Authenticated Data Structures and untrusted DB applications

Bernardo Palazzi
Maurizio Pizzonia

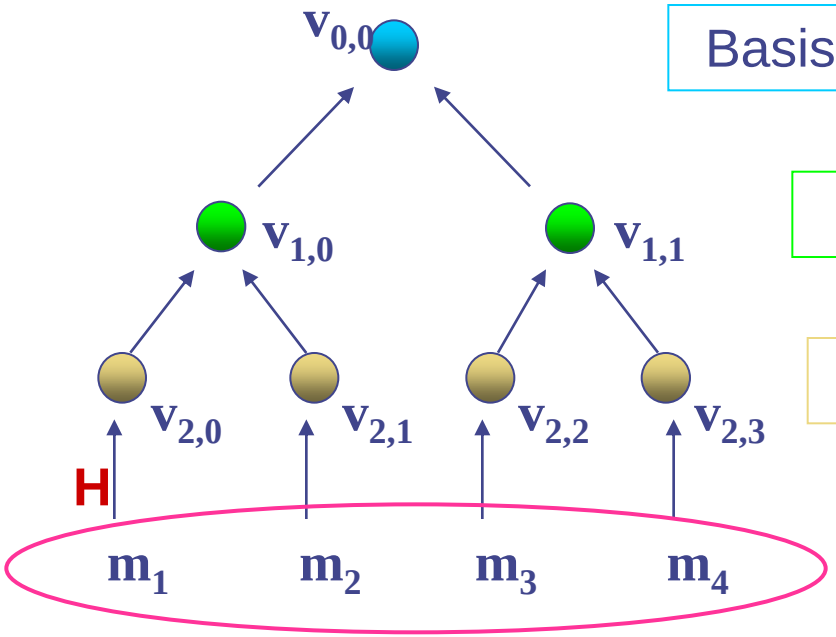
DIA – Roma Tre University, IT

Cryptographic Hash Function

- Preimage resistance (one-way)
  - Given hash value x, hard to find plaintext P such that h(P) = x
- Second preimage resistance (weak collision resistance)
  - Hard to find pair of plaintexts P and Q such that h(Q) = h(P)
- Collision resistance (strong collision resistance)
  - Given plaintext P, hard to find plaintext Q such that h(Q) = h(P)
- Collision resistance implies second preimage resistance
- Domain of hash values should be large to prevent exhaustive search

Hash Tree (Merkle): building

- Basis: authenticated tree root
- Authentication structure
- Data hash



Basis = $V_{0,0}$ = H [($V_{1,0}$) || ($V_{1,1}$)]

$V_{1,1}$ = H [($V_{2,2}$) || ($V_{2,3}$)]

$V_{2,2}$ = H(m_3)

$V_{2,3}$ = H(m_4)

data must be ordered

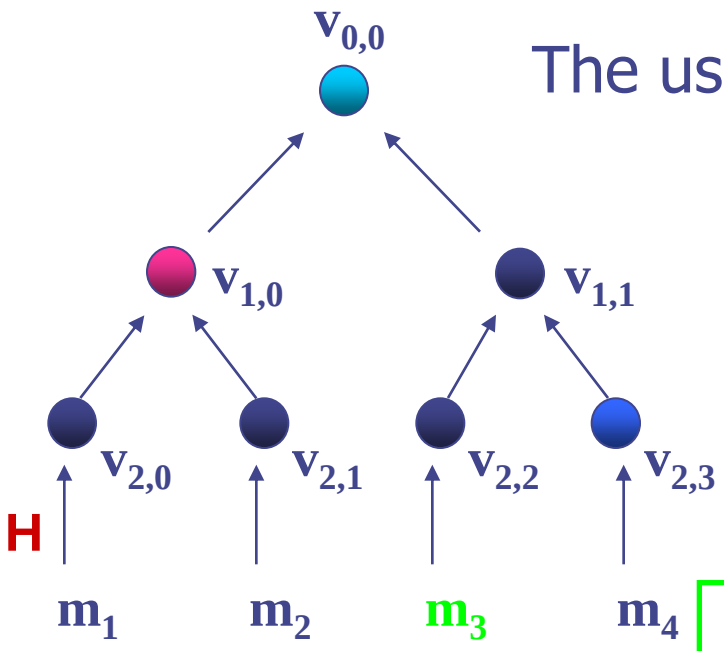
H is a Hash function

Hash Tree (Merkle): test

A user would verify data authenticity of m_3

Authenticated answer is made by: $\mathbf{m_3}$, $V_{2,3}$, $V_{1,0}$

And from the Basis signed by a CA.



The user can verify if m_3 is authentic:

$V_{2,2} = H\ m_3$

$V_{1,1} = H\ (\ V_{2,2} \| \ V_{2,3})$

$V_{0,0} = \ H\ (\ V_{1,0} \| \ V_{1,1})$

If **Basis** == $V_{0,0}$ then m_3 is authentic

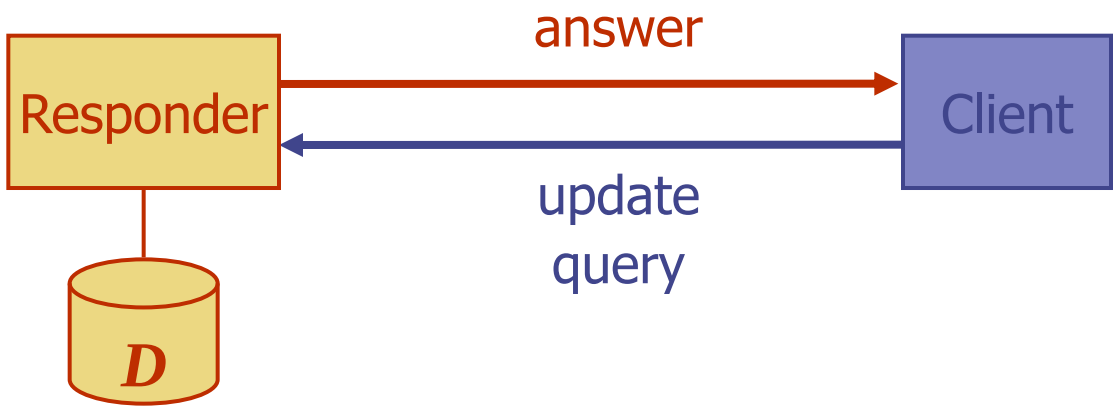
dynamicity

- updating the Hash Tree require keeping it balanced
  - to have O(log n) depth
- techniques similar to AVL and RB trees

Data Outsourcing Models

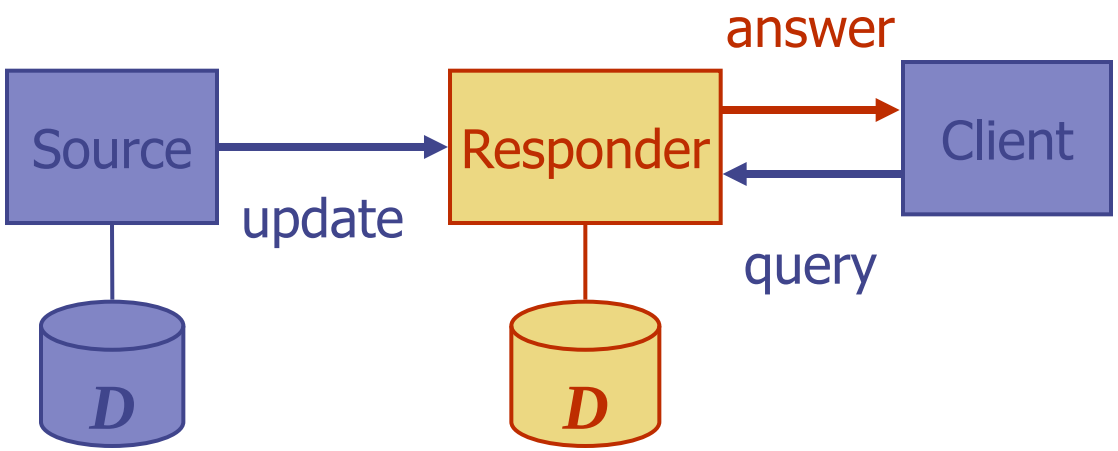
Two –party model:

- Client issues queries and updates
- Responder stores data structure



Three-party model:

- Source issues updates
- Source and responder store data structure
- Client issues queries



Two-Party Example

- Remote file system

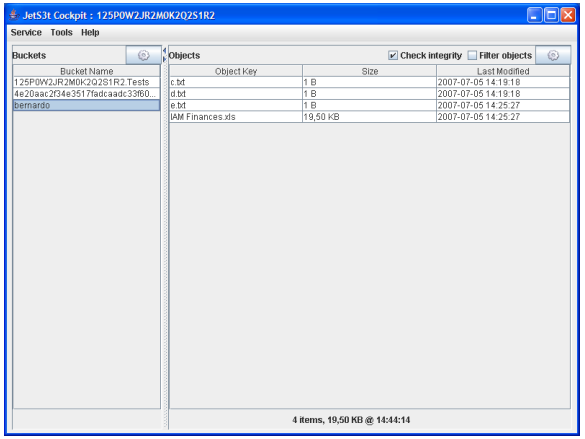
Responder

Client

Amazon Simple Storage Service (S3)

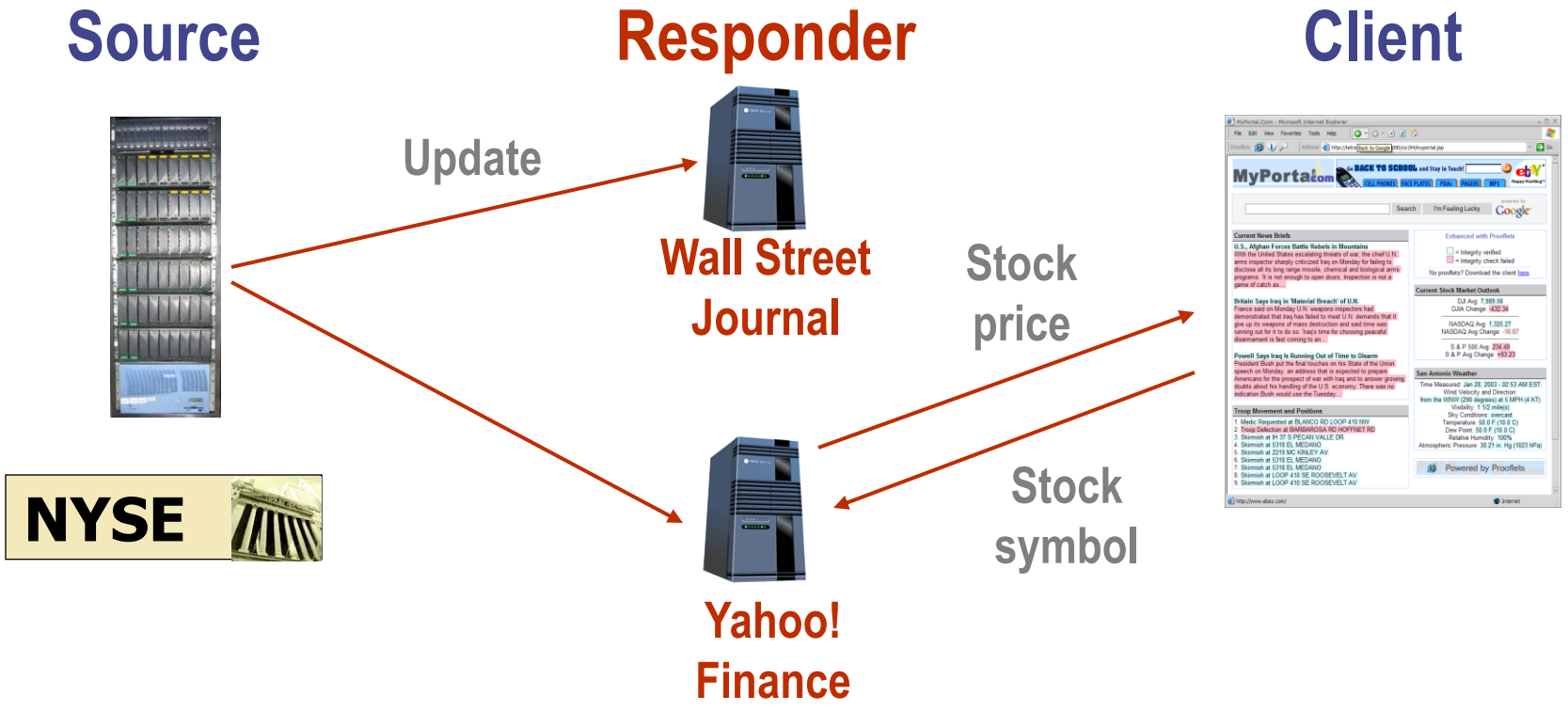
Download

Query
Upload



Three-Party Example

- Dissemination of stock quotes by financial portals
- Content delivery network (e.g., Akamai)



Security Problems

Authentication

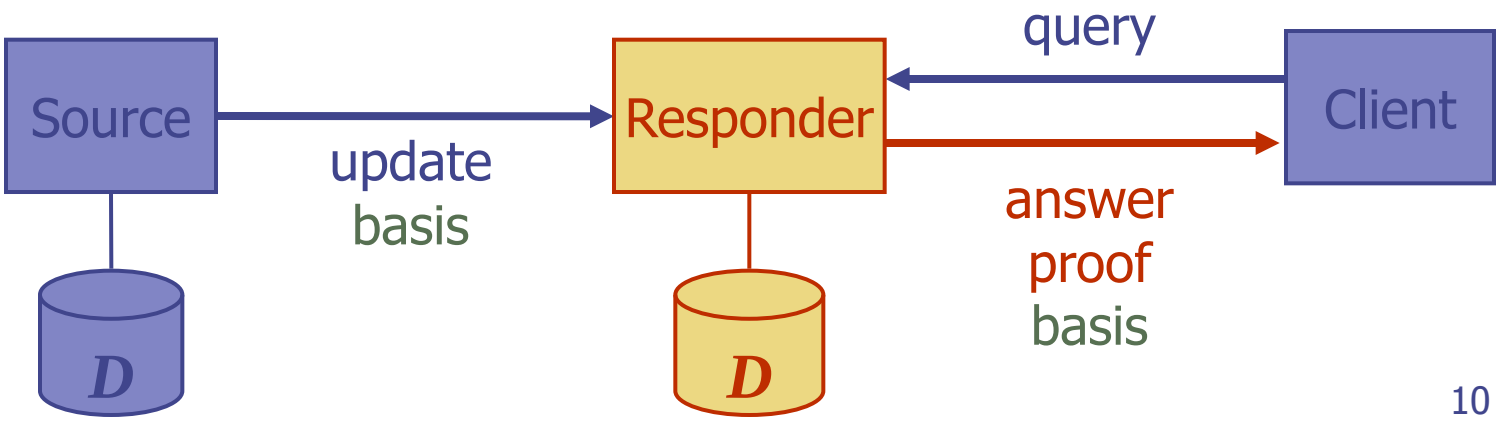
- Integrity: is the item retrieved the same as the one inserted?
- Soundness: was an item in the answer previously deleted?
- Completeness: is there a missing item in the answer?

Confidentiality

- Does the responder see the data?
- Does an eavesdropper see the data?
- Standard encryption techniques can be used
- Not addressed in this talk

Authenticated Data Structure

- Update includes basis (signed statements)
- Answer includes basis (forwarded from source) and proof (computed by responder)
- Client verifies answer using proof and relying on basis
- Regular updates and timestamp to avoid replay attacks
- A polynomial-time responder cannot find an incorrect answer and a proof for it that passes verification



Benefits and Applications

Benefits

- Centralized trust:
  - Users trust only the source
- Distributed service
  - Responders serve as authentication caches on the network
- Low deployment cost
  - Responders do not need secure installations
- Resiliency to denial-of-service attacks
  - The source does not answer queries

Applications

- Investing
  - Data: stock quotes
  - Source: NASDAQ, NYSE
- Credit Cards
  - Data: valid card numbers
  - Source: Visa, AmEx
- Public Key Infrastructure
  - Data: Web server certificates
  - Source: Verisign, Entrust
- Intrusion Detection
  - Data: fingerprints of OS files
  - Source: Microsoft, Sun

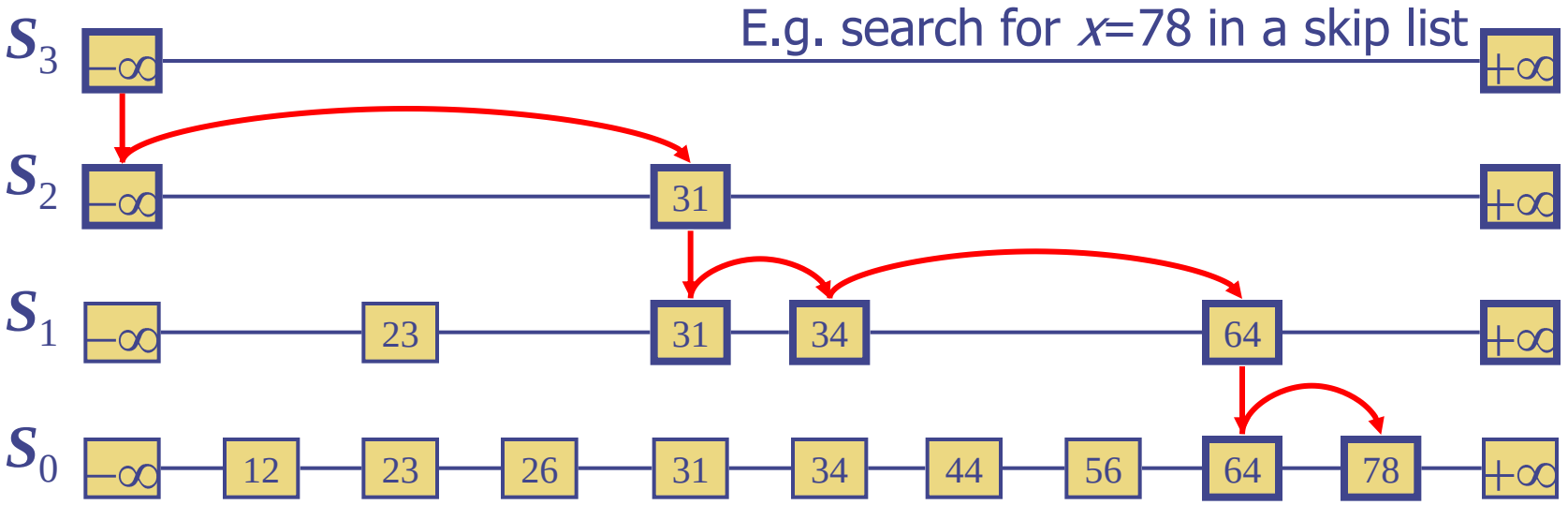
Skip List

A **skip list**, introduced by Pugh '89, for a set S of distinct (key, element) items is a series of lists $S_0, S_1, \ldots, S_h$

- List S_0 contains the keys of S in non-decreasing order
- Each list is a subsequence of the previous one, with probability ½

i.e. $S_0 \supseteq S_1 \supseteq \ldots \supseteq S_h$

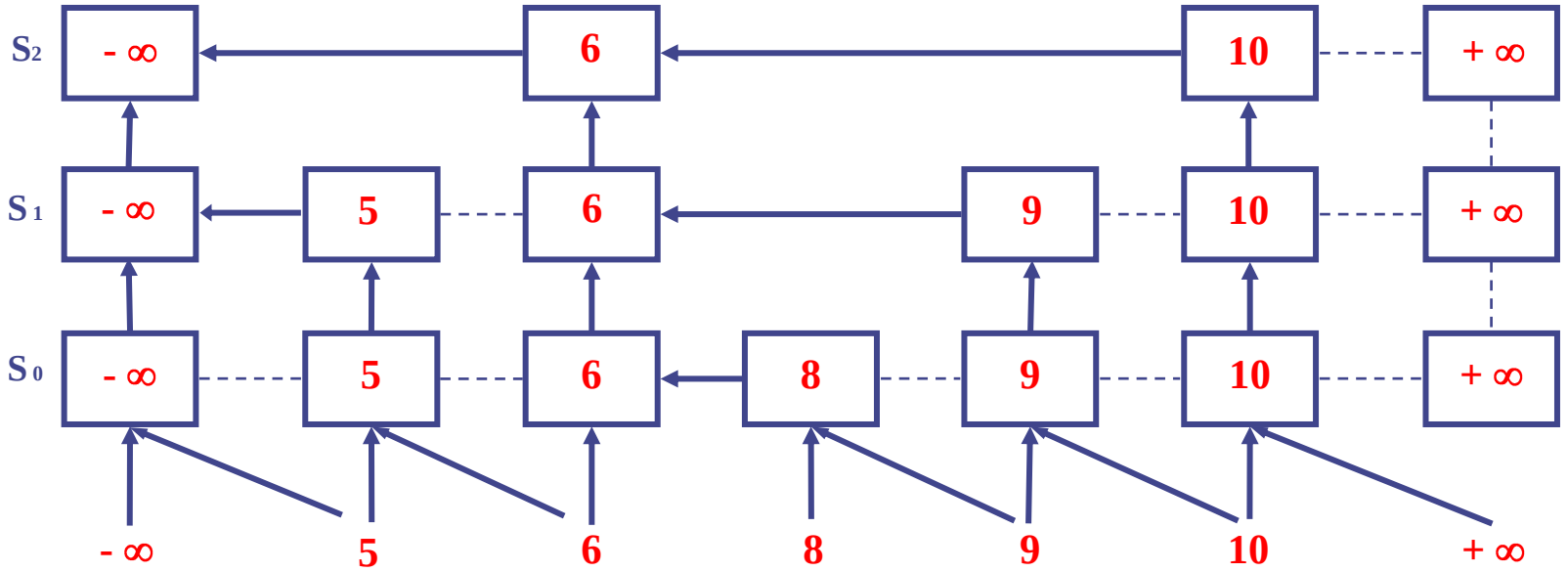
The **search** for a key x starts at first position of the top list and if $x > y$: we "scan forward" or if $x < y$: we "drop down"



skiplist and DBs

- skip lists are easy to represent in a db
  - e.g. set a maximu «tower» height
    - feasible because height is O(log n)
  - represent a «tower» in one record
- random version does not need balancing

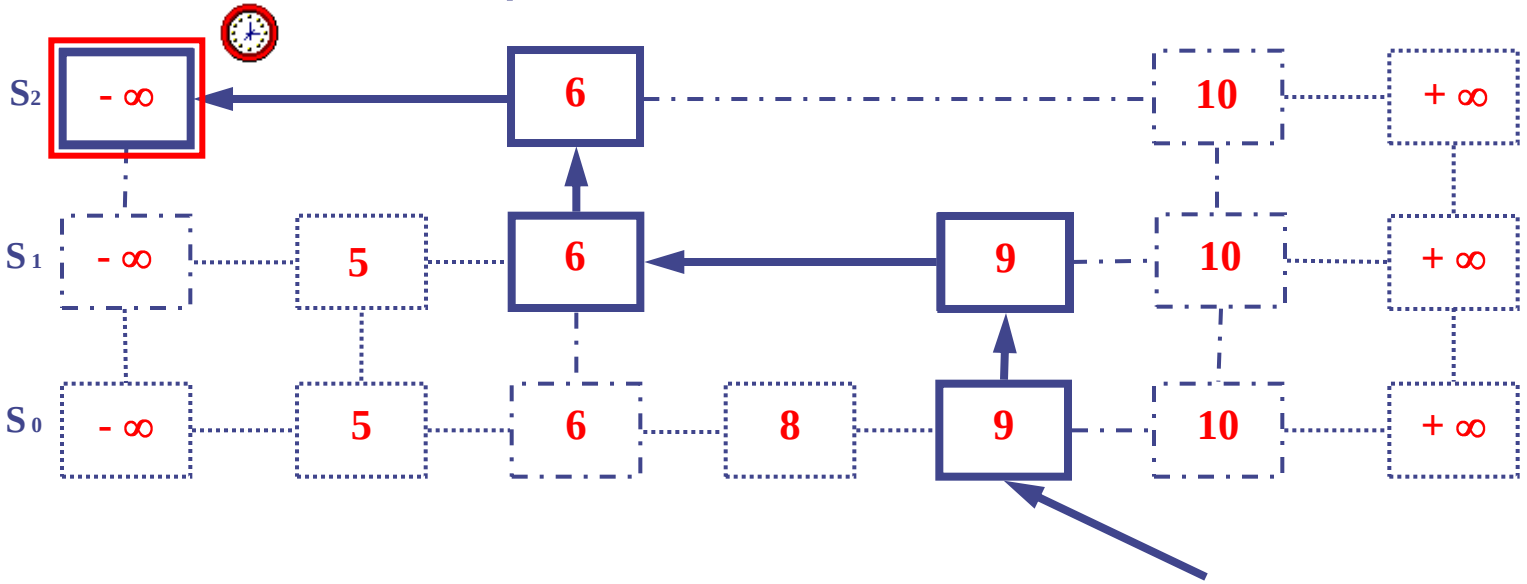
Authenticated Skip List



- Hierarchical hashing over a directed acyclic graph.
- Commutative Hash *H(x, y) == H(y, x)* [Goodrich, Tamassia 2000]
- The arrows denote the flow of authenticated information.

Authenticated Skip List: Main Concepts

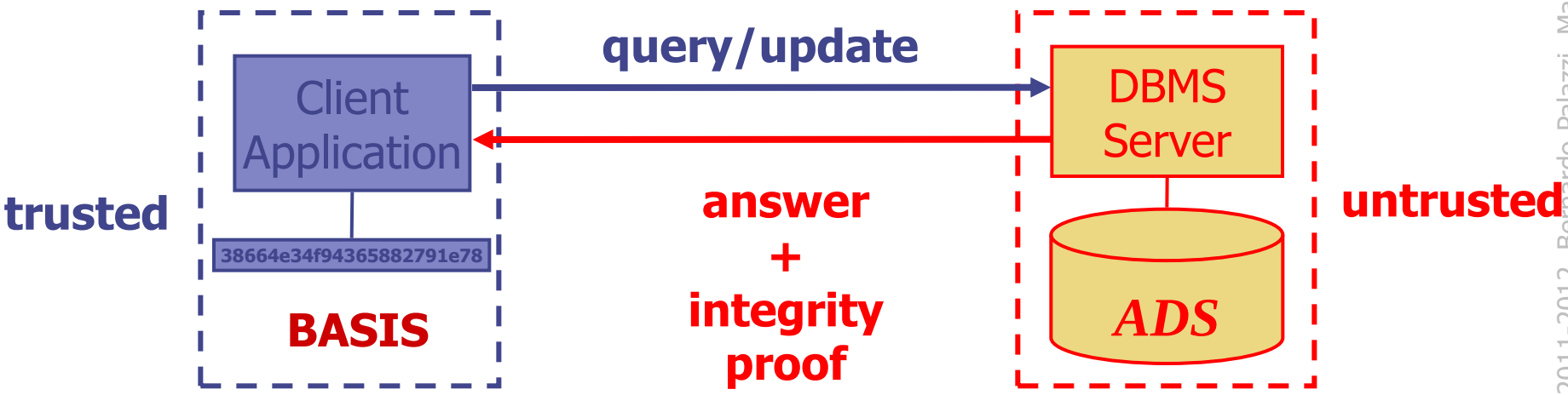
Basis: signed time-stamped hash of the start node
Proof: hashes of neighbours of nodes on search path
Verification: computation of a hash chain



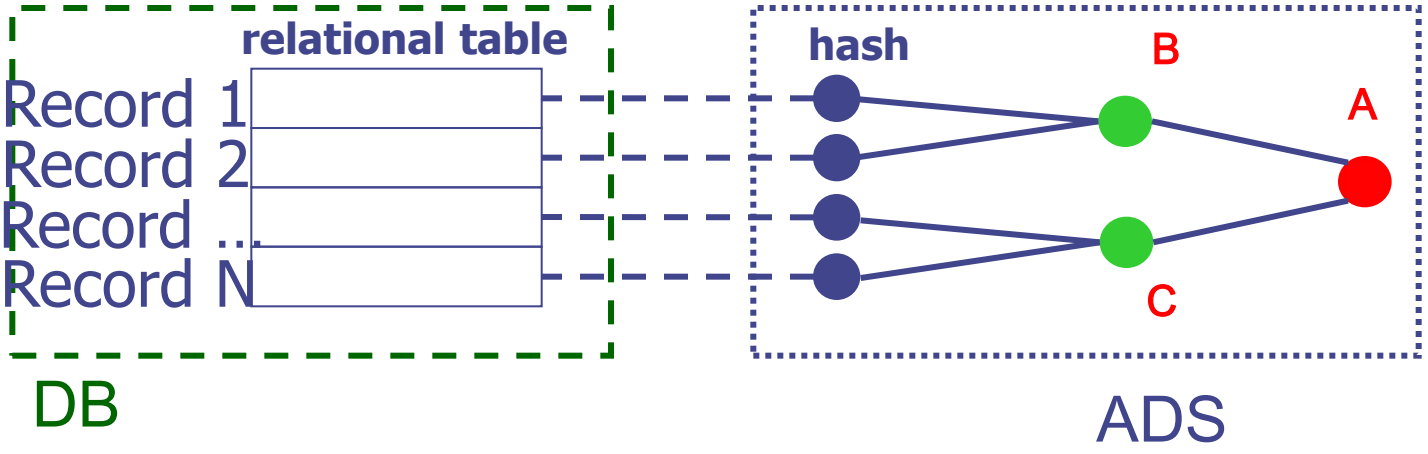
The user **Verification** for element 9 is obtained by simply hashing the values of the **Proof**, and comparing the result with the signed **BASIS**.

Reference Model Overview

- This method **does not require trust in the db manager or DBMS**
- A client application, external to the DBMS, is the source of data.
- To manage authentication the **client stores only the basis**
- Secure storage of tables is possible by the use of techniques to represent hierarchical data structures in relational tables and special queries that allow an efficient selection of the authentication information.

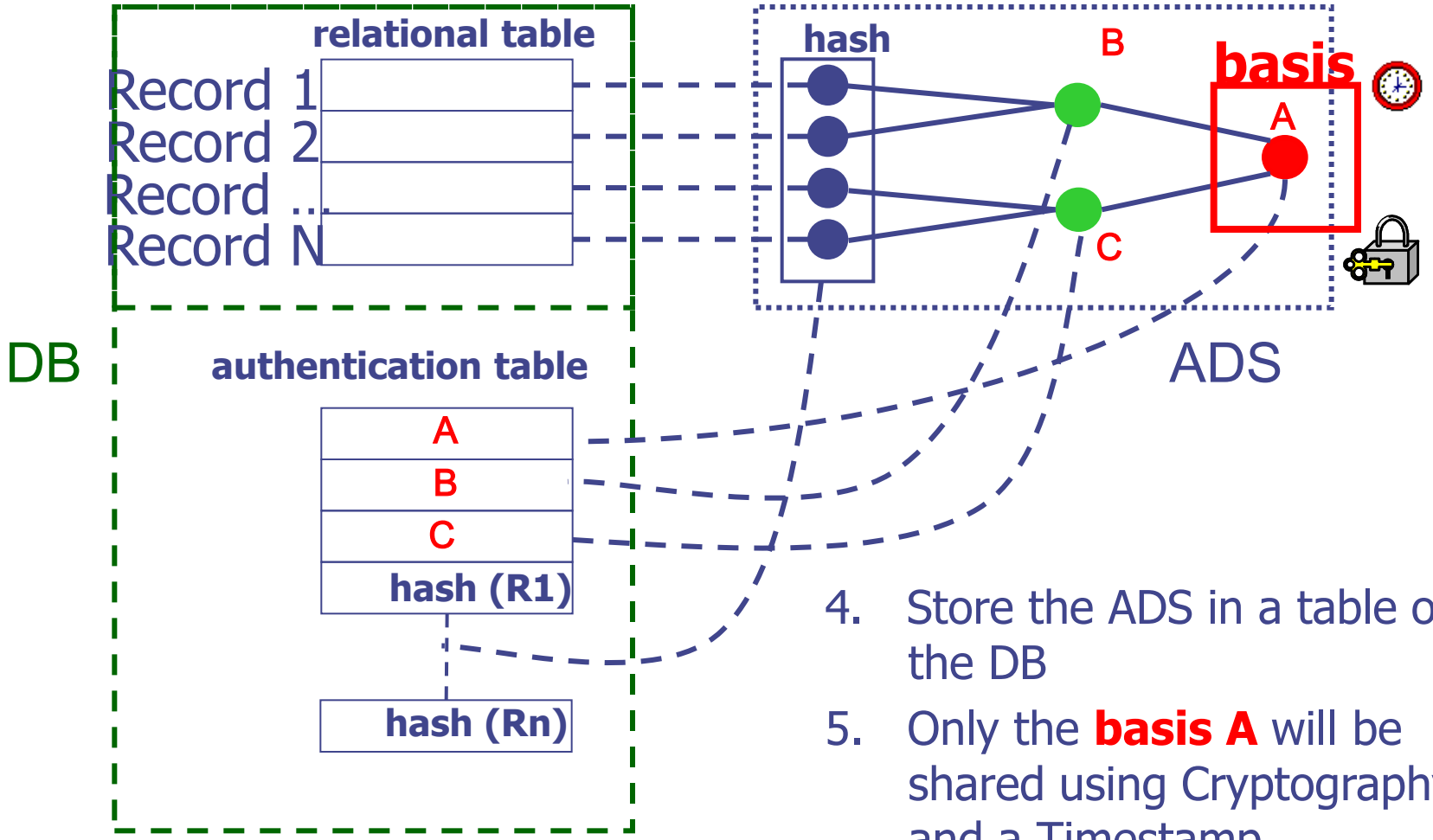


Technical Overview



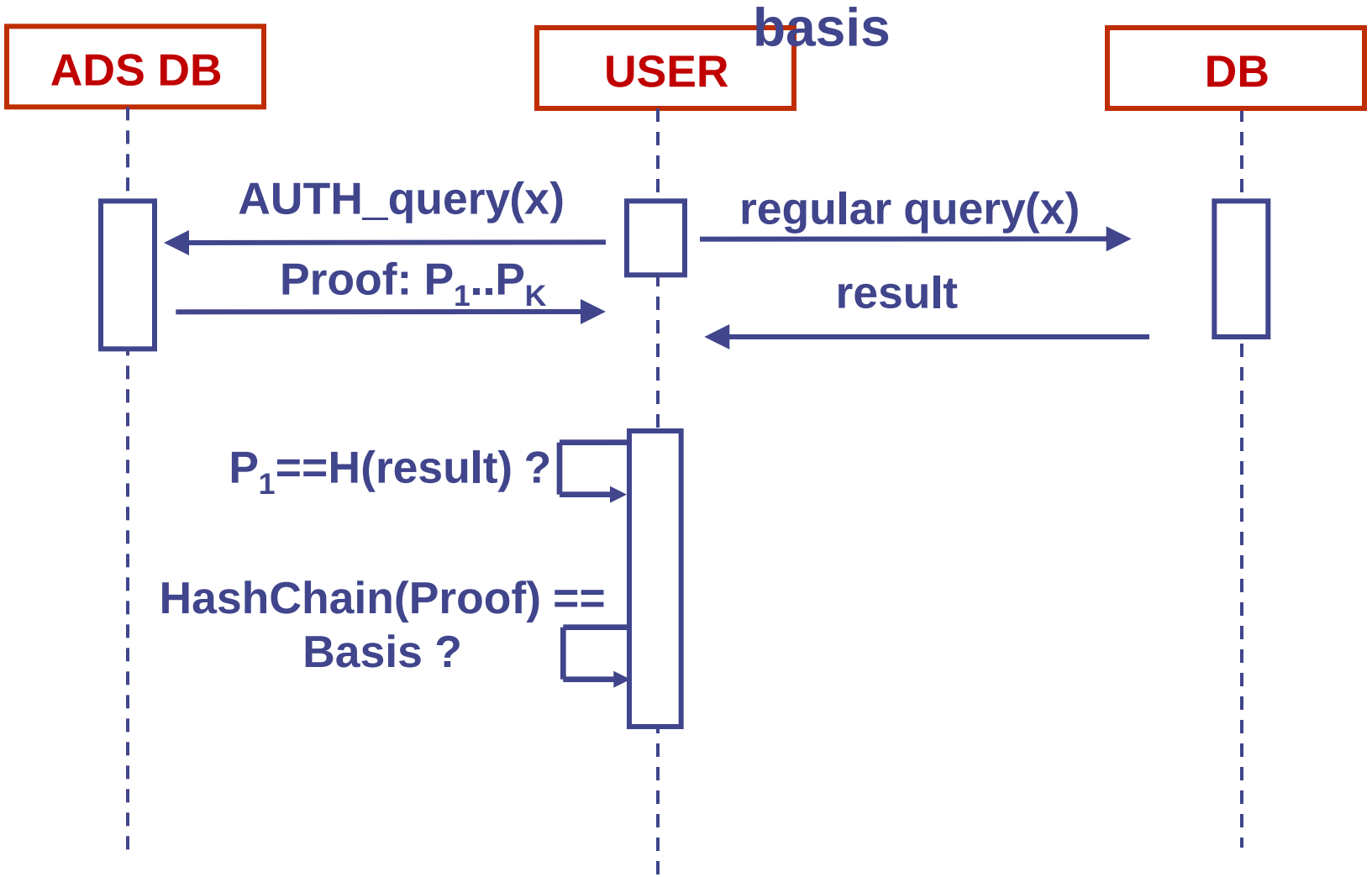
1. Start with a relational table in a DB
2. Compute a hash for each record, it is possible to use different methods:
    1. Concatenation of value in all fields
    2. Build an ADS for each record to allow the use of projection operator
    3. …
3. Build an ADS using the hashes

Technical Overview(2)

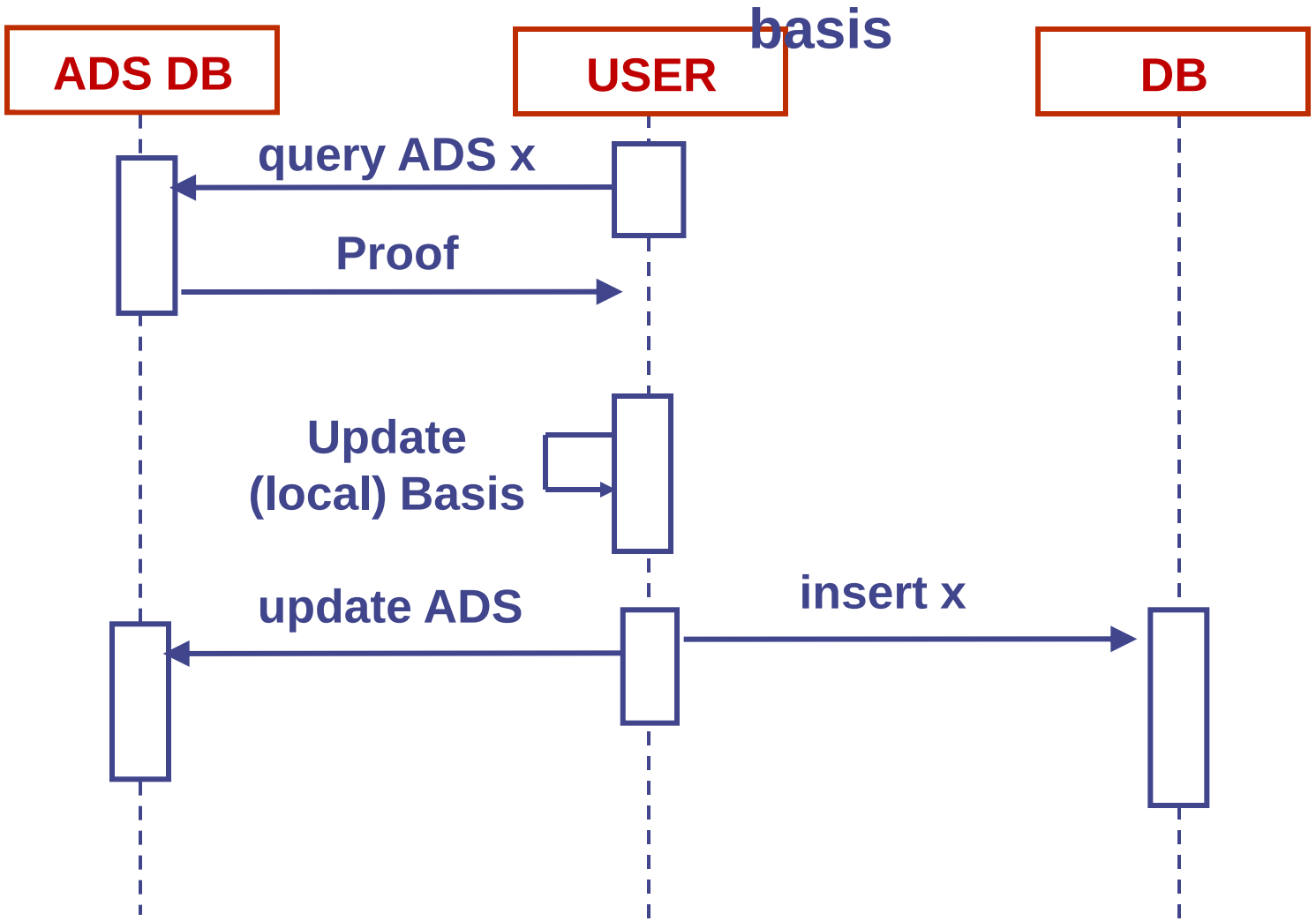


4. Store the ADS in a table on the DB
5. Only the **basis A** will be shared using Cryptography and a Timestamp

Authenticated query



Authenticated update



Acknowledgements

- Roberto Tamassia at Brown University