Cryptographic RBAC Compiler

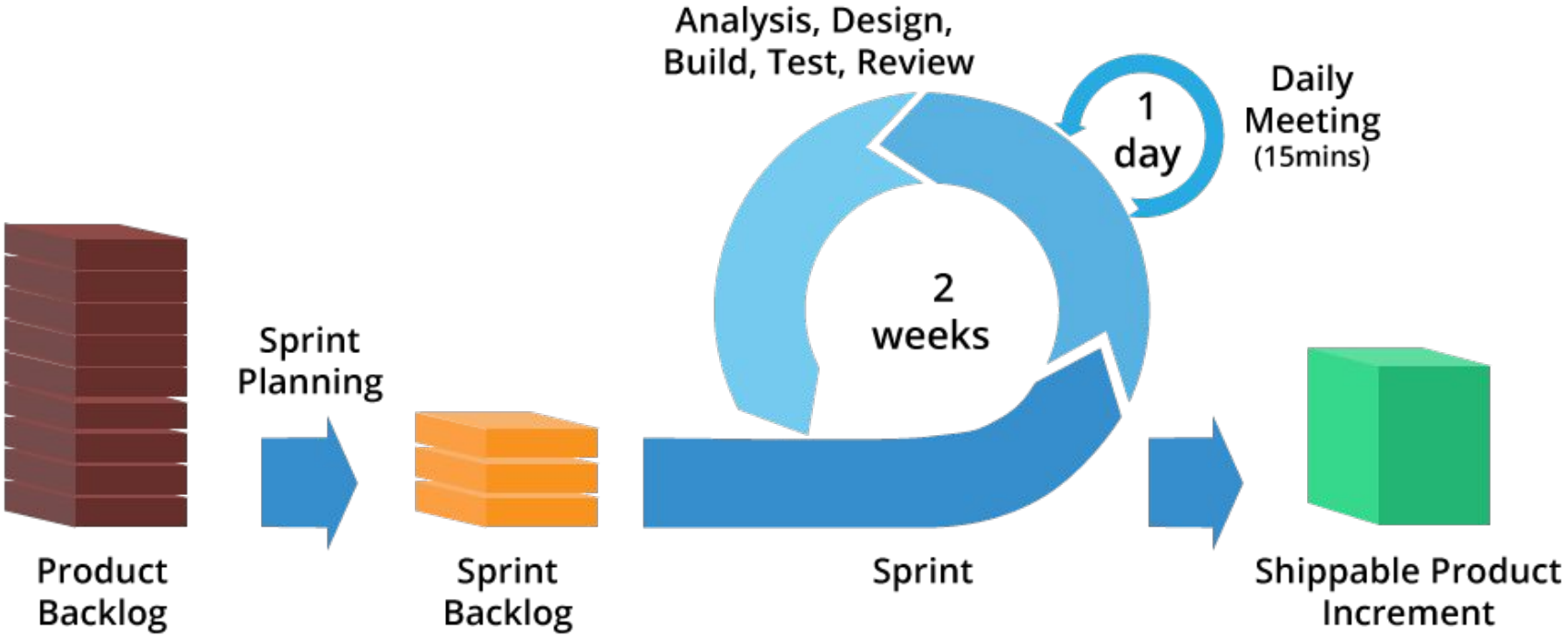
General Setup

Layout of the slides by Coman Catalin Andrei

Agenda

- Adopted Process
- Product Backlog
- Next Two Sprints
- Questions

Adopted Process


Analysis, Design,
Build, Test, Review
1
day
Daily
Meeting
(15mins)
2
weeks
Sprint
Planning
Product
Backlog
Sprint
Backlog
Sprint
Shippable Product
Increment

- UML as designing language
- Java (for later integration with SecurePG)
- IntelliJ as Java IDE
- MySQL as database for permanent storage

Product Backlog

Product Backlog

$addU(u)$
- Add u to USERS
- Generate IBE private key $k_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(u)$ and IBS private key $s_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(u)$ for the new user u
- Give k_u and s_u to u over private and authenticated channel

$delU(u)$
- For every role r that u is a member of:
  - $revokeU(u, r)$

$addP_u(fn, f)$
- Generate symmetric key $k \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$
- Send $\langle F, fn, 1, \mathbf{Enc}_{SU}^{\mathbf{Sym}}(f), u, \mathbf{Sign}_u^{\mathbf{IBS}} \rangle$ and $\langle FK,\ SU,\ \langle fn, RW \rangle, 1, \mathbf{Enc}_u^{\mathbf{IBE}}(k), u, \mathbf{Sign}_u^{\mathbf{IBS}} \rangle$ to R.M.
- The R.M. receives $\langle F, fn, 1, c, u, sig \rangle$ and $\langle FK,\ SU,\ \langle fn, RW \rangle, 1, c', u, sig' \rangle$ and verifies that the tuples are well-formed and the signatures are valid, i.e., $\mathbf{Ver}_u^{\mathbf{IBS}}(\langle F, fn, 1, c, u \rangle, sig) = 1$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ SU,\ \langle fn, RW \rangle, 1, c', u \rangle, sig') = 1$.
- If verification is successful, the R.M. adds $\langle fn, 1 \rangle$ to FILES and stores $\langle F, fn, 1, c, u, sig \rangle$ and $\langle FK,\ SU,\ \langle fn, RW \rangle, 1, c', u, sig' \rangle$

$delP(fn)$
- Remove $\langle fn, v_{fn} \rangle$ from FILES
- Delete $\langle F, fn, -, -, -, - \rangle$ and all $\langle FK, -, \langle fn, - \rangle, -, -, -, - \rangle$

$addR(r)$
- Add $\langle r, 1 \rangle$ to ROLES
- Generate IBE private key $k_{\langle r,1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(\langle r, 1 \rangle)$ and IBS private key $s_{\langle r,1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(\langle r, 1 \rangle)$ for role $\langle r, 1 \rangle$
- Send $\langle RK,\ SU,\ \langle r, 1 \rangle, \mathbf{Enc}_{SU}^{\mathbf{IBE}}(k_{\langle r,1 \rangle}), s_{\langle r,1 \rangle} \rangle, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.

$delR(r)$
- Remove $\langle r, v_r \rangle$ from ROLES
- Delete all $\langle RK, -, \langle r, v_r \rangle, -, - \rangle$
- For all permissions $p = \langle fn, op \rangle$ that r has access to:
  - $revokeP(r, \langle fn, RW \rangle)$

$assignU(u, r)$
- Find $\langle RK,\ SU,\ \langle r, v_r \rangle, c, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle RK,\ SU,\ \langle r, v_r \rangle, c \rangle, sig) = 1$
- Decrypt keys $(k_{\langle r,v_r \rangle}, s_{\langle r,v_r \rangle}) = \mathbf{Dec}_{kSU}^{\mathbf{IBE}}(c)$
- Send $\langle RK,\ u,\ \langle r, v_r \rangle, \mathbf{Enc}_u^{\mathbf{IBE}}(k_{\langle r,v_r \rangle}), s_{\langle r,v_r \rangle} \rangle, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.

$revokeU(u, r)$
- Generate new role keys $k_{\langle r,v_r+1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(\langle r, v_r + 1 \rangle)$, $s_{\langle r,v_r+1 \rangle} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(\langle r, v_r + 1 \rangle)$
- For all $\langle RK,\ u',\ \langle r, v_r \rangle, c, sig \rangle$ with $u' \neq u$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle RK,\ u',\ \langle r, v_r \rangle, c \rangle, sig) = 1$:
  - Send $\langle RK,\ u',\ \langle r, v_r + 1 \rangle, \mathbf{Enc}_{u'}^{\mathbf{IBE}}(k_{\langle r,v_r+1 \rangle}, s_{\langle r,v_r+1 \rangle}), \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
- For every fn such that there exists $\langle FK,\ \langle r, v_r \rangle, \langle fn, op \rangle, v_{fn}, c, SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, p, v_{fn}, c, SU \rangle, sig) = 1$:
  - For every $\langle FK,\ \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU \rangle, sig) = 1$:
    - Decrypt key $k = \mathbf{Dec}_{k_{\langle r,v_r \rangle}}^{\mathbf{IBE}}(c')$
    - Send $\langle FK,\ \langle r, v_r + 1 \rangle, \langle fn, op' \rangle, v, \mathbf{Enc}_{\langle r,v_r+1 \rangle}^{\mathbf{IBE}}(k), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
  - Generate new symmetric key $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$ for p
  - For all $\langle FK,\ id,\ \langle fn, op' \rangle, v_{fn}, c'', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ id,\ \langle fn, op' \rangle, v_{fn}, c'', SU \rangle, sig) = 1$:
    - Send $\langle FK,\ id,\ \langle fn, op' \rangle,\ v_{fn} + 1,\ \mathbf{Enc}_{id}^{\mathbf{IBE}}(k'),\ SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
  - Increment v_{fn} in FILES, i.e., set $v_{fn} := v_{fn} + 1$
- Increment v_r in ROLES, i.e., set $v_r := v_r + 1$
- Delete all $\langle RK, -, \langle r, v_r \rangle, -, - \rangle$
- Delete all $\langle FK,\ \langle r, v_r \rangle, -, -, -, -, - \rangle$

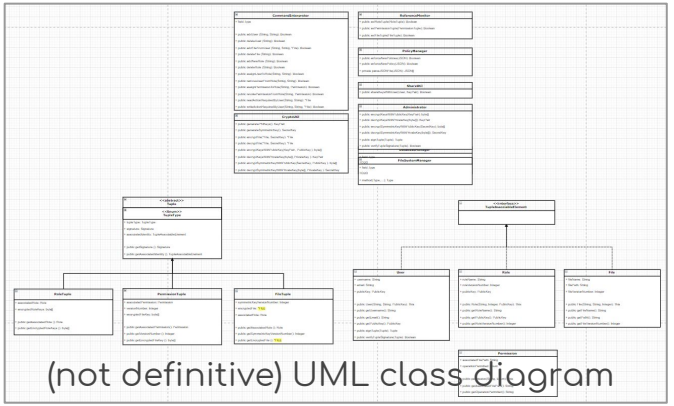
$assignP(r, \langle fn, op \rangle)$
- For all $\langle FK,\ SU,\ \langle fn, RW \rangle, v, c, id, sig \rangle$ with $\mathbf{Ver}_{id}^{\mathbf{IBS}}(\langle FK,\ SU,\ \langle fn, RW \rangle, v, c, id \rangle, sig) = 1$:
  - If this adds Write permission to existing Read permission, i.e., $op = RW$ and there exists $\langle FK,\ \langle r, v_r \rangle, \langle fn, \mathbf{Read} \rangle, v, c', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU \rangle, sig) = 1$:
    - Send $\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v, c', SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
    - Delete $\langle FK,\ \langle r, v_r \rangle, \langle fn, \mathbf{Read} \rangle, v, c', SU, sig \rangle$
  - If the role has no existing permission for the file, i.e., there does not exist $\langle FK,\ \langle r, v_r \rangle, \langle fn, op' \rangle, v, c', SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, op' \rangle, v, c, SU \rangle, sig) = 1$:
    - Decrypt key $k = \mathbf{Dec}_{kSU}^{\mathbf{IBE}}(c)$
    - Send $\langle FK,\ \langle r, v_r \rangle, \langle fn, op \rangle, v, \mathbf{Enc}_{\langle r,v_r \rangle}^{\mathbf{IBE}}(k), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.

$revokeP(r, \langle fn, op \rangle)$
- If $op =$ Write:
  - For all $\langle FK,\ \langle r, v_r \rangle,\ \langle fn, RW \rangle,\ v,\ c,\ SU,\ sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v, c, SU \rangle, sig) = 1$:
    - Send $\langle FK,\ \langle r, v_r \rangle, \langle fn, \mathbf{Read} \rangle, v, c, SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
    - Delete $\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v, c, SU, sig \rangle$
- If $op =$ RW:
  - Delete all $\langle FK,\ \langle r, v_r \rangle, \langle fn, - \rangle, -, -, - \rangle$
  - Generate new symmetric key $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$
  - For all $\langle FK,\ r',\ \langle fn, op' \rangle, v_{fn}, c, SU, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ r',\ \langle fn, op' \rangle, v, c, SU \rangle, sig) = 1$:
    - Send $\langle FK,\ r',\ \langle fn, op' \rangle,\ v_{fn} + 1,\ \mathbf{Enc}_{r'}^{\mathbf{IBE}}(k'),\ SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$ to R.M.
  - Increment v_{fn} in FILES, i.e., set $v_{fn} := v_{fn} + 1$

$read_u(fn)$
- Find $\langle F, fn, v, c, id, sig \rangle$ with valid ciphertext c and valid signature sig, i.e., $\mathbf{Ver}_{id}^{\mathbf{IBS}}(\langle F, fn, 1, c, id \rangle, sig) = 1$
- Find a role r such that the following hold:
  - u is in role r, i.e., there exists $\langle RK,\ u,\ \langle r, v_r \rangle, c', sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle RK,\ u,\ \langle r, v_r \rangle, c' \rangle, sig) = 1$
  - r has read access to version v of fn, i.e., there exists $\langle FK,\ \langle r, v_r \rangle, \langle fn, op \rangle, v, c'', SU, sig' \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, op \rangle, v, c'', SU \rangle, sig') = 1$
- Decrypt role key $k_{\langle r,v_r \rangle} = \mathbf{Dec}_{k_u}^{\mathbf{IBE}}(c')$
- Decrypt file key $k = \mathbf{Dec}_{\langle r,v_r \rangle}^{\mathbf{IBE}}(c'')$
- Decrypt file $f = \mathbf{Dec}_k^{\mathbf{Sym}}(c)$

$write_u(fn, f)$
- Find a role r such that the following hold:
  - u is in role r, i.e., there exists $\langle RK,\ u,\ \langle r, v_r \rangle, c, sig \rangle$ with $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle RK,\ u,\ \langle r, v_r \rangle, c \rangle, sig) = 1$
  - r has write access to the newest version of fn, i.e., there exists $\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v_{fn}, c', SU, sig' \rangle$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v, c', SU \rangle, sig') = 1$
- Decrypt role key $k_{\langle r,v_r \rangle} = \mathbf{Dec}_{k_u}^{\mathbf{IBE}}(c)$
- Decrypt file key $k = \mathbf{Dec}_{\langle r,v_r \rangle}^{\mathbf{IBE}}(c')$
- Send $\langle F, fn, v_{fn}, \mathbf{Enc}_k^{\mathbf{Sym}}(f), \langle r, v_r \rangle, \mathbf{Sign}_{\langle r,v_r \rangle}^{\mathbf{IBS}} \rangle$ to R.M.
- The R.M. receives r and $\langle F, fn, v, c'', \langle r, v_r \rangle, sig'' \rangle$ and verifies the following:
  - The tuple is well-formed with $v = v_{fn}$
  - The signature is valid, i.e., $\mathbf{Ver}_{\langle r,v_r \rangle}^{\mathbf{IBS}}(\langle F,\ fn,\ v,\ c'', \langle r, v_r \rangle \rangle, sig'') = 1$
  - r has write access to the newest version of fn, i.e., there exists $\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v_{fn}, c', SU, sig' \rangle$ and $\mathbf{Ver}_{SU}^{\mathbf{IBS}}(\langle FK,\ \langle r, v_r \rangle, \langle fn, RW \rangle, v_{fn}, c', SU \rangle, sig') = 1$
- If verification is successful, the R.M. replaces $\langle F, fn, -, -, -, - \rangle$ with $\langle F, fn, v_{fn}, c'', \langle r, v_r \rangle, sig'' \rangle$

Implementation of RBAC0 using IBE and IBS



(not definitive) UML class diagram

- **CommandInterpreter.revokePermissionFromRole (role name, permission)**
  If is it a RW permission:
  - Fetch all tuples <FK, (role, role version number), (filename, RW), version, cyphertext, SU, sig>. For all tuples:
    - Call **Administrator.verifyTupleSignature**
    - Create tuple <FK, (role, role version number), (filename, R), version, cyphertext, SU> and sign it through **Administrator.signTuple**
    - Call **ReferenceMonitor.setPermissionTuple** and pass it the tuple
    - Delete the old tuple <FK, (role, role version number), (filename, RW), version, cyphertext, SU, sig>

  Else:
  - Delete all tuples <FK, (role, role version number), (filename, -), -, -, -, ->
  - Generate new symmetric key K' = **Crypto.Util.generateSymmetricKey**
  - Fetch all tuples <FK, (r', role' version number), (filename, operation'), filename version, cyphertext, SU, sig>. For all tuples:
    - Call **Administrator.verifyTupleSignature**
    - Obtain cyphertext' = **CryptoUtil.encryptSymmetricKeyWithPublicKey** with the key of the identity
    - Create tuple <FK, (role', role' version number), (filename, operation'), version +1, cyphertext', SU> and sign it through **Administrator.signTuple**
    - Call **ReferenceMonitor.setPermissionTuple** and pass it the tuple
  - Increment the version number in the FILES SQL table

- **CommandInterpreter.readActionRequiredByUser (username, filename)**
  Find a File Tuple <F, filename, version number, cyphertext, id, sig>

(not definitive) Description of the functions

Classes

- PolicyManager (Receive and parse policies)
- ReferenceMonitor (DAO)
- DatabaseManager (DAO implementation - DB)
- FileSystemManager (DAO implementation - FS)
- Administrator (RBAC administrator)
- CommandInterpreter (Handle Access Control)
- ShareUtil (Share keys with users)
- CryptoUtil (Generate keys, ...)

- Permission (pair action,file)
- TupleAssociableElement (abstract for:)
  - User (User object)
  - Role (User object)
  - File (User object)
- Tuple (abstract for:)
  - FileTuple ("F" file tuple)
  - RoleTuple ("RK" role tuple)
  - PermissionTuple ("FK" permission tuple)

Next Two Sprints

- Choose the **Sprint Backlog** from the Product Backlog
- Specify the **Requirements** through user stories
- Create Sequence UML Diagram (**Design**)
- Create tests from user stories (optional)
- **Implementation**
- **Testing**
- **Maintenance** and refactoring
- **Sprint Review**

Next Two Sprints

First: 15/10 - 28/10

Sprint Backlog: start by adding **User** and **Role** classes to the system. Generate and distribute the keys. No permanent data storage (integration with SecurePG will come later), thus **no RoleTuple** creation.

Related components:

- CryptoUtil (generatePublicAndPrivateKeys)
- User (getUsername, getEmail, getPublicKey)
- CommandInterpreter (addNewUser, addNewRole)
- Administrator (addNewUser, addNewRole)
- Role (getRoleName, getPublicKey, getRoleVersioNumber)
- TupleAssociableElement
- ShareUtil (shareKeysWithUser)

Next Two Sprints

Second: 29/10 - 11/11

Sprint Backlog: create the **Administrator** class with all its methods and the class **RoleTuple**. Implement the association between users and roles through the **CommandInterpreter** methods. No permanent data storage (integration with the DB of SecurePG will come later).

Related components:

- Administrator (<all>)
- CommandInterpreter (assignUserToRole, removeUserFromRole)
- Tuple (<all>)
- RoleTuple (<all>)

Questions

Questions

- In the permission tuple <FK>, what is the utility of having “SU” as element of the tuple?
- Is it preferable to implement the RBAC0 algorithm using IBE/IBS or using PKI? And why so?

Thanks

Layout of the slides by Coman Catalin Andrei