Cloud Crypto

Sushmita Ruj

Recap

- Need for Zero-knowledge proofs
- ZKP Foundations
- SNARKS: Building Blocks and Design
- ZKP Applications Conclusion and open problems

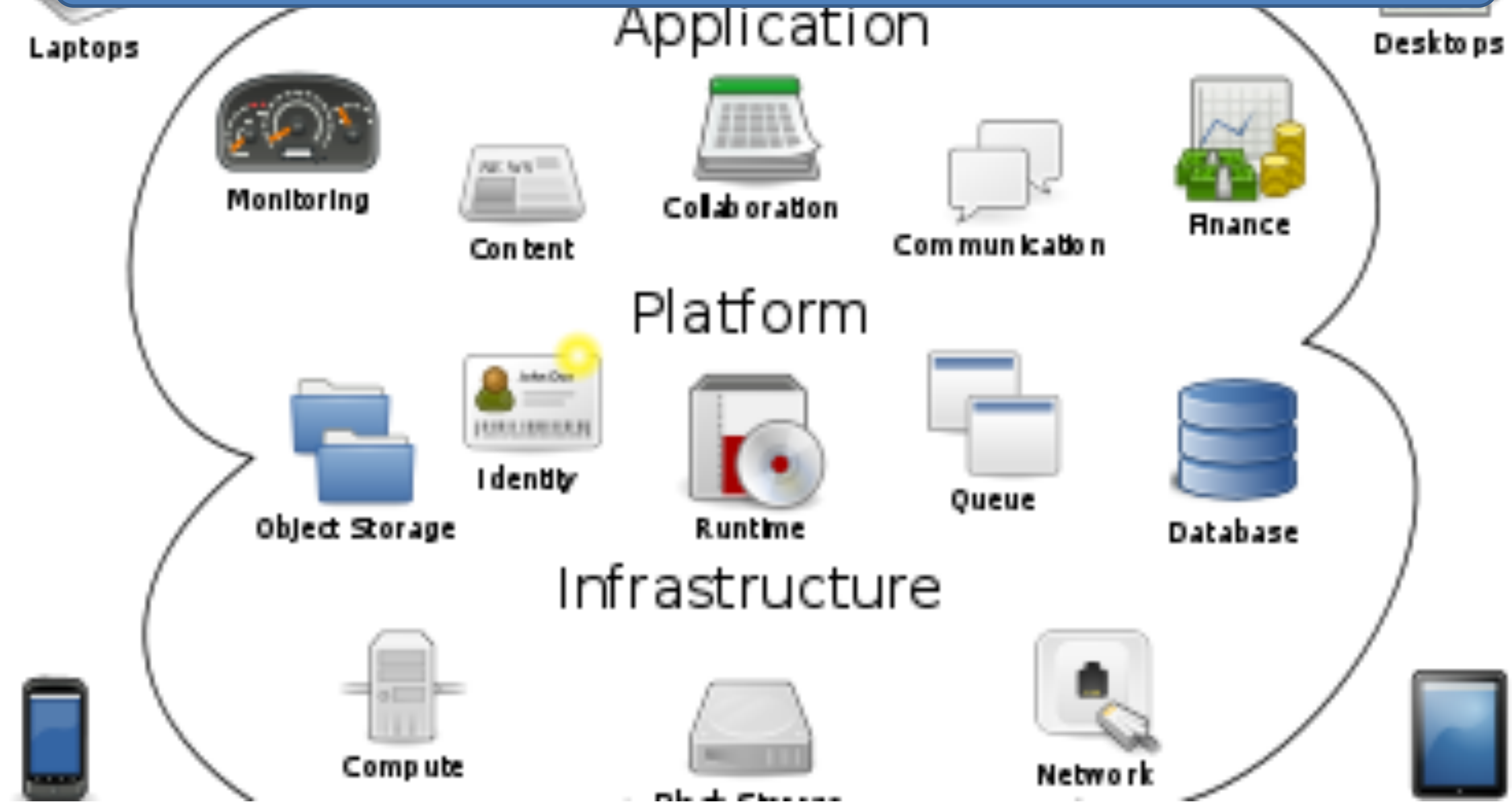
This Lecture

- Crypto for cloud
- Bilinear Pairings
- BLS signatures
- Shamir Secret sharing
- Data Audits
- Attribute based access control

Clouds

Clouds

Why buy when we can rent?



Security issues in Cloud Computing

- A user's data should be protected against adversaries or other users
- Cloud should be oblivious to the data stored
- Cloud should be oblivious to data it is computing
- Cloud should be accountable for its services

Cloud service provider as adversary

- Read/modify data
- CSP might not provide the desired amount of redundancy
- Might not provide the amount of storage as specified in the SLA
- Might not provide enough computational resources as specified in the SLA

Privacy issues in Cloud Computing

- Cloud service providers should not be able to track the position of a user/mobile device
- Legal issues in privacy protection

  - Data might be stored in different servers across different countries

  - Different privacy laws across different nations

Different faces of cloud security

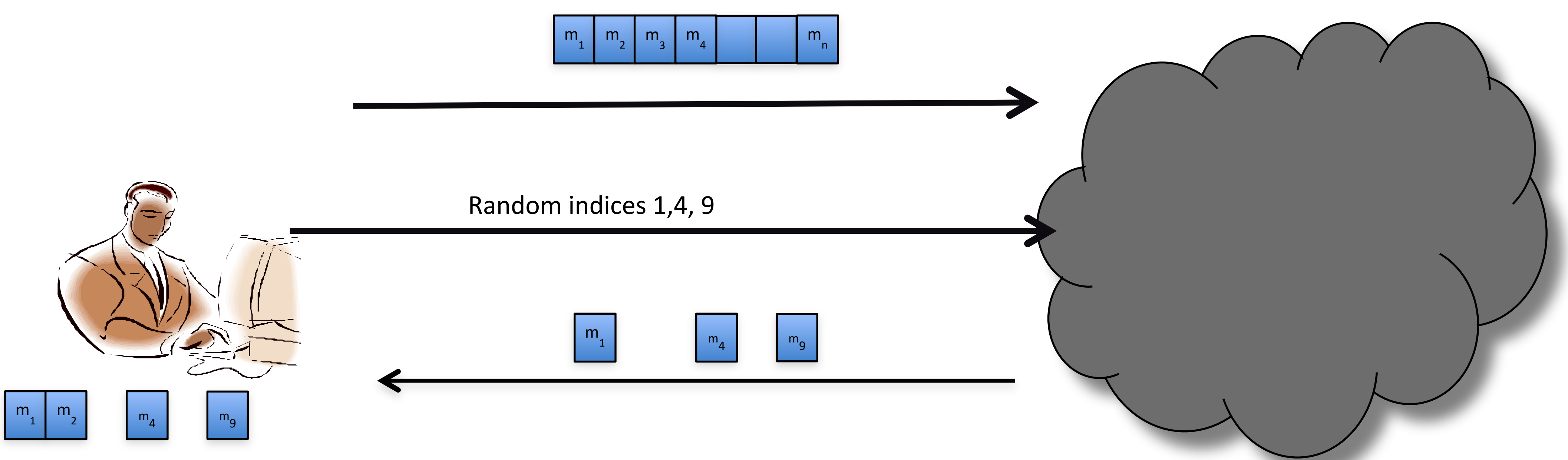
- Cryptographic security
  - Authenticating users
  - Hiding data from cloud: computing and searching on encrypted data
  - Access control
  - Data auditing for integrity verification
- Network Security
  - Ensure that all communication channels are secure
- Operating system security
  - Virtualization security

Cryptographic techniques for Cloud computing

- Data auditing: Verify data integrity
- Fine-grained access control: Grants authorized access to user who have paid for service and denies access to unauthorized users
- Homomorphic encryption: Cloud does not know what data it is operating on, just gives back the result
- Searchable encryption: Cloud returns result of a query without knowing what the query is
- Secure Multi parti Computation: Computing on data that resides in two/more servers

Cloud Audits

Data Auditing (Attempt)



Owner verifies

Traditional Proofs of Storage

- Provable Data Possession (PDP):

CPS (prover) proves that it possesses the client's original data. Client (verifier) is able to verify this proof.

Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, Song, "Provable data possession at untrusted stores", CCS '07.

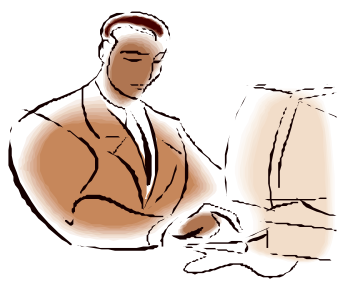
Proofs of Retrievability (PoR):

The client (verifier) runs an efficient data audit proof in which the data storage server (prover) proves that it still possesses the client's data and client can recover entire file

Jules & Kaliski, "Pors: proofs of retrievability for large files", CCS '07

This is achieved using Erasure codes.

Proof of Data Possession



$F = [(m_i)]_{1 \leq i \leq n}$

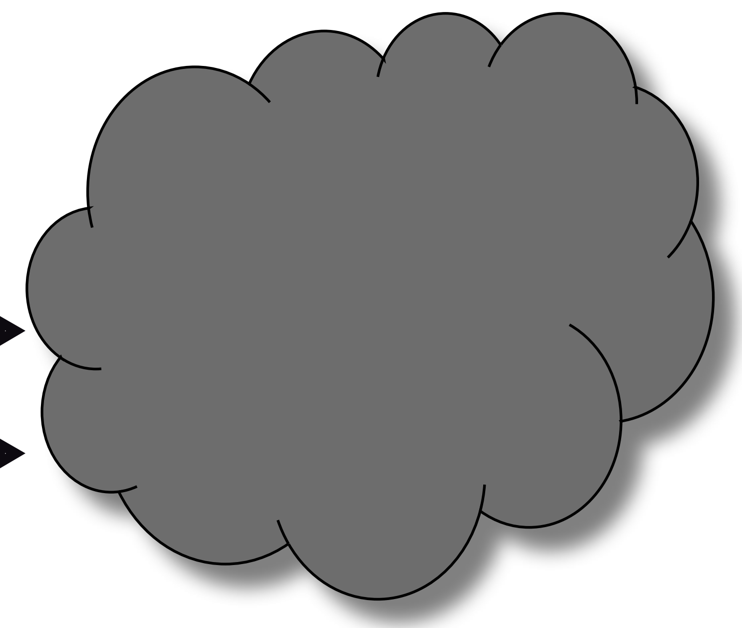
For every block m_i
generate (m_i, σ_i)

Upload $[(m_i, \sigma_i)]_{1 \leq i \leq n}$ →

Random indices i →

← (m_i, σ_i)

Owner verifies



σ_i is the authenticator (unforgeable)

Private verification : message authentication code

Preprocessing for auditing: Erasure Codes

- An $(n; f; d)_\Sigma$ erasure code over finite alphabet Σ is an error-correcting code that consists of
- Enc: $\Sigma^f \rightarrow \Sigma^n$ An encoding algorithm
- Dec: $\Sigma^n \rightarrow \Sigma^f$ - decoding algorithm

d is the minimum distance (Hamming distance between any two codewords is at least d) of the code.

- An $(n; f; d)$ erasure code can tolerate up to $d-1$ erasures.
- If $d = n - f + 1$, we call the code a maximum distance separable (MDS) code.
- For an MDS code, the original message can be reconstructed from any f out of n symbols of the codeword.
- Examples: Reed-Solomon codes

Proof of Retrievability

Given a file F_0, it is erasure coded to F

$$F = [(m_i)]_{1 \leq i \leq n}$$

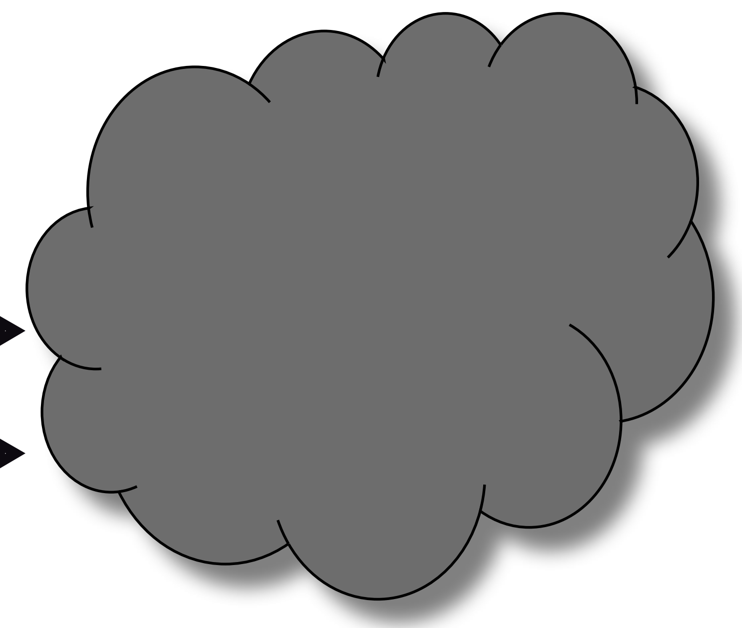
For every block m_i generate (m_i, σ_i)

Upload $[(m_i, \sigma_i)]_{1 \leq i \leq n}$

Random indices i

(m_i, σ_i)

Owner verifies



σ_i is the authenticator (unforgeable)

Private verification : message authentication code

Proof size is proportional to query size

(n, k, d) erasure code can tolerate up to d -1 erasures.

If d = n - k + 1 , we call the code a maximum distance separable (MDS) code.

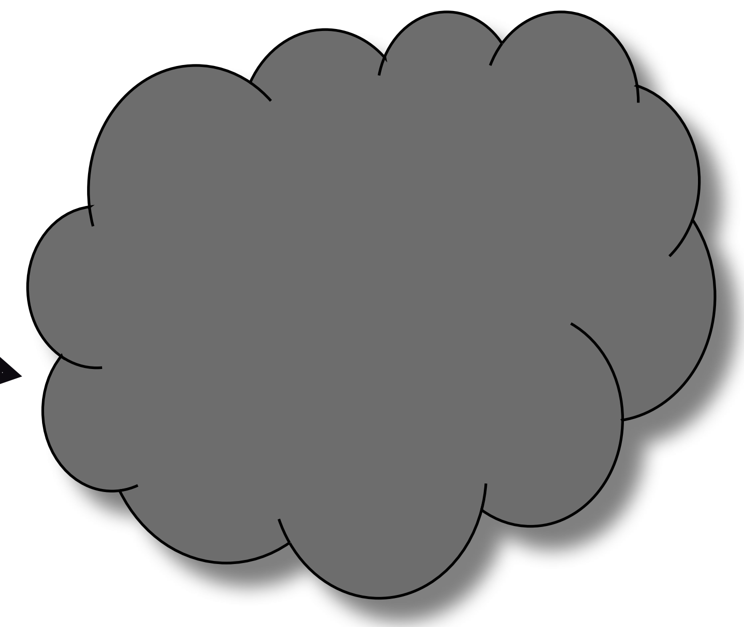
Proofs of Storage with Third Party Auditors

Given a file F_0 , it is erasure coded to F

$F = [(m_i)]_{1 \leq i \leq n}$

For every block m_i generate (m_i, σ_i)

Upload $[(m_i, \sigma_i)]_{1 \leq i \leq n}$



Random indices i

(m_i, σ_i)

Auditor verifies

σ_i is the authenticator (unforgeable)

Public verification: signature

Requirements of an Auditing Scheme

- Verification should be fast
- Proof should be short (low communication cost)
- Anyone can verify (public verifiability)
- A third party performing the audit should have no knowledge of the data (Privacy preserving)

Constructing a desirable auditing scheme

Bilinear Pairings

↓

BLS signatures

↓

Homomophic Linear Authenticators (HLA)

↓

Compact Proofs of Retrievability (PoR)

↓

Privacy-preserving data auditing

Bilinear Pairings

- G, G_T are groups of order p (prime)
- $e : G \times G \rightarrow G_T$ is an a bilinear map if:

-Non degenerate
$e(g,g) \neq 1$

-Bilinear: $e(g^a,g^b) = e(g,g)^{ab}$, $a,b \in Z_p^*$, $g \in G$

-e can be computed efficiently

Boneh-Lynn-Shacham Signature (BLS)

- H: {0,1}* -> G
- Private signing key sk = $x \varepsilon Z_p$
- Public verification key pk = g^x
- Sign(M,sk): $\sigma = H(M)^x$
- Verify(M, σ,pk) : Valid iff $e(\sigma,g) = e(H(M),pk)$
- Correctness: $e(\sigma, g) = e(H(M)^x, g) = e(H(M),g^x)$
- Short Signatures
- Aggregated easily

Homomorphic Linear Authenticator

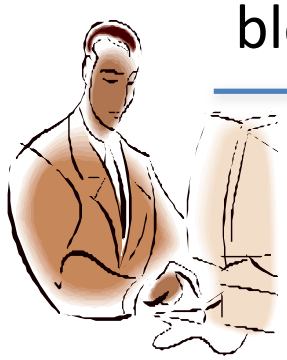
- Let σ_1, σ_2 be 2 authenticators on m_1, m_2 resp.
- $(\sigma_1)^a(\sigma_2)^b$ is an "authenticator" on $(m_1)^a(m_2)^b$
- Easily forgeable?
- "Linear combination"
- BLS signature: $[H(m)]^x$
- H's (pseudo-)randomness gives unforgeabiliity
- Easier if the message is an exponent

Data Auditing

$Q = \{(i, \nu_i)\}$

Randomly select blocks 1,4, 9

m_1	m_2	m_3	m_4	m_5			m_n
σ_1	σ_2	σ_3	σ_4	σ_5	..	…	σ_n



$$\sigma = \prod_{(i,\nu_i)\in Q} {\sigma_i}^{\nu_i}$$

$$\mu = \sum_{(i,\nu_i)\in Q} \nu_i m_i \mod p$$

μ
σ

Linear homomorphic authenticator

Prover computes a combined value of blocks and authenticators

Send μ and σ

verifies using only μ and σ

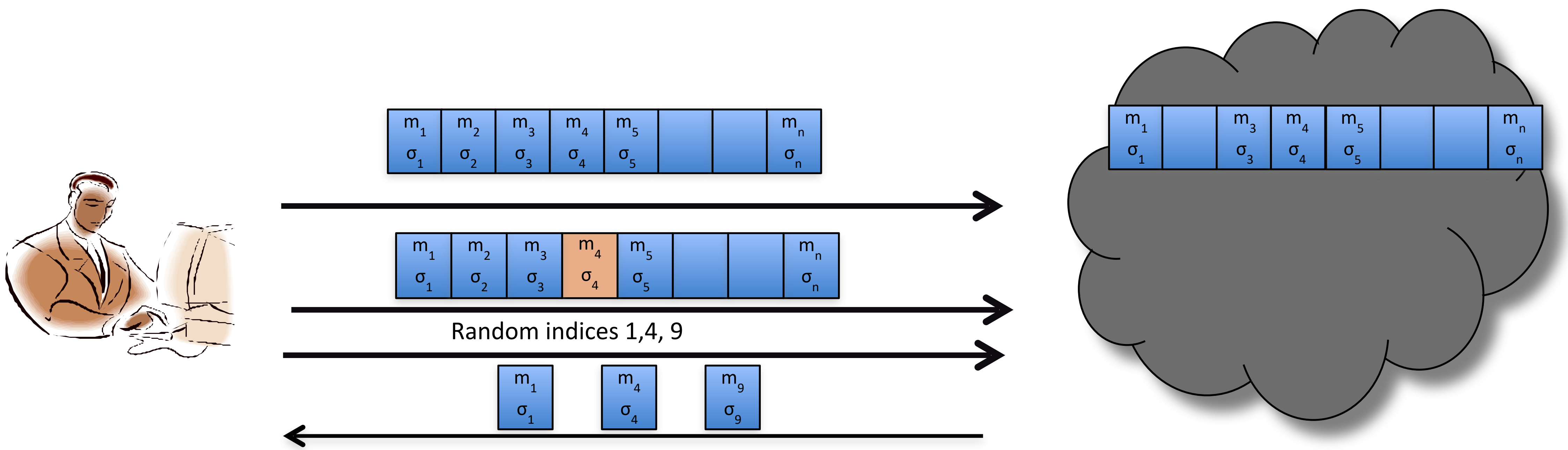
$$e(\sigma, g) \stackrel{?}{=} e\left(\prod_{(i,\nu_i)\in Q} H(F_{id}||i)^{\nu_i} \cdot u^{\mu}, g^{\alpha}\right)$$

For private auditing, σ_i is a MAC, $\sigma_i = f_{k_{prf}}(i) + \alpha m_i \mod p$

for public auditing, σ_i is signature $\sigma_i = \left(H(F_{id}||i)u^{m_i}\right)^{\alpha}$

Shacham, Waters, "Compact proofs of retrievability", JoC'12

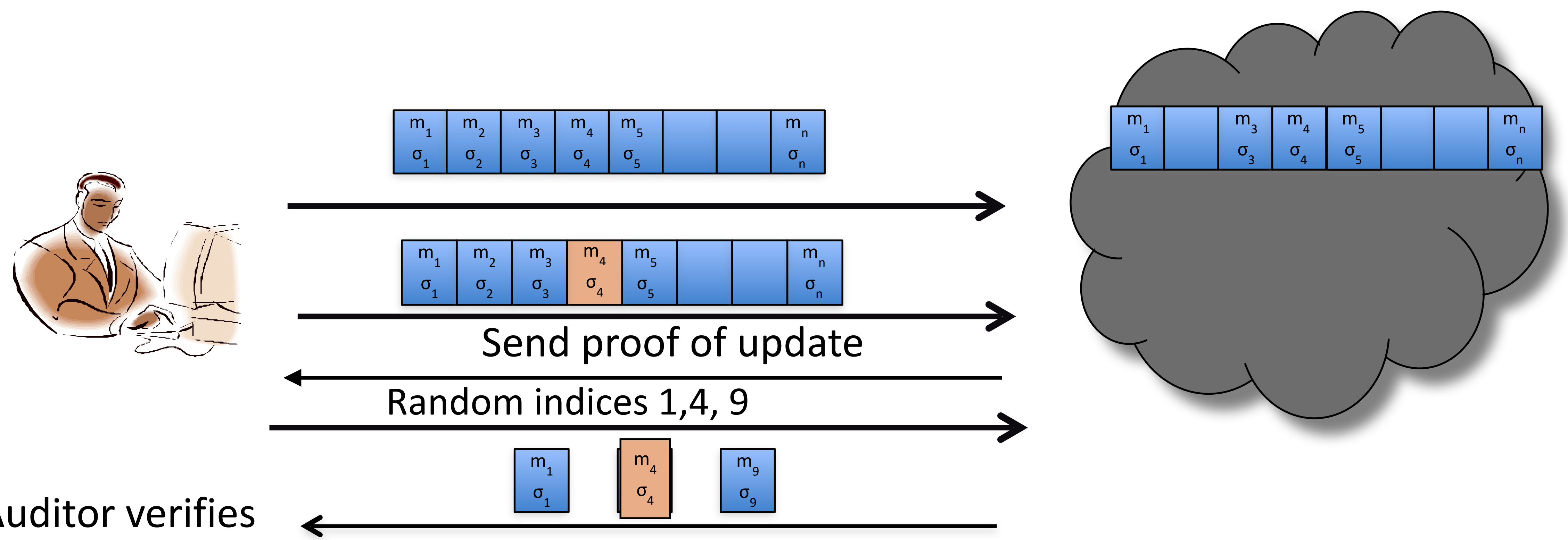
Construction of Dynamic Auditing Schemes



Owner verifies

Even if server does not update file, it passes the audit

Construction of Dynamic Auditing Schemes



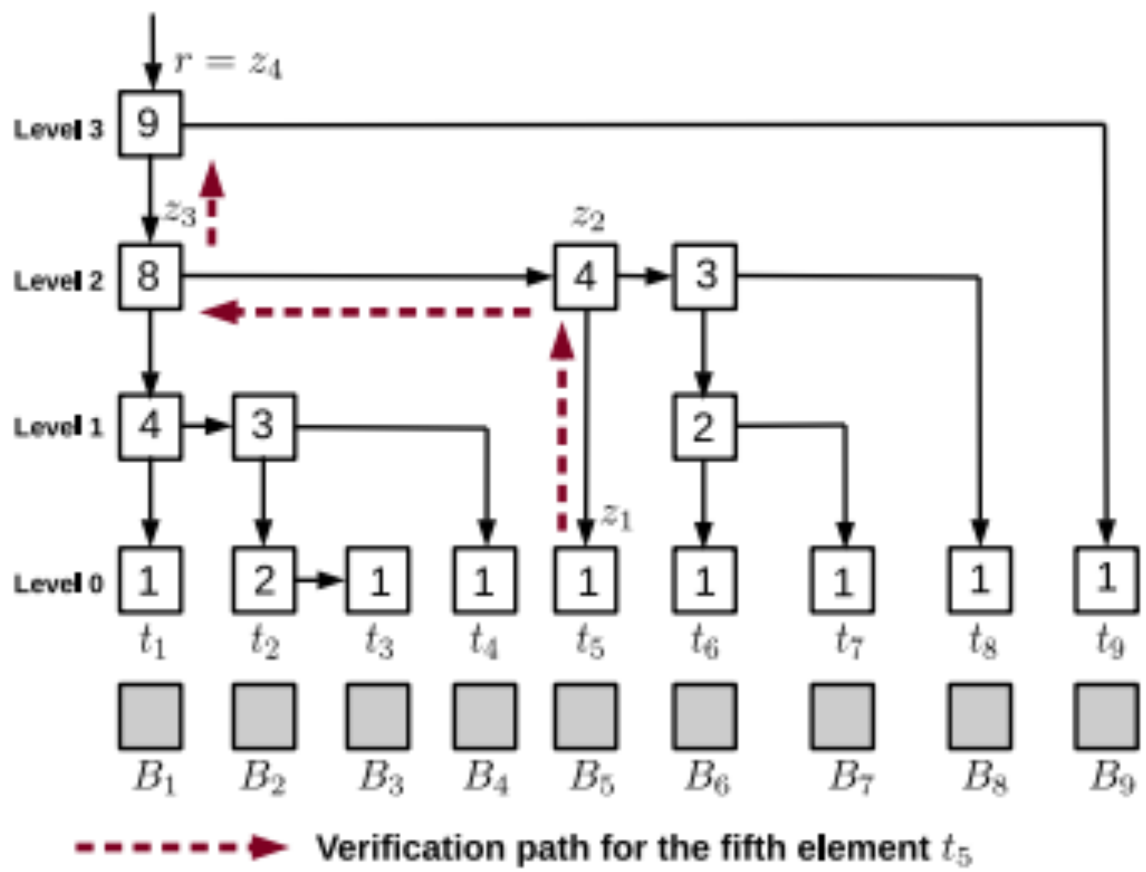
σ_i is the authenticator (unforgeable)

Data Freshness: Proof of update
How does it look like?

Guarantee of Freshness

- There should be a data structure (DS) with the server containing all tags
- When a block is modified, the tag is modified and the change is made in the data structure in the server
- When audits are run, then this data structure is used along with data blocks for a complete proof.
- What Data structures to use?
- Merkle trees are good for static data
- Alternate data structure are Skiplists.

Rank Based Skip Lists

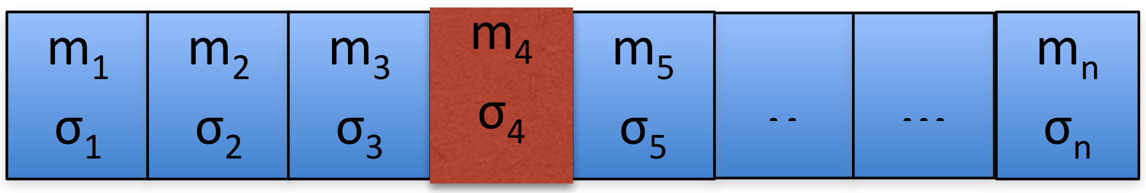


$$f(z) = \begin{cases} 0, & \text{if } z \text{ is } \texttt{null} \\ h(l(z)||rank(z)||x(z)||f(right(z))), & \text{if } l(z) = 0 \\ h(l(z)||rank(z)||f(down(z))||f(right(z))), & \text{if } l(z) > 0. \end{cases}$$

$$\Pi(i) = (A(z_1), \ldots, A(z_k)) : A(z) = (l(z), q(z), d(z), f(z))$$

Inspired by Erway, Kupcu, Papamanthou, Tamassia, “Dynamic Provable Data Possession”, ACM TISEC’15

General Construction for Dynamic Proofs of Storage



$$\sigma_i = f_{k_{prf}}(i) + \alpha m_i \mod p$$
$$\sigma_i = \left(H(F_{id}||i)u^{m_i}\right)^{\alpha}$$

- If insertions are made after the i-th position of the file, then rest of the tags at positions i+1 to n have to be recomputed

  Use signatures/MACs which do not embed the index i

- The freshness of data must be guaranteed.

  Use a dynamic data structure like Skiplist

- Public verifiability

  Use signatures instead of MACs

Wang, Chow, Wang, Ren, Lou, "Privacy-preserving public auditing for secure cloud storage", IEEE Transactions on Computers, 2013

Network Coding Based Dynamic Cloud Storage

$$F = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}$$

- Present general construction of secure cloud auditing scheme from secure network coding
- Present a concrete construction using rank based skip list and network coding signature scheme
- Secure in the standard model
- Efficient than existing schemes
- Privacy-preserving

Security

- Signature scheme of Catalano, Fiore, Warinschi, “ Efficient network coding signatures in the standard model”, PKC 2012.
- Authenticity: Given that underlying signature scheme is unforgeable, the server cannot create fake signatures on messages.
- Freshness: Skip list is updated when data is modified
- Extractability: Given the authentication and freshness guarantee, an adversary can extract the file, by solving linear equations.
- For vector v_i authentication tag is (s_i, t_i), where,
- $s_i \xleftarrow{R} F_e$ $\qquad x_i^e = g^{s_i}(\prod_{j=1}^{n} g_j^{v_{ij}}) h_i \bmod N$

Dynamic Auditing

$\mathrm{KeyGen}(1^\lambda, m, n) \to K = (sk, pk)$

Compute file with tags F’
Build Authenticated rank-based skip list M (on the tags),
root of skip list

d_M

$\mathrm{Outsource}(F, K, \texttt{fid}) \to (F', d_M, M)$

$\mathrm{InitUpdate}(i, \texttt{Insert}, d_M, pk, \texttt{fid})$

i

Get $d_M, \Pi(i)$

Verify d_M
Verify proof $\Pi(i)$

$i, F', M, h', \mathrm{v}', t', pk, \texttt{fid}$

$\mathrm{PerformUpdate}(i, \texttt{Insert}, F', M, h', \mathrm{v}', t', pk, \texttt{fid})$

Get $d'_M, \Pi(i)$

Verify $d'_M, \Pi(i)$

Update d_M

$\mathrm{Challenge}(pk, l, \texttt{fid})$

$Q = \{(i, \nu_i)\}_{i \in I}$

$\mathrm{Prove}(Q, pk, F', M, \texttt{fid})$

$s = \sum_{i \in I} \nu_i s_i \bmod e$

$T_1 = (\mathrm{y}, t), T_2 = \{(t_i, \Pi(i))\}_{i \in I}$

Aggregate tag x, aggregate data y

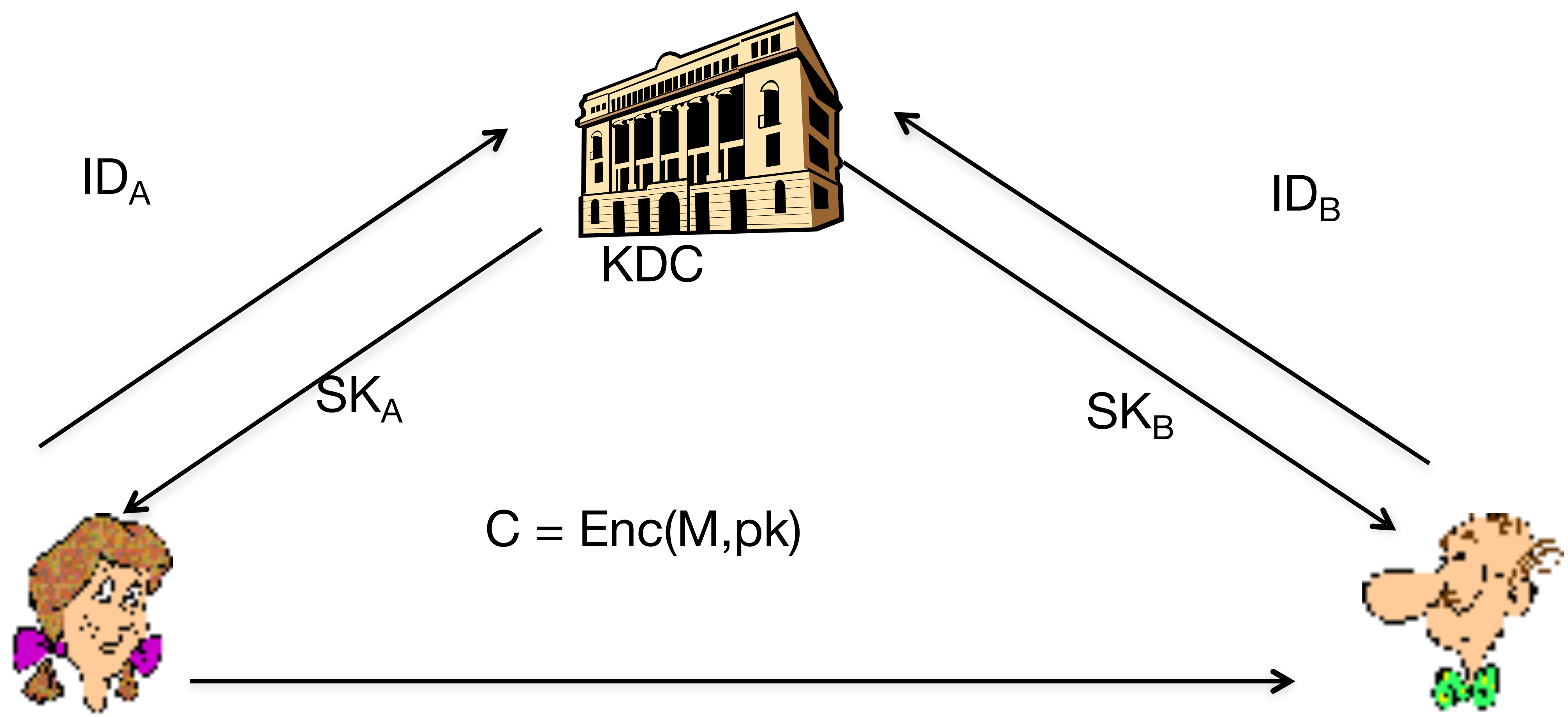
$t = (s, x)$

$\mathrm{Verify}(Q, T, pk, \texttt{fid})$

Access Control

Identity Based Encryption (IBE)

Generates public parameters pk and master secret key MSK



Alice

Bob

$M' = Dec(C, SK_B)$

Assumption: KDC is trusted

Decryption successful if M=M'

IBE: Algorithms

- Setup: Generate system parameters, public key pk and master secret key MSK
- KeyGen: Using MSK and identity of user generates secret key sk
- Encrypt: Given message M and public key pk of receiver, generates ciphertext C
- Decrypt: Given ciphertext C and secret key sk, generates M

Algorithm can be probabilistic, for the same message and same public key, can encrypt() produces different ciphertext C

Social Networks

- Share personal information with certain members
- Hide information from others

Ways to achieve Access Control

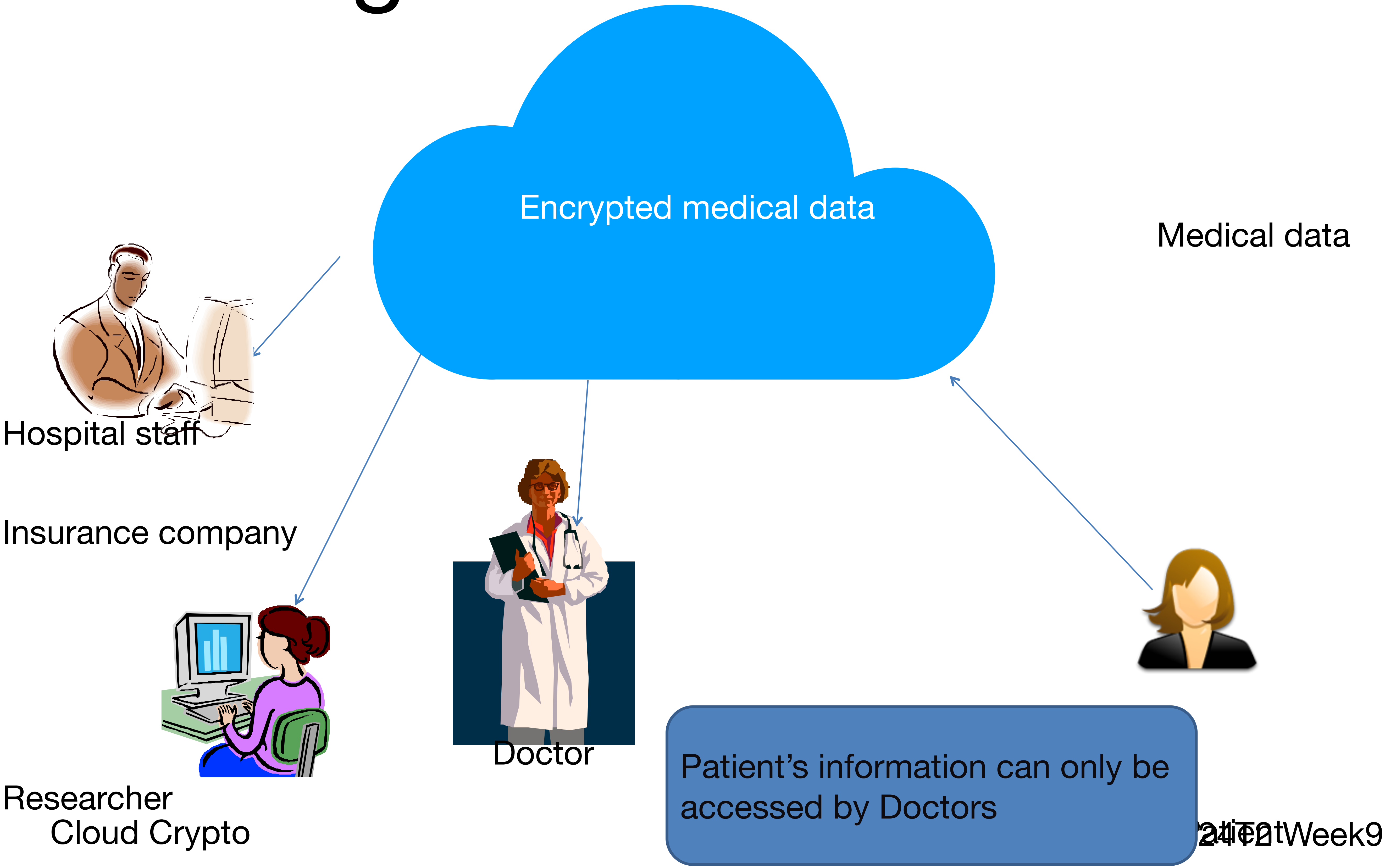
- User based Access Control
  - Access control list attached to data
  - Not a feasible solution when there are many users, example clouds
- Role-Based Access Control
  - Access based on specific role
  -Does not support fine grained access control

- Give each user a public/secret key pair

- Encrypt each message with public key of authorized user
  - Same data has to be encrypted multiple times.

- Attribute based access control
  - Provides fine-grained access control

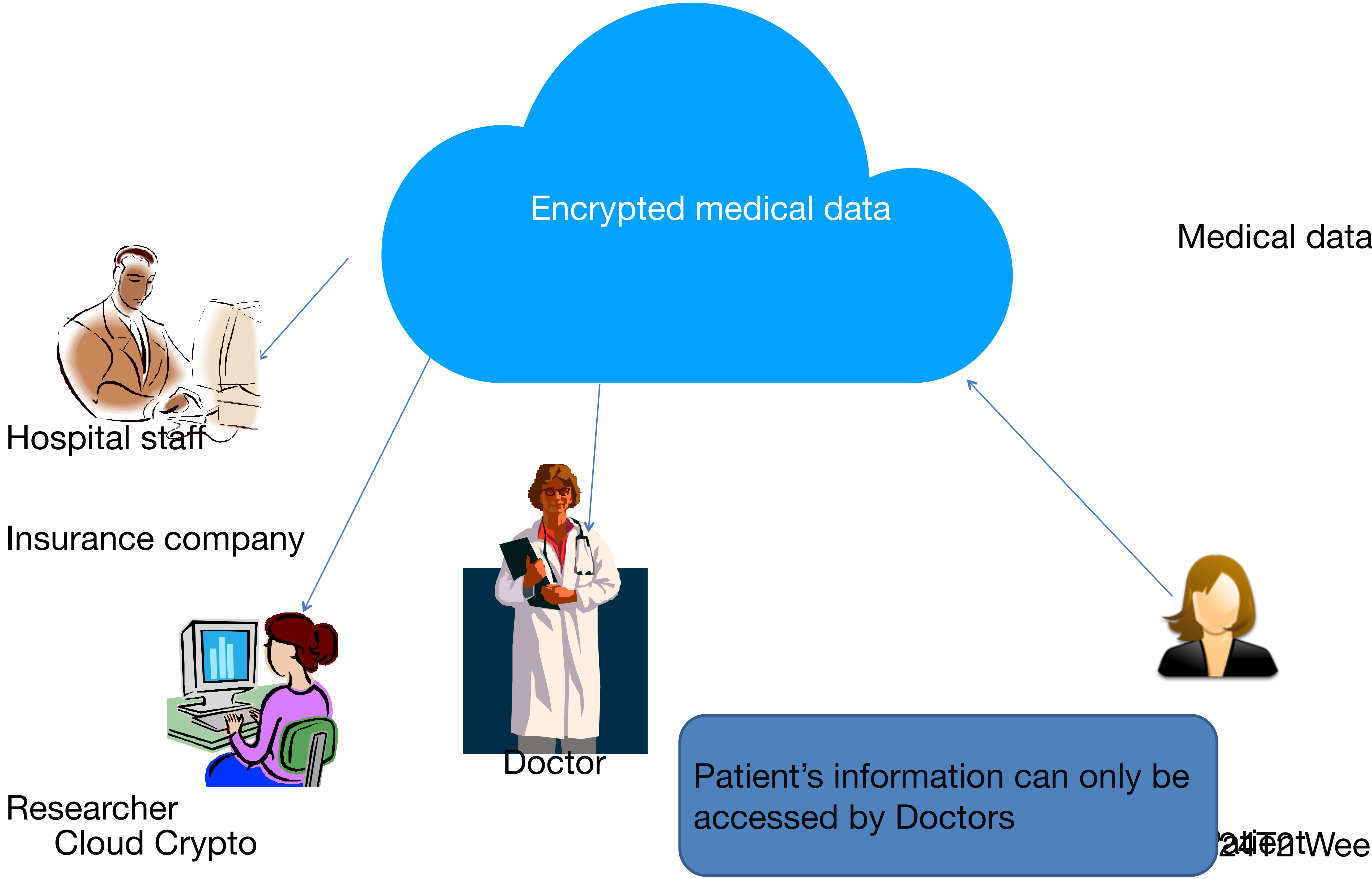
Storage of medical records



What is Attribute Based Access Control?

- Users have attributes rather than roles
- Doctor working between 9 am to 5 pm
- Doctor specializes at cardiology
- Has 10 years experience
- Works in Hospital A and research lab R

Attributes


Encrypted medical data
Medical data
Hospital staff
Insurance company
Researcher
Doctor
Patient
Patient's information can only be accessed by Doctors

Secret Sharing

- Threshold secret sharing (Shamir)
- Dealer has a secret s
- Dealer gives each part a share s
- n parties, at least t parties must collude to construct the secret

Share generation and secret reconstruction

- Dealer randomly choose a polynomial of degree c-1 whose constant term is s and other coefficients in Z_p
- $P(x) = s + c_1x + c_2x^2 + .. + c_{c-1}x^{c-1}$
- Evaluate the polynomial at each point i
- Give share P(i) to party i
- c parties can then collude to compute the polynomial P and s
- Use Lagrange's interpolation to compute P(0)=s

ABE: First Attempt

- c-out-of-n threshold structure
- Choose a secret s
- Blind message M using s, Ciphertext C=Ms
- Choose a polynomial P(x) of degree c-1, such that P(0)=s
- For every attribute i, calculate P(i) and send it to all users who have the attribute i,
- An authorized user with c attributes can compute P(x) and P(0) = s
- Problem: two users can collude and combine their shares and compute s, can compute M

ABE: Second Attempt

- c-out-of-n threshold structure
- Choose a secret s
- Blind message M using s, Ciphertext C=Ms
- For each user u choose a polynomial $P_u(x)$ of degree c-1, such that $P_u(0)=s$
- For every attribute i of user u, calculate $P_u(i)$ and send it securely
- If a user has c attributes, it can construct $P_u(x)$ and $P_u(0) =s$
- Two users can collude, because P_u values are different
- Problems with the two approaches: For every message a secret s is generated and secret keys are distributed accordingly. Highly inefficient.

ABE: Sahai and Water's scheme

- SetUp: Choose prime p group G and G_T of order p, and pairing function e.

  MSK $=\{ t_1, t_2, \ldots t_W , u \varepsilon\ Zp\}$,

$PP=\{pk_1 = T_1=g^{t1}, pk_2 = T_2=g^{t2}\ , \ldots., pk_w=T_w = g^{tw}\ ,Y=e(g,g)^u\}$

- KeyGen: Degree c polynomial p(x), s.t. p(0) = u

Secret keys: $sk_i\ =g$^($p(i)/t_i$), i is an attribute of user

- Encrypt: Choose s, Ciphertext $C = MY^s$, $<c_i = T_i^s,>$ i is an attribute
- Decrypt: For matching set of attributes, calculate

$e(sk_i, c_i) = e(g^{p(i)/t_i}, g\quad) = e(g,g)^{p(i)s}$

If atleast c attributes matching, then calculate $e(g,g)^{p(0)s}$ Lagrange interpolation. So, $Y^s = e(g,g)^{us}$

M can be obtained

Searching on Encrypted Data

- Searching large data bases is a difficult problem
- Searching on encrypted databases is even a bigger challenge
- Known techniques include:

  -property-preserving encryption

  -functional encryption

  -fully-homomorphic encryption

  -searchable symmetric encryption

  -oblivious RAMs

  -secure two-party computation

Important problems not discussed

- Searchable Encryption techniques
- Fully Homomorphic encryption
- Verifiable computation

Thank you!