communication Client-Server

- Server handles several requests via **threads**
- Follows a **protocol** and a defined set of **commands** (ProtocolCommands)
- Has defined **status** and **errors** (ProtocolStatus)

(see code for details)

Testing

Wrote tests for:

- ProtocolStatus Enumerator
- ProtocolCommands Enumerator
- Client and Server Communication Socket
- Client and Server Communication Protocol

(see code for details)

Communication code (sockets) is now complete

but took longer than expected.

Functionalities not implemented during the sprint:

- Creation of a Keystore (server side for the admin, client side for the users) for storing keys, but it needs certificates... Alternatives?
- Add symmetric keys to the environment (generation,encrypt, decrypt...)

Questions

- The **KeyGen** function requires the **msk** parameter (master secret key by the administrator). Why is it necessary?
  It is necessary only with **IBE/IBS**

- Is it preferable to implement the RBAC0 algorithm using **IBE/IBS** or using **PKI**? And why so? (keep in mind that is a prototype)
  No differences (**PKI** more flexible and quick)

- Why can't the **keys** be generated by the Users themselves?
  In **IBE/IBS** the **msk** is needed. In **PKI** is possible and recommended

- Where should we **store PKI keys**?
  Not important for now

Thanks

Layout of the slides by Coman Catalin Andrei