Secure Joins with MapReduce

Xavier Bultel[1] Radu Ciucanu[2] Matthieu Giraud[3]
Pascal Lafourcade[3] Lihua Ye[4]

[1]IRISA, Univeristé de Rennes 1, France
[2]INSA, Université Orléans, France
[3]LIMOS, Université Clermont Auvergne, France
[4]Harbin Institute of Technology, China

Foundations & Practice of Security – November 13, 2018

Joins

Name	City
Alice	Montreal
Bob	London
Cesar	Tokyo

$\bowtie$

Name	Disease
Alice	Diabetes
Bob	Flu
Bob	Cancer

$=$

Name	City	Disease
Alice	Montreal	Diabetes
Bob	London	Flu
Bob	London	Cancer

Cascade Joins

$R_1 =$

Name	City
Alice	Montreal
Bob	London
Cesar	Tokyo

$R_2 =$

Name	**Disease**
Alice	Diabetes
Bob	Flu
Bob	Cancer

$R_3 =$

Disease	Specialist
Cancer	Hopkins
Diabetes	Jude

Cascade Joins

$R_1 =$

Name	City
Alice	Montreal
Bob	London
Cesar	Tokyo

$R_2 =$

Name	**Disease**
Alice	Diabetes
Bob	Flu
Bob	Cancer

$R_3 =$

Disease	Specialist
Cancer	Hopkins
Diabetes	Jude

1. $R_1 \bowtie R_2 =$

Name	City	**Disease**
Alice	Montreal	Diabetes
Bob	London	Flu
Bob	London	Cancer

Cascade Joins

$R_1 =$

Name	City
Alice	Montreal
Bob	London
Cesar	Tokyo

$R_2 =$

Name	**Disease**
Alice	Diabetes
Bob	Flu
Bob	Cancer

$R_3 =$

Disease	Specialist
Cancer	Hopkins
Diabetes	Jude

1. $R_1 \bowtie R_2 =$

Name	City	**Disease**
Alice	Montreal	Diabetes
Bob	London	Flu
Bob	London	Cancer

2. $(R_1 \bowtie R_2) \bowtie R_3 =$

Name	City	Disease	Specialist
Alice	Montreal	Diabetes	Jude
Bob	London	Cancer	Hopkins

Hypercube Joins

Relation R_1:

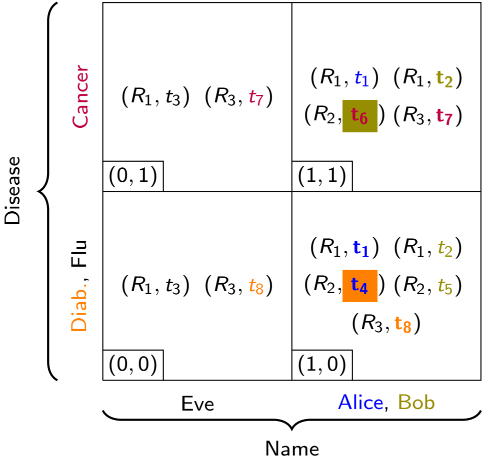
- $t_1 = $ (Alice, Montreal)
- $t_2 = $ (Bob, London)
- $t_3 = $ (Eve, Tokyo)

Relation R_2:

- $t_4 = $ (Alice, Diabetes)
- $t_5 = $ (Bob, Flu)
- $t_6 = $ (Bob, Cancer)

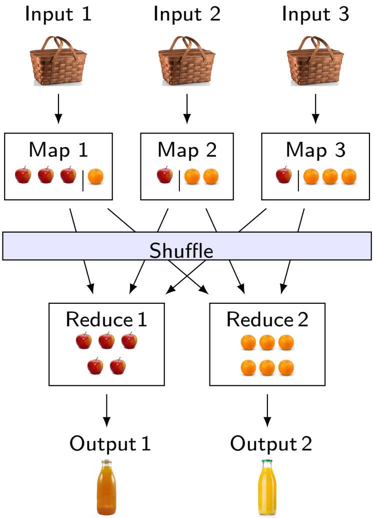
Relation R_3:

- $t_7 = $ (Cancer, Hopkins)
- $t_8 = $ (Diabetes, Jude)



MapReduce

- Partitioning input data
- Scheduling program execution on machines
- Performing the shuffle
- Handling machine failures



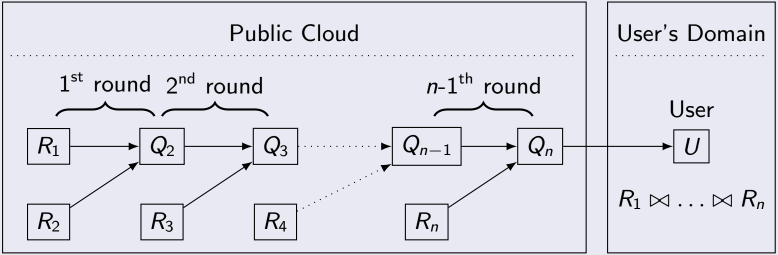
Programmer gives:

- Input files
- **Map** and **Reduce**

Joins with MapReduce

Cascade Joins

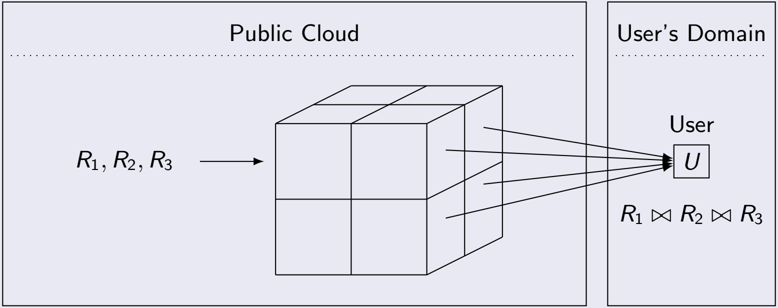
n relations $\Rightarrow$ $n - 1$ MapReduce rounds



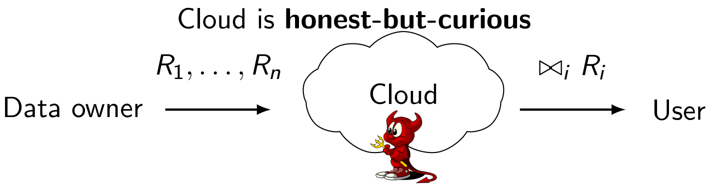
Joins with MapReduce

Hypercube Joins

n relations $\Rightarrow$ 1 MapReduce round



Security Model



Security properties

- Secrecy of $R_1, \ldots, R_n$ and $\bowtie_i R_i$
- User queries $\bowtie_i R_i$ but cannot learn $R_1, \ldots, R_n$

Contributions

Secure MapReduce Algorithms

- Cascade
- Hypercube

Secure-Private (SP) approach

- Cloud nodes do not learn $R_1, \ldots, R_n$
- Cloud nodes do not learn $\bowtie_i R_i$

Collision-Resistant-Secure-Private (CRSP) approach

- Prevent collision between cloud and user

Outline

Outline

Pseudo-Random Function

Definition

$$f : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$$

- Deterministic
- Indistinguishable from a random function

Notation

$$f_k(m) = f(k, m)$$

Public-Key Encryption

Definition

- $(pk, sk) \leftarrow \mathcal{G}(\lambda)$
- $c \leftarrow \mathcal{E}_{pk}(m)$
- $m \leftarrow \mathcal{D}_{sk}(c)$

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$$

Notation

$$\{m\} = \mathcal{E}_{pk}(m)$$

Outline

SP Preprocessing

Example

$R_1 =$

Name	City
Alice	Montreal
Bob	London
Cesar	Tokyo

$\Rightarrow$ $\hat{R}_1 =$

f_k(Name)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$R_2 =$

Name	Disease
Alice	Diabetes
Bob	Flu
Bob	Cancer

$\Rightarrow$ $\hat{R}_2 =$

f_k(Name)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diabetes)	{Diabetes}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

$R_3 =$

Disease	Specialist
Cancer	Hopkins
Diabetes	Jude

$\Rightarrow$ $\hat{R}_3 =$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diabetes)	{Jude}

SP Cascade $(\hat{R}_1 \bowtie \hat{R}_2) \bowtie \hat{R}_3$

$\Bigg($

f_k(Name)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(Name)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

$\Bigg) \bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

SP Cascade $(\hat{R}_1 \bowtie \hat{R}_2) \bowtie \hat{R}_3$

(

f_k(Name)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(Name)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

) $\bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

(

f_k(**Name**)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(**Name**)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

) $\bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

SP Cascade $(\hat{R}_1 \bowtie \hat{R}_2) \bowtie \hat{R}_3$

$($

f_k(Name)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(Name)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

$) \bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

$($

f_k(**Name**)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(**Name**)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

$) \bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

f_k(Name)	{Name}	{City}	f_k(**Disease**)	{Disease}
f_k(Alice)	{Alice}	{Montreal}	f_k(Diab.)	{Diab.}
f_k(Bob)	{Bob}	{London}	f_k(Flu)	{Flu}
f_k(Bob)	{Bob}	{London}	f_k(Cancer)	{Cancer}

$\bowtie$

f_k(**Disease**)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

SP Cascade $(\hat{R}_1 \bowtie \hat{R}_2) \bowtie \hat{R}_3$

$\Big($

f_k(Name)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(Name)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

$\Big) \bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

$\Big($

f_k(**Name**)	{Name}	{City}
f_k(Alice)	{Alice}	{Montreal}
f_k(Bob)	{Bob}	{London}
f_k(Cesar)	{Cesar}	{Tokyo}

$\bowtie$

f_k(**Name**)	f_k(Disease)	{Disease}
f_k(Alice)	f_k(Diab.)	{Diab.}
f_k(Bob)	f_k(Flu)	{Flu}
f_k(Bob)	f_k(Cancer)	{Cancer}

$\Big) \bowtie$

f_k(Disease)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

f_k(Name)	{Name}	{City}	f_k(**Disease**)	{Disease}
f_k(Alice)	{Alice}	{Montreal}	f_k(Diab.)	{Diab.}
f_k(Bob)	{Bob}	{London}	f_k(Flu)	{Flu}
f_k(Bob)	{Bob}	{London}	f_k(Cancer)	{Cancer}

$\bowtie$

f_k(**Disease**)	{Specialist}
f_k(Cancer)	{Hopkins}
f_k(Diab.)	{Jude}

f_k(Name)	{Name}	{City}	f_k(Disease)	{Disease}	{Specialist}
f_k(Alice)	{Alice}	{Montreal}	f_k(Diab.)	{Diab.}	{Jude}
f_k(Bob)	{Bob}	{London}	f_k(Cancer)	{Cancer}	{Hopkins}

SP Cascade

Map function

If $i = 1$: $\texttt{emit}\left(\pi_{\mathbb{Q}_1^f \cap \mathbb{R}_2^f}(t), (\texttt{Q}_1, t_r)\right)$

Else: $\texttt{emit}\left(\pi_{\mathbb{Q}_i^f \cap \mathbb{R}_{i+1}^f}(t), (\texttt{R}_{i+1}, t_q)\right)$

Reduce function

If $i \neq n-1$: $\texttt{emit}\left(\pi_{\mathbb{Q}_{i+1}^f \cap \mathbb{R}_{i+2}^f}(t_r \times t_q), t_r \times t_q\right)$

Else: $\texttt{emit}\ (t_r \times t_q, t_r \times t_q)$

SP Hypercube

Relation R_1:

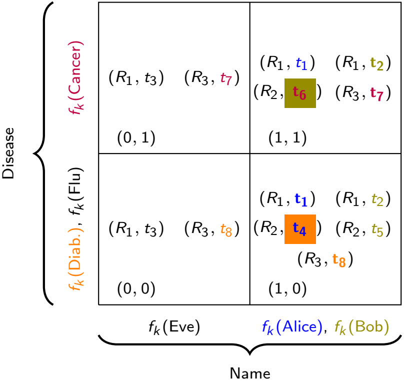
- $t_1 = (f_k(\text{Alice}), \{\text{Alice}\}, \{\text{Montreal}\})$
- $t_2 = (f_k(\text{Bob}), \{\text{Bob}\}, \{\text{London}\})$
- $t_3 = (f_k(\text{Eve}), \{\text{Eve}\}, \{\text{Tokyo}\})$

Relation R_2:

- $t_4 = (f_k(\text{Alice}), f_k(\text{Diab.}), \{\text{Diab.}\})$
- $t_5 = (f_k(\text{Bob}), f_k(\text{Flu}), \{\text{Flu}\})$
- $t_6 = (f_k(\text{Bob}), f_k(\text{Cancer}), \{\text{Cancer}\})$

Relation R_3:

- $t_7 = (f_k(\text{Cancer}), \{\text{Hopkins}\})$
- $t_8 = (f_k(\text{Diab.}), \{\text{Jude}\})$



SP Hypercube

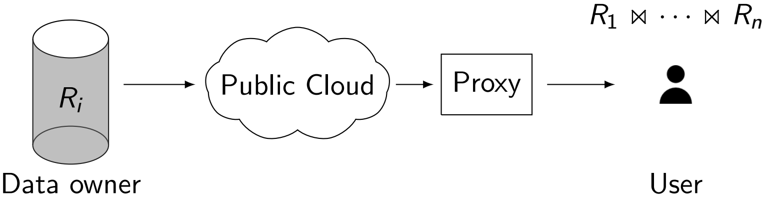
Map function

$$\texttt{emit}\ \Big((h_1(\pi_{X_1^f}(t_r)), \ldots, h_d(\pi_{X_d^f}(t_r))), t_r\Big)$$

Reduce function

$$\texttt{emit}\ (t, t)$$

CRSP Approach



$$\mathcal{E}_{pk_P}(\{m\}) = \mathcal{E}_{pk_P}(\mathcal{E}_{pk_U}(m))$$

CRSP Preprocessing

Example

$R_1 =$

Name	City
Alice	Montreal
Bob	London
Cesar	Tokyo

$\Rightarrow$ $\hat{R}_1 =$

f_k(Name)	$\mathcal{E}_{pk_P}$({Name})	$\mathcal{E}_{pk_P}$({City})
f_k(Alice)	$\mathcal{E}_{pk_P}$({Alice})	$\mathcal{E}_{pk_P}$({Montreal})
f_k(Bob)	$\mathcal{E}_{pk_P}$({Bob})	$\mathcal{E}_{pk_P}$({London})
f_k(Cesar)	$\mathcal{E}_{pk_P}$({Cesar})	$\mathcal{E}_{pk_P}$({Tokyo})

$R_2 =$

Name	Disease
Alice	Diabetes
Bob	Flu
Bob	Cancer

$\Rightarrow$ $\hat{R}_2 =$

f_k(Name)	f_k(Disease)	$\mathcal{E}_{pk_P}$({Disease})
f_k(Alice)	f_k(Diabetes)	$\mathcal{E}_{pk_P}$({Diabetes})
f_k(Bob)	f_k(Flu)	$\mathcal{E}_{pk_P}$({Flu})
f_k(Bob)	f_k(Cancer)	$\mathcal{E}_{pk_P}$({Cancer})

$R_3 =$

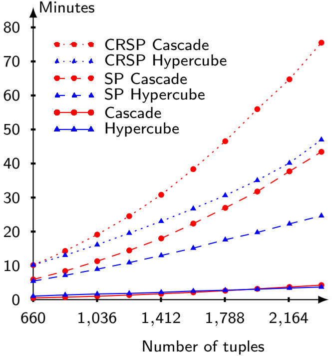
Disease	Specialist
Cancer	Hopkins
Diabetes	Jude

$\Rightarrow$ $\hat{R}_3 =$

f_k(Disease)	$\mathcal{E}_{pk_P}$({Specialist})
f_k(Cancer)	$\mathcal{E}_{pk_P}$({Hopkins})
f_k(Diabetes)	$\mathcal{E}_{pk_P}$({Jude})

Outline

Performances



Hadoop implementation

- 1 master + 3 data nodes
- 4/2 CPUs @ 2.4GHz
- 8/4Gb RAM

- Higgs Twitter dataset
- RSA-OAEP 2048 bits
- AES-CTR 128 bits

Outline

Conclusion

- *Secure* Cascade and Hypercube algorithms (SP & CRSP)
- Honest-but-curious adversay
- Practical implementation

Future Works

- Avoid leakage on same values
- Security in *standard model*
- *Cloud-User Collision resistant* without trusted-third party

Thank you for your attention!

Any questions?

Montreal by Pascal Lafourcade (flickr.com/pascalafourcade).

Keep in touch

email matthieu.giraud@uca.fr

web http://sancy.univ-bpclermont.fr/~giraud/