Data Security and Privacy in the Cloud

Sara Foresti
Dipartimento di Informatica
Università degli Studi di Milano
sara.foresti@unimi.it

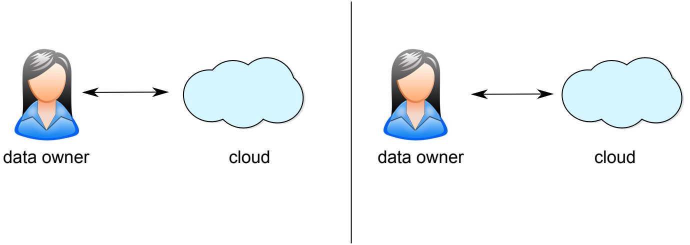
Secure Cloud Services and Storage Workshop 2017
September 10, 2017 – Oslo, Norway

Cloud computing

- The Cloud allows users and organizations to rely on external providers for storing, processing, and accessing their data
  - \+ high configurability and economy of scale
  - \+ data and services are always available
  - \+ scalable infrastructure for applications
- Users lose control over their own data
  - – new security and privacy problems
- Need solutions to protect data and to securely process them in the cloud

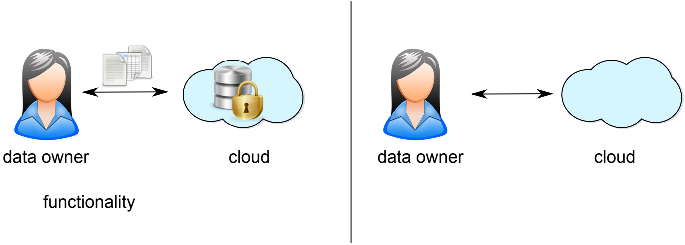
Cloud computing: Today

Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders



Cloud computing: Today

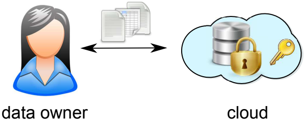
Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders



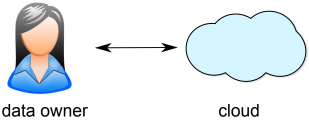
- functionality

Cloud computing: Today

Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders



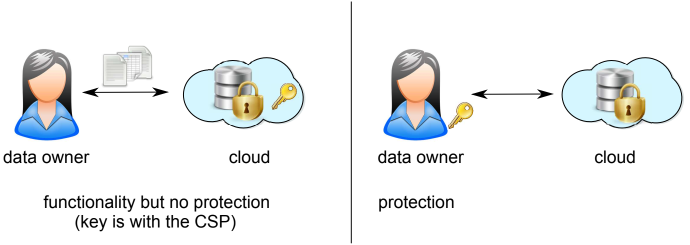
functionality but no protection
(key is with the CSP)



- functionality implies full trust in the CSP that has full access to the data (e.g., Google Cloud Storage, iCloud)

Cloud computing: Today

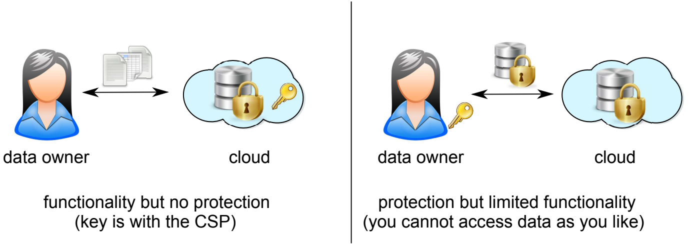
Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders



- functionality implies full trust in the CSP that has full access to the data (e.g., Google Cloud Storage, iCloud)
- protection

Cloud computing: Today

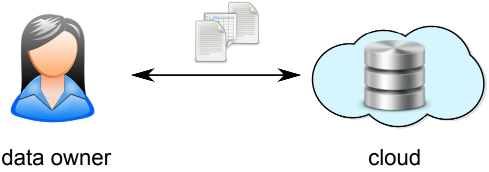
Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders



- functionality implies full trust in the CSP that has full access to the data (e.g., Google Cloud Storage, iCloud)
- protection but limited functionality since the CSP cannot access data (e.g., Boxcryptor, SpiderOak)

Cloud computing: ESCUDO-CLOUD's vision

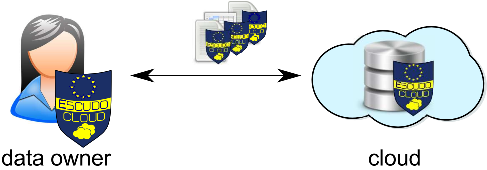
Solutions that provide protection guarantees giving the data owners both: full control over their data and cloud functionality over them



H2020 project "Enforceable Security in the Cloud to Uphold Data Ownership" (ESCUDO-CLOUD).

Cloud computing: ESCUDO-CLOUD's vision

Solutions that provide protection guarantees giving the data owners both: full control over their data and cloud functionality over them



- client-side trust boundary: only the behavior of the client should be considered trusted

  $\Longrightarrow$ techniques and implementations supporting direct processing of encrypted data in the cloud

H2020 project "Enforceable Security in the Cloud to Uphold Data Ownership" (ESCUDO-CLOUD).

Some challenges in data protection

- Protection of and fine-grained access to outsourced data
  - confidentiality (and integrity) of data at rest
  - fine-grained retrieval and query execution
- Selective information sharing
  - access control on resources in the cloud
- Confidentiality of data access
  - privacy of users' actions (access and pattern confidentiality)
- Integrity
  - integrity of stored data and query results

P. Samarati, S. De Capitani di Vimercati, "Cloud Security: Issues and Concerns," in *Encyclopedia on Cloud Computing,* S. Murugesan, I. Bojanova (eds.), Wiley, 2016.

Some challenges in data protection

- Protection of and fine-grained access to outsourced data
  - confidentiality (and integrity) of data at rest
  - fine-grained retrieval and query execution
- Selective information sharing
  - access control on resources in the cloud
- Confidentiality of data access
  - privacy of users' actions (access and pattern confidentiality)
- Integrity
  - integrity of stored data and query results

P. Samarati, S. De Capitani di Vimercati, "Cloud Security: Issues and Concerns," in *Encyclopedia on Cloud Computing,* S. Murugesan, I. Bojanova (eds.), Wiley, 2016.

Selective Information Sharing

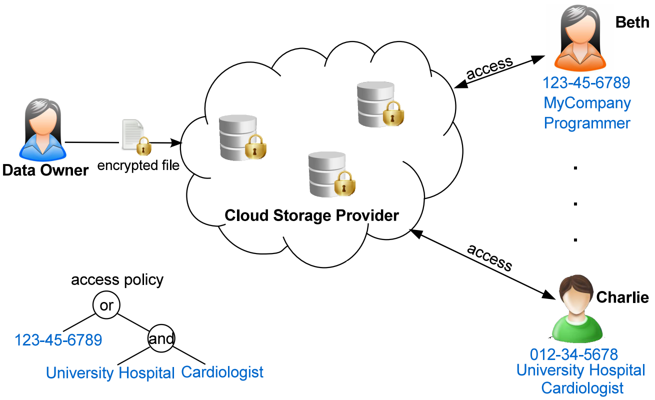
S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Encryption Policies for Regulating Access to Outsourced Data," in *ACM Transactions on Database Systems (TODS)*, vol. 35, n. 2, April 2010, pp. 12:1-12:46.

Selective information sharing

- Different users might need to enjoy different views on the outsourced data
- Enforcement of the access control policy requires the data owner to mediate access requests
  $\Longrightarrow$ impractical (if not inapplicable)
- Authorization enforcement may not be delegated to the provider
  $\Longrightarrow$ data owner should remain in control

Selective information sharing: Approaches – 1

- Attribute-based encryption (ABE): allow derivation of a key only by users who hold certain attributes (based on asymmetric cryptography)



Selective information sharing: Approaches – 2

- Selective (policy-based) encryption: the authorization policy defined by the data owner is translated into an equivalent encryption policy
  - users will be able to access only the resources for which they have the key

Selective encryption – 1

- Selective encryption: different keys are used to encrypt different data and users can know (or can derive) the keys of the data they can access
  - data themselves need to directly enforce access control
  - authorization to access a resource translated into knowledge of the key with which the resource is encrypted

	r_1	r_2	r_3	r_4	r_5
A	1	1	0	0	0
B	1	1	1	0	0
C	1	1	1	0	0
D	0	1	1	1	1
E	0	0	0	1	1

A knows the keys of r_1, r_2
B knows the keys of r_1, r_2, r_3
C knows the keys of r_1, r_2, r_3
D knows the keys of r_2, r_3, r_4, r_5
E knows the keys of r_3, r_5

Selective encryption – 2

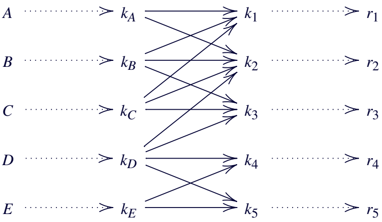
Requirements:

- one version of data (no replication)
- one key per user

Basic idea:

- key derivation method: via public tokens a user can derive all keys of the resources she is allowed to access

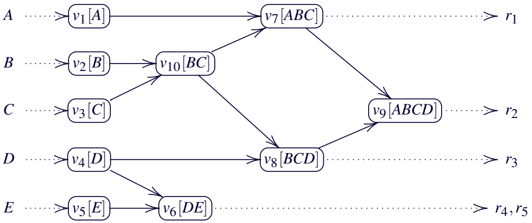
	r_1	r_2	r_3	r_4	r_5
A	1	1	0	0	0
B	1	1	1	0	0
C	1	1	1	0	0
D	0	1	1	1	1
E	0	0	0	1	1



Selective encryption – 3

Exploit ACLs to minimize number of keys and tokens

- Keys:
  - one key per user
  - an additional key for each non-singleton ACL
- Resources are encrypted with the key of their ACLs
- Tokens allow users to derive the keys of the ACLs to which they belong



Policy updates

- When authorizations dynamically change, the data owner needs to:
  - download the resource from the provider
  - create a new key for the resource
  - decrypt the resource with the old key
  - re-encrypt the resource with the new key
  - upload the resource to the provider and communicate the public catalog updates

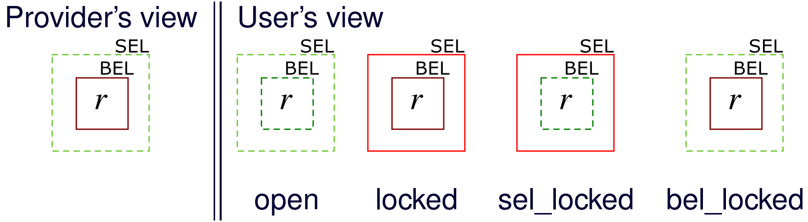
  $\Longrightarrow$ inefficient

- Possible solution: over-encryption

Over-encryption – 1

- Resources are encrypted twice
  - by the owner, with a key shared with the users and unknown to the provider (Base Encryption Layer - BEL level)
  - by the provider, with a key shared with authorized users (Surface Encryption Layer - SEL level)
- To access a resource a user must know both the corresponding BEL and SEL keys
- Grant and revoke operations may require
  - the addition of new tokens at the BEL level
  - the re-encryption of resources at the SEL level to guarantee the enforcement of policy updates

Over-encryption – 2

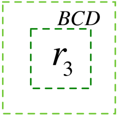


- Each layer is depicted as a fence
  - discontinuous, if the key is known
  - continuous, if the key is not known (protection cannot be passed)

Over-encryption – 3

- Revoke
  to protect resources for which the revokee has the BEL key

  EXAMPLE
  r_3 is encrypted with a key known to B, C, D at BEL
  r_3 is not encrypted at SEL



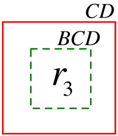
user B view

Over-encryption – 3

- Revoke
  to protect resources for which the revokee has the BEL key

  EXAMPLE
  r_3 is encrypted with a key known to B, C, D at BEL
  r_3 is not encrypted at SEL

  revoke B access to r_3:
  - over-encrypt r_3, using a key at SEL known to C, D only

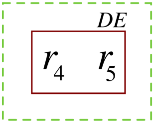


user B view

Over-encryption – 4

- Grant
  if a BEL key protects multiple resources and access is to be granted only to a subset of them, there is the need to protect at SEL level the resources on which access is not being granted

  EXAMPLE
  r_4, r_5 are encrypted with the same key known to D, E at BEL
  r_4, r_5 are not encrypted at SEL



user C view

Over-encryption – 4

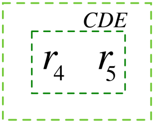
- Grant
  if a BEL key protects multiple resources and access is to be granted only to a subset of them, there is the need to protect at SEL level the resources on which access is not being granted

  EXAMPLE

  r_4, r_5 are encrypted with the same key known to D, E at BEL

  r_4, r_5 are not encrypted at SEL

  grant C access to r_4
  - add a token at BEL enabling C to derive the key of r_4



user C view

Over-encryption – 4

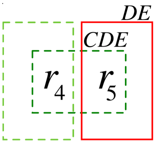
- Grant
  if a BEL key protects multiple resources and access is to be granted only to a subset of them, there is the need to protect at SEL level the resources on which access is not being granted

  EXAMPLE

  r_4, r_5 are encrypted with the same key known to D, E at BEL

  r_4, r_5 are not encrypted at SEL

  grant C access to r_4
  - add a token at BEL enabling C to derive the key of r_4
  - over-encrypt r_5, using a key at SEL known to D, E only



user C view

Mix&Slice for Policy Revocation

E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, “Mix&Slice: Efficient Access Revocation in the Cloud,” in *Proc. of the 23rd ACM Conference on Computer and Communications Security (CCS 2016)*, Vienna, Austria, October 2016.

Mix&Slice

- Over-encryption requires support by the server (i.e., the server implements more than simple get/put methods)

- Alternative solution to enforce revoke operations: Mix&Slice

- Use different rounds of encryption to provide complete mixing of the resource

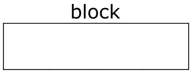
  $\Longrightarrow$ unavailability of a small portion of the encrypted resource prevents its (even partial) reconstruction

- Slice the resource into fragments and, every time a user is revoked access to the resource, re-encrypt a randomly chosen fragment

  $\Longrightarrow$ lack of a fragment prevents resource decryption

Resource organization

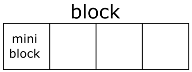
- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits



Resource organization

- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits

- Mini-block: sequence of bits in a block
  it is our atomic unit of protection
  mini-blocks of 32 bits imply a cost of
  2^{32} for brute-force attacks

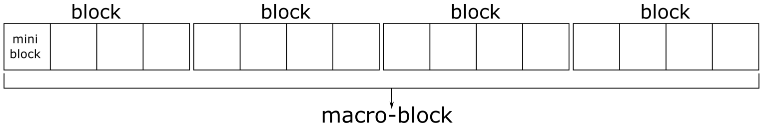


Resource organization

- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits

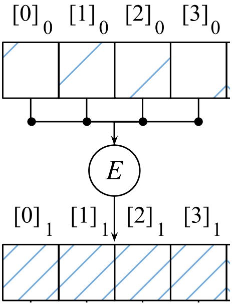
- Mini-block: sequence of bits in a block
  it is our atomic unit of protection
  mini-blocks of 32 bits imply a cost of
  2^{32} for brute-force attacks

- Macro-block: sequence of blocks
  mixing operates at the level of macro-block
  a macro-block of 1KB includes 8 blocks



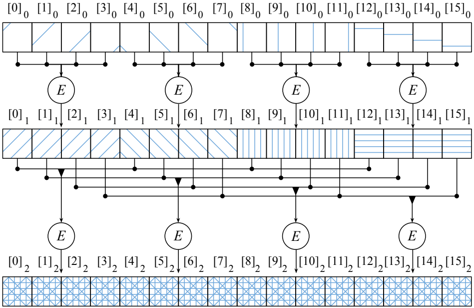
Mixing – 1

- When encryption is applied to a block, all the mini-blocks are mixed
  + absence of a mini-block in a block from the result prevents reconstruction of the block
  – does not prevent the reconstruction of other blocks in the resource



Mixing – 2

- Extend mixing to a macro-block
  - iteratively apply block encryption
  - at iteration i, each block has a mini-block for each encrypted block obtained at iteration $i-1$ (at distance 2^i)
  - x rounds mix 4^x mini-blocks

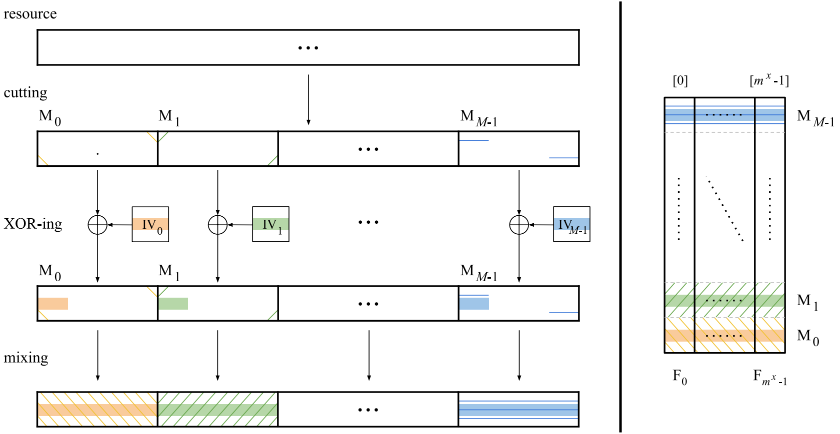


Slicing – 1

- To be mixed, large resources require large macro-blocks
  - many rounds of encryption
  - considerable computation and data transfer overhead
- Large resources are split in different macro-blocks for encryption
- Absence of a mini-block for each macro-block prevents the (even partial) reconstruction of the resource

Slicing – 2

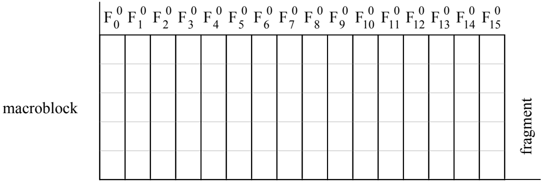
- Slice resources in fragments having a mini-block for each macro-block (the ones in the same position)
  - absence of a fragment prevents reconstruction of the resource



Revoke

To revoke user u access to a resource r

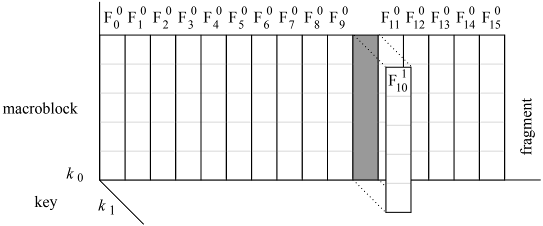
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

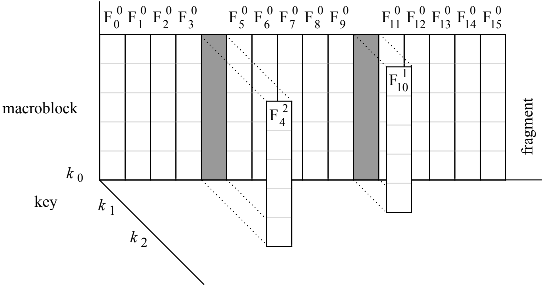
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

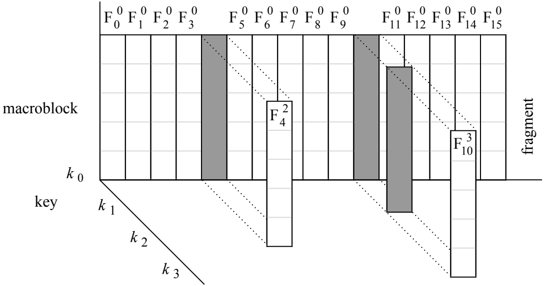
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Effectiveness of the approach

- A revoked user does not know the encryption key of at least one fragment
  - necessary a brute force attack to reconstruct the fragment (and the resource)
  - 2^{msize} attempts, with msize the number of bits in a mini-block
- A user can locally store f_{loc} of the f fragments of a resource
- Probability to be able to reconstruct the resource after f_{miss} fragments have been re-encrypted: $P = (f_{\text{loc}}/f)^{f_{\text{miss}}}$
  - proportional to the number of locally stored fragments
  - decreases exponentially with the number of policy updates

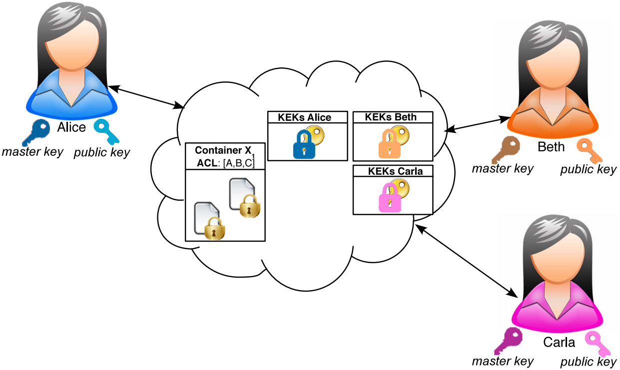
Applying Selective Encryption and Over-encryption in OpenStack Swift

E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, “Access Control Management for Secure Cloud Storage,” in *Proc. of SecureComm 2016*, Guangzhou, China, October 10-12, 2016.

Policy-based encryption in OpenStack Swift – 1

- Swift module: an object storage service allowing users to store and access data in the form of objects
- Swift enforces access control associating an Access Control List (ACL) with each container
- Policy-based encryption:
  - associates a DEK (Data Encryption Key) with each container, used to encrypt objects in the container
  - associates a MEK (Master Encryption Key) and an asymmetric encryption key pair with each user
  - stores a KEK (Key Encryption Key) for each user authorized for a container, enabling her to derive the container DEK from her private or master key

Policy-based encryption in OpenStack Swift – 2

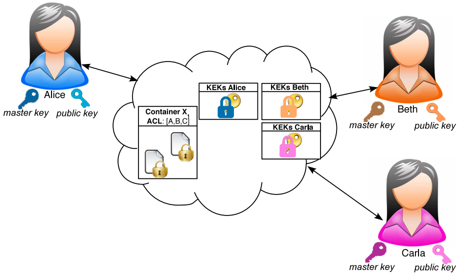


Alice generates a container X_1 and grants Beth and Carla access to it

Policy changes: Grant

User u grants to user u_j access to a container C

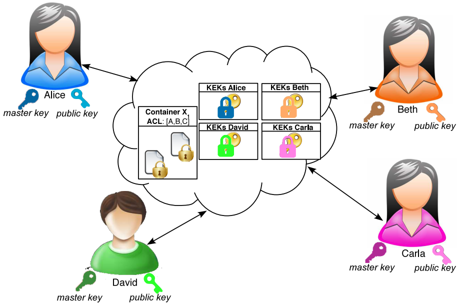
- User u_j is added to the ACL of container C
- User u computes a new KEK for u_j, which allows u_j to derive the DEK of container C



Policy changes: Grant

User u grants to user u_j access to a container C

- User u_j is added to the ACL of container C
- User u computes a new KEK for u_j, which allows u_j to derive the DEK of container C

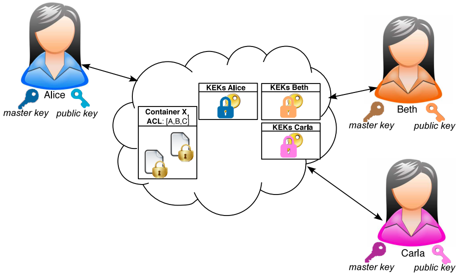


Alice grants to David access to container X_1

Policy changes: Revoke with Over-encryption

User u revokes access to container C from user u_j

- User u removes u_j from the ACL of container C
- User u asks the storing server to over-encrypt the objects in container C with a SEL key that only non-revoked users can derive

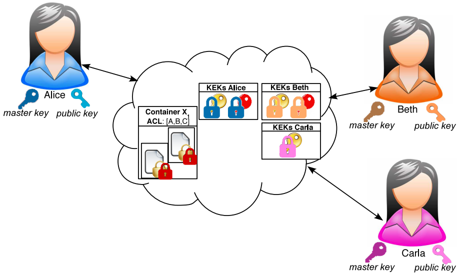


Policy changes: Revoke with Over-encryption

User u revokes access to container C from user u_j

- User u removes u_j from the ACL of container C

- User u asks the storing server to over-encrypt the objects in container C with a SEL key that only non-revoked users can derive



Alice revokes from Carla access to container X_1

Conclusions and future directions

Solutions based on policy-based encryption

- enable users to regulate access to their resources
- guarantee that resources self-enforce access restrictions
- support efficient policy updates through over-encryption and mix&slice approaches
- can be integrated with current cloud technology

Open issues include:

- support for write authorizations
- combine with techniques for efficient query evaluation
- address collusion
- ...