Data Protection in Cloud Scenarios

Sabrina De Capitani di Vimercati
Dipartimento di Informatica
Università degli Studi di Milano
sabrina.decapitani@unimi.it

Information and Communication Technologies

Advancements in ICTs enable the development of better and more efficient infrastructures and services (often for less cost than in the past)

- improve communication and information services
- facilitate the creation and collection of big data from different sources (e.g., satellite imagery and sensors, smart phones, surveys and census)

Smart home, smart grid, . . .

NEWS

Home | Video | World | UK | Business | Tech | Science | Magazine | Entertainment & Arts

Building cities of the future now

By Jane Wakefield
Technology reporter

21 February 2013 | Technology

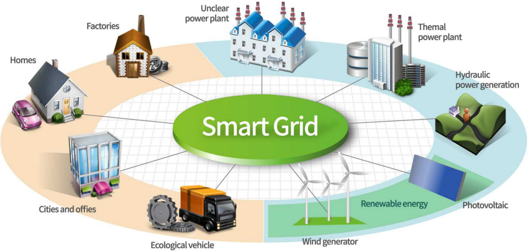
Share

Around the world new cities are being built while those we have lived in for centuries are being upgraded for the future.

It is partly a reaction to over-crowding and pollution and partly because in an ever-connected world it makes increasing sense to hook entire cities up to the network.

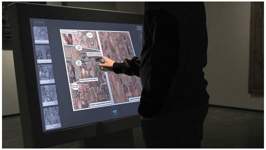
A smarter city may mean one that uses data on traffic to ease congestion or one that aims to join up services to provide better information for citizens. For many it is about

Technology being used in urban communities around the world hints at how we may live in



… Everything is getting smart …


Smart car


Museum and exhibitions


Health Care


Augmented reality


Smart e-commerce

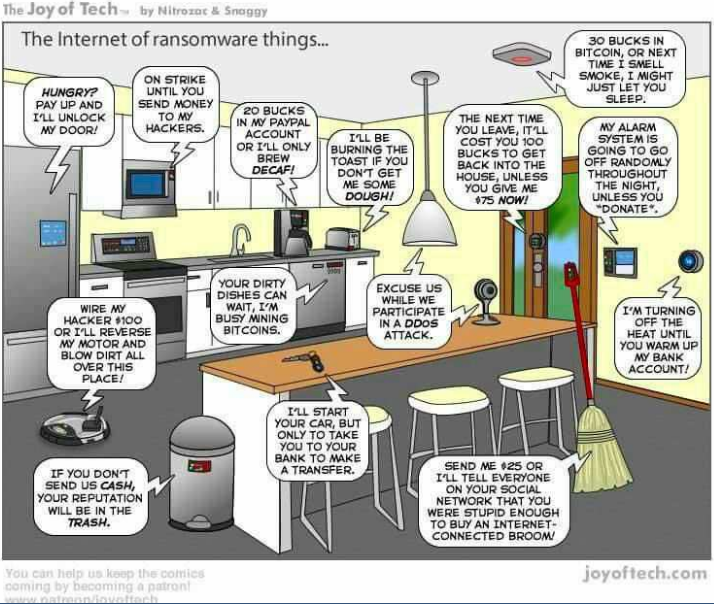

Intelligent shops


Smart entertainment systems


Smart governance


Smart transportation

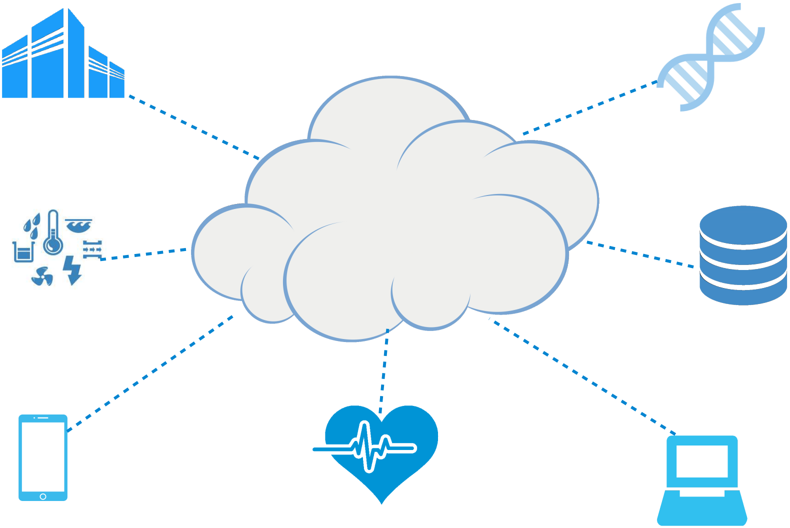
… Maybe too smart?



The data protection challenge

- Huge amount of data collected, generated, and shared
- Growing use of SaaS business applications
- Growing amount of pervasive and mobile applications relying on data availability anytime anywhere

Huge amount of data stored at external providers



Impact on data protection and privacy – 1

theguardian
all sections
port football opinion culture business lifestyle fashion environment tech travel

Samsung's voice-recording smart TVs breach privacy law, campaigners claim

US consumer rights group Epic claims Samsung has breached the privacy of its users, and is demanding an FCC investigation

Safety, Security And Privacy Risks Of Fitness Tracking And 'Quantified Self'

JUL 31, 2014 @ 02:45 PM 11,599 VIEWS

The Little Black Book of Bill

Larry Magid, CONTRIBUTOR
I write about consumer technology: the good, the bad and the ugly. FULL BIO
Opinions expressed by Forbes Contributors are their own.

Symantec SYMC -1.98% is out with a report that raises questions about the safety and security of wearable technology.

BBC Sign in News Sport Weather iPlayer TV Radio
NEWS
Home UK World Business Politics Tech Science Health Education
Technology

Warning over smart meters privacy risk

12 June 2012 | Technology

An EU data watchdog has warned of the "considerable risks" to privacy posed by new energy smart meters.

The European Data Protection Supervisor said safeguards were needed over how firms used the "massive collection" of consumers' data uploaded by meters.

The technology is able to track when

Smart meters are able to upload data about consumers' energy use to third parties

ED MARKEY
United States Senator *for* Massachusetts

Home / News / Press Releases / Press Release

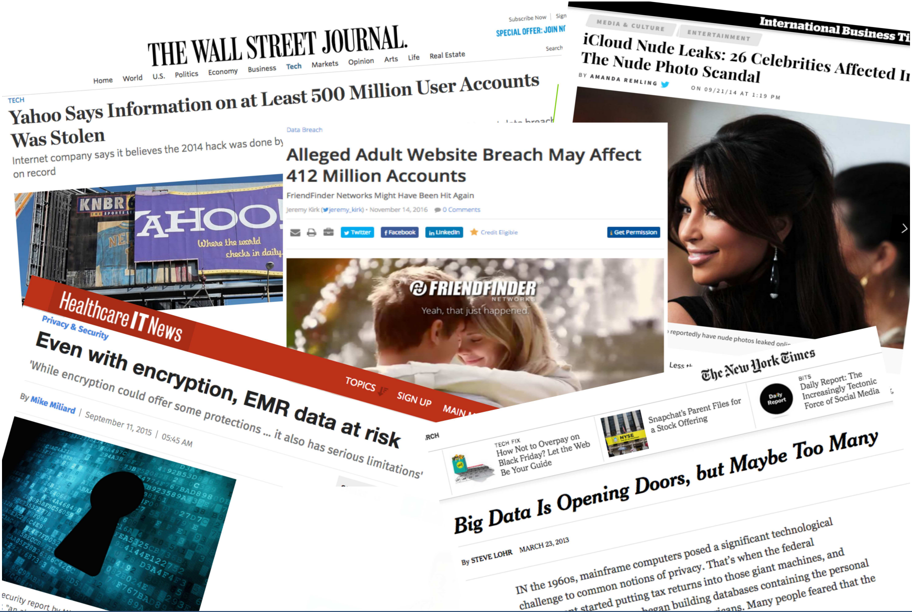
Markey Report Reveals Automobile Security and Privacy Vulnerabilities

Monday, February 9, 2015

Wireless technologies leave vehicles exposed to hackers; Information collected on driver locations, habits

WASHINGTON (February 9, 2014) – New standards are needed to plug security and privacy gaps in our cars and trucks, according to a report released today by Senator Edward J. Markey (D-Mass.). The report, called *Tracking & Hacking: Security & Privacy Gaps Put American Drivers at Risk* and first reported on by CBS News' 60 Minutes, reveals how sixteen major automobile manufacturers responded to questions from Senator Markey in 2014 about how vehicles may be vulnerable to hackers, and how driver information is collected and protected.

Impact on data protection and privacy – 2

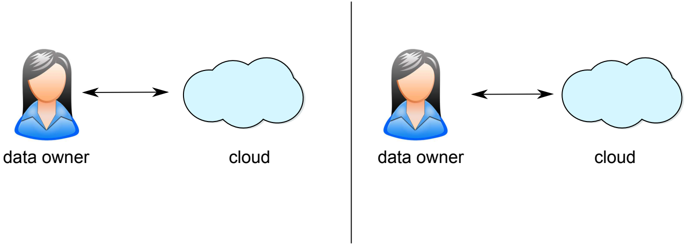


Cloud computing

- 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

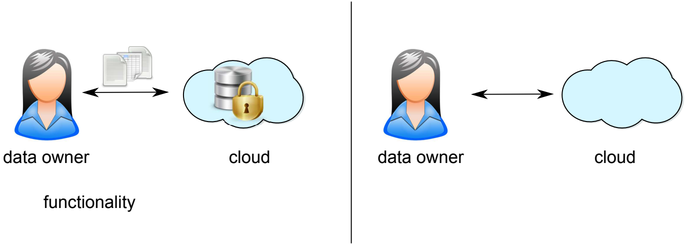
Cloud computing: Today

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

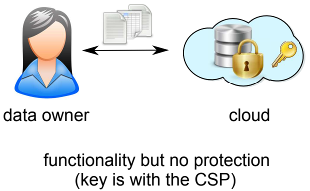
Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders

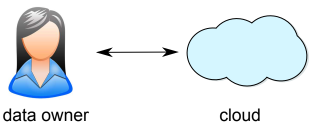


- functionality

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

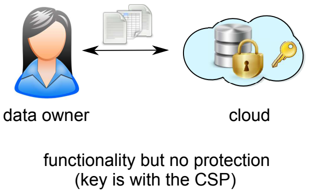


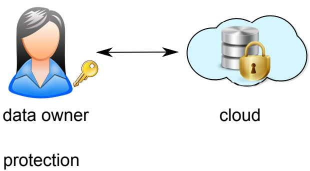


- functionality implies full trust in the CSP that has full access to the data (e.g., Google Cloud Storage, iCloud)

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

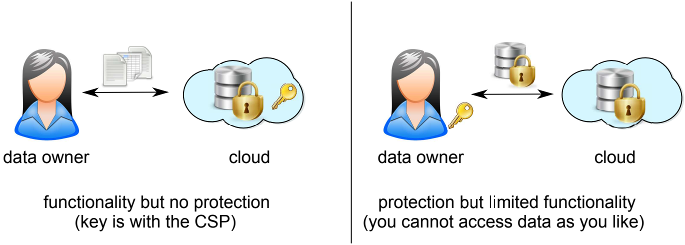




- functionality implies full trust in the CSP that has full access to the data (e.g., Google Cloud Storage, iCloud)
- protection

Cloud computing: Today

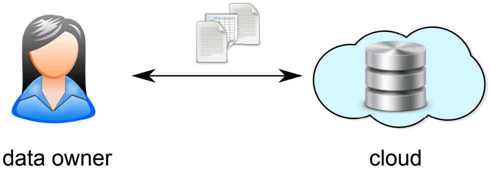
Cloud Service Providers (CSPs) apply security measures in the services they offer but these measures protect only the perimeter and storage against outsiders



- 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

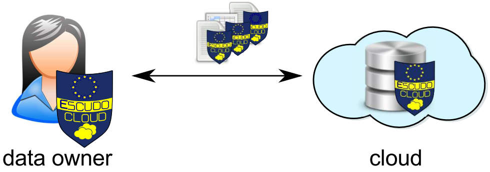


H2020 project "Enforceable Security in the Cloud to Uphold Data Ownership" (ESCUDO-CLOUD).

http://www.escudocloud.eu/

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

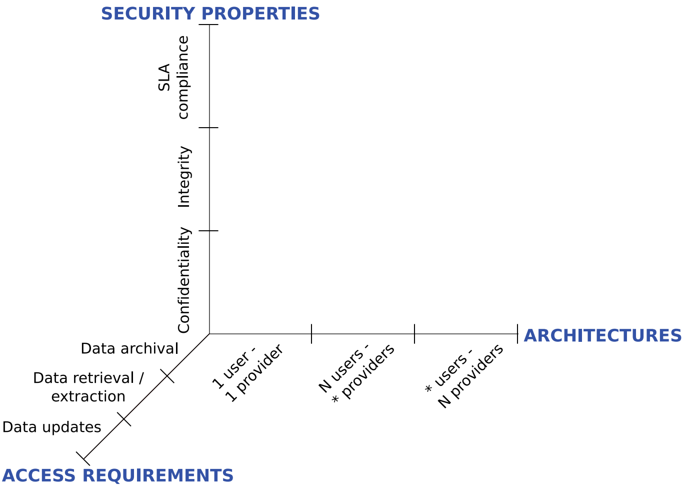
---

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

http://www.escudocloud.eu/

Scientific and technical challenges

Three dimensions characterize the problems and challenges



Security properties

Confidentiality
- data externally stored
- users identities
- actions that users perform on the data

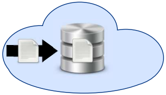
Integrity
- data externally stored
- computation and query results

SLA compliance
- assurance and certification

Access requirements

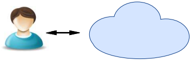
Data archival
- upload/download
- protection of data in storage

Data retrieval/extraction
- support for fine-grained data retrieval and queries
- protection of computations and query results



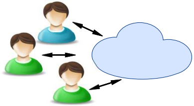
Data update
- support for access retrieval and enforcement of updates
- protection of the actions and of their effects on the data

Architectures



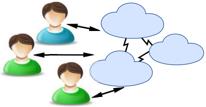
1 user - 1 provider

- protection of data at rest
- fine-grained retrieval
- query privacy/integrity



n users - * providers

- authorizations and access control
- multiple writers



*** users - n providers**

- controlled data sharing and computation

Combinations of the dimensions

- Every combination of the different instances of the dimensions identifies new problems and challenges
- The security properties to be guaranteed can depend on the access requirements and on the trust assumption on the providers involved in storage and/or processing of data
- Providers can be:
  - curious
  - lazy
  - malicious

Some Challenges in Data Protection

Some issues and opportunities

- 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
- Integrity
  - integrity of stored data and query results
- Cloud providers selection

Protection of and Fine-Grained Access to Outsourced Data

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

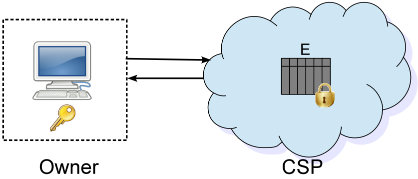
S. De Capitani di Vimercati et al., "Encryption and Fragmentation for Data Confidentiality in the Cloud," in *Foundations of Security Analysis and Design VII,* A. Aldini, J. Lopez, F. Martinelli (eds.), Springer, 2014.

S. De Capitani di Vimercati, S. Foresti, P. Samarati, "Selective and Fine-Grained Access to Data in the Cloud," in *Secure Cloud Computing,* S. Jajodia, K. Kant, P. Samarati, V. Swarup, C. Wang (eds.), Springer, 2014.

The role of encryption in protecting data

- The Cloud Service Provider (CSP) can be honest-but-curious and should not have access to the resource content

- Data confidentiality typically provided by wrapping a layer of encryption around sensitive data (e.g., Boxcryptor, SpiderOak)



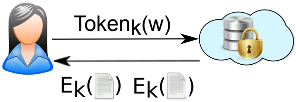
Fine-grained access to data in the cloud

- For confidentiality reasons, CSPs storing data cannot decrypt them for data processing/access

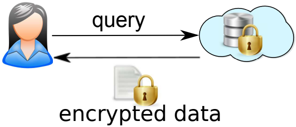
- Need mechanisms to support access to the outsourced data
  - effective and efficient
  - should not open the door to inferences

Fine-grained access: Approaches – 1

- Keyword-based searches directly on the encrypted data: supported by specific cryptographic techniques (e.g., [CWLRL-11])

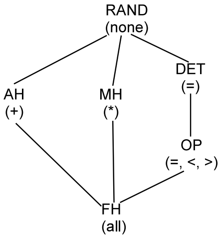


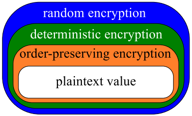
- Homomorphic encryption: supports the execution of operations directly on the encrypted data (e.g., [BV-11,G-09,GSW-13])

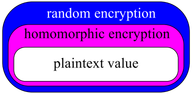


Fine-grained access: Approaches – 2

- Onion encryption (CryptDB): different onion layers each of which supports the execution of a specific SQL operation (e.g., HanaDB SEEED framework) [PRZB-11]
- Encryption schemas: each column can be encrypted with a different encryption schema, depending on the conditions to be evaluated on it (e.g., Google encrypted BigQuery)

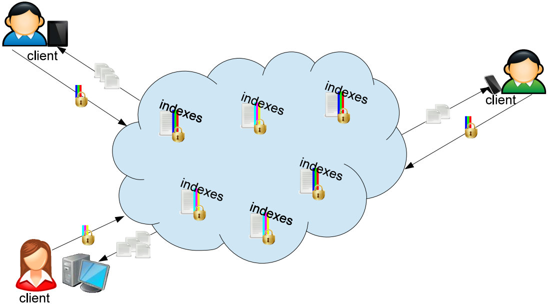






Fine-grained access: Approaches – 3

Indexes (direct 1:1; with collision n:1; flattened 1:n): metadata attached to the data and used for fine-grained information retrieval and query execution (e.g., [SD-16])



Fine-grained access: Approaches – 4

- Indexes associated with attributes are used by the provider to select data to be returned in response to a query

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Query on plaintext translated to a query on indexes and some postprocessing at the client

Fine-grained access: Approaches – 4

- Indexes associated with attributes are used by the provider to select data to be returned in response to a query

Patients

SSN	Name	Illness	Illness
123...89	Alice	**Asthma**	Angel
234...91	Bob	**Asthma**	Angel
345...12	Carol	**Asthma**	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Fred	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Query on plaintext translated to a query on indexes and some postprocessing at the client

Original query

```
SELECT *
FROM Patients
WHERE Illness = 'Asthma'
```

Fine-grained access: Approaches – 4

- Indexes associated with attributes are used by the provider to select data to be returned in response to a query

Patients

SSN	**Name**	**Illness**	**Doctor**
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Fred	Stroke	Ellis

Patients^k

Tid	**Etuple**	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	$\boldsymbol{\alpha}$	δ
2	mNHg1oC010p8w	ϖ	ω	$\boldsymbol{\alpha}$	δ
3	WsIaCvfyF1Dxw	ξ	λ	$\boldsymbol{\alpha}$	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	$\boldsymbol{\alpha}$	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Query on plaintext translated to a query on indexes and some postprocessing at the client

Original query
SELECT *
FROM Patients
WHERE Illness ='Asthma'

At server
r = SELECT Etuple
FROM Patients^k
WHERE $I_I = \alpha$

Fine-grained access: Approaches – 4

- Indexes associated with attributes are used by the provider to select data to be returned in response to a query

Patients

SSN	Name	Illness	Doctor
123…89	Alice	**Asthma**	Angel
234…91	Bob	**Asthma**	Angel
345…12	Carol	**Asthma**	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Fred	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	$\boldsymbol{\alpha}$	δ
2	mNHg1oC010p8w	ϖ	ω	$\boldsymbol{\alpha}$	δ
3	WsIaCvfyF1Dxw	ξ	λ	$\boldsymbol{\alpha}$	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	$\boldsymbol{\alpha}$	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Query on plaintext translated to a query on indexes and some postprocessing at the client

Original query

SELECT Name,Illness
FROM Patients
WHERE Illness ='Asthma'

At server

r = SELECT Etuple
FROM Patients^k
WHERE $I_I = \alpha$

At client

SELECT *
FROM Decrypt(r, *key*)
WHERE Illness = 'Asthma'

Fine-grained access: Approaches – 4

- Indexes associated with attributes are used by the provider to select data to be returned in response to a query

Patients

SSN	Name	Illness	Doctor
123...89	Alice	**Asthma**	Angel
234...91	Bob	**Asthma**	Angel
345...12	Carol	**Asthma**	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Fred	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	$\boldsymbol{\alpha}$	δ
2	mNHg1oC010p8w	ϖ	ω	$\boldsymbol{\alpha}$	δ
3	WsIaCvfyF1Dxw	ξ	λ	$\boldsymbol{\alpha}$	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	$\boldsymbol{\alpha}$	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Query on plaintext translated to a query on indexes and some postprocessing at the client

Original query	At server	At client
SELECT Name,Illness	r = SELECT Etuple	SELECT *
FROM Patients	FROM Patients^k	FROM Decrypt(r, *key*)
WHERE Illness ='Asthma'	WHERE $I_I = \alpha$	WHERE Illness = 'Asthma'

Indexes for queries: Direct (1:1)

Actual value or coding

+ simple and precise for equality queries
– preserves plaintext value distinguishability (inference attacks)

Indexes for queries: Direct (1:1)

Actual value or coding

+ simple and precise for equality queries
– preserves plaintext value distinguishability (inference attacks)

Patients

SSN	Name	Illness	**Doctor**
123…89	Alice	Asthma	Angel
234…91	Bob	Asthma	Angel
345…12	Carol	Asthma	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	$\mathbf{I_D}$
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Direct (1:1)

Actual value or coding

+ simple and precise for equality queries
– preserves plaintext value distinguishability (inference attacks)

Patients

SSN	Name	Illness	**Doctor**
123…89	Alice	Asthma	Angel
234…91	Bob	Asthma	Angel
345…12	Carol	Asthma	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	$\mathbf{I_D}$
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WsIaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Bucket (n:1)

Partition-based or hash-based

+ supports for equality queries
+ collisions remove plaintext distinguishability
– result may contain spurious tuples (postprocessing query)
– still vulnerable to inference attacks

Indexes for queries: Bucket (n:1)

Partition-based or hash-based

- \+ supports for equality queries
- \+ collisions remove plaintext distinguishability
- – result may contain spurious tuples (postprocessing query)
- – still vulnerable to inference attacks

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Bucket (n:1)

Partition-based or hash-based

- \+ supports for equality queries
- \+ collisions remove plaintext distinguishability
- – result may contain spurious tuples (postprocessing query)
- – still vulnerable to inference attacks

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Flattened (1:n)

Flat indexes

- + decreases exposure to inference attacks
- – remains vulnerabile to dynamic observations

Indexes for queries: Flattened (1:n)

Flat indexes

+ decreases exposure to inference attacks
– remains vulnerabile to dynamic observations

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Flattened (1:n)

Flat indexes

+ decreases exposure to inference attacks
– remains vulnerabile to dynamic observations

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Fragmentation and Encryption

V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Selective Data Outsourcing for Enforcing Privacy,” in *Journal of Computer Security (JCS)*, vol. 19, n. 3, 2011.

V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Combining Fragmentation and Encryption to Protect Privacy in Data Storage,” in *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 3, July 2010.

Fragmentation and encryption

- Encryption makes query evaluation and application execution more expensive or not always possible

- Often what is sensitive is the association between values of different attributes, rather than the values themselves
  - e.g., association between employee's names and salaries

  $\Longrightarrow$ protect associations by breaking them, rather than encrypting

- Alternative solutions limit encryption by coupling:
  - encryption
  - data fragmentation

Confidentiality constraints

- Sets of attributes such that the (joint) visibility of values of the attributes in the sets should be protected

- Sensitive attributes: the values of some attributes are considered sensitive and should not be visible
  $\Longrightarrow$ singleton constraints

- Sensitive associations: the associations among values of given attributes are sensitive and should not be visible
  $\Longrightarrow$ non-singleton constraints

Confidentiality constraints – Example

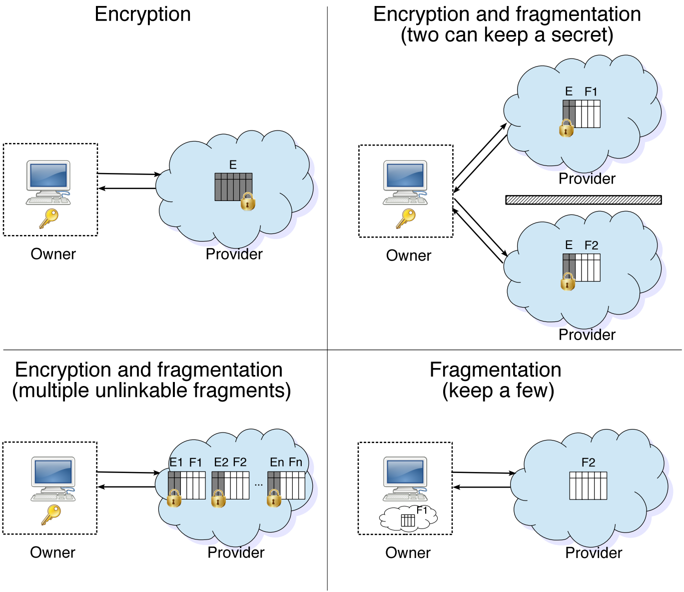
R = (Name,DoB,Gender,Zip,Disease,Doctor,Email,Job,Telephone)

- {Telephone}, {Email}
  - attributes Telephone and Email are sensitive (cannot be stored in the clear)
- {Name,Doctor}, {Name,Disease}, {Name,DoB}
  - attributes Doctor, Disease, and DoB are private of an individual and cannot be stored in the clear in association with the Name
- {DoB,Gender,Zip,Doctor}, {DoB,Gender,Zip,Disease}
  - attributes DoB, Gender, Zip can work as quasi-identifier
- {Zip,Disease}, {Job,Disease}
  - association rules between Zip and Disease and between Job and Disease need to be protected from an adversary

Fragmentation

- Fragmentation partitions attributes of original relation to provide (maximal) availability of attributes in plaintext form for access
  - no sensitive attribute visible in external fragments
  - no sensitive association visible in external fragments
  - ensure unlinkability of fragments (no attribute in common)
- Different approaches:
  - Two can keep a secret splits information over two independent servers that cannot communicate [ABGGKMSTX-05]
  - Multiple unlinkable fragments allows for more than two non-linkable fragments [CDFJPS-10]
  - Keep a few involves the data owner as a trusted party to maintain a limited amount of data [CDFJPS-09, CDFJPS-11]

Fragmentation and encryption: Approaches



Fragmentation and encryption – Examples

P

SSN	Name	YoB	Job	Disease	Doctor

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Two can keep a secret

F_1

tid	Name	YoB	SSN^k	$Disease^k$

F_2

tid	Job	Doctor	SSN^k	$Disease^k$

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Multiple unlinkable fragments

F_1

$salt_1$	enc_1	Name	YoB

F_2

$salt_2$	enc_2	Job	Doctor

F_3

$salt_3$	enc_3	Disease

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Keep a few

F_o

tid	SSN	Name	Disease

F_s

tid	YoB	Job	Doctor

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Fragmentation and encryption – Examples

P

SSN	Name	YoB	Job	Disease	Doctor

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Two can keep a secret

F_1

tid	Name	YoB	SSN^k	$Disease^k$

F_2

tid	Job	Doctor	SSN^k	$Disease^k$

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Multiple unlinkable fragments

F_1

$salt_1$	enc_1	Name	YoB

F_2

$salt_2$	enc_2	Job	Doctor

F_3

$salt_3$	enc_3	Disease

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Keep a few

F_o

tid	SSN	Name	Disease

F_s

tid	YoB	Job	Doctor

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Fragmentation and encryption – Examples

P | SSN | Name | YoB | Job | Disease | Doctor |

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Two can keep a secret

F_1 | tid | Name | YoB | SSN^k | $Disease^k$ |

F_2 | tid | Job | Doctor | SSN^k | $Disease^k$ |

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Multiple unlinkable fragments

F_1 | $salt_1$ | enc_1 | Name | YoB |

F_2 | $salt_2$ | enc_2 | Job | Doctor |

F_3 | $salt_3$ | enc_3 | Disease |

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Keep a few

F_o | tid | SSN | Name | Disease |

F_s | tid | YoB | Job | Doctor |

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Fragmentation and encryption – Examples

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Two can keep a secret

F_1

<u>tid</u>	Name	YoB	SSN^k	$Disease^k$

F_2

<u>tid</u>	Job	Doctor	SSN^k	$Disease^k$

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Multiple unlinkable fragments

F_1

<u>$salt_1$</u>	enc_1	Name	YoB

F_2

<u>$salt_2$</u>	enc_2	Job	Doctor

F_3

<u>$salt_3$</u>	enc_3	Disease

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Keep a few

F_o

<u>tid</u>	SSN	Name	Disease

F_s

<u>tid</u>	YoB	Job	Doctor

c_0 = {SSN}
c_1 = {Name,Doctor}
c_2 = {Name,Disease}
c_3 = {Doctor,Disease}

Query: Two can keep a secret

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

q := SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

Query: Two can keep a secret

P

SSN	Name	YoB	Job	Disease	Doctor

$q :=$ SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

F_1

tid	Name	YoB	SSN^k	$Disease^k$

F_2

tid	Job	Doctor	SSN^k	$Disease^k$

Query: Two can keep a secret

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

q := SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

F_1

<u>tid</u>	Name	YoB	SSN^k	$Disease^k$

F_2

<u>tid</u>	Job	Doctor	SSN^k	$Disease^k$

q_1 := SELECT tid, SSN^k, $Disease^k$
FROM F_1
WHERE Name="Alice"

q_2 := SELECT tid
FROM F_2
WHERE Doctor="Angel"

Query: Two can keep a secret

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

q := SELECT SSN, Disease FROM Patients
WHERE Name=“Alice” AND Doctor=“Angel”

F_1

<u>tid</u>	Name	YoB	SSN^k	$Disease^k$

F_2

<u>tid</u>	Job	Doctor	SSN^k	$Disease^k$

q_1 := SELECT tid, SSN^k, $Disease^k$
FROM F_1
WHERE Name=“Alice”

q_2 := SELECT tid
FROM F_2
WHERE Doctor=“Angel”

Query at client

q_{12} := SELECT Decrypt(SSN^k, k), Decrypt($Disease^k$, k)
FROM q_1 JOIN q_2 ON q_1.tid=q_2.tid

Query: Multiple unlinkable fragments

P

SSN	Name	YoB	Job	Disease	Doctor

$q :=$ SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

Query: Multiple unlinkable fragments

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

$q :=$ SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

F_1

<u>salt$_1$</u>	enc$_1$	Name	YoB

F_2

<u>salt$_2$</u>	enc$_2$	Job	Doctor

F_3

<u>salt$_3$</u>	enc$_3$	Disease

Query: Multiple unlinkable fragments

P

SSN	Name	YoB	Job	Disease	Doctor

q := SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

F_1

salt$_1$	enc$_1$	Name	YoB

F_2

salt$_2$	enc$_2$	Job	Doctor

F_3

salt$_3$	enc$_3$	Disease

q_1 := SELECT salt$_1$, enc$_1$
FROM F_1
WHERE Name="Alice"

Query at client

q' := SELECT SSN, Disease
FROM Decrypt(q_1,*key*)
WHERE Doctor="Angel"

Query: Keep a few

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

q := SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

Query: Keep a few

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

$q :=$ SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

F_o

<u>tid</u>	SSN	Name	Disease

F_s

<u>tid</u>	YoB	Job	Doctor

Query: Keep a few

P

<u>SSN</u>	Name	YoB	Job	Disease	Doctor

q := SELECT SSN, Disease FROM Patients
WHERE Name="Alice" AND Doctor="Angel"

F_o

<u>tid</u>	SSN	Name	Disease

F_s

<u>tid</u>	YoB	Job	Doctor

q_s := SELECT tid
FROM F_s
WHERE Doctor="Angel"

Query at client

q_o := SELECT SSN, Disease
FROM F_o JOIN q_s ON F_o.tid=q_s.tid
WHERE Name="Alice"

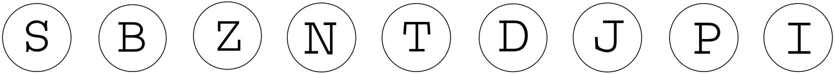
Fragmentation and inference

- Fragmentation assumes attributes to be independent
- In presence of data dependencies:
  - sensitive attributes/associations may be indirectly exposed
  - fragments may be indirectly linkable

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, “Fragmentation in Presence of Data Dependencies,” in *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 11, n. 6, November/December 2014, pp. 510-523.

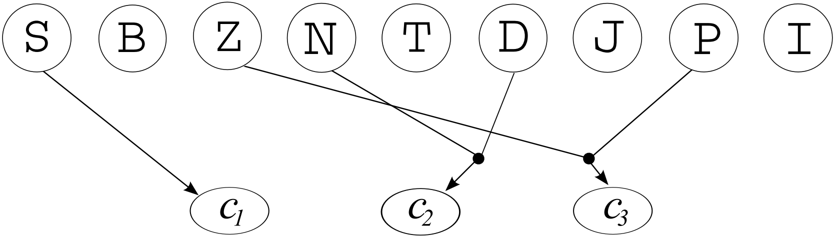
Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)

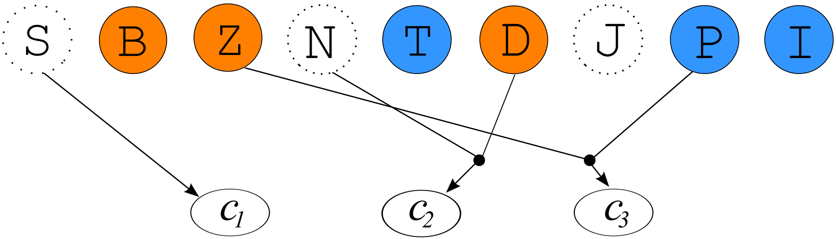


Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)

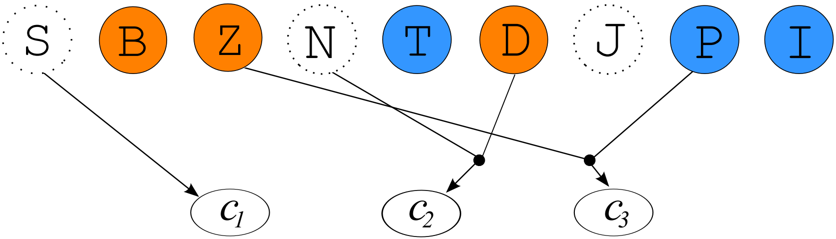


Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

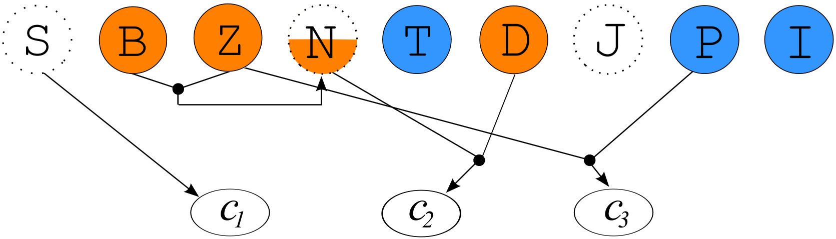
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

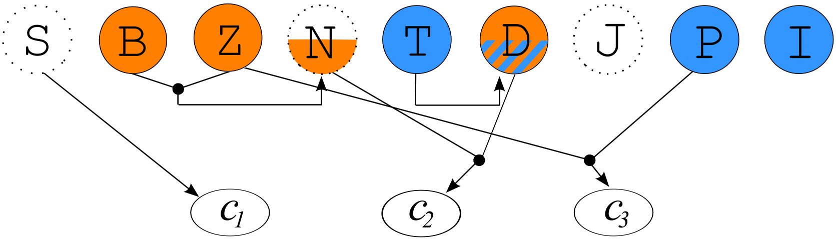
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP}⇝ Name
d_2 = {Treatment}⇝ Disease
d_3 = {Disease}⇝ Job
d_4 = {Insurance, Premium}⇝ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

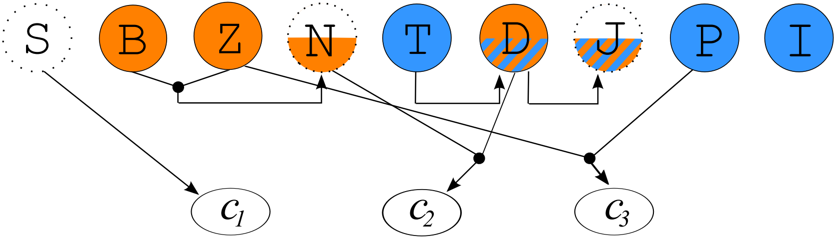
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP}$\rightsquigarrow$ Name
d_2 = {Treatment}$\rightsquigarrow$ Disease
d_3 = {Disease}$\rightsquigarrow$ Job
d_4 = {Insurance, Premium}$\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

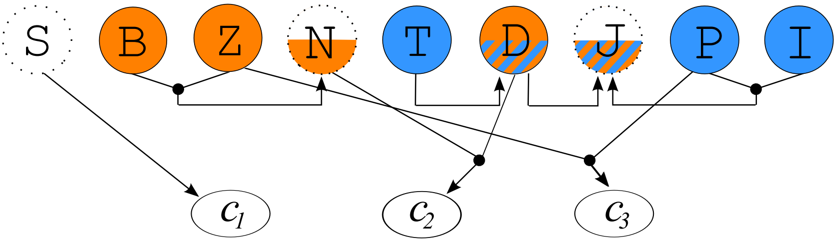
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

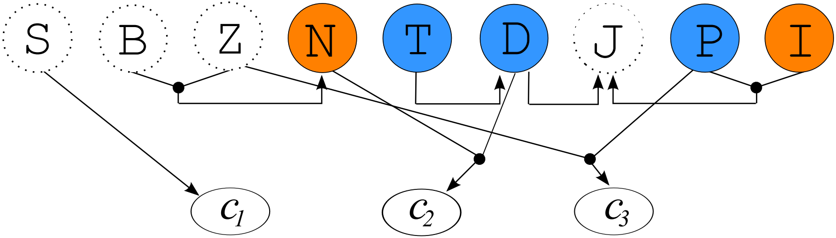
Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmenting with data dependencies

Take into account data dependencies in fragmentation

- Fragments should not contain sensitive attributes/associations neither directly nor indirectly



Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

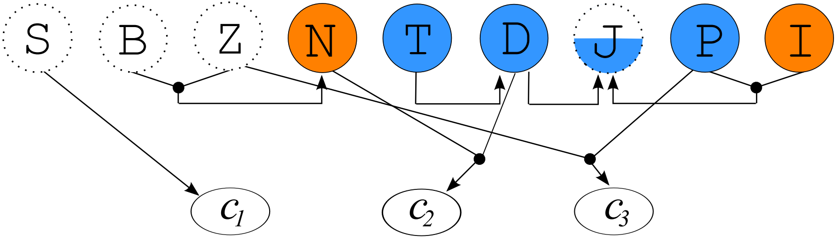
Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmenting with data dependencies

Take into account data dependencies in fragmentation

- Fragments should not contain sensitive attributes/associations neither directly nor indirectly



Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP}$\rightsquigarrow$ Name
d_2 = {Treatment}$\rightsquigarrow$ Disease
d_3 = {Disease}$\rightsquigarrow$ Job
d_4 = {Insurance, Premium}$\rightsquigarrow$ Job

Selective Information Sharing

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 ACM Conference on Computer and Communications Security (CCS 2016)*, Vienna, Austria, October 2016.

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.

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Over-encryption: Management of Access Control Evolution on Outsourced Data,” in *Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007)*, Vienna, Austria, September 2007.

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

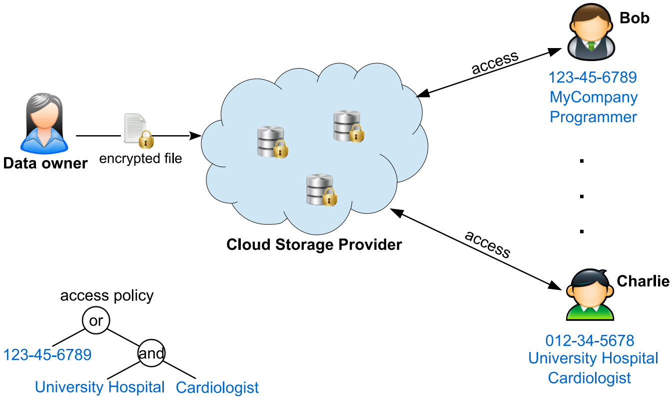
  $\Longrightarrow$ impractical (if not inapplicable)

- 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)



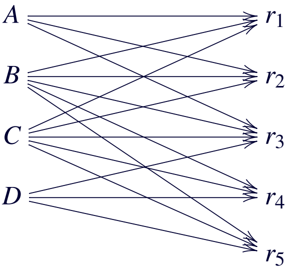
Selective information sharing: Approaches – 2

- Selective 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

Authorization policy

- An authorization policy regulates read access to the resources
- It can be represented as an access matrix or a directed and bipartite graph

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



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 [DFJPS-10, DFJPS-07]
  - 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
- 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

Selective encryption – 2

Token-based key derivation method

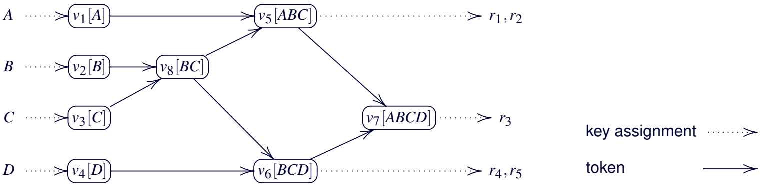
- Keys are arbitrarily assigned to vertices
- A public label l_i is associated with each key k_i
- A piece of public information $t_{i,j}$, called token, is associated with each edge in the hierarchy
- Given an edge (k_i,k_j), token $t_{i,j}$ is computed as $k_j \oplus h(k_i, l_j)$ where
  - $\oplus$ is the n-ary `xor` operator
  - h is a secure hash function
- Advantages of tokens:
  - they are public and allow users to derive multiple encryption keys, while having to worry about a single one
  - they can be stored on the remote server (just like the encrypted data), so any user can access them

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 (to limit the number of tokens additional keys might be inserted for 'factoring' derivation paths)

Selective encryption – Example



- user A can access $\{r_1, r_2, r_3\}$
- users B and C can access $\{r_1, r_2, r_3, r_4, r_5\}$
- user D can access $\{r_3, r_4, r_5\}$

Construction of the key and token graph

Start from an authorization policy $\mathscr{A}$

1. Create a vertex/key for each user and for each non-singleton *acl* (initialization)

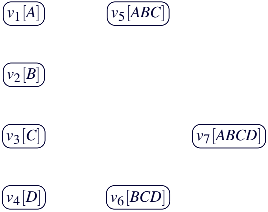
2. For each vertex v corresponding to a non-singleton *acl*, find a cover without redundancies (covering)
   - for each user u in v.*acl*, find an ancestor v' of v with $u \in v'$.*acl*

3. Factorize common ancestors (factorization)

Key and token graph – Example

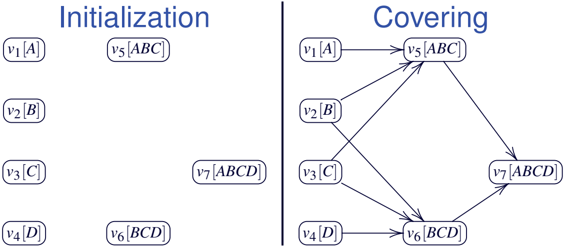
	r_1	r_2	r_3	r_4	r_5
A	1	1	1	0	0
B	1	1	1	1	1
C	1	1	1	1	1
D	0	0	1	1	1

Initialization



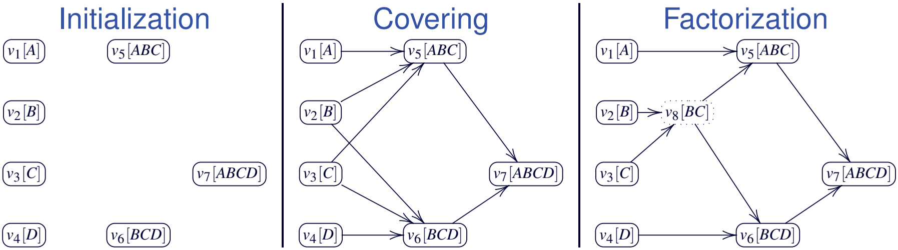
Key and token graph – Example

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

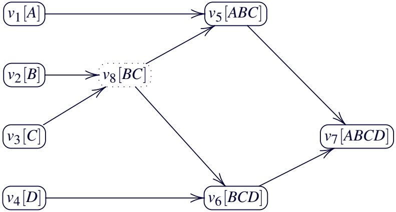


Key and token graph – Example

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



Key assignment and encryption schema ϕ and catalog



u	$\phi(u)$
A	$v_1.l$
B	$v_2.l$
C	$v_3.l$
D	$v_4.l$

r	$\phi(r)$
r_1	$v_5.l$
r_2	$v_5.l$
r_3	$v_7.l$
r_4,r_5	$v_6.l$

source	destination	token_value
$v_1.l$	$v_5.l$	$t_{1,5}$
$v_2.l$	$v_8.l$	$t_{2,8}$
$v_3.l$	$v_8.l$	$t_{3,8}$
$v_4.l$	$v_6.l$	$t_{4,6}$
$v_5.l$	$v_7.l$	$t_{5,7}$
$v_6.l$	$v_7.l$	$t_{6,7}$
$v_8.l$	$v_5.l$	$t_{8,5}$
$v_8.l$	$v_6.l$	$t_{8,6}$

Over-encryption

Policy updates

- When authorizations dynamically change, the data owner needs to:
  - download the resource from the server
  - 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 server and communicate the public catalog updates

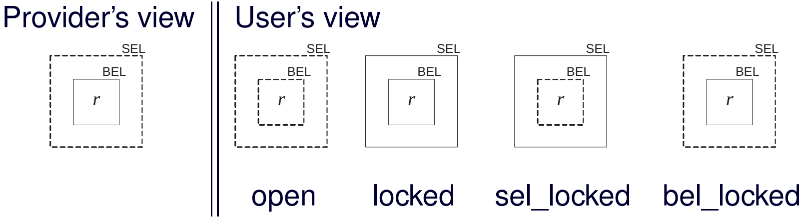
  $\Longrightarrow$ inefficient

- Possible solution: over-encryption [DFJPS-10a, DFJPS-07]

Over-encryption – 1

- Resources are encrypted twice:
  - by the owner, with a key shared with the users and unknown to the server (Base Encryption Layer - BEL level)
  - by the server, 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 update of the SEL level according to the operations performed

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

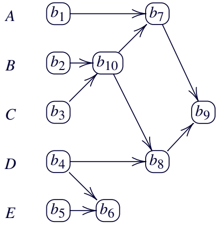
  to protect resources for which the revokee has the BEL key

- 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

An example of revoke operation

BEL	SEL



r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4,r_5	$b_6.l_a$

$s_1[A]$

$s_2[B]$

$s_3[C]$

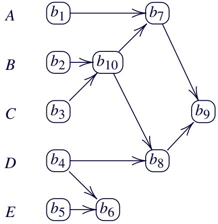
$s_4[D]$

$s_5[E]$

r	$\phi_S(r)$
r_1,r_2,r_3,r_4,r_5	NULL

An example of revoke operation

BEL	SEL
revoke(B, r_3)	

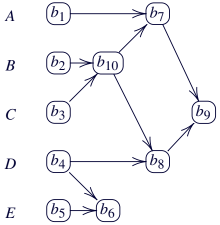


r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4,r_5	$b_6.l_a$

r	$\phi_S(r)$
r_1,r_2,r_3,r_4,r_5	NULL

An example of revoke operation

BEL	SEL
revoke(B, r_3)	**over_encrypt**(CD, r_3)

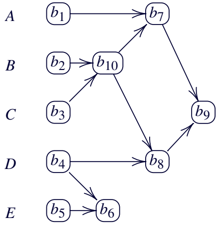


r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4,r_5	$b_6.l_a$

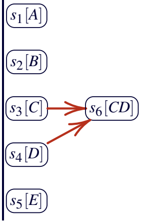
r	$\phi_S(r)$
r_1,r_2,r_3,r_4,r_5	NULL

An example of revoke operation

BEL	SEL
revoke(B, r_3)	**over_encrypt**(CD, r_3)



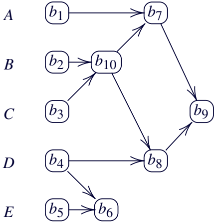
r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4, r_5	$b_6.l_a$



r	$\phi_s(r)$
$r_1, r_2, \sout{r_3}, r_4, r_5$	NULL
r_3	$s_6.l$

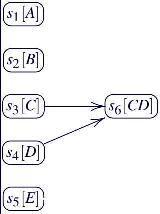
An example of grant operation

BEL



r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4,r_5	$b_6.l_a$

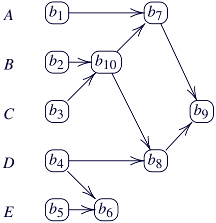
SEL



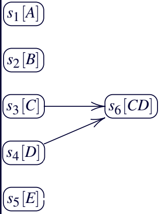
r	$\phi_S(r)$
r_1,r_2,r_4,r_5	NULL
r_3	$s_6.l$

An example of grant operation

BEL	SEL
grant(C, r_4)	



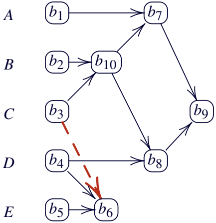
r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4, r_5	$b_6.l_a$



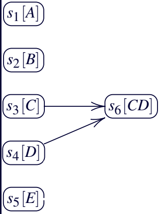
r	$\phi_S(r)$
r_1, r_2, r_4, r_5	NULL
r_3	$s_6.l$

An example of grant operation

BEL	SEL
grant(C, r_4)	



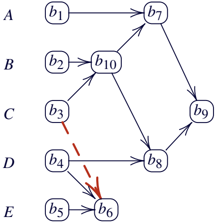
r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4, r_5	$b_6.l_a$



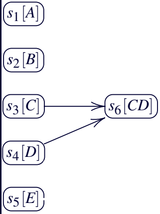
r	$\phi_S(r)$
r_1, r_2, r_4, r_5	NULL
r_3	$s_6.l$

An example of grant operation

BEL	SEL
grant(C, r_4)	**over_encrypt**(DE, r_5)



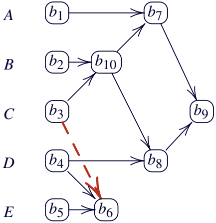
r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4,r_5	$b_6.l_a$



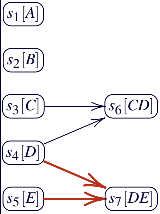
r	$\phi_S(r)$
r_1,r_2,r_4,r_5	NULL
r_3	$s_6.l$

An example of grant operation

BEL	SEL
grant(C, r_4)	**over_encrypt**(DE, r_5)



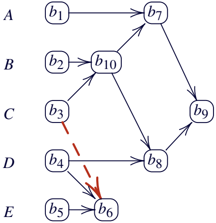
r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4, r_5	$b_6.l_a$



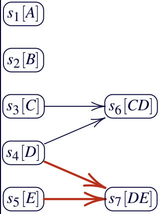
r	$\phi_S(r)$
$r_1, r_2, r_4,$ ~~r_5~~	NULL
r_3	$s_6.l$
r_5	$s_7.l$

An example of grant operation

BEL	SEL
grant(C, r_4)	**over_encrypt**(DE, r_5) **over_encrypt**(ALL,r_4)



r	$\phi_b(r)$
r_1	$b_7.l_a$
r_2	$b_9.l_a$
r_3	$b_8.l_a$
r_4, r_5	$b_6.l_a$



r	$\phi_S(r)$
$r_1, r_2, r_4,$ ~~r_5~~	NULL
r_3	$s_6.l$
r_5	$s_7.l$

Variations/open issues…

- Support of write authorizations [DFJLPS-13]
- Support of multi-owners scenario [DFJPPS-10]
- Integration with current cloud technology [BDFGPRSS-16]
- Different approaches for over-encryption enforcement (immediate, on-the-fly, opportunistic) [BDFPRS-16b]
- Selective encryption for access control combined with indexes for query execution [DFJPS-11]

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

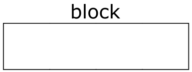
  $\Longrightarrow$ unavailability of a small portion of the encrypted resource prevents its (even partial) reconstruction

- 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

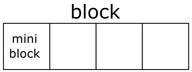
- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits



Resource organization

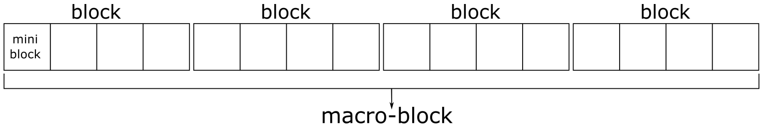
- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits

- 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



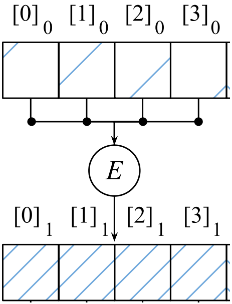
Resource organization

- 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



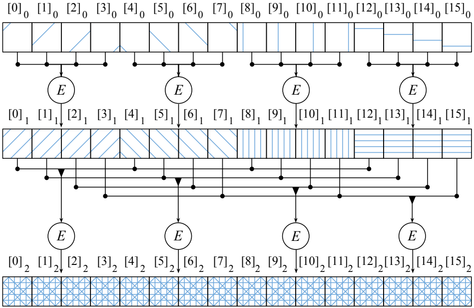
Mixing – 1

- 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

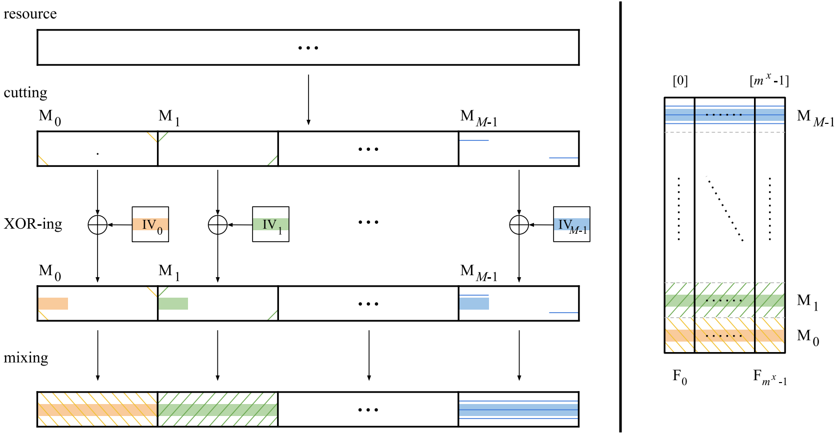


Slicing – 1

- 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

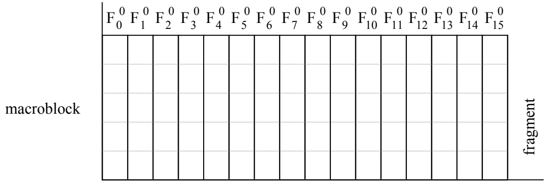
- Slice resources in fragments having a mini-block for each macro-block (the ones in the same position)
  - absence of a fragment prevents reconstruction of the resource



Revoke

To revoke user u access to a resource r

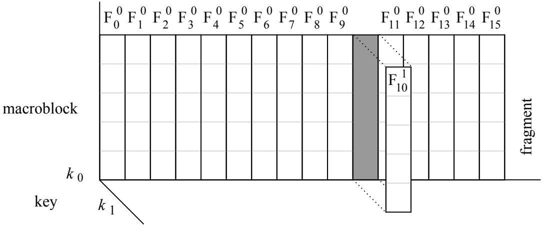
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

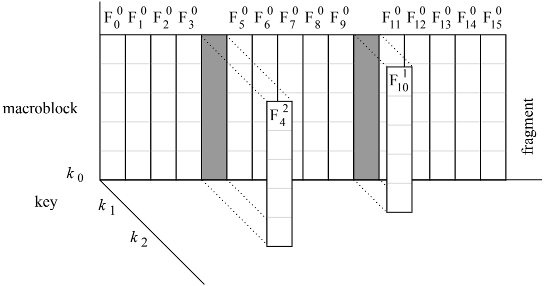
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

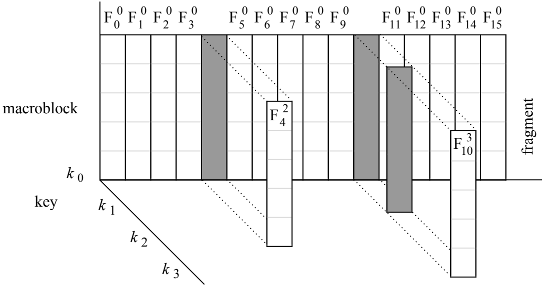
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



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 (generated with key regression and seed encrypted with new ACL)
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
  - a brute force attack is needed 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

Integrity of Data Storage and Computation

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, "Integrity for Distributed Queries,"in *Proc. of the 2nd IEEE Conference on Communications and Network Security (CNS 2014)*, CA, USA, October 2014.

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Integrity for Join Queries in the Cloud," in *IEEE Transactions on Cloud Computing (TCC),* vol. 1, n. 2, July-December 2013, pp. 187-200.

Integrity of storage and query computation

- Data owner and users need mechanisms that provide integrity for query results:
  - correctness: computed on genuine data
  - completeness: computed on the whole data collection
  - freshness: computed on the most recent version of the data
- Two approaches:
  - deterministic: uses authenticated data structures (e.g., signature chains, Merkle hash trees, skip lists) or encryption-based solutions (e.g., verifiable homomorphic encryption schema [LDPW-14])
  - probabilistic: exploits insertion of fake tuples in query results, replication of tuples in query results, pre-computed tokens (e.g., [DFJPS-13b,DFJPS-14,DFJLPS-14b,XWYM-07])

Merkle hash tree

- Binary tree where:
  - each leaf contains the hash of one tuple
  - each internal node contains the result of the hash of the concatenation of its children
- The hash function used to build the tree is collision-resistant
- The root is signed by the data owner and communicated to authorized users
- Tuples in the leaves are ordered according to the value of the attribute A on which the tree is defined
- The tree is created by the data owner and stored at the server

Merkle hash tree – Example

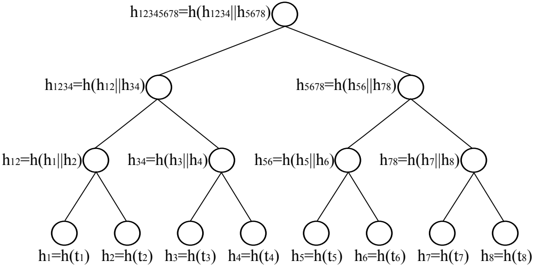
Accounts

	<u>Account</u>	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500

Merkle hash tree – Example

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500



Merkle hash tree over attribute Account

Merkle hash tree verification

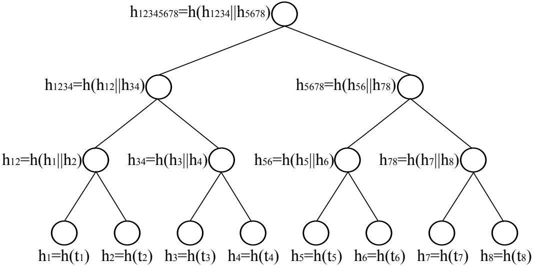
- The Merkle hash tree defined over A supports the verification of equality and range queries over A
- The server returns, together with the query result, a verification object (hash of other tuples allowing the derivation of the hash of the root)
- The client uses the verification object and query result to recompute the root of the tree
- The query result is correct and complete iff the computed root is the same as the one she knows
  - if a tuple is not correct or is missing from the query result, the recomputed root value is not the same as the one known to the client

Merkle hash tree verification – Example

SELECT *
FROM Accounts
WHERE Account = 'Acc3'

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500

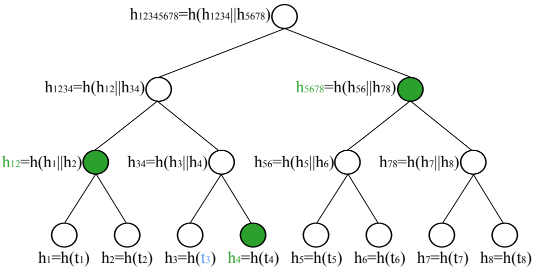


Merkle hash tree verification – Example

SELECT *
FROM Accounts
WHERE Account = 'Acc3'

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500



Result: t_3

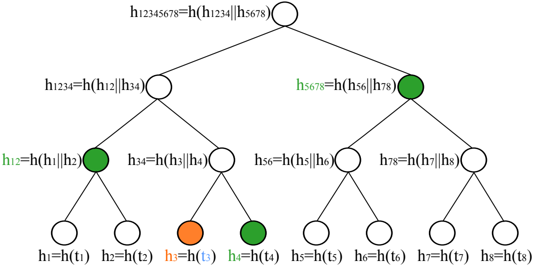
Verification Object: h_4, h_{12}, h_{5678}

Merkle hash tree verification – Example

SELECT *
FROM Accounts
WHERE Account = 'Acc3'

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500



Result: t_3

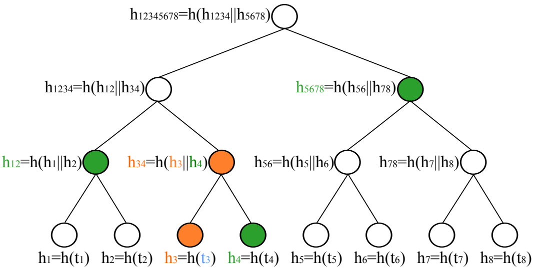
Verification Object: h_4, h_{12}, h_{5678}
$h_3 = h(t_3)$

Merkle hash tree verification – Example

SELECT *
FROM Accounts
WHERE Account = 'Acc3'

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500



Result: t_3

Verification Object: h_4, h_{12}, h_{5678}

$h_3 = h(t_3)$

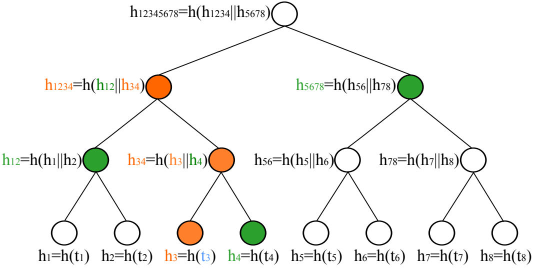
$h_{34} = h(h_3||h_4)$

Merkle hash tree verification – Example

SELECT *
FROM Accounts
WHERE Account = 'Acc3'

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500



Result: t_3

Verification Object: h_4, h_{12}, h_{5678}

$h_3 = h(t_3)$

$h_{34} = h(h_3||h_4)$

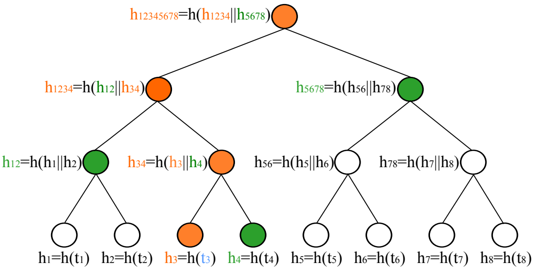
$h_{1234} = h(h_{12}||h_{34})$

Merkle hash tree verification – Example

SELECT *
FROM Accounts
WHERE Account = 'Acc3'

Accounts

	Account	Customer	Balance
t_1	Acc1	Alice	100
t_2	Acc2	Alice	200
t_3	Acc3	Bob	300
t_4	Acc4	Chris	200
t_5	Acc5	Donna	400
t_6	Acc6	Elvis	200
t_7	Acc7	Frank	100
t_8	Acc8	Gary	500



Result: t_3

Verification Object: h_4, h_{12}, h_{5678}

$h_3 = h(t_3)$

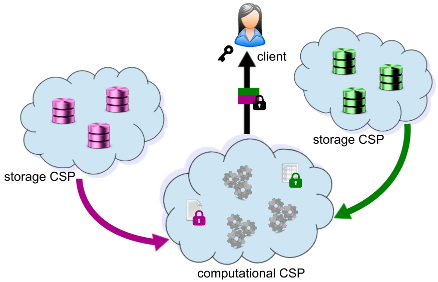
$h_{34} = h(h_3||h_4)$

$h_{1234} = h(h_{12}||h_{34})$

$h_{12345678} = h(h_{1234}||h_{5678})$

Computation with multiple providers

- Different CSPs are available on the market, offering a variety of services (e.g., storage, computation) at different prices
- Users can select the CSP that better matches their security, economic, and functional requirements
- Multiple CSPs can help enhancing security but
  $\Longrightarrow$ need solutions to verify the correct behavior of these CSPs



Probabilistic approach for join queries

- A client, with the cooperation of the storage servers, can assess the integrity of joins performed by a computational cloud
- Protection techniques [DFJPS-13b,DFJPS-14]:
  - encryption makes data unintelligible
  - markers, fake tuples not recognizable as such by the computational cloud (and not colliding with real tuples)
  - twins, replication of existing tuples
- A marker missing or a twin appearing solo $\Longrightarrow$ integrity violation
- Probabilistic guarantee depending on the amount of control (markers and twins) inserted

On-the-fly encryption

- Server S encrypts $B(I, \mathit{Att})$, obtaining $B_k(I_k, B.\mathit{Tuple}_k)$
  - For each t in B, there is τ in B_k: $\tau[I_k]=E_k(t[I])$ and $\tau[B.\mathit{Tuple}_k]=E_k(t)$
  - E is a symmetric encryption function with key k
  - k is defined by the client and changes at every query
- Encryption provides data confidentiality

L

	I	Attr
l_1	a	Ann
l_2	b	Beth
l_3	c	Cloe

R

	I	Attr
r_1	a	flu
r_2	a	asthma
r_3	b	ulcer
r_4	e	hernia
r_5	e	flu
r_6	e	cancer

J

	L.I	L.Attr	R.I	R.Attr	
l_1	a	Ann	a	flu	r_1
l_1	a	Ann	a	asthma	r_2
l_2	b	Beth	b	ulcer	r_3

On-the-fly encryption

- Server S encrypts $B(I, Att)$, obtaining $B_k(I_k, B.Tuple_k)$
  - For each t in B, there is τ in B_k: $\tau[I_k]=E_k(t[I])$ and $\tau[B.Tuple_k]=E_k(t)$
  - E is a symmetric encryption function with key k
  - k is defined by the client and changes at every query
- Encryption provides data confidentiality

L_k

I_k	L.Tuple$_k$
α	λ_1
β	λ_2
γ	λ_3

R_k

I_k	R.Tuple$_k$
α	ρ_1
α	ρ_2
β	ρ_3
ε	ρ_4
ε	ρ_5
ε	ρ_6

J_k

L.I$_k$	L.Attr$_k$	R.I$_k$	R.Attr$_k$
α	λ_1	α	ρ_1
α	λ_1	α	ρ_2
β	λ_2	β	ρ_3

Markers

- Artificial tuples injected into L by S_l and R by S_r
  - not recognizable by the computational server
  - do not generate spurious tuples
  - inserted in a concerted manner to guarantee that they belong to the join result
- The absence of markers signals incompleteness of the join result

L

	I	Attr
l_1	a	Ann
l_2	b	Beth
l_3	c	Cloe

R

	I	Attr
r_1	a	flu
r_2	a	asthma
r_3	b	ulcer
r_4	e	hernia
r_5	e	flu
r_6	e	cancer

J

	L.I	L.Attr	R.I	R.Attr	
l_1	a	Ann	a	flu	r_1
l_1	a	Ann	a	asthma	r_2
l_2	b	Beth	b	ulcer	r_3

Markers

- Artificial tuples injected into L by S_l and R by S_r
  - not recognizable by the computational server
  - do not generate spurious tuples
  - inserted in a concerted manner to guarantee that they belong to the join result
- The absence of markers signals incompleteness of the join result

L^*

	I	Attr
l_1	a	Ann
l_2	b	Beth
l_3	c	Cloe
m_1	x	*marker$_1$*

R^*

	I	Attr
r_1	a	flu
r_2	a	asthma
r_3	b	ulcer
r_4	e	hernia
r_5	e	flu
r_6	e	cancer
m_2	x	*marker$_2$*

J^*

	L.I	L.Attr	R.I	R.Attr	
l_1	a	Ann	a	flu	r_1
l_1	a	Ann	a	asthma	r_2
l_2	b	Beth	b	ulcer	r_3
m_1	x	*marker$_1$*	x	*marker$_2$*	m_2

Twins

- Duplicates of tuples that satisfy condition C_{twin} that
  - is defined on the join attribute I
  - tunes the percentage p_t of twins
  - is defined by the client and communicated to S_l and S_r
- Twin pairs are not recognizable by the computational server (join attribute concatenated with a flag set to 1)
- A twin appearing solo signals incompleteness of the join result

L

	I	Attr
l_1	a	Ann
l_2	b	Beth
l_3	c	Cloe

R

	I	Attr
r_1	a	flu
r_2	a	asthma
r_3	b	ulcer
r_4	e	hernia
r_5	e	flu
r_6	e	cancer

J

	L.I	L.Attr	R.I	R.Attr	
l_1	a	Ann	a	flu	r_1
l_1	a	Ann	a	asthma	r_2
l_2	b	Beth	b	ulcer	r_3

Twins

- Duplicates of tuples that satisfy condition C_{twin} that
  - is defined on the join attribute I
  - tunes the percentage p_t of twins
  - is defined by the client and communicated to S_l and S_r
- Twin pairs are not recognizable by the computational server (join attribute concatenated with a flag set to 1)
- A twin appearing solo signals incompleteness of the join result

L^*

	I	Attr
l_1	a	Ann
l_2	b	Beth
l_3	c	Cloe
$\bar{l}_2$	$\bar{\text{b}}$	Beth

R^*

	I	Attr
r_1	a	flu
r_2	a	asthma
r_3	b	ulcer
r_4	e	hernia
r_5	e	flu
r_6	e	cancer
$\bar{r}_3$	$\bar{\text{b}}$	ulcer

J^*

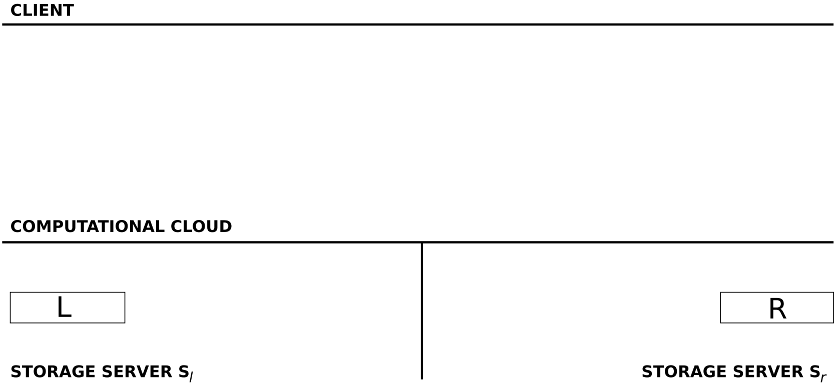
	L.I	L.Attr	R.I	R.Attr	
l_1	a	Ann	a	flu	r_1
l_1	a	Ann	a	asthma	r_2
l_2	b	Beth	b	ulcer	r_3
$\bar{l}_2$	$\bar{\text{b}}$	Beth	$\bar{\text{b}}$	ulcer	$\bar{r}_3$

Query evaluation

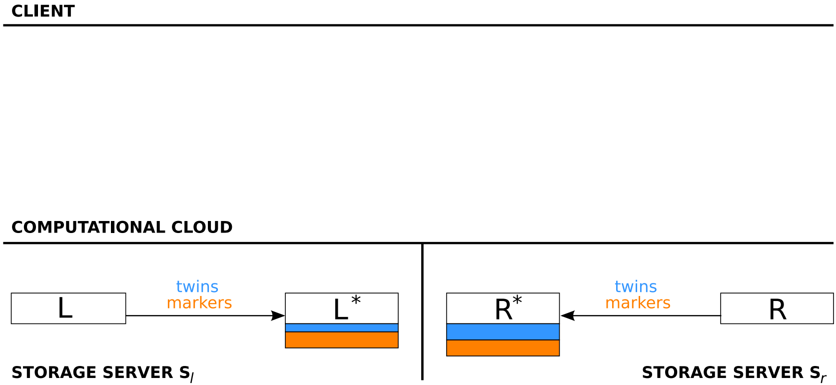
The client shares with each server S_i a symmetric key k_i

- The client sends to the computational cloud a request to execute a join between the relations produced by S_l and S_r
- The relations to be produced by S_l and S_r are represented as two strings, encrypted with keys k_l and k_r, respectively, and to be forwarded by the computational cloud to the respective storage server, containing:
  - subquery to be executed by the storage server
  - query key k (on-the-fly encryption) to be used by the storage server to encrypt the relation sent to the computational cloud
  - number m of markers (random generator known to the servers)
  - percentage p_t of twins (condition over hash values)

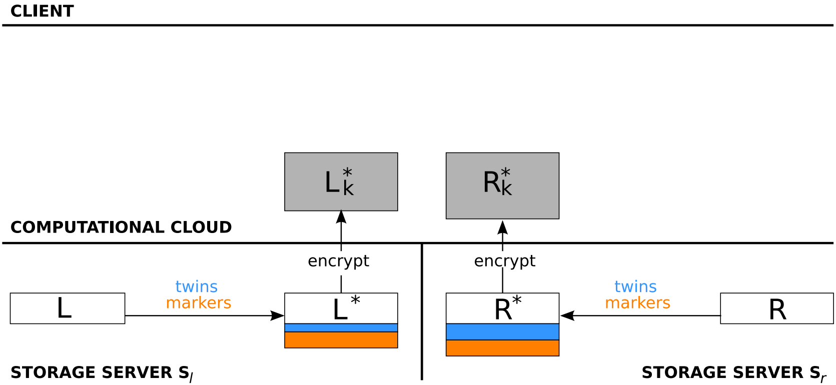
Query execution – Example



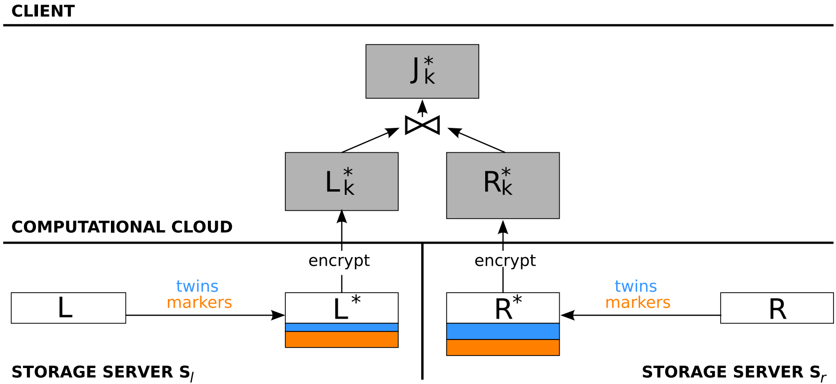
Query execution – Example



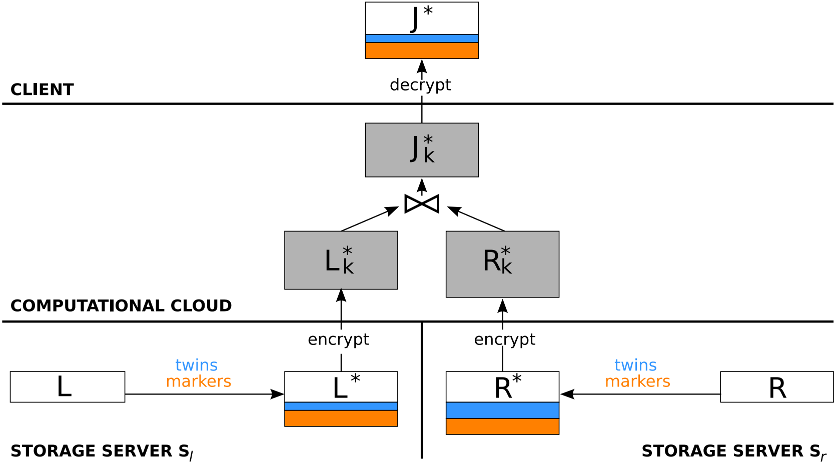
Query execution – Example



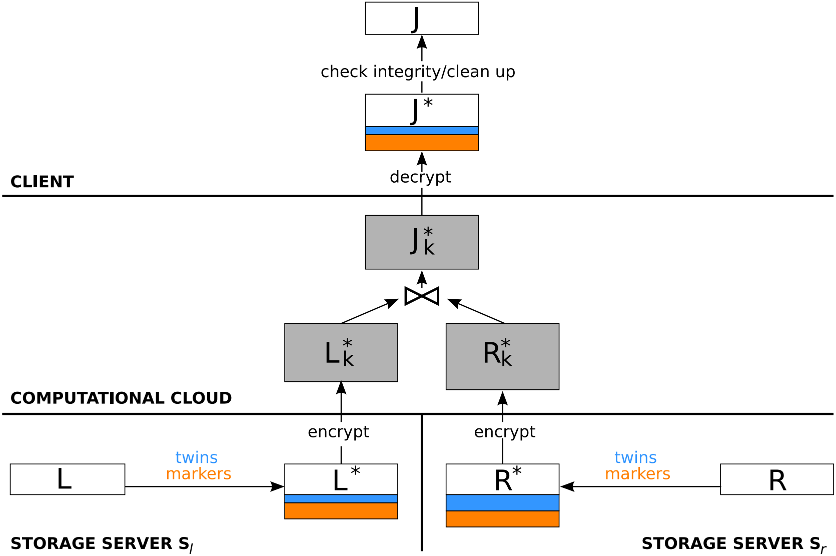
Query execution – Example



Query execution – Example



Query execution – Example



Join execution – Example

L

	I	Attr
l_1	a	Alice
l_2	b	Bob
l_3	c	Carol

R

I	Attr	
a	300	r_1
b	800	r_2
e	200	r_3

Storage servers

Join execution – Example

L

	I	Attr
l_1	a	Alice
l_2	b	Bob
l_3	c	Carol
$\bar{l}_1$	ā	Alice
$\bar{l}_3$	c̄	Carol
m_1	x	$marker_1$
m_3	y	$marker_3$

R

I	Attr	
a	300	r_1
b	800	r_2
e	200	r_3
ā	300	$\bar{r}_1$
x	$marker_2$	m_2
y	$marker_4$	m_4

Storage servers

Join execution – Example

L

	I	Attr
l_1	a	Alice
l_2	b	Bob
l_3	c	Carol
$\bar{l}_1$	ā	Alice
$\bar{l}_3$	c̄	Carol
m_1	x	$marker_1$
m_3	y	$marker_3$

R

I	Attr	
a	300	r_1
b	800	r_2
e	200	r_3
ā	300	$\bar{r}_1$
x	$marker_2$	m_2
y	$marker_4$	m_4

Storage servers

L_k^*

I_k	$L^*.Tuple_k$
α	λ_1
β	λ_2
γ	λ_3
$\bar{\alpha}$	$\bar{\lambda}_1$
$\bar{\gamma}$	$\bar{\lambda}_3$
χ	μ_1
ψ	μ_3

R_k^*

I_k	$R^*.Tuple_k$
α	ρ_1
β	ρ_2
ε	ρ_3
$\bar{\alpha}$	$\bar{\rho}_1$
χ	μ_2
ψ	μ_4

Computational server

Join execution – Example

L

	I	Attr
l_1	a	Alice
l_2	b	Bob
l_3	c	Carol
$\bar{l}_1$	ā	Alice
$\bar{l}_3$	c̄	Carol
m_1	x	$marker_1$
m_3	y	$marker_3$

R

I	Attr	
a	300	r_1
b	800	r_2
e	200	r_3
ā	300	$\bar{r}_1$
x	$marker_2$	m_2
y	$marker_4$	m_4

Storage servers

L_k^*

I_k	L^*.$Tuple_k$
α	λ_1
β	λ_2
γ	λ_3
$\bar{\alpha}$	$\bar{\lambda}_1$
$\bar{\gamma}$	$\bar{\lambda}_3$
χ	μ_1
ψ	μ_3

R_k^*

I_k	R^*.$Tuple_k$
α	ρ_1
β	ρ_2
ε	ρ_3
$\bar{\alpha}$	$\bar{\rho}_1$
χ	μ_2
ψ	μ_4

J_k^*

I_k	L^*.$Tuple_k$	R^*.$Tuple_k$
α	λ_1	ρ_1
β	λ_2	ρ_2
$\bar{\alpha}$	$\bar{\lambda}_1$	$\bar{\rho}_1$
χ	μ_1	μ_2
ψ	μ_3	μ_4

Computational server

Join execution – Example

Storage servers

L

	I	Attr
l_1	a	Alice
l_2	b	Bob
l_3	c	Carol
$\bar{l}_1$	ā	Alice
$\bar{l}_3$	c̄	Carol
m_1	x	$marker_1$
m_3	y	$marker_3$

R

I	Attr	
a	300	r_1
b	800	r_2
e	200	r_3
ā	300	$\bar{r}_1$
x	$marker_2$	m_2
y	$marker_4$	m_4

Client

J^*

	$L^*.I$	$L^*.Attr$	$R^*.I$	$R^*.Attr$	
l_1	a	Alice	a	300	r_1
l_2	b	Bob	b	800	r_2
$\bar{l}_1$	ā	Alice	ā	300	$\bar{r}_1$
m_1	x	$marker_1$	x	$marker_2$	m_2
m_3	y	$marker_3$	y	$marker_4$	m_4

Computational server

L^*_k

I_k	$L^*.Tuple_k$
α	λ_1
β	λ_2
γ	λ_3
$\bar{\alpha}$	$\bar{\lambda}_1$
$\bar{\gamma}$	$\bar{\lambda}_3$
χ	μ_1
ψ	μ_3

R^*_k

I_k	$R^*.Tuple_k$
α	ρ_1
β	ρ_2
ε	ρ_3
$\bar{\alpha}$	$\bar{\rho}_1$
χ	μ_2
ψ	μ_4

J^*_k

I_k	$L^*.Tuple_k$	$R^*.Tuple_k$
α	λ_1	ρ_1
β	λ_2	ρ_2
$\bar{\alpha}$	$\bar{\lambda}_1$	$\bar{\rho}_1$
χ	μ_1	μ_2
ψ	μ_3	μ_4

Join execution – Example

Storage servers

L

	I	Attr
l_1	a	Alice
l_2	b	Bob
l_3	c	Carol
$\bar{l}_1$	ā	Alice
$\bar{l}_3$	c̄	Carol
m_1	x	$marker_1$
m_3	y	$marker_3$

R

I	Attr	
a	300	r_1
b	800	r_2
e	200	r_3
ā	300	$\bar{r}_1$
x	$marker_2$	m_2
y	$marker_4$	m_4

Client

J

	$R_l.I$	$R_l.Attr$	$R_r.I$	$R_r.Attr$	
l_1	a	Alice	a	300	r_1
l_2	b	Bob	b	800	r_2

Computational server

L^*_k

I_k	$L^*.Tuple_k$
α	λ_1
β	λ_2
γ	λ_3
$\bar{\alpha}$	$\bar{\lambda}_1$
$\bar{\gamma}$	$\bar{\lambda}_3$
χ	μ_1
ψ	μ_3

R^*_k

I_k	$R^*.Tuple_k$
α	ρ_1
β	ρ_2
ε	ρ_3
$\bar{\alpha}$	$\bar{\rho}_1$
χ	μ_2
ψ	μ_4

J^*_k

I_k	$L^*.Tuple_k$	$R^*.Tuple_k$
α	λ_1	ρ_1
β	λ_2	ρ_2
$\bar{\alpha}$	$\bar{\lambda}_1$	$\bar{\rho}_1$
χ	μ_1	μ_2
ψ	μ_3	μ_4

Markers and twins: Integrity guarantees

- The guarantee offered by markers and twins can be measured as the probability of the computational cloud to go undetected when omitting tuples

- Markers and twins offer complementary protection:
  - Twins are twice as effective as markers, but loose their effectiveness when the computational cloud omits a large fraction of tuples (extreme case: all tuples omitted)
  - Markers allow detecting extreme behavior (all tuples omitted) and provide effective when the computational cloud omits a large fraction of tuples

Variations/open issues . . .

- Use of salts and buckets to protect join profile [DFJPS-16]
- Application of the salts and buckets only to twins and markers (verification object) [DFJPS-16]
- Execution of the join as a semi-join to support n:m joins and protect join profile [DFJPS-16]
- Application of the techniques in a distributed computation scenario (e.g., MapReduce) [DFJLPS-14b]
- Evaluation of approximate joins [DFJPS-15]
- Consideration of different trust levels
- Removal of trust assumptions in the storage servers

Users Requirements and Preferences for Cloud Plan Selection

S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, P. Samarati, “Supporting User Requirements and Preferences in Cloud Plan Selection,” in *IEEE Transactions on Services Computing (TSC)*, November 2017.

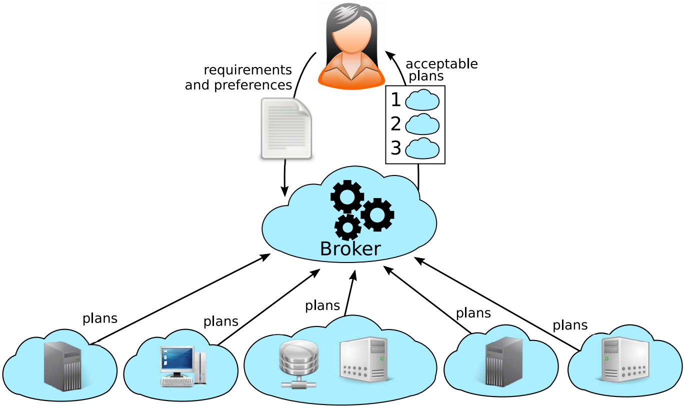
Supporting user requirements and preferences

- Rich and diversified cloud market
  + more possibilities for perspective customers
  – selection process can be difficult
- Several problems [DFLPS-18a, DFLS-17]
  - identification of Quality of Service attributes
  - definition of metrics for determining preferred plans
  - consideration of security properties in SLAs
  - user support for
    - defining (hard) security/privacy requirements and (soft) preferences
    - defining high-level specifications supporting reasoning over security/privacy requirements

Supporting user requirements and preferences

- Rich and diversified cloud market
  - + more possibilities for perspective customers
  - – selection process can be difficult
- Several problems [DFLPS-18a, DFLS-17]
  - identification of Quality of Service attributes
  - definition of metrics for determining preferred plans
  - consideration of security properties in SLAs
  - user support for
    - defining (hard) security/privacy requirements and (soft) preferences
    - defining high-level specifications supporting reasoning over security/privacy requirements

A brokerage-based approach



- + identification of possible requirements and preferences
- + expressive and user-friendly language for requirements and preferences
- + ranking of acceptable plans based on preferences

Abstract model of cloud plans

- A plan P is modeled as a set $[a_1, \ldots, a_n]$ of attributes of interest

	P_1	P_2	P_3	P_4	P_5	
prov	Ghost	Ghost	GoGo	GoGo	GoGo	(provider)
loc	US	US	EU	US	EU	(server location)
encr	3DES	AES	3DES	AES	AES	(encryption algo)
avail	M	H	VH	H	VH	(availability)
test	authC	authB	authB	authA	authA	(pen test authority)
cert	certB	certC	certB	certC	certA	(security certification)
aud	1Y	_	_	_	_	(audit frequency)

Requirement specification language

- Requirements restrict the values that can be assumed by a plan
- Expressed through a user-friendly and expressive language offering simple constructs

Requirement specification language

- Requirements restrict the values that can be assumed by a plan
- Expressed through a user-friendly and expressive language offering simple constructs

simple	prov IN {Ghost, GoGo, MHard} avail NOT IN {VL, L}
alternatives	ANY({test IN {authA, authB}, cert IN {certA, certB}})
conjunctions	ALL({loc IN {EU, US}, encr NOT IN {DES}})
implications	IF ALL({loc IN {US}, encr IN {3DES}}) THEN ANY({audit IN {3M, 6M}, cert IN {certA}})
exclusions	FORBIDDEN({loc NOT IN {EU}, test IN {authC}})
at least n	AT_LEAST(2, {loc IN {EU}, encr IN {AES}, prov IN {GoGo, Ghost}})
at most n	AT_MOST(2, {prov IN {Ghost}, avail IN {M, MH}, encr IN {3DES}})

Acceptable plan

A plan is acceptable iff it satisfies all user requirements

	P_1	P_2	P_3	P_4	P_5
prov	Ghost	Ghost	GoGo	GoGo	GoGo
loc	US	US	EU	US	EU
encr	3DES	AES	3DES	AES	AES
avail	M	H	VH	H	VH
test	authC	authB	authB	authA	authA
cert	certB	certC	certB	certC	certA
aud	1Y	_	_	_	_

c_1 : prov IN {Ghost, GoGo, MHard}
c_2 : avail NOT IN {VL, L}
c_3 : ALL({loc IN {EU, US}, encr NOT IN {DES}})
c_4 : FORBIDDEN({loc NOT IN {EU}, test IN {authC}})
c_5 : AT_LEAST(2, {loc IN {EU}, encr IN {AES}, prov IN {Gogo, Ghost}})

Acceptable plan

A plan is acceptable iff it satisfies all user requirements

	P_1	P_2	P_3	P_4	P_5
prov	Ghost	Ghost	GoGo	GoGo	GoGo
loc	US	US	EU	US	EU
encr	3DES	AES	3DES	AES	AES
avail	M	H	VH	H	VH
test	authC	authB	authB	authA	authA
cert	certB	certC	certB	certC	certA
aud	1Y	_	_	_	_
	×				

c_1 : prov IN {Ghost, GoGo, Mhard}
c_2 : avail NOT IN {VL, L}
c_3 : ALL({loc IN {EU, US}, encr NOT IN {DES}})
c_4 : FORBIDDEN({loc NOT IN {EU}, test IN {authC}})
c_5 : AT_LEAST(2, {loc IN {EU}, encr IN {AES}, prov IN {Gogo, Ghost}})

Acceptable plan

A plan is acceptable iff it satisfies all user requirements

	P_1	P_2	P_3	P_4	P_5
prov	Ghost	Ghost	GoGo	GoGo	GoGo
loc	US	US	EU	US	EU
encr	3DES	AES	3DES	AES	AES
avail	M	H	VH	H	VH
test	authC	authB	authB	authA	authA
cert	certB	certC	certB	certC	certA
aud	1Y	_	_	_	_
	×	✓	✓	✓	✓

c_1 : prov IN {Ghost, GoGo, Mhard}
c_2 : avail NOT IN {VL, L}
c_3 : ALL({loc IN {EU, US}, encr NOT IN {DES}})
c_4 : FORBIDDEN({loc NOT IN {EU}, test IN {authC}})
c_5 : AT_LEAST(2, {loc IN {EU}, encr IN {AES}, prov IN {Gogo, Ghost}})

Preferences

- Soft requirements: some characteristics are preferred to others
- Enforced on acceptable plans to rank them
- Two kinds of preferences
  - on attribute values
  - on attributes themselves

Preferences on attribute values

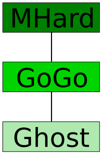
- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values

Preferences on attribute values

- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values for `prov`: $\{$ MHard $\} \succ \{$ GoGo $\} \succ \{$ Ghost $\}$

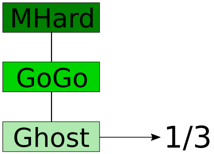
Preferences on attribute values

- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values for `prov`: $\{$ MHard $\} \succ \{$ GoGo $\} \succ \{$ Ghost $\}$
  - score function reflecting the relative position of the values



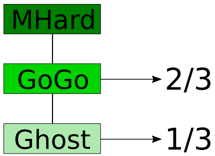
Preferences on attribute values

- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values for `prov`: {MHard} $\succ$ {GoGo} $\succ$ {Ghost}
  - score function reflecting the relative position of the values



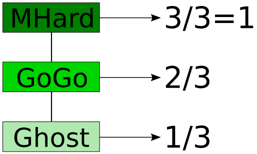
Preferences on attribute values

- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values for `prov`: $\{$ MHard $\} \succ \{$ GoGo $\} \succ \{$ Ghost $\}$
  - score function reflecting the relative position of the values



Preferences on attribute values

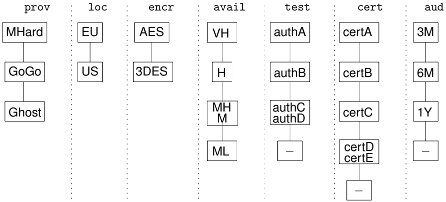
- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values for `prov`: $\{$ MHard $\} \succ \{$ GoGo $\} \succ \{$ Ghost $\}$
  - score function reflecting the relative position of the values



Preferences on attribute values

- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values for `prov`: {MHard} $\succ$ {GoGo} $\succ$ {Ghost}
  - score function reflecting the relative position of the values
- Scoring vector Π_i includes the scores of the values of P_i

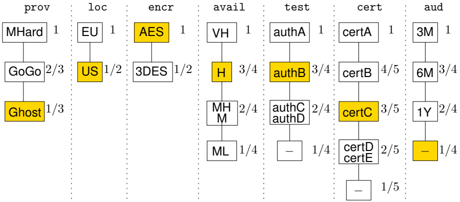
	P_2
prov	Ghost
loc	US
encr	AES
avail	H
test	authB
cert	certC
aud	_



Preferences on attribute values

- Certain values are preferred to other values for an attribute
- Modeled through a preference relationship
  - total ordering relationship over sets of acceptable values
    for prov: {MHard} ≻ {GoGo} ≻ {Ghost}
  - score function reflecting the relative position of the values
- Scoring vector Π_i includes the scores of the values of P_i

	P_2	Π_2
prov	Ghost	1/3
loc	US	1/2
encr	AES	1
avail	H	3/4
test	authB	3/4
cert	certC	3/5
aud	_	1/4



Preferences on attributes

- Certain attributes are more important than other ones
- Modeled through a weight function
  - higher weights imply higher importance

    $w(\texttt{prov}) = 1, w(\texttt{avail}) = 10$

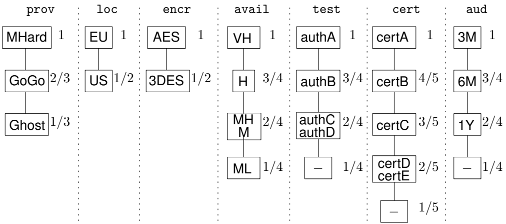
Ranking

- Three possible approaches
  - Pareto dominance
  - D-dominance (distance-based)
  - WD-dominance (weighted distance-based)
- Each strategy defines dominance among pairs of plans
  - P_i dominates $P_j \Longrightarrow P_i$ is preferred to P_j

Pareto dominance

- P_i dominates P_j iff P_i has:
  - for all attributes, values that are equally or more preferred than those in P_j and
  - for at least one attribute, a more preferred value than the one in P_j

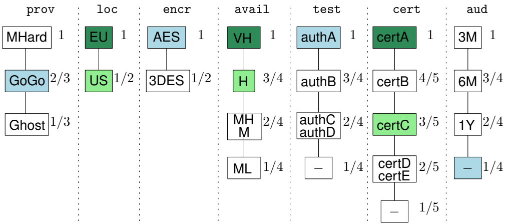
	P_4	P_5
prov	GoGo	GoGo
loc	US	EU
encr	AES	AES
avail	H	VH
test	authA	authA
cert	certC	certA
aud	_	_



Pareto dominance

- P_i dominates P_j iff P_i has:
  - for all attributes, values that are equally or more preferred than those in P_j and
  - for at least one attribute, a more preferred value than the one in P_j

	P_4	P_5
prov	GoGo	GoGo
loc	US	EU
encr	AES	AES
avail	H	VH
test	authA	authA
cert	certC	certA
aud	_	_

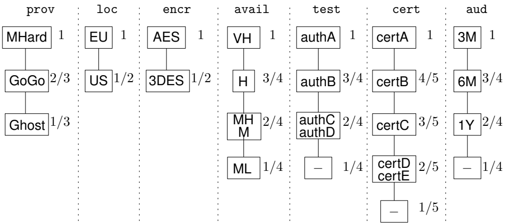


P_5 dominates P_4

Pareto dominance

- P_i dominates P_j iff P_i has:
  - for all attributes, values that are equally or more preferred than those in P_j and
  - for at least one attribute, a more preferred value than the one in P_j

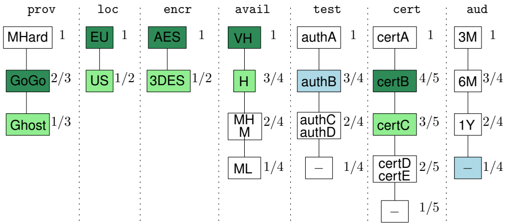
	P_2	P_3
prov	Ghost	GoGo
loc	US	EU
encr	AES	3DES
avail	H	VH
test	authB	authB
cert	certC	certB
aud	_	_



Pareto dominance

- P_i dominates P_j iff P_i has:
  - for all attributes, values that are equally or more preferred than those in P_j and
  - for at least one attribute, a more preferred value than the one in P_j

	P_2	P_3
prov	Ghost	GoGo
loc	US	EU
encr	AES	3DES
avail	H	VH
test	authB	authB
cert	certC	certB
aud	–	–

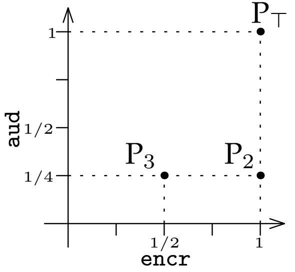


P_2 and P_3 are not comparable

Distance-based dominances

- Plans are represented as points in an n-dimensional space
  - coordinates of plan P are the values in scoring vector Π
- Dominance is given by the distance from an ideal plan $\mathrm{P}_\top$
  - for all attributes, $\mathrm{P}_\top$ has top values ($\Pi_\top$ has 1)

	P_2	Π_2	P_3	Π_3
encr	AES	1	3DES	1/2
aud	_	1/4	_	1/4

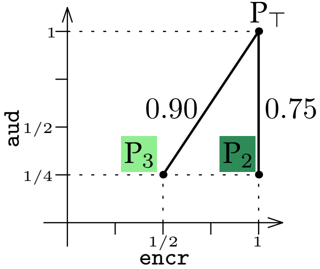


$$\mathrm{dist}(\Pi_h, \Pi_k) = \sqrt{\sum_{i=1}^{m} (\Pi_h[a_i] - \Pi_k[a_i])^2}$$

Distance-based dominances

- Plans are represented as points in an n-dimensional space
  - coordinates of plan P are the values in scoring vector Π
- Dominance is given by the distance from an ideal plan $\mathrm{P}_\top$
  - for all attributes, $\mathrm{P}_\top$ has top values ($\Pi_\top$ has 1)

	P_2	Π_2	P_3	Π_3
encr	AES	1	3DES	1/2
aud	_	1/4	_	1/4



$$\mathrm{dist}(\Pi_h,\Pi_k)=\sqrt{\sum_{i=1}^{m}(\Pi_h[a_i]-\Pi_k[a_i])^2}$$

$$\mathrm{dist}(\Pi_\top,\Pi_2)=\sqrt{(1-1)^2+(1-1/4)^2}=0.75;\quad \mathrm{dist}(\Pi_\top,\Pi_3)=\sqrt{(1-1/2)^2+(1-1/4)^2}=0.90$$

Distance-based dominances

- Plans are represented as points in an n-dimensional space
  - coordinates of plan P are the values in scoring vector Π
- Dominance is given by the distance from an ideal plan $\mathrm{P}_\top$
  - for all attributes, $\mathrm{P}_\top$ has top values ($\Pi_\top$ has 1)

	P_2	Π_2	P_3	Π_3
prov	Ghost	1/3	GoGo	2/3
loc	US	1/2	EU	1
encr	AES	1	3DES	1/2
avail	H	3/4	VH	1
test	authB	3/4	authB	3/4
cert	certC	3/5	certB	4/5
aud	_	1/4	_	1/4

dist(Π_2,$\Pi_\top$) = 1.24

dist(Π_3,$\Pi_\top$) = 1.01

Distance-based dominances

- Plans are represented as points in an n-dimensional space
  - coordinates of plan P are the values in scoring vector Π
- Dominance is given by the distance from an ideal plan $\mathrm{P}_\top$
  - for all attributes, $\mathrm{P}_\top$ has top values ($\Pi_\top$ has 1)

	P_2	Π_2	P_3	Π_3
prov	Ghost	1/3	GoGo	2/3
loc	US	1/2	EU	1
encr	AES	1	3DES	1/2
avail	H	3/4	VH	1
test	authB	3/4	authB	3/4
cert	certC	3/5	certB	4/5
aud	_	1/4	_	1/4

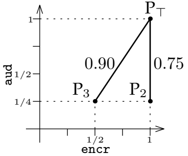
dist(Π_2,$\Pi_\top$) = 1.24

dist(Π_3,$\Pi_\top$) = 1.01

P_3 dominates P_2

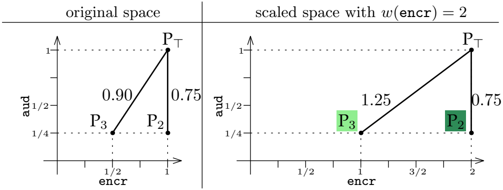
Distance-based dominances

- Plans are represented as points in an n-dimensional space
  - coordinates of plan P are the values in scoring vector Π
- Dominance is given by the distance from an ideal plan $\mathrm{P}_\top$
  - for all attributes, $\mathrm{P}_\top$ has top values ($\Pi_\top$ has 1)
- Priorities among attributes $\Longrightarrow$ scaling the n-dimensional space
  - scaling factor along the dimension for attribute *a* is $w(a)$



Distance-based dominances

- Plans are represented as points in an n-dimensional space
  - coordinates of plan P are the values in scoring vector Π
- Dominance is given by the distance from an ideal plan $\mathrm{P}_\top$
  - for all attributes, $\mathrm{P}_\top$ has top values ($\Pi_\top$ has 1)
- Priorities among attributes $\Longrightarrow$ scaling the n-dimensional space
  - scaling factor along the dimension for attribute *a* is $w(a)$



Variations/open issues ...

- Dependencies among requirements and/or cloud plan characteristics [DLP-16, DLPSS-16]
- Combination of requirements from multiple applications [AFLS-16, AFLS-18]
- Support for fuzzy specifications and reasoning [DFLPS-18a, DFLPS-18b, DFLS-17, FPS-15]

Other open issues

Selective access

Protection of data at rest

Security metrics

Providers/plans selection

Multi providers for security

Access confidentiality

Private collaborative computation

Query privacy

Fine-grained access

Query and computation integrity

Data publication and utility

User privacy

Conclusions

- Novel scenarios provide great convenience and benefit in the management and access to the information but require solutions to protect data
- Need to provide users and data owners with control over their data
- Data protection solutions are beneficial to both:
  - users and data owners (empowered with control)
  - CSPs and data controllers (increased confidence of users, decreased liability)

References

References – 1

- [ABGGKMSTX-05] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, Y. Xu, “Two Can Keep a Secret: A Distributed Architecture for Secure Database Services,” in *Proc. of CIDR*, Asilomar, CA, USA, January 2005.
- [AFLS-16] A. Arman, S. Foresti, G. Livraga, P. Samarati, “A Consensus-based Approach for Selecting Cloud Plans,” in *Proc. of RTSI*, Bologna, Italy, September 2016.
- [AFLS-18] A. Arman, S. Foresti, G. Livraga, P. Samarati, “Cloud Plan Selection Under Requirements of Multiple Applications,” in *Security and Privacy Journal,* 2018 (to appear).
- [AFB-05] M. Atallah, K. Frikken, M. Blanton, “Dynamic and Efficient Key Management for Access Hierarchies,” in *Proc. of CCS*, Alexandria, VA, USA, November 2005.
- [ASB-12] M. Askari, R. Safavi-Naini, K. Barker, “An Information Theoretic Privacy and Utility Measure for Data Sanitization Mechanisms,” in *Proc. of CODASPY*, San Antonio, TX, USA, February 2012.

References – 2

- [AT-83] S. Akl, P. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," *ACM TOCS*, vol. 1, no. 3, August 1983.
- [BDFGPRSS-16] E. Bacis, S. De Capitani di Vimercati, S. Foresti, D. Guttadoro, S. Paraboschi, M. Rosa, P. Samarati, A. Saullo, "Managing Data Sharing in OpenStack Swift with Over-Encryption," in *Proc. of WISCS*, Vienna, Austria, October 2016.
- [BDFPRS-16a] 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 CCS*, Vienna, Austria, October 2016.
- [BDFPRS-16b] 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*, Guangzhou, China, October 2016.
- [BS-04] S.M. Bellovin, B. Cheswick, "Privacy-enhanced Searches Using Encrypted Bloom Filters," in *Cryptology ePrint*, 2004.
- [B-70] B.H. Bloom, "Trade-offs in Hash Coding with Allowable Error," in *Communication of the ACM*, vol. 13, no. 7, July 1970.

References – 3

- [BV-11] Z. Brakerski, V. Vaikuntanathan, “Efficient Fully Homomorphic Encryption from (standard) LWE,” in Proc. of FOCS, Palm Springs, CA, USA, October 2011.
- [CDFJPS-09a] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Fragmentation Design for Efficient Query Execution over Sensitive Distributed Databases,” in *Proc. of ICDCS*, Montreal, Quebec, Canada, June 2009.
- [CDFJPS-09b] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Keep a Few: Outsourcing Data while Maintaining Confidentiality,” in *Proc. of ESORICS*, Saint-Malo, France, September 2009.
- [CDFJPS-10] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Combining Fragmentation and Encryption to Protect Privacy in Data Storage,” in *ACM TISSEC*, vol. 13, no. 3, July 2010.
- [CDFJPS-11] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Selective Data Outsourcing for Enforcing Privacy,” in *JCS*, vol. 19, n. 3, 2011.
- [CKGS-98] B. Chor, E. Kushilevitz, O. Goldreich, M. Sudan, “Private Information Retrieval,” in *JACM*, vol. 45, no. 6, 1998.

References – 4

- [CMW-06] J. Crampton, K. Martin, P. Wild, “On Key Assignment for Hierarchical Access Control,” in *Proc. of CSFW*, Venice, Italy, July 2006.
- [CWLRL-14] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-Preserving Multikeyword Ranked Search over Encrypted Cloud Data,” in *IEEE TPDS*, 25(1):222-233, January 2014.
- [CDDJPS-05] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, P. Samarati, “Modeling and Assessing Inference Exposure in Encrypted Databases,” in *ACM TISSEC*, vol. 8, no. 1, February 2005.
- [CSYZ-08] G. Cormode, D. Srivastava, T. YU, Q. Zhang, “Anonymizing Bipartite Graph Data Using Safe Groupings,” in *Proc. of VLDB*, Auckland, New Zealand, August 2008.
- [DFJLPS-13] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, “Enforcing Dynamic Write Privileges in Data Outsourcing,” in *Computers & Security*, vol. 39, November 2013.
- [DFJLPS-14a] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, “Fragmentation in Presence of Data Dependencies,” in *IEEE TDSC*, 11(6):510-523, November/December 2014.

References – 5

- [DFJLPS-14b] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, “Integrity for Distributed Queries,” in *Proc. of CNS*, San Francisco, CA, USA, October 2014.
- [DFJLPS-15] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, “Loose Associations to Increase Utility in Data Publishing,” in *JCS*, vol. 23, no. 1, 2015.
- [DFJPPS-10] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Pelosi, P. Samarati, “Encryption-based Policy Enforcement for Cloud Storage,” in *Proc. of SPCC*, Genova, Italy, June 2010.
- [DFJPS-07] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Over-encryption: Management of Access Control Evolution on Outsourced Data,” in *Proc. of VLDB*, Vienna, Austria, September 2007.
- [DFJPS-10a] 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.

References – 6

- [DFJPS-10b] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Fragments and Loose Associations: Respecting Privacy in Data Publishing," in *Proc. of the VLDB Endowment*, vol. 3, no. 1, September 2010.
- [DFJPS-11] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Private Data Indexes for Selective Access to Outsourced Data," in *Proc. of WPES*, Chicago, Illinois, USA, October 2011.
- [DFJPS-13a] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "On Information Leakage by Indexes over Data Fragments," in *Proc. of PrivDB*, Brisbane, Australia, April 2013.
- [DFJPS-13b] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Integrity for Join Queries in the Cloud," in *IEEE TCC*, vol. 1, n. 2, July-December 2013.
- [DFJPS-15] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Integrity for Approximate Joins on Untrusted Computational Servers," in *Proc. of SEC*, Hamburg, Germany, May 2015.

References – 7

- [DFJPS-16] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Efficient Integrity Checks for Join Queries in the Cloud," in *JCS*, vol. 24, no. 3, 2016.
- [DFLPS-17] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, P. Samarati, "Supporting User Requirements and Preferences in Cloud Plan Selection," in *IEEE TSC*, November 2017.
- [DFLPS-18a] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, P. Samarati, "Supporting users in cloud plan selection," in *From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of his 70th Birthday*, I. Ray, I. Ray, and P. Samarati (eds.), Springer, 2018.
- [DFLPS-18b] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, P. Samarati, "A Fuzzy-based Brokering Service for Cloud Plan Selection"
- [DFLS-17] S. De Capitani di Vimercati, S. Foresti, G. Livraga, P. Samarati, "Supporting Users in Data Outsourcing and Protection in the Cloud," in *Proc. of CLOSER*, Rome, Italy, April 2017.
- [DFM-04] A. De Santis, A.L. Ferrara, B. Masucci, "Cryptographic Key Assignment Schemes for any Access Control Policy," in *Inf. Process. Lett.*, vol. 92, no. 4, 2004.

References – 8

- [DFMPPS-16] S. De Capitani di Vimercati, S. Foresti, R. Moretti, S. Paraboschi, G. Pelosi, P. Samarati, "A Dynamic Tree-based Data Structure for Access Privacy in the Cloud," in *Proc. of CloudCom*, Luxembourg, Luxembourg, December 2016.
- [DFPPS-11] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, P. Samarati, "Efficient and Private Access to Outsourced Data," in *Proc. of ICDCS,* Minneapolis, Minnesota, USA, June 2011.
- [DFPPS-13a] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, P. Samarati, "Supporting Concurrency and Multiple Indexes in Private Access to Outsourced Data," in *JCS*, vol. 21, n. 3, 2013.
- [DFPPS-13b] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, P. Samarati, "Distributed Shuffling for Preserving Access Confidentiality," in *Proc. of ESORICS*, Egham, U.K., September 2013.
- [DFPPS-15] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, P. Samarati, "Shuffle Index: Efficient and Private Access to Outsourced Data," in *ACM TOS*, vol. 1, no. 4, 2015.

References – 9

- [DFPPS-18] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, P. Samarati, “Three-Server Swapping for Access Confidentiality,” in *IEEE TCC*, vol. 6, no. 2, April-June 2018.
- [DFPPS-16] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, P. Samarati, “Access Control for the Shuffle Index,” in *Proc. of DBSec*, Trento, Italy, July 2016.
- [DLP-16] S. De Capitani di Vimercati, G. Livraga, V. Piuri, “Application Requirements with Preferences in Cloud-based Information Processing,” in *Proc. of RTSI*, Bologna, Italy, September 2016.
- [DLPSS-16] S. De Capitani di Vimercati, G. Livraga, V. Piuri and P. Samarati, G.A. Soares, “Supporting Application Requirements in Cloud-based IoT Information Processing,” in *Proc. of IoTBD*, Rome, Italy, April 2016
- [FPS-15] S. Foresti, V. Piuri, G.A. Soares, “On the Use of Fuzzy Logic in Dependable Cloud Management,” in *Proc. of CNS,* Florence, Italy, September 2015.
- [G-03] E. Goh, “Secure Indexes,” in *IACR Cryptology ePrint Archive*, 2003.
- [G-80] E. Gudes, “The Design of a Cryptography based Secure File System,” in *IEEE TSE*, vol. 6, no. 5, September 1980.

References – 10

- [G-09] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of STOC*, Bethesda, MD, USA, May-June 2009.
- [GSW-13] C. Gentry, A. Sahai, B. Waters, "Homomorphic Encryption from Learning with Errors: Conceptually-simpler, Asymptotically-faster, Attribute-based," in Proc. of CRYPTO, Santa Barbara, CA, USA, August 2013.
- [HIML-02] H. Hacigümüş, B. Iyer, S. Mehrotra, C. Li, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," in *Proc. of the ACM SIGMOD*, Madison, Wisconsin, USA, June 2002.
- [HL-90] L. Harn, H. Lin, "A Cryptographic Key Generation Scheme for Multilevel Data Security," *Computers and Security*, vol. 9, no. 6, October 1990.
- [HY-03] M. Hwang, W. Yang, "Controlling Access in Large Partially Ordered Hierarchies using Cryptographic Keys," *The Journal of Systems and Software*, vol. 67, no. 2, August 2003.
- [JP-13] R. Jhawar, V. Piuri, "Adaptive Resource Management for Balancing Availability and Performance in Cloud Computing", in *Proc. of SECRYPT,* Reykjavik, Iceland, July 2013.
- [JP-12] R. Jhawar, V. Piuri, "Fault Tolerance Management in IaaS Clouds", in *Proc. of ESTEL,* Rome, Italy, October 2012.

References – 11

- [JPS-12a] R. Jhawar, V. Piuri, P. Samarati, “Supporting Security Requirements for Resource Management in Cloud Computing,” in *Proc. of CSE*, Paphos, Cyprus, December 2012.
- [JPS-12b] R. Jhawar, V. Piuri, M. Santambrogio, “A Comprehensive Conceptual System-Level Approach to Fault Tolerance in Cloud Computing”, in *Proc. of SysCon*, Vancouver, BC, Canada, March 2012.
- [JPS-13] R. Jhawar, V. Piuri, M. Santambrogio, “Fault Tolerance Management in Cloud Computing: A System-Level Perspective”, in *IEEE Systems Journal*, vol. 7, n. 2, June 2013.
- [JP-13] R. Jhawar, V. Piuri, “Fault Tolerance and Resilience in Cloud Computing Environments,” in Vacca, J. (ed.) Computer and Information Security Handbook, 2nd Edition, pp. 125-141. Morgan Kaufmann, 2013.
- [LDGW-13] J. Lai, R.H. Deng, C. Guan, J. Weng, “Attribute-Based Encryption With Verifiable Outsourced Decryption,” in *IEEE TIFS*, 8(8):1343-1354, August 2013.
- [LDL-12] J. Lai, R.H. Deng, Y. Li, “Expressive CP-ABE with Partially Hidden Access Structures,” in *Proc. of ASIACCS*, Seoul, Korea, May 2012.

References – 12

- [LDLW-14] J. Lai, R.H. Deng, Y. Li, J. Weng, “Fully Secure Key-Policy Attribute-based Encryption with Constant-Size Ciphertexts and Fast Decryption,” in *Proc. of ASIACCS*, Kyoto, Japan, June 2014.
- [LDPW-14] J. Lai, R.H. Deng, H. Pang, J. Weng, “Verifiable Computation on Outsourced Encrypted Data,” in *Proc. of ESORICS*, Wroclaw, Poland, September 2014.
- [LC-04] P. Lin, K.S. Candan, “Hiding Traversal of Tree Structured Data from Untrusted Data Stores,” in *Proc. of WOSIS*, Porto, Portugal, April 2004.
- [LCLJML-13] J. Li, X. Chen, J. Li, C, Jia, J. Ma, W. Lou, “Fine-grained access control system based on outsourced attribute-based encryption,” in *Proc. of ESORICS*, Egham, U.K., September 2013.
- [LHKR-06] F. Li, M. Hadjieleftheriou, G. Kollios, L. Reyzin, “Dynamic Authenticated Index Structures for Outsourced Databases,” in *Proc. of SIGMOD*, Chicago, IL, USA, June 2006.
- [LWL-89] H. Liaw, S. Wang, C. Lei, “On the Design of a Single-Key-Lock Mechanism Based on Newton’s Interpolating Polynomial,” in *IEEE TSE*, vol. 15, no. 9, September 1989.

References – 13

- [LWWCRL-10] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," in PROC. OF IEEE INFOCOM, San Diego, CA, USA, March 2010.
- [M-85] S. MacKinnon et al., "An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy," in *IEEE TC,* vol. 34, no. 9, September 1985.
- [PRZB-11] R.A. Popa, C.M.S. Redfield, N. Zeldovich, H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," in *Proc. of SOSP*, Cascais, Portugal, October 2011.
- [RFKSSvD-15] L. Ren, C. Fletcher, A. Kwon, E. Stefanov, E. Shi M. van Dijk, S. Devadas, "Constants Count: Practical Improvements to Oblivious RAM" in *Proc. of USENIX*, Washington, USA, August 2015.
- [RVBM-09] M. Raykova, B. Vo, S.M. Bellovin, T. Malkin, "Secure Anonymous Database Search," in *Proc. of ACM CCSW*, Chicago, IL, USA, November 2009.
- [S-14] P. Samarati, "Data Security and Privacy in the Cloud," in *Proc. of ISPEC*, Fuzhou, China, May 2014.

References – 14

- [SD-16] P. Samarati, S. De Capitani di Vimercati, “Cloud Security: Issues and Concerns,” in *Encyclopedia on Cloud Computing*, S. Murugesan, I. Bojanova (eds.), Wiley, 2016.
- [SD-10] P. Samarati, S. De Capitani di Vimercati, “Data Protection in Outsourcing Scenarios: Issues and Directions,” in *Proc. of ASIACCS*, Beijing, China, April 2010.
- [S-87] R. Sandhu, “On Some Cryptographic Solutions for Access Control in a Tree Hierarchy,” in *Proc. of the 1987 Fall Joint Computer Conference on Exploring Technology: Today and Tomorrow*, Dallas, TX, USA, October 1987.
- [S-88] R. Sandhu, “Cryptographic Implementation of a Tree Hierarchy for Access Control,” in *Information Processing Letters*, vol. 27, no. 2, 1988.
- [SC-02] V. Shen, T. Chen, “A Novel Key Management Scheme Based on Discrete Logarithms and Polynomial Interpolations,” in *Computers and Security*, vol. 21, no. 2, March 2002.
- [SC-07] R. Sion, B. Carbunar, “On the Computational Practicality of Private Information Retrieval,” in *Proc. of NDSS*, San Diego, CA, USA, February/March 2007.

References – 15

- [SLLDW-14] J. Shi, J. Lai, Y. Li, R.H. Deng, J. Weng, “Authorized Keyword Search on Encrypted Data,” in *Proc. of ESORICS*, Wroclaw, Poland, September 2014.
- [SvSFRYD-13] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, S. Devadas, “Path ORAM: An Extremely Simple Oblivious RAM Protocol,” in *Proc. of CCS*, Berlin, Germany, November 2013.
- [XWYM-07] M. Xie, H. Wang, J. Yin, X. Meng, “Integrity Auditing of Outsourced Data,” in *Proc. of VLDB*, Vienna, Austria, September 2007.
- [WCRL-12] C. Wang, N. Cao, K. Ren, W. Lou, “Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data,” in *IEEE TPDS*, vol. 23, no. 8, pp. 1467-1479, August 2012.
- [WL-06] H. Wang, Laks V. S. Lakshmanan, “Efficient Secure Query Evaluation over Encrypted XML Databases,” in *Proc. of VLDB*, Seoul, Korea, September 2006.
- [WSC-08] P. Williams, R. Sion, B. Carbunar, “Building Castles Out of Mud: Practical Access Pattern Privacy and Correctness on Untrusted Storage,” in *Proc of CCS*, Alexandria, USA, October 2008.

References – 16

- [WS-12] P. Williams, R. Sion, “Single Round Access Privacy on Outsourced Storage,” in *Proc. of CCS*, Raleigh, NC, USA, October 2012.
- [YPPK-09] Y. Yang, D. Papadias, S. Papadopoulos, P. Kalnis, “Authenticated Join Processing in Outsourced Databases,” in *Proc. of SIGMOD*, Providence, RI, USA, June 2009.
- [YWRL-10] S. Yu, C. Wang, K. Ren, W. Lou, “Attribute Based Data Sharing with Attribute Revocation,” in *Proc. of ASIACCS*, Beijing, China, April 2010.
- [ZS-14] L.F. Zhang, R. Safavi-Naini, “Verifiable Delegation of Computations with Storage-Verification Trade-off,” in *Proc. of ESORICS*, Wroclaw, Poland, September 2014.