Dynamic Authenticated Index Structures for Outsourced Databases

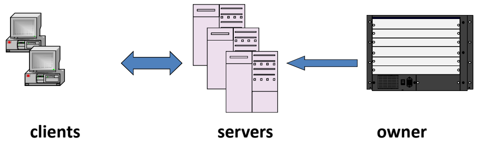
Feifei Li, Marios Hadjieleftheriou, George Kollios, Leonid Reyzin
Boston University
AT&T Labs-Research
Presenter : Nima Najafian

Outline

- ☺ The Model
- ☺ Motivation
- ☺ Problem
- ☺ Solution
- ☺ Background
- ☺ Papers contributions
- ☺ Experimental validation

Outsourced Database Model

Owner: publish data
Servers: host the data and provide query services
Clients: query the owner’s data through servers



Motivation

- Advantages
  - The data owner does not need the hardware / software / personnel to run a DBMS
  - The ownerachieves economies of scale
  - The client enjoys better quality of service
- A main challenge
  - The service provider is not trusted, and may return incorrect query results

---

Problem

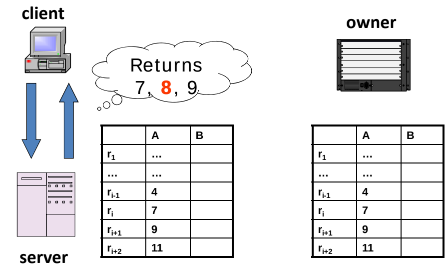
☺ Un-trusted Servers

Un-trusted server

- Lazy: incentives to perform less
- Curious: incentives to acquire information
- Malicious
  - Incorrect results (could be bugs)
  - Possibly compromised

Problem 1: Injection

Select * from T where 5<A<11

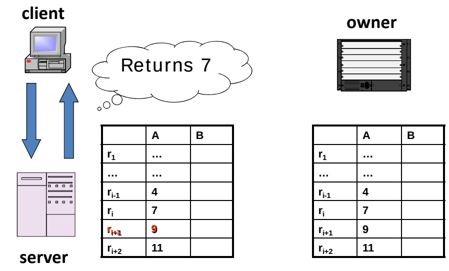


	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	7	
r_{i+1}	9	
r_{i+2}	11	

	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	7	
r_{i+1}	9	
r_{i+2}	11	

Problem 2: Drop

Select * from T where 5<A<11



	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	7	
r_{i+1}	9	
r_{i+2}	11	

	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	7	
r_{i+1}	9	
r_{i+2}	11	

Solution

☺ The Model

☺ Motivation

☺ Problem

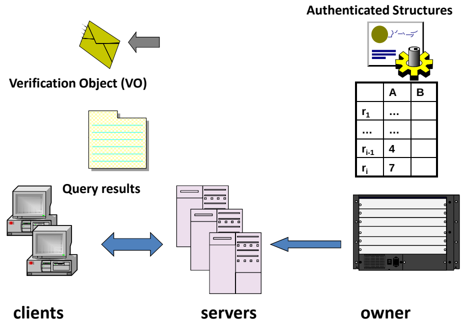
☺ Ability to authenticate without trusting the server
(Query Authentication)

Query Authentication: (the dimensions)

- Query Correctness
  results do exist in the owner's database ~ injection
- Query Completeness
  no records have been omitted from the result ~ drop
- Query Freshness ★
  results are based on the most current version of the database (this will bring a third problem into the picture) ~omission

---

General Approach



	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	7	

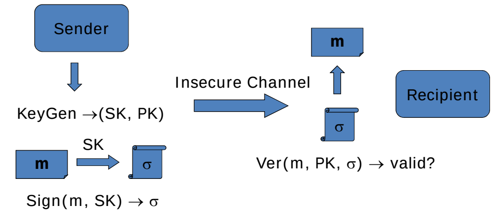
Background

☺Cryptographic essentials

1: Collision-resistant hash functions

- It is computational hard to find x_1 and x_2 s.t. $h(x_1)=h(x_2)$
- Computational hard? Based on well established assumptions such as discrete logarithms
- SHA1
- Observations:
  - variable input size → 20 bytes
  - Computation cost: 2-3 µs (for up to 500 bytes input)
  - Storage cost: 20 bytes
  - Under Crypto++ [crypto] and OpenSSL [openssl]

2: Public key digital signature schemes



2: Public Key Digital Signature Schemes

- Formally defined by [GMR88]
  - The message has not been changed in any way
  - The message is indeed from the sender (corresponding to the public key)
  - No one except the secret key owner could produce a signature
- One such scheme: RSA [RSA78]
- Observations
  - Computation cost: about 3-4 ms for signing and more than 100 µs for verifying
  - Storage cost: 128 bytes

3: Signature Aggregation (Condensed RSA)

- Checking one aggregated signature is almost as fast as an individual signature

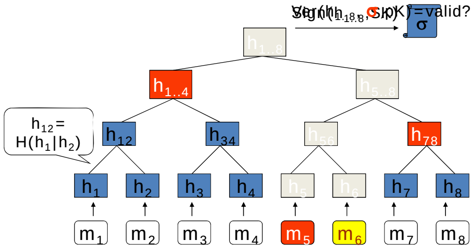
4: Merkle Hash Tree[M89]-Amortizing Signature Cost

Collision resistant of hash function → any change in the tree will lead to a different hash value for the root

Digital signature of the root → no one except the owner could produce the signature

Single signature to sign many messages

Hash function is publicly known



Correctness and Completeness

- Correctness, Completeness:
  - Any change in the tree will lead to different hash
  - Relative position of values is authenticated
- Authentication:
  - Signing the root with SK

Contributions

☺ Proposed authenticated structures

- Getting to know B+ trees
- The idea of changing
- ASB Tree (based on existing work)
- MB tree (based on existing work)
- EMB tree
- Freshness (third dimension of query Authentication)

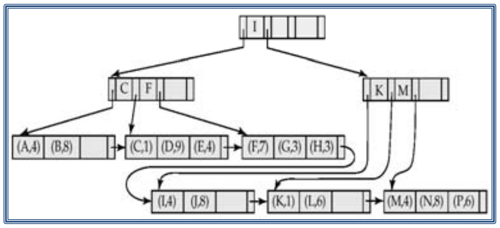
B+ - Tree Structure

- A typical node contains up to n – 1 search key values K1, K2,…, Kn-1, and n pointers P1, P2,…, Pn. The search key values are kept in sorted order.
- The pointer Pi can point to either a file record or a bucket of pointers which each point to a file record.

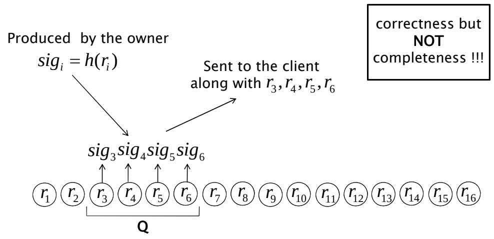
P1	K1	P2	…	Pn-1	Kn-1	Pn

B+ - Tree File Organization

In a B+ - Tree file organization, the leaf nodes of the tree stores the actual record rather than storing pointers to records.



Range Authentication – A Simple Approach



Produced by the owner

$sig_i = h(r_i)$

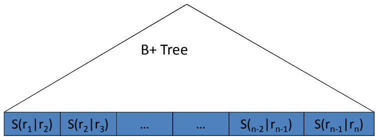
Sent to the client along with r_3, r_4, r_5, r_6

correctness but **NOT** completeness !!!

$sig_3 sig_4 sig_5 sig_6$

Signature-Based Approach: ASB Tree

based on [PJR05]

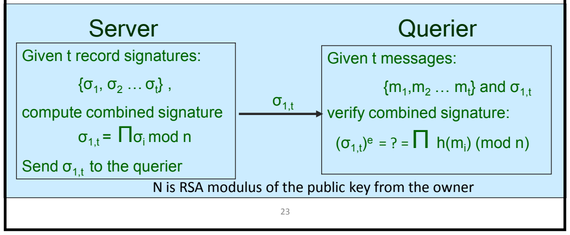


$S(r_1\|r_2)$	$S(r_2\|r_3)$	…	…	$S(_{n-2}\|r_{n-1})$	$S(r_{n-1}\|r_n)$

1. order database tuples w.r.t query attribute
2. sign consecutive pairs
3. build B+ tree on top of it
4. return tuples [a-1, b+1] together with signatures in [a-1, b]. (query is [a, b]) (a, b here are index)
5. verify any two consecutive pairs

Condensed RSA (NDSS'04)

- Server:
  - Selects records matching posed query
  - Multiplies corresponding RSA signatures
  - Returns single signature to querier



Server

Given t record signatures:

$\{\sigma_1, \sigma_2 \ldots \sigma_t\}$,

compute combined signature

$\sigma_{1,t} = \prod \sigma_i \bmod n$

Send $\sigma_{1,t}$ to the querier

$\xrightarrow{\sigma_{1,t}}$

Querier

Given t messages:

$\{m_1, m_2 \ldots m_t\}$ and $\sigma_{1,t}$

verify combined signature:

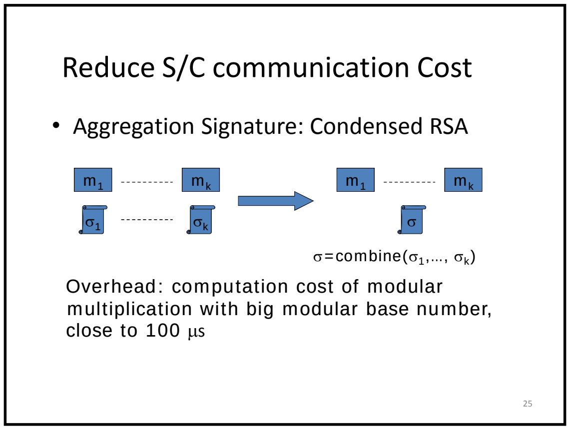
$(\sigma_{1,t})^e = ? = \prod h(m_i) \pmod n$

N is RSA modulus of the public key from the owner

Comparing Cryptographic OP

- one hashing takes 2-3 µs
  - Modular Multiplication -100 times slower
  - Verifying -1000 times slower
  - Signing -10000 times slower

$t_{Hashing} < t_{mod\_M} < t_{ver} < t_{Sign}$



Reduce S/C communication Cost

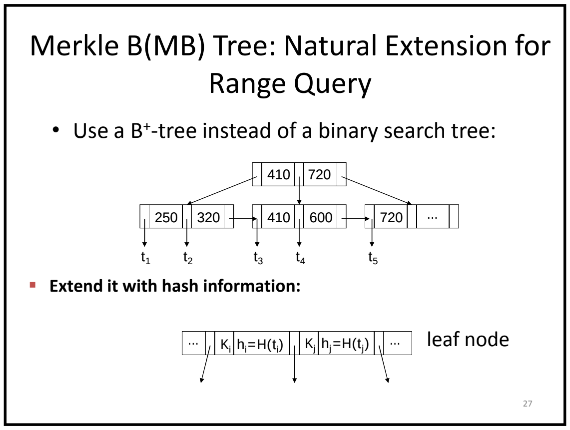
- Aggregation Signature: Condensed RSA

σ=combine(σ_1,…, σ_k)

Overhead: computation cost of modular multiplication with big modular base number, close to 100 μs

Signature Chaining Issues

- A heavy burden on the owner to produce the signatures
- Overhead on the client to verify the aggregated signature
- Storage overhead at the server to store the signatures (which potentially leads to higher computational cost to retrieve them)
- High communication overhead on both the server and the owner, in order to exchange the signatures

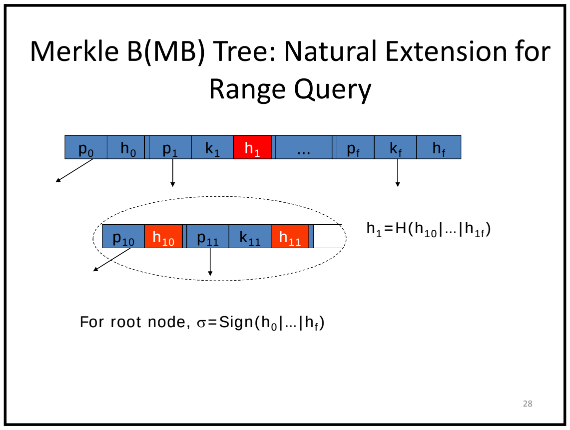


Merkle B(MB) Tree: Natural Extension for Range Query

- Use a B$^+$-tree instead of a binary search tree:

- **Extend it with hash information:**

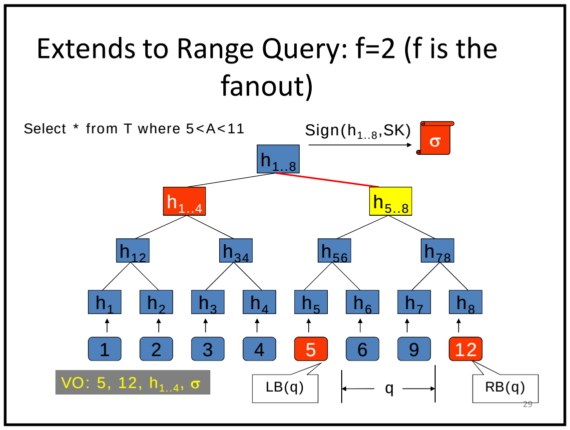
leaf node



Merkle B(MB) Tree: Natural Extension for Range Query

h_1=H(h_{10}|…|h_{1f})

For root node, σ=Sign(h_0|…|h_f)

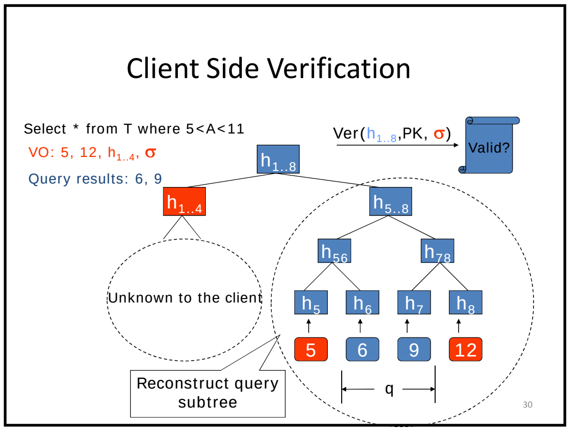


Extends to Range Query: f=2 (f is the fanout)

Select * from T where 5<A<11

Sign($h_{1..8}$,SK)

VO: 5, 12, $h_{1..4}$, σ



Client Side Verification

Select * from T where 5<A<11

VO: 5, 12, $h_{1..4}$, σ

Query results: 6, 9

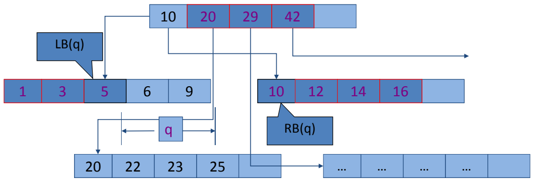
Ver($h_{1..8}$,PK, σ)

Query Example: f=5

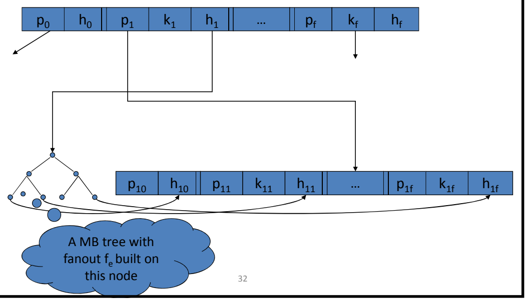
VO: tuple 5, 10, hash of 1, 3, 12, 14, 16,

hash of entry 20, 29, 42

8 hashes



Embedded Merkle B (EMB) tree: A fractal structure



EMB tree Analysis

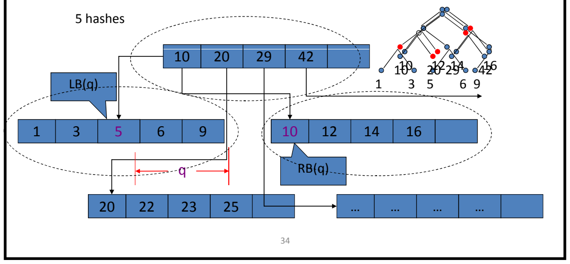
- We can show that:
  - Query cost is as a MB tree with fanout f_k
  - Authentication cost (c/s comm. cost and client verification cost) is as a MB tree with fanout f_e, intuition:
  - f_k is smaller than a normal MB tree given a page size P

Query Example: f=5

VO: tuple 5, 10, hash of red circle node,

hash of red circle nodes(2), hash of red circle nodes(2),

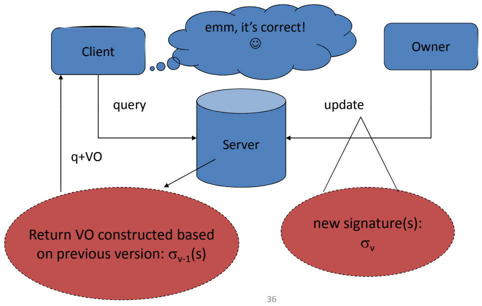
5 hashes



EMB tree's variants

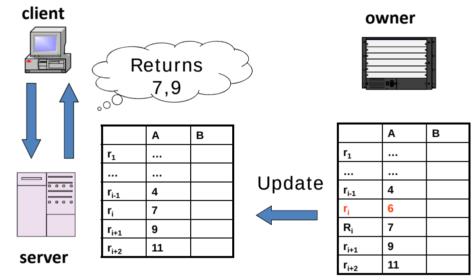
- Don't store the embedded tree, build it on the fly – EMB^- tree
  - Fanout f_k is as a normal MB tree, better query performance, better storage performance
- Use multi-way search tree instead of B^+ tree as embedded tree – EMB^* tree
  - Hash path in the embedded tree could stop in index level, not necessary to go to the leaf level, hence reduce the VO size

Freshness?



Problem 3: Omission

Select * from T where 5<A<11



	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	7	
r_{i+1}	9	
r_{i+2}	11	

	A	B
r_1	…	
…	…	
r_{i-1}	4	
r_i	6	
R_i	7	
r_{i+1}	9	
r_{i+2}	11	

Solution to Freshness

- Must have client-owner communication
  - Reduce this communication cost is the key issue
  - Observation: this cost is correlated with the number of signatures maintained in the authentication structure used by the owner

Other Query Types

- Join
- Projection
- Aggregate

Tradeoff: query vs. authentication efficiency

- Key observations:
  - Query efficiency vs. authentication efficiency
  - Impossible to have one solution that optimizes all cost metrics

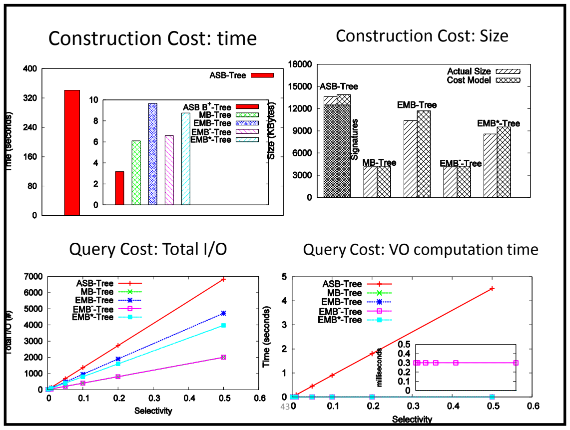
Comparing Cryptographic OP

- one hashing takes 2-3 μs
  - Modular Multiplication -100 times slower
  - Verifying -1000 times slower
  - Signing -10000 times slower
- Why is verifying faster?!

$$t_{Hashing}<t_{mod\_M}<t_{ver}<t_{Sign}$$

Experiments

- Experiment setup
  - Crypto function – Crypto++ and OpenSSL
  - Pagesize: 1KB
  - 100,000 tuples
  - 2.8GHz Intel Pentium 4 CPU
  - Linux Machine

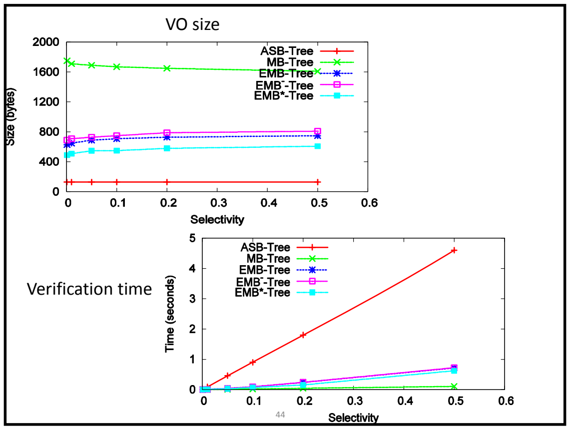


Construction Cost: time

Construction Cost: Size

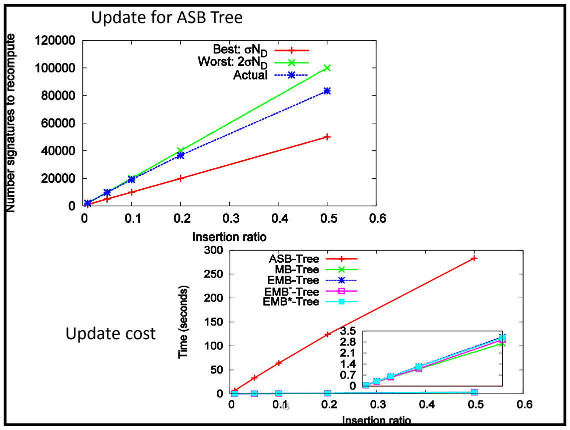
Query Cost: Total I/O

Query Cost: VO computation time



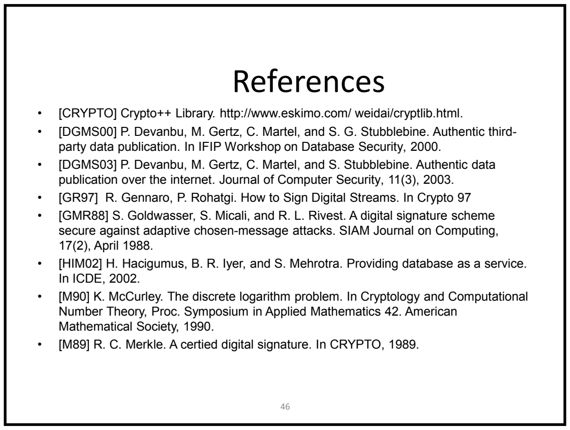
VO size

Verification time



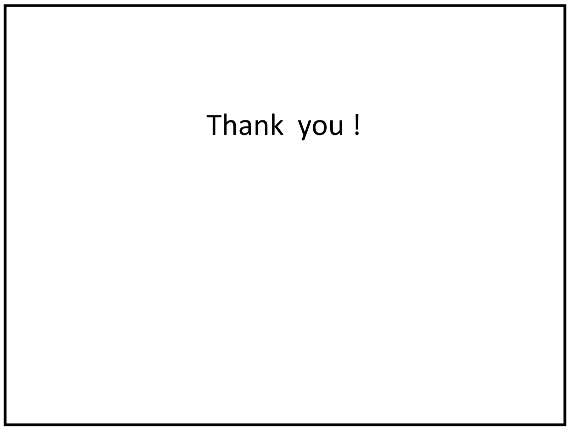
Update for ASB Tree

Update cost



References

- [CRYPTO] Crypto++ Library. http://www.eskimo.com/ weidai/cryptlib.html.
- [DGMS00] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic third-party data publication. In IFIP Workshop on Database Security, 2000.
- [DGMS03] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic data publication over the internet. Journal of Computer Security, 11(3), 2003.
- [GR97] R. Gennaro, P. Rohatgi. How to Sign Digital Streams. In Crypto 97
- [GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing, 17(2), April 1988.
- [HIM02] H. Hacigumus, B. R. Iyer, and S. Mehrotra. Providing database as a service. In ICDE, 2002.
- [M90] K. McCurley. The discrete logarithm problem. In Cryptology and Computational Number Theory, Proc. Symposium in Applied Mathematics 42. American Mathematical Society, 1990.
- [M89] R. C. Merkle. A certied digital signature. In CRYPTO, 1989.



Thank you !