Cloud Storage Security
Cryptographic tools

Abdallah M'HAMED

R3S /SAMOVAR (CNRS)

TELECOM SUDPARIS – Evry – France

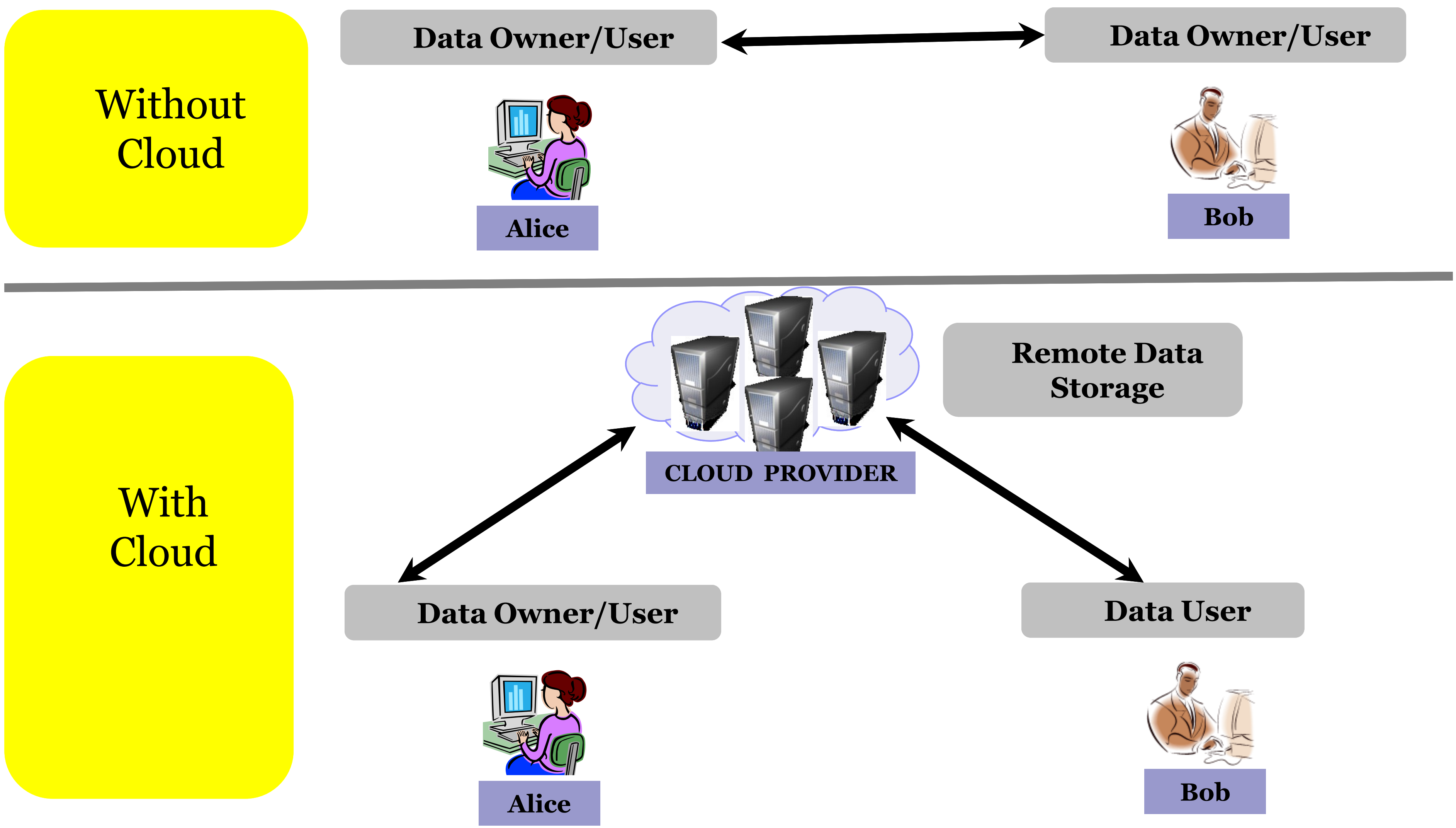
Abdallah.mhamed@telcom-sudparis.eu

CONTENTS

I INTRODUCTION

- **Communication Security**
  - Internal Com.
    - Shared infrastructure
    - Virtual network
    - Security misconfigurations
  - External Com.
- **Architectural Security**
  - Data storage
    - Privacy & Integrity
    - Data recovery
    - Data backup
    - Improper media sanitisation
  - Web applications & API
  - Identity management & Access Control
  - Virtualization
    - VM Image sharing
    - VM isolation
    - VM escape
    - VM migration
    - VM rollback
    - Hypervisor issues
- **Contractual & Legal Aspects**
  - Service Level Agreements
  - Legal issues

Cryptography for Data protection



Data Storage Security Requirements

- **Confidentiality:** the CSP does not learn any information about customer data
- **Privacy**: prevent any disclosure of private information
- **Integrity**: any unauthorized modification of customer data by the CSP should be detected by the customer
- **Availability**: customer data should be accessible from any machine and all times
- **Data sharing**: customers can share their data with trusted parties

Cloud Integrity

- **Remote Data checking**:
- Interactive assurance that remotely stored data is available and intact
- Let a client efficiently verify that a remotely stored data is available (without downloading it) and can be fully recovered on demand.
- Establish trust in CS services by providing proofs to clients that the data is consistent and available at all times

Cloud Integrity

- **Data preparation for storage:**

• Data indexing
• Data encryption using **symmetric** scheme (ie: AES)
• Index encryption using **searchable encryption (SE)**
• Encrypted data and Index are encoded in such a way that the data integrity can be verified using a ***proof of storage*** **(PoS)**

Cloud Privacy

- Client's private data is stored only in its encrypted form
- Offering different levels of privacy to cloud customers
- Enable to compute arbitrary functions on data in its encrypted form
- The computation leaks no information and is verifiable

Cloud Privacy

- **Attribute based Encryption (ABE):**

  Each user is provided with a decryption key using a set of ***attributes*** associated with him/her ***credentials*** (name, age, address, job, etc.)

- **Fully Homomorphic Encryption (FHE)**

  - Enabling ***computation on encrypted data*** stored in distributed servers of CP.
  - Cloud server have no knowledge about : input data, processing function, result values
  - Outsourced computation occurs in a fully privacy-preserving way

IBE: Identity Based Encryption

- Public-key is derived from public data string (identity, email, IP adress)
- Each client acts as its own PKG and generates its IBC-PE (Public Elements) and its own private key
- Easier Key management: Certificate-free concept
- Derivation of public key does not depend on previous computation of previous private keys

ABE: Attribute Based Encryption

- A generalization of IBE scheme
- Public-key derived from a combination of ***attributes***
- Restrict decryption privileges to entities with a defined set of ***attributes*** (not to particular identity)
- Allows decryption for anyone carrying a chosen set of attributes satisfying a policy defined over ***attributes***
- Data remains inaccessible unless the provider's or user's role or privileges adhere to the policy

ABE: Attribute Based Encryption

- Examples of access restriction:

  - *Encrypted traffic log file* : a particular range of dates, subnet IP addresses)

  - *Medical record*: most recent medication

- Let Ω be a universe of ***attributes***
- User U is characterized by a subset $U \subseteq \Omega$
- Anyone encrypt under an identity $V \subseteq \Omega$
- Everyone having at least $d \geq 1$ attributes in common with V will be able to decrypt
- Formally $V \subseteq \Omega$ ***and*** $|U \cap V| \geq 1$

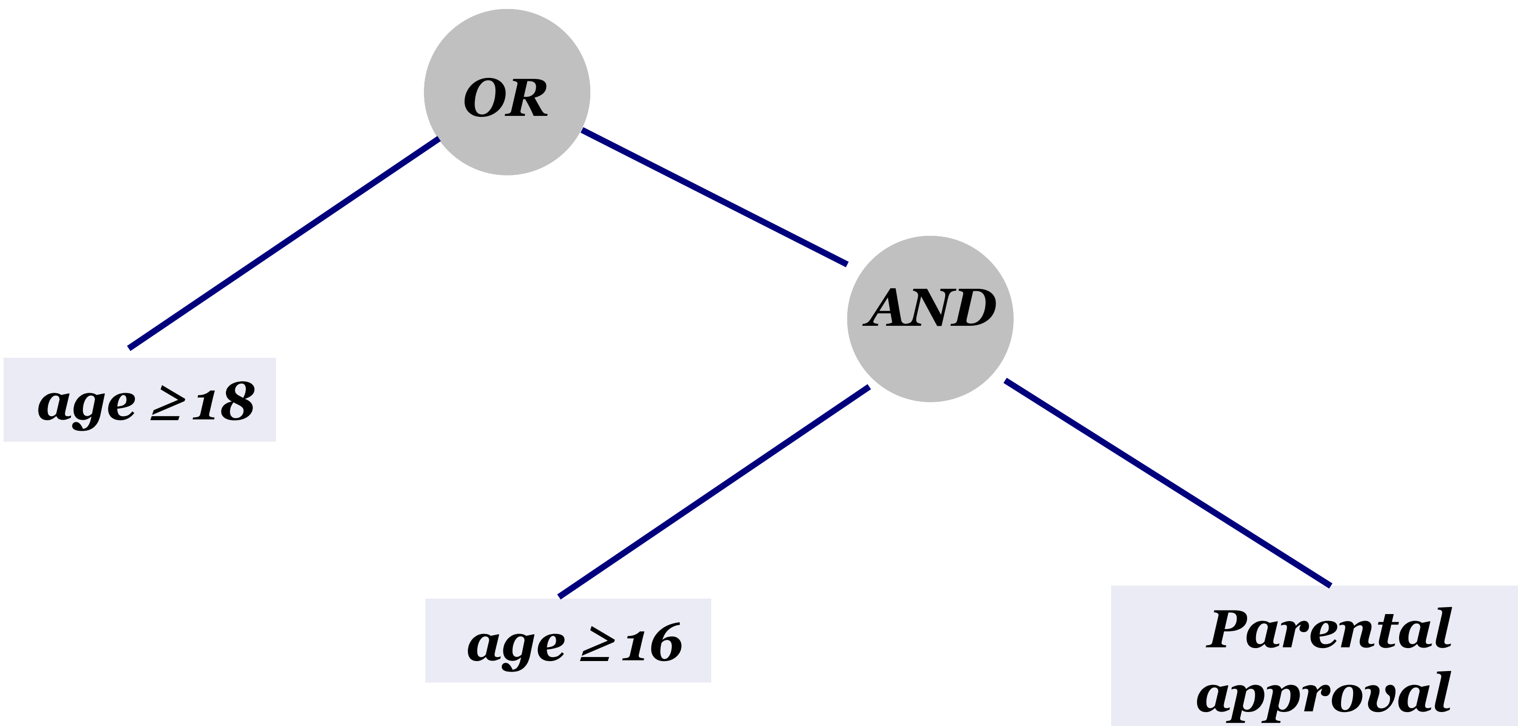
ABE: Attribute Based Encryption

- **Encryption**: $\mathbf{m * g^y}$
  The exponent $\boldsymbol{y}$ constructed from at least $\boldsymbol{d}$ values
  Decryption : reconstructing the secret $\boldsymbol{y}$ and dividing out by the factor $\mathbf{g^y}$

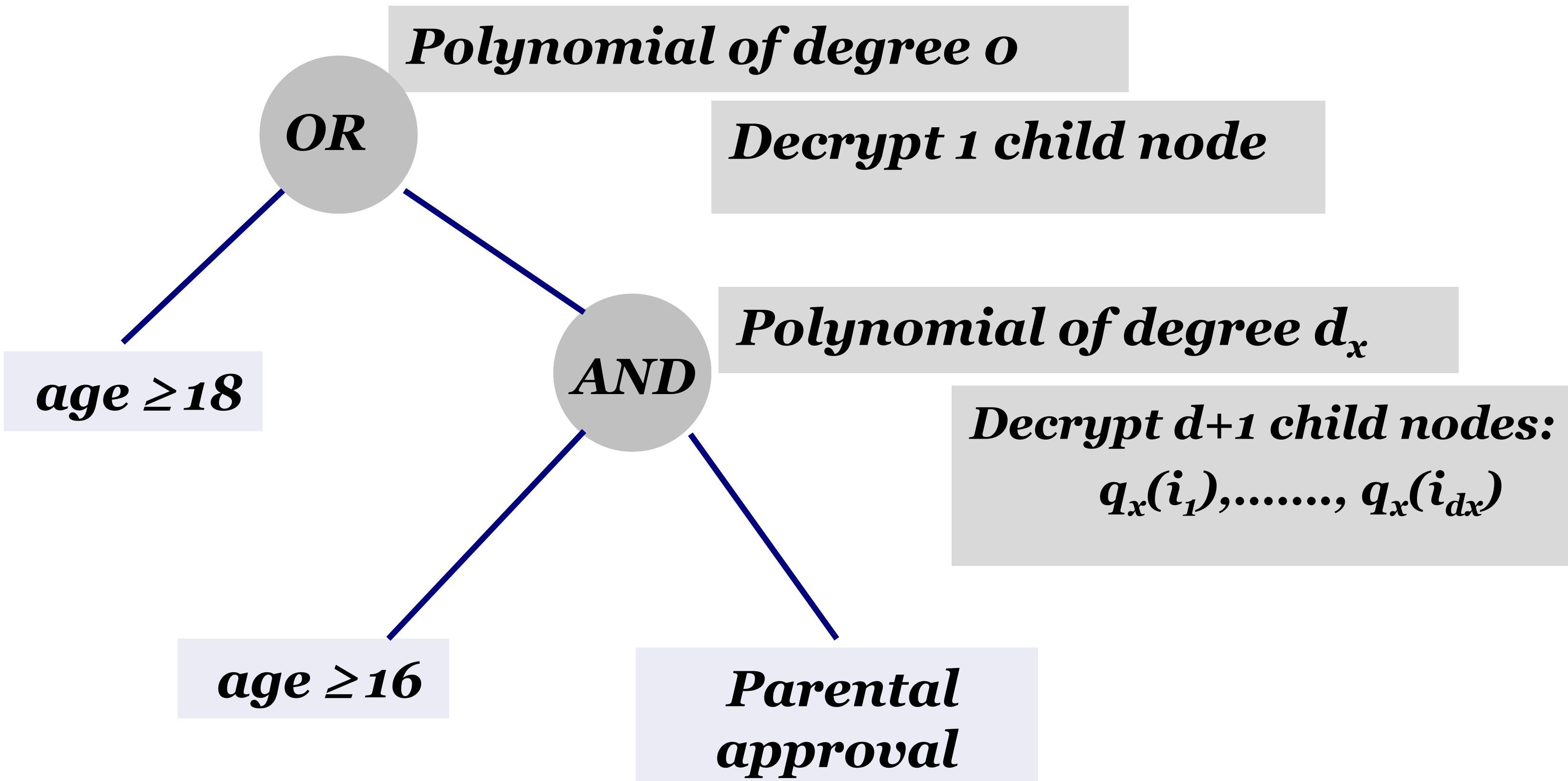
- **Access policy**: Boolean expression (AND/OR) involving a subset of attributes.
  The Access granted if at least d attributes are present
  The access structure: $\{U \subseteq \Omega : |U| \geq d\} \subseteq 2^{\Omega}$

ABE: Attribute Based Encryption

- ***Access policy***: decryption granted if a boolean expression is satisfied



ABE: Attribute Based Encryption



PE: Predicate Encryption

- Ciphertext permit the evaluation of predicates on the hidden plaintext
- Encrypt M such as M remains hidden but a set of predefined predicates can be evaluated to check the hidden plaintext for certain properties (access permissions) prior to decryption
- Support access control at a very fine-grained level
- Allow M to be tested for certain properties

PE: Predicate Encryption

- **Predicates**: general function that evaluates on strings into the set ***{0, 1}***
- An entity seeking to decrypt would request a decryption key for the predicate ***f***
- The plaintext ***M*** can be recovered if and only if ***f*** evaluates to ***1*** for ***M***.
- Predicates are constructed using logical expressions or polynomial equations:

HE: Homomorphic Encryption

- Allows CP to compute an operation on plaintext while only having access to ciphertexts, without any knowledge on secret key and plaintext
- ***Group homomorphic*** scheme:

Plaintext space : $m \in (M, *)$

Cyphertext space : $c \in (C, \oplus)$

$$m_1 , m_2 \in M$$

$$Enc(m_1 * m_2 ; P_k) = Enc(m_1 ; P_k) \oplus Enc(m_2 ; P_k) = c_1 \oplus c_2$$

$$Dec(c_1 \oplus c_2; S_k) = m_1 * m_2$$

HE: Homomorphic Encryption

- Multiplicative HE: RSA

$$E(m_1 \times m_2 ; P_k) = E(m_1 ; P_k) \cdot E(m_2 ; P_k) = c_1 \cdot c_2$$

$$Dec(c_1 \cdot c_2 ; S_k) = m_1 \times m_2$$

- Additive HE: El Gamal

$$E(m_1 + m_2 ; P_k) = E(m_1 ; P_k) \cdot E(m_2 ; P_k) = c_1 \cdot c_2$$

$$Dec(c_1 \cdot c_2 ; S_k) = m_1 + m_2$$

- **Somewhat HE**: specific class of functions [BGN: Boneh-Goh-Nissim]
- **Full HE**: any functions []

HE: Homomorphic Encryption

- **Somewhat HE**: [BGN: Boneh-Goh-Nissim]

specific class of functions

- **Fully HE**: [Gentry]

- Most sophisticated class of HE schemes
- Allow arbitrary functions to be evaluated on cyphertexts

Class	Scheme	Additions	Multiplications
SHE	**RSA**	None	Unlimited
	ElGamal	Unlimited	None
SHE	**BGN**	Unlimited	Limited to 1
FHE	**Gentry**	Unlimited	Unlimited

Remote Data checking

- To let a file owner verify the existence of remotely stored file at CSP (without downloading it).
- Establish trust by providing proofs to client about the file **consistency and availability** at anytime.
  - *Proofs of retrievability* ***(PoR)***
  - *Provable Data possession* ***(PDP)***

- ***PoR*** and ***PDP*** achieve the same proof with :

  - significantly less communication /computational overhead
  - little storage requirement for **V**
  - small number of memory access for **P**

Remote Data checking

- ***Simple scheme***:
- File Owner choose a set of keyed hash function:

  $H(x;k_1), H(x;k_2), \ldots\ldots\ldots, H(x;k_t)$

  x : *input data*, H : *hash function*, $k_1, k_2, \ldots\ldots, k_t$: *keys*
- The Verifier stores the t hash values
- A backup server can be challenged to provide the hash $H(x;k_i)$ under a given key k_i

- **Drawbacks**:

- Storage cost for **V** /proportional to t
- Need for **SP** to process **F** /every challenge
- Limited number of challenges

Remote Data checking

- ***Efficient scheme***:
- Use a hash function H and a block cipher like AES.

  $\boldsymbol{F} = (f_1, f_2, \ldots, f_n)$ *of n blocks,* $\boldsymbol{k}$ *: key,* $\boldsymbol{H}$ *: hash function*

  t random challenges : $r_1, r_2, \ldots, r_t$

  t random index sets: $I_1, I_2, \ldots, I_t \subset \{1, \ldots, n\}$ *of size c*
- The owner computes **tokens**

$\boldsymbol{v_i} = Encrypt\,[\boldsymbol{H}(r_i || f_{i1} || f_{i2} || \ldots || f_{ic}) ; \boldsymbol{k}]$ for $1 \leq i \leq t$ and $i_j \in I_i$

- Outsources $(\boldsymbol{F}, v_1, v_2, \ldots, v_t)$ for $1 \leq i \leq t$
- The Client send r_i and I_i to the server
- The Server computes $h_i = \boldsymbol{H}(r_i || f_{i1} || f_{i2} || \ldots\ldots || f_{ic})$ *and sends* (v_i, h_i) *to the client*
- The client check *if* $h_i = Decrypt\,(v_i ; \boldsymbol{k})$

Remote Data checking

- *Simple modification of* ***Efficient scheme*** to support updates on outsourced files on arbitrary positions

$$\boldsymbol{v_i} = Encrypt\,[\boldsymbol{H}(r_i || i_1 || f_{i1}) \oplus \ldots \oplus \boldsymbol{H}(r_i || i_c | f_{ic}) ; \boldsymbol{k}]$$

If some block $\boldsymbol{f_j}$ *needs to be modified to* $\boldsymbol{f'_j}$ *, then all values* $\boldsymbol{v_i}$ *that includes* $\boldsymbol{f_j}$ *can be updated to* $\boldsymbol{v'_i}$

$$\boldsymbol{v'_i} = Enc\,[\,Dec\,(\boldsymbol{v_i} ; \boldsymbol{k}) \oplus \boldsymbol{H}(r_i || j || \boldsymbol{f_j}) \oplus \boldsymbol{H}(r_i || j || \boldsymbol{f'_j}) ; \boldsymbol{k}]$$

Block $\boldsymbol{f_j}$ *will be replaced by the Block* $\boldsymbol{f'_j}$

Proofs of Retrievabality (PoR)

- Basic idea: to embed small (randomly values) data chunks called ***sentinels*** in F
- **Spot checks** can be made in order to detect corruptions of F.

  The server is asked to return the sentinel values at some specific positions.
- To protect against small corruptions of F, **error correcting block codes** are applied
- To prevent the server from deleting portions of F when retaining the sentinels only, **encryption and permutations** are employed to scatter and hide the position of the sentinels across F.

Provable Data Possession (PDP)

- The client having $\boldsymbol{F} = (f_1, f_2, \ldots, f_n)$ computes ***tags*** $\boldsymbol{T} = (T_1, T_2, \ldots, T_n)$ *and stores* ***(F, T)*** at the server.

  The client generates a challenge related to a random subset of blocks *{1, …, n}*
- The prover responds by computing ***homomorphic tag*** aggregating all **tags** corresponding to the challenged file blocks and some linear combination of the file blocks.

Secure Data Deduplication (SDD)

- Help cloud providers to **save storage space** while at the same time preserving **confidentiality** of client data
- Physical storage only happens when initial first user uploads the file
- For any subsequent storage request for the same file, the server create only a **reference** to the initial file
- Encrypting files before uploading them to a storage service allows us to guarantee the **confidentiality** in respect to insider attacks of SP or outsider attacks
- Neither SP and outsiders gets to know the **decryption keys** (managed by the client)

Secure Data Deduplication (SDD)

- Applied on *file* or *block* levels and performed on Client or Server sides
- **Client side DD**: the server look-up before uploading F:

asking the server if he knows a file with a **given hash value**

- **Server side DD**: upload always happens, letting the Server handling redundant copies of the same file.
- Client side DD results in enormous bandwidth and disk savings
- Client side DD is the most rewarding solution, applied by popular storage services (DropBox and Mozy)

Searchable Encryption (SE)

- Combines ***confidentiality*** with ***search*** functionality
- Encryption allows **keyword** search over encrypted files
- Use of highly available cloud storage without revealing the files in plaintext to the SP
- Enable SP to search encrypted data by virtue of **a *trapdoor information*** that the Client provides
- The server neither learns the query nor the underlying data
- The storage server can retrieve and respond with all the data (encrypted documents) that match the query
- The client can decrypt the results locally
- A *full homomorphic encryption* schemes allow the construction of *searchable encryption*

Searchable Encryption (SE)

- **Symmetric/Asymmetric SE**

  - *Symmetric SE*: Owner of secret key (data owner) can store encrypted data and generate encrypted queries

  - Asymmetric SE: Everyone can store encrypted data

  Only the owner can issue encrypted queries and decrypt data

- **Fulltext/Index SE**

  - *Fulltext search*: the entire content of data file is searched item by item where every item is tested (i.e: equality with keyword)

  - *Index search*: runs a query on as separate encrypted index file

  Forward Index: a list of keywords per file

  Inverted Index: a list of matching files for every keywords

V CONCLUSION

- A huge specific cryptographic tools dedicated to cloud storage security
- To ensure both integrity and privacy of outsourced data targeting

  - reduce the computational and cost overhead

  - release of data on remote CP servers
- Some schemes (Like HE) are still costly for both CP and client sides
- Research work is quite promising for the remaining challenging tasks

Bibliographical sources

- **Surveyson Cloud security**

1. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms, Nesrine Kaaniche, Maryline Laurent, Computer Communications, 111 (2017), pp. 120-141

- **Cryptographic background of Cloud security**

1. Cryptography for security and Privacy in Cloud Computing, Stepan Rass, Daniel Slamanig,, Artech House, Boston, 2014, 242 pages