

Title	Proving Properties of Incremental Merkle Trees
Author(s)	Ogawa, Mizuhito; Horita, Eiichi; Ono, Satoshi
Citation	
Issue Date	2009-09-22
Type	Presentation
Text version	publisher
URL	http://hdl.handle.net/10119/8333
Rights	
Description	1st VERITE : JAIST/TRUST-AIST/CVS joint workshop on VERification TEchnologyでの発表資料, 開催 : 2005年9月21日 ~ 22日, 開催場所 : 金沢市文化ホール 3F

Proving Properties of Incremental Merkle Trees

Mizuhito Ogawa , Eiichi Horita, Satoshi Ono

September 22nd, 2005

JAIST-AIST joint workshop, VERITE

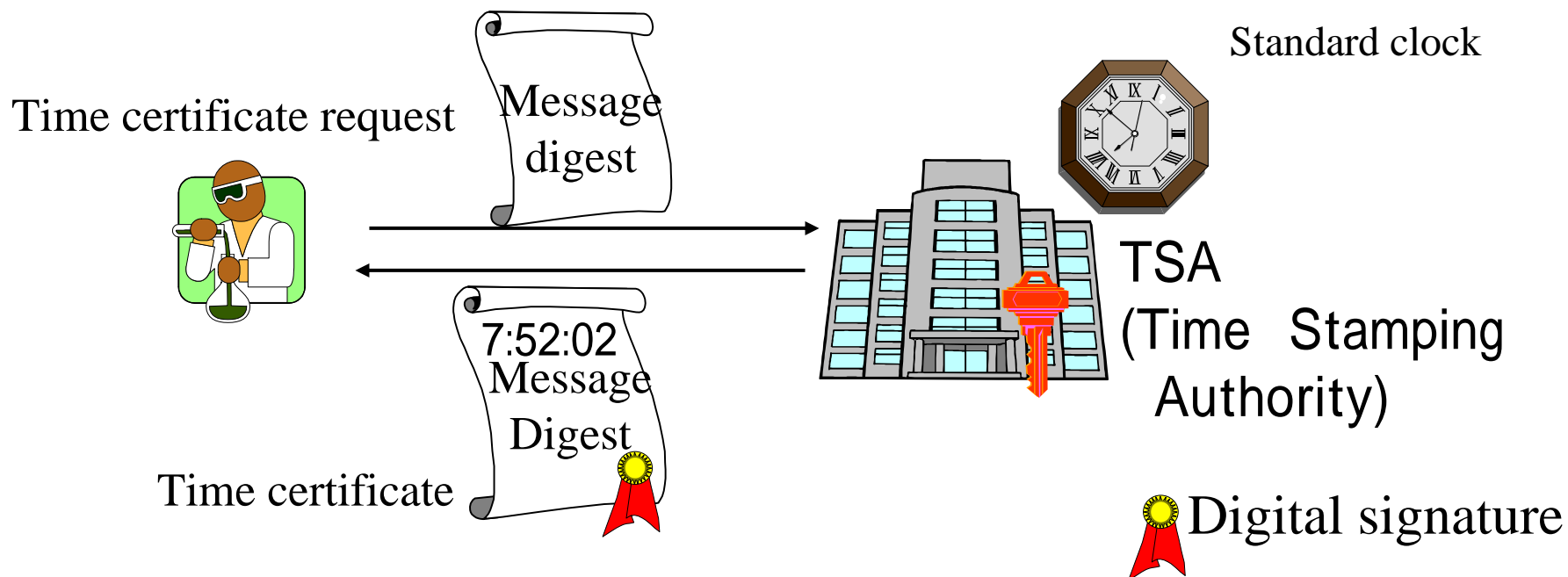
(presented at CADE-20, July 27th 2005)

What is temporal authentication ?

Certificate an occurrence of an transaction at “*time*”

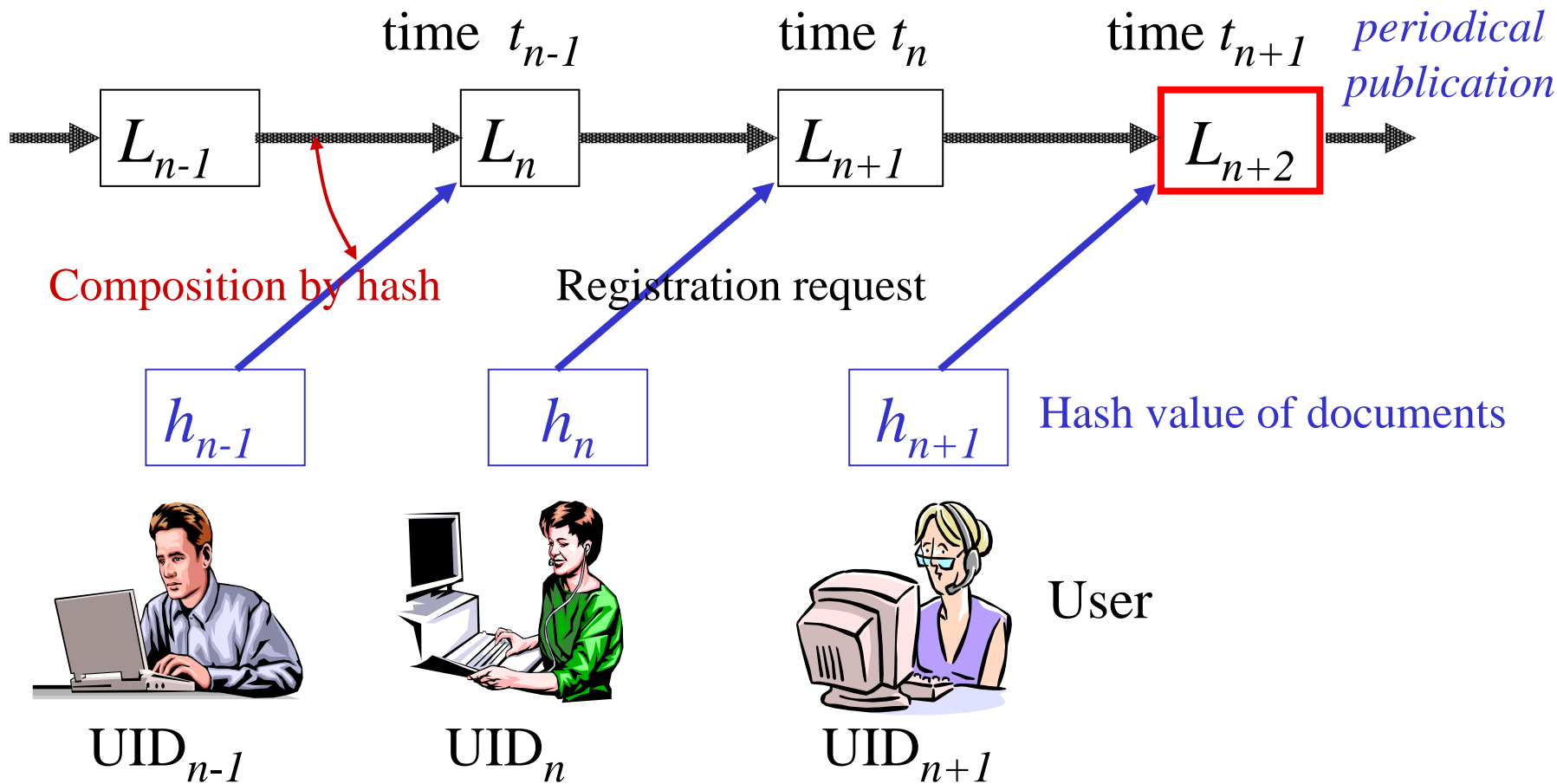
- Time stamp by digital signature (rfc-3161)
- Linking and publication by hash (ISO 18014-3)

Time stamp by digital signature (rfc-3161)



Linking and publication by hash function (ISO18014-3)

Compose past time stamps by a hash function



We assume collision-resistance, one-way hash function.

Time stamp by digital signature v.s. Linking by hash

Time stamp by digital signature

First certificate

Pros:

- Relatively safe for intra-dishonesty by Hardware Secure Module.
- Fine precision (< sec).

Cons:

- Contamination of crypto system invalidates *all* certificates.
- Relatively short life span : ~ 5years.

Linking and publication by hash function

Secondary certificate

Pros:

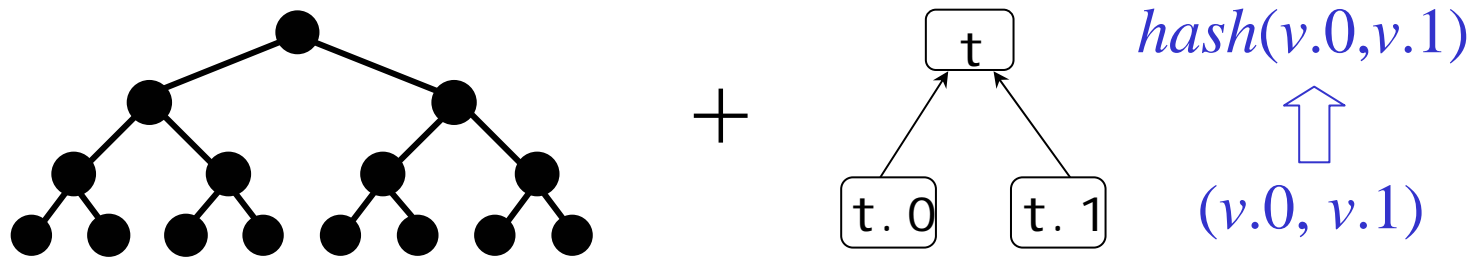
- Relies only on hardness of a hash function (e.g., SHA-1).
- Relatively long life span : ~ 30years.

Cons:

- Hash function has no key; **guarantee required for intra-dishonesty.**
- **During publication period, no auditor can check.**

Merkle tree (Merkle 1979)

- *Merkle tree = Binary tree + hash function*
- Each node has its hash value, computed from a pair of hash values of its children.



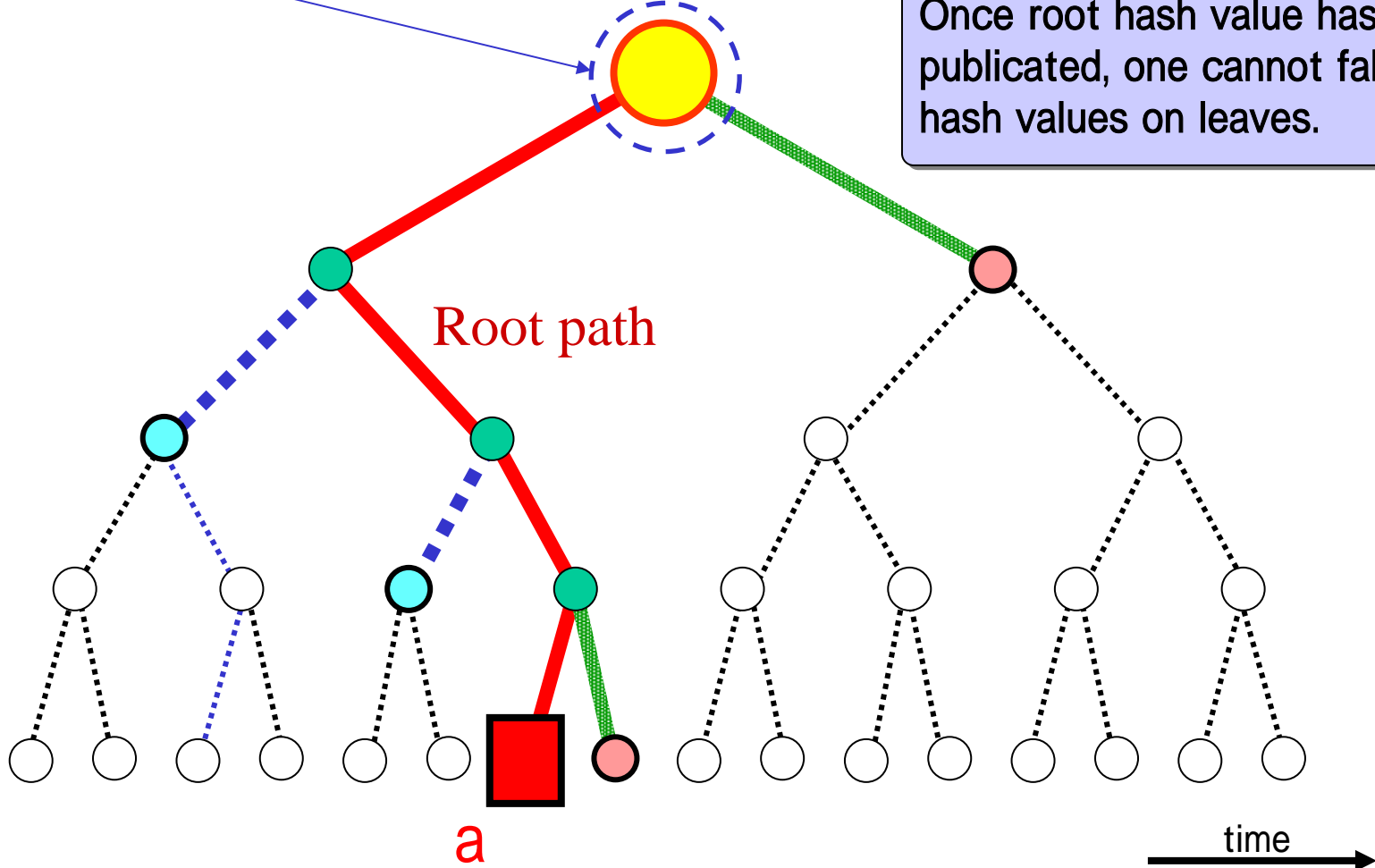
We assume collision-resistance, one-way hash function.

Basic idea : Merkle tree (1)

Public witness
(to be publicated)

Root hash value

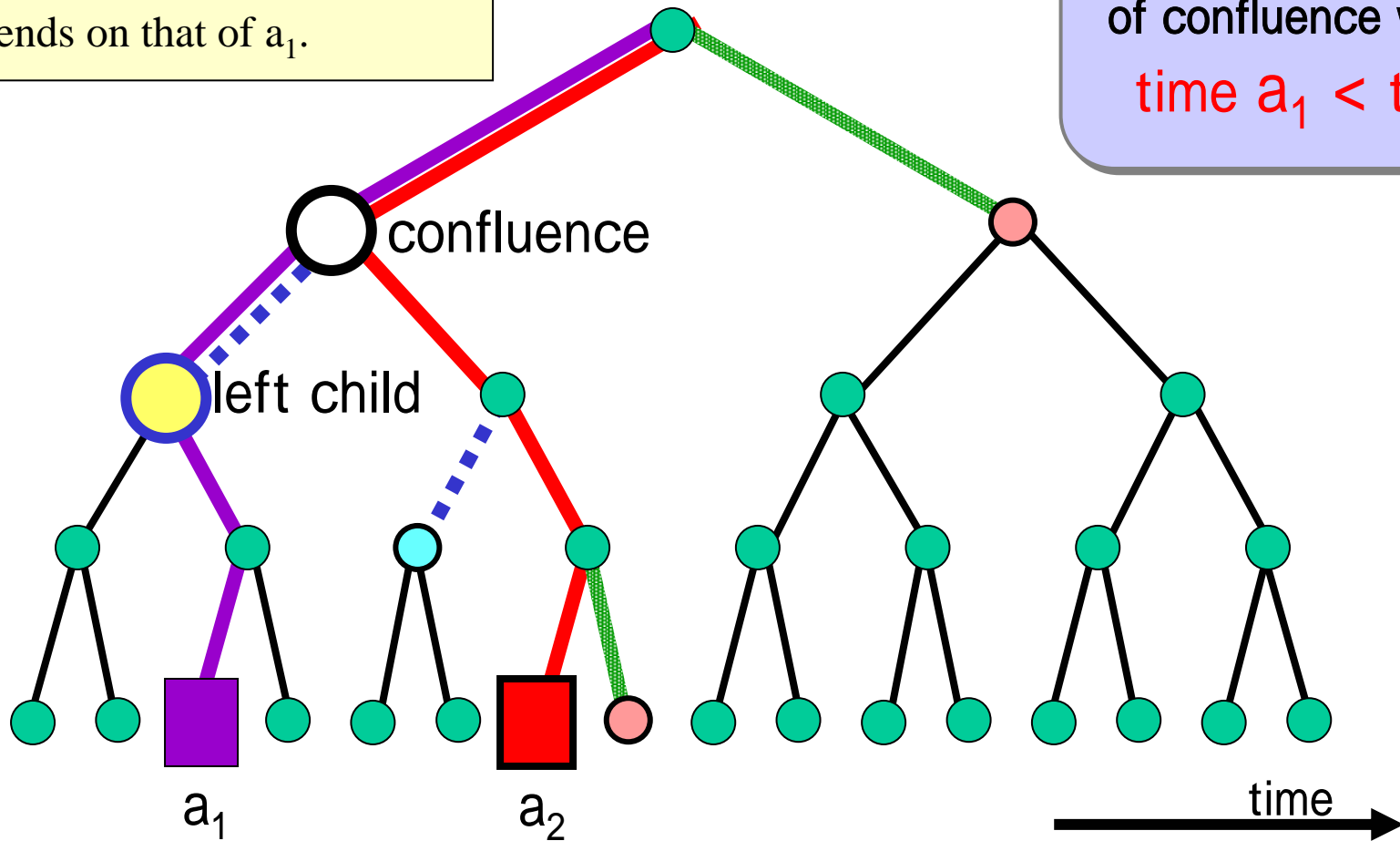
Once root hash value has been publicated, one cannot false hash values on leaves.



Basic idea: Merkle tree (2)

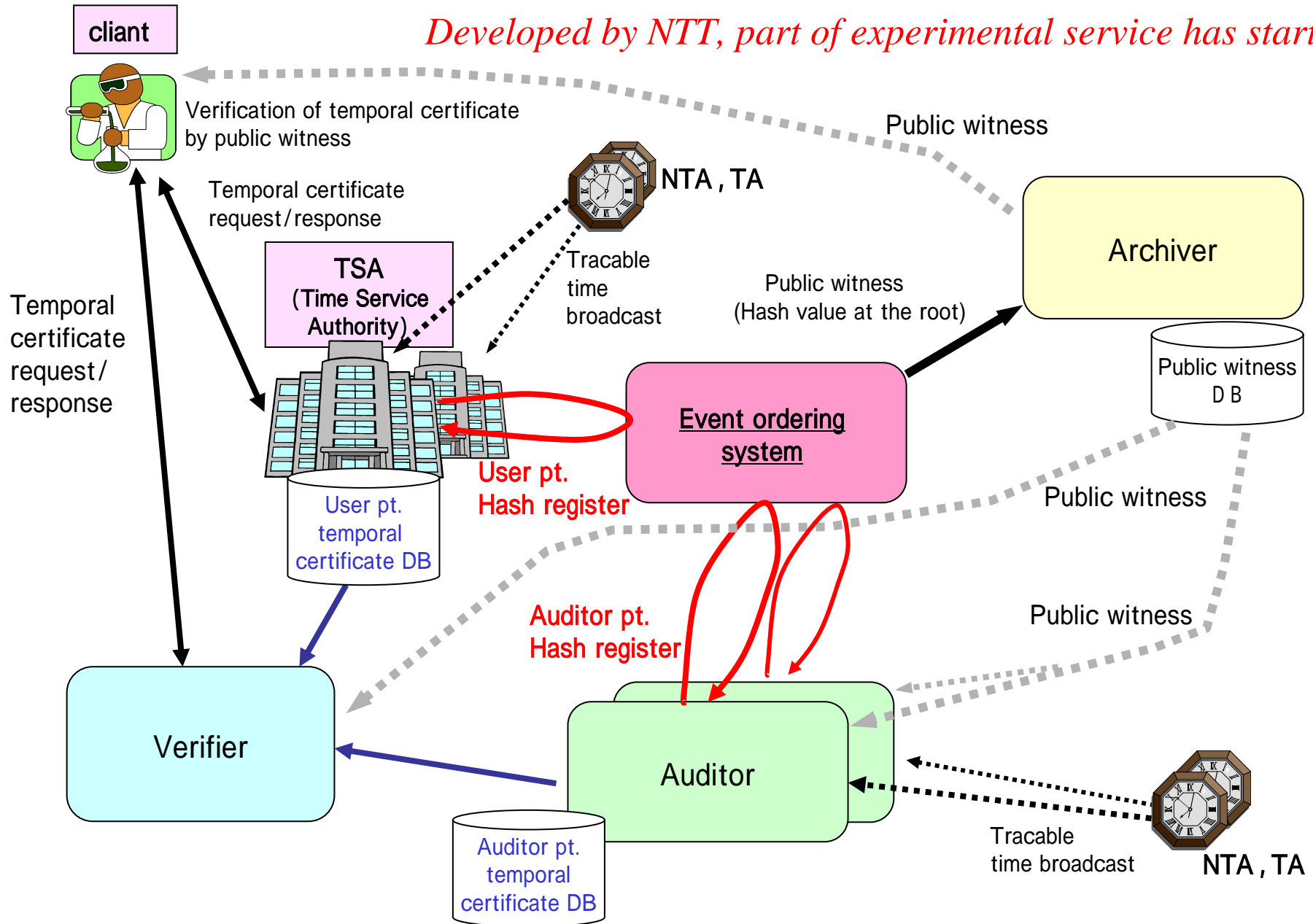
The hash value of the left child depends on that of a_1 .

If the root path of a_1 contains the left child of confluence with a_2
time $a_1 < \text{time } a_2$

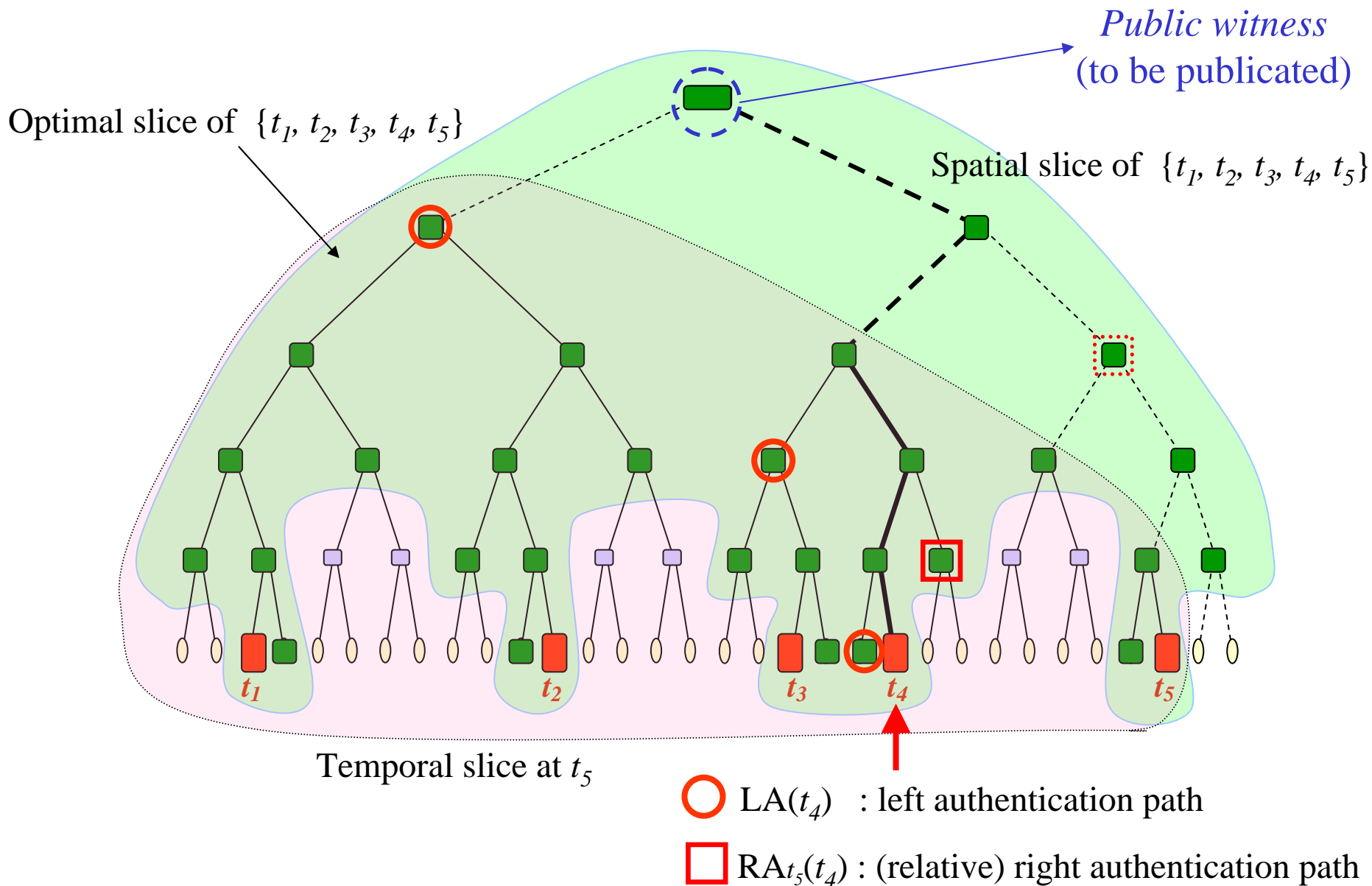


Reference model of event ordering system based on Incremental Merkle trees

Developed by NTT, part of experimental service has started



Incremental Merkle trees construction for registration requests at t_1, t_2, t_3, t_4, t_5



Protocol between users and an event-ordering system

- We assume that each user will register reasonably frequent.
- Assume a user registers at t_1, t_2, \dots, t_n and receives:
 - $(\text{RA}_{t_1}(t_1), \text{LA}(t_1), \{t_1\})$ at t_1 .
 - $(\text{RA}_{t_i}(t_{i-1}), \text{LA}(t_i), \{t_i\})$ at t_i with $0 < i < n$.

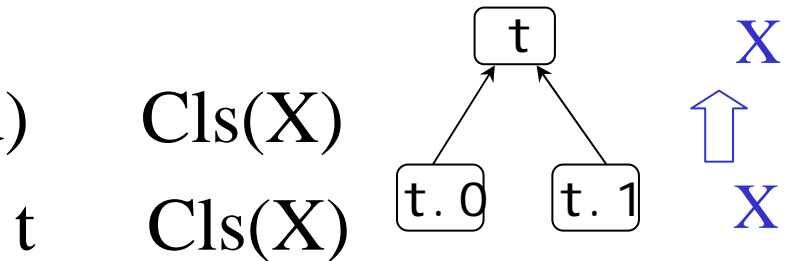
Size $O(n)$
- We denote $\text{LS}(t_i) = \text{LA}(t_i), \{t_i\}$
 $\text{LSR}_{t_{i+1}}(t_i) = \text{LS}(t_i), \text{RA}_{t_{i+1}}(t_i)$

Incremental scheme for Optimal Slice replication

- **Def.** Closure $\text{Cls}(X)$ is the minimum set such that

- $X \subseteq \text{Cls}(X)$

- $t.0$ (left child), $t.1$ (right child)



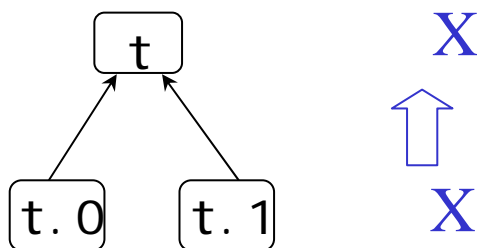
- **Th 1.**

$$\text{OptimalSlice}(\{t_1, t_2, \dots, t_n\}) = \left(\bigcup_{1 \leq i < n} \text{Cls}(\text{LSR}_{t_{i+1}}(t_i)) \right) \cup \text{Cls}(\text{LS}(t_n))$$

Described in WS2S

(incomparable(A) & opt_slice(A,X) & LSRclosure_union(A,Y)) => X = Y;

MONA:WS2S satisfiability checker



```
MeadowNT.exe@CALIBAN
Buffers  Files  Tools  Edit  Search  Mule  Help

ws2s;

var2 X,Y,Z;
var1 s,t,u;

pred preclosure(var2 X,Y) = X sub Y &
  (all1 t: ((t.0 in Y & t.1 in Y) => t in Y));

pred closure(var2 X,Y) = preclosure(X,Y) &
  (all2 Z : preclosure(X,Z) => Y sub Z);

(closure(X,Y) & closure(Y,Z)) => closure(X,Z);
[--]S:** example.mona (MONA Encoded-kbd)-
```

```
~/papers/ono2/mona
$ mona example.mona
MONA v1.4-5 for WS1S/WS2S
Copyright (C) 1997-2002 BRICS

PARSING
Time: 00:00:00.05

CODE GENERATION
DAG hits: 34, nodes: 34
Time: 00:00:00.03

REDUCTION
Projections removed: 0 (of 4)
Products removed: 1 (of 16)
Other nodes removed: 0 (of 13)
DAG nodes after reduction: 32
Time: 00:00:00.02

AUTOMATON CONSTRUCTION
100% completed

Time: 00:00:00.22

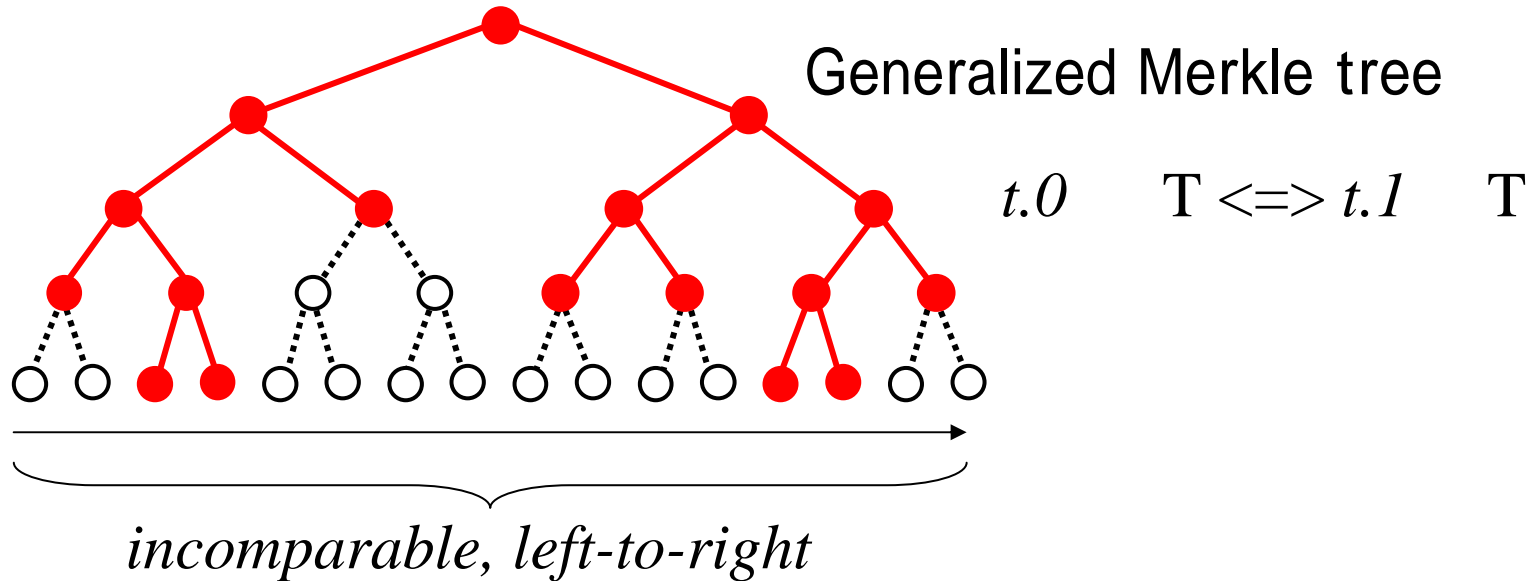
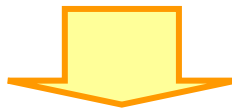
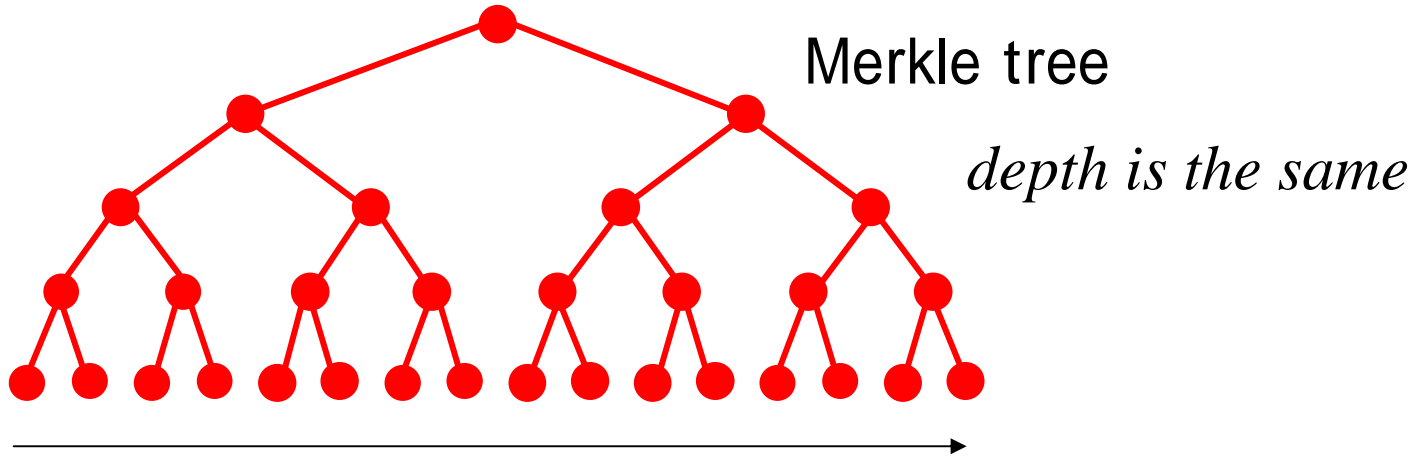
Automaton has 12 states and 36 BDD nodes

ANALYSIS
Formula is valid
```

Trick 1 : Generalized Merkle tree

- MONA **cannot** describe that :
 “a binary tree has the same depth”
 (i.e., *each root path has the same length*)
- We have been implicitly assuming that :
 “a Merkle tree has the same depth”
- We relax *“the same depth”* to just *“don’t care depth”*.

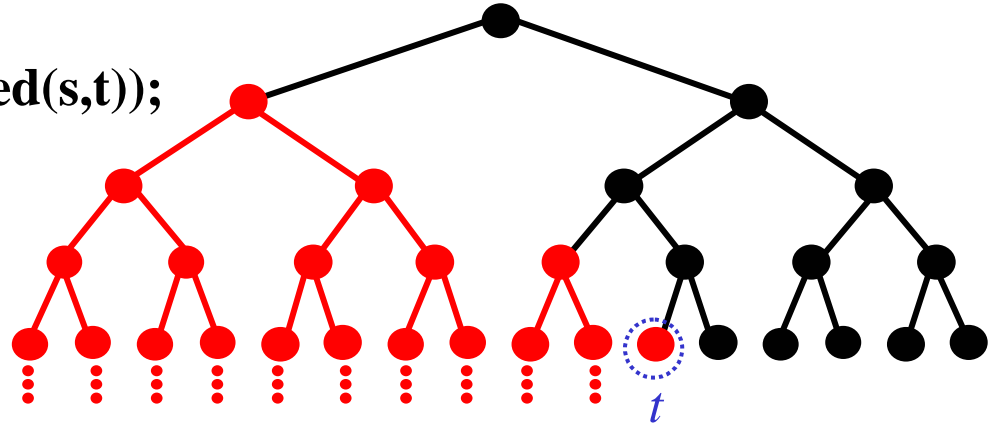
Generalized Merkle tree example



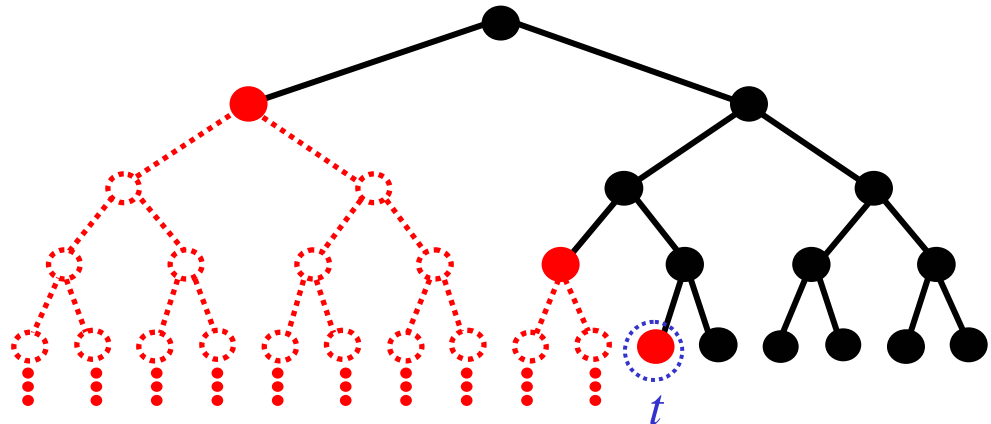
Trick 2: *Temporal slice* in WS2S

- First attempt : “*becomes an infinite set.*”

all1 s: (s in X \Leftrightarrow defined(s,t));

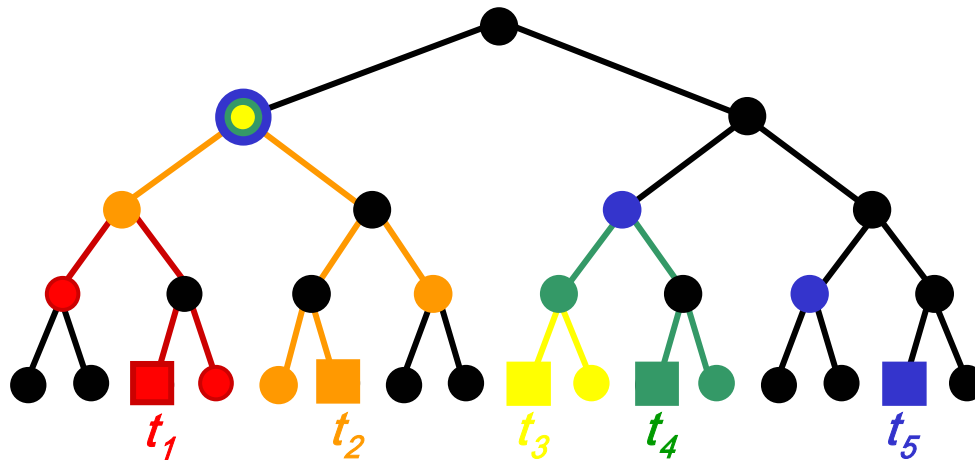


- Second attempt : “*temporal slice as its roots.*”



Second challenge: Sanity Check

- Hash values are computed from different $\text{LSR}_{t_{i+1}}(t_i)$'s. If multiple computations at each node coincide (i.e., consistent), *it suggests no internal-failures*.



Consistency

- Let (U_i, ℓ_i) such that
 - U_i : a set of incomparable nodes,
 - ℓ_i : labeling function on U_i
 - (extended $\ell_i(t) = \text{hash}(\ell_i(t.0), \ell_i(t.1))$ on $\text{Cls}(U_i)$)
- **Def.** $\{(U_i, \ell_i)\}$ is *weakly consistent* if

$$\ell_i(t) = \ell_j(t) \text{ for each } t \in \text{Cls}(U_i) \cap \text{Cls}(U_j)$$
- **Def.** $\{(U_i, \ell_i)\}$ is *consistent* if ℓ is well-defined for

$$\ell(t) = \begin{cases} \ell_i(t) & \text{when } t \in U_i \\ \text{hash}(\ell(t.0), \ell(t.1)) & \text{when } \neg t \in \text{leaves}(U_i) \end{cases}$$

Correctness of incremental sanity check

Cannot be described in WS2S

- **Th 2.** *If $\{ (\text{LSR}_{t_{i+1}}(t_i), \quad_i), (\text{LS}(t_{i+1}), \quad_{i+1}) \}$ is weakly consistent for each i with $1 \leq i < n$, $\{ (\text{LSR}_{t_{i+1}}(t_i), \quad_i) \mid 1 \leq i < n \} \cup \{ (\text{LS}(t_n), \quad_n) \}$ are consistent.*

Checked by large-scale experiment, but has not been proved !

- **Key Lemma.** Let $i+1 \leq k \leq j$. Then,
 $\text{Cls}(\text{LSR}_{t_{i+1}}(t_j)) \cap \text{Cls}(\text{LSR}_{t_{j+1}}(t_j)) \cap \text{Cls}(\text{LS}(t_k))$

Described in WS2S

$(\text{left}(s,t) \ \& \ (t = u \mid \text{left}(t,u)) \ \& \ (u = v \mid \text{left}(u,v)) \ \& \ (v = w \mid \text{left}(v,w)) \ \& \ \text{LSRclosure}(s,t,X) \ \& \ \text{LSclosure}(u,Y) \ \& \ \text{LSRclosure}(v,w,Z)) \Rightarrow X \text{ inter } Z \text{ sub } Y;$

Conclusion

- Case study of proving new properties of an event-ordering system developed by NTT, using MONA.
- Once clarified, they are not difficult; but when finding the *first* proofs (there are pitfalls), MONA assists very well.
- Found bug in “*where*”-sentence in WS2S mode of MONA:-)
- *Future work*: combine MONA & Isabelle/HOL, e.g., Th.2 (I have been saying this; but recent my focus is on Math...)

Thank you