RSA CONFERENCE Where The World Talks Security

San Francisco, Feb 27-Mar 2 2012

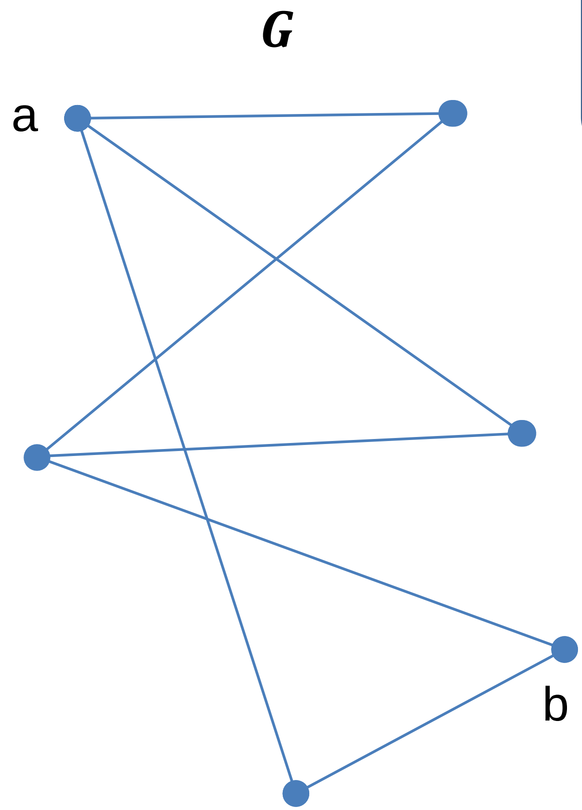
Short Transitive Signatures for Directed Trees

Philippe Camacho and Alejandro Hevia

University of Chile

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

How do we sign a graph?



Trivial solutions

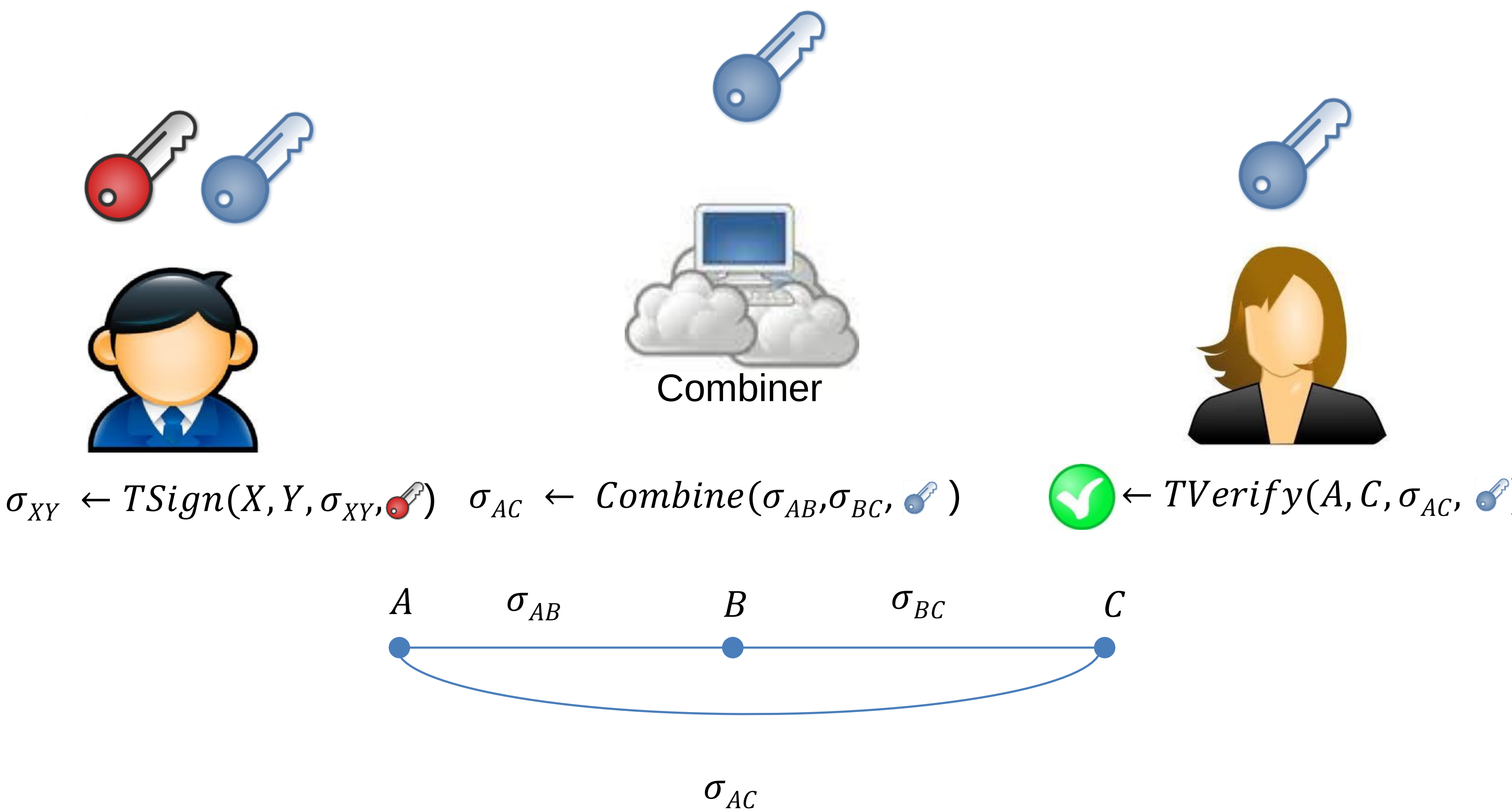
Let $\boldsymbol{n} = |\boldsymbol{G}|$, security parameter $\boldsymbol{\kappa}$

When adding a new node…

- Sign each edge
  - Time to sign: $\boldsymbol{O(1)}$
  - Size of signature: $\boldsymbol{O(n\kappa)}$ bits

- Sign each path
  - Time to sign (new paths): $\boldsymbol{O(n)}$
  - Size of signature: $\boldsymbol{O(\kappa)}$ bits

Transitive signature schemes [MR02,BN05,SMJ05]



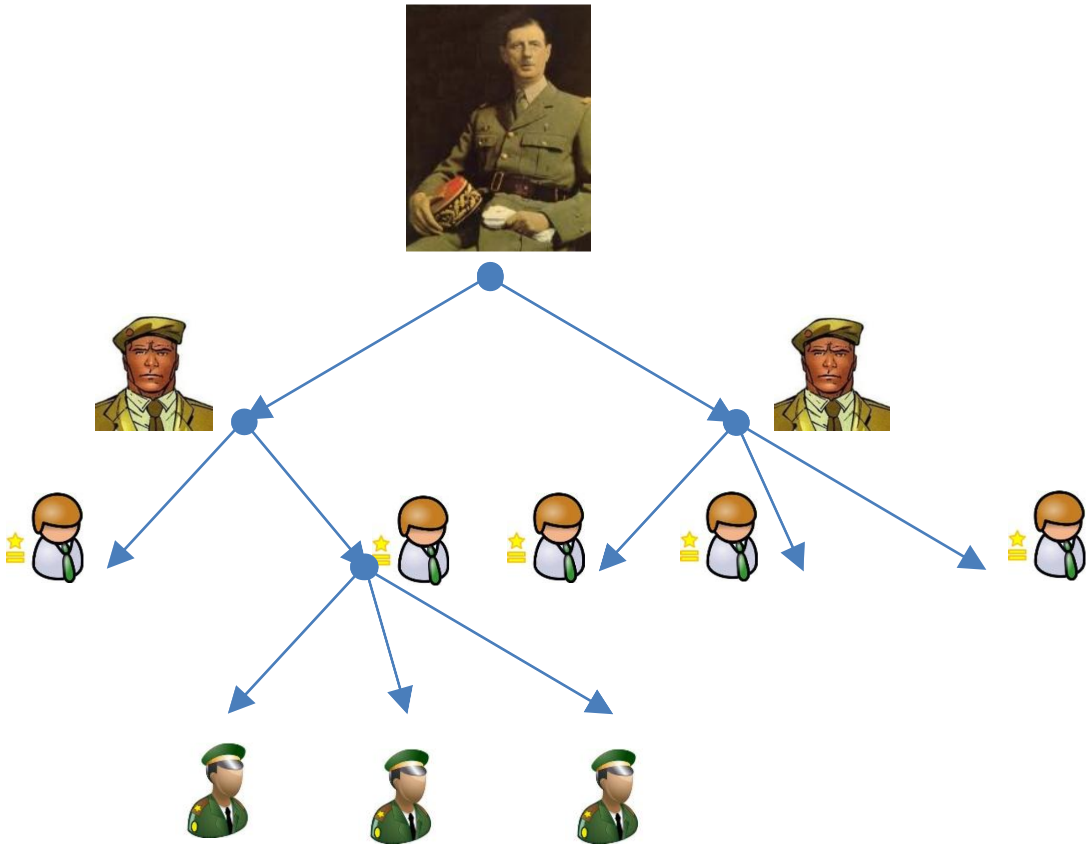
Landscape

- **[MR02,BN05,SMJ05]**
  for UNDIRECTED graphs

- Transitive Signatures for
  Directed Graphs (DTS) still OPEN

- **[Hoh03]**
  DTS $\Rightarrow$ Trapdoor Groups with
  Infeasible Inversion

Transitive Signatures for Directed Trees



Previous Work

- **[Yi07]**
  - Signature size: $\boldsymbol{O}(\boldsymbol{n} \log(\boldsymbol{n} \log \boldsymbol{n}))$ bits
    - Better than $\boldsymbol{O}(\boldsymbol{n\kappa})$ bits for the trivial solution
  - RSA related assumption

- **[Neven08]**
  - Signature size: $\boldsymbol{O}(\boldsymbol{n} \log \boldsymbol{n})$ bits
  - Standard Digital Signatures

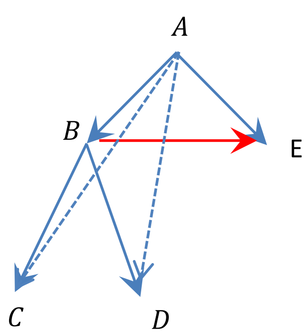
$\boldsymbol{O}(\boldsymbol{n} \log \boldsymbol{n})$ bits still impractical

Our Results

- For $\boldsymbol{\lambda} \geq \mathbf{1}$
  - Time to sign edge / verify path signature: $\boldsymbol{O}(\boldsymbol{\lambda})$
  - Time to compute a path signature: $\boldsymbol{O}(\boldsymbol{\lambda}(\boldsymbol{n}/\boldsymbol{\kappa})^{1/\lambda})$
  - Size of path signature: $\boldsymbol{O}(\boldsymbol{\lambda}\ \boldsymbol{\kappa})$ bits

Examples	$\lambda = 1$	$\lambda = 2$	$\lambda = \log(n)$
Time to sign edge / verify path signature	$\boldsymbol{O}(\mathbf{1})$	$\boldsymbol{O}(\mathbf{1})$	$\boldsymbol{O}(\log \boldsymbol{n})$
Time to compute a path signature	$\boldsymbol{O}(\boldsymbol{n}/\boldsymbol{\kappa})$	$\boldsymbol{O}(\sqrt{\boldsymbol{n}/\boldsymbol{\kappa}})$	$\boldsymbol{O}(\log \boldsymbol{n})$
Size of path signature	$\boldsymbol{O}(\boldsymbol{\kappa})$	$\boldsymbol{O}(\boldsymbol{\kappa})$	$\boldsymbol{O}(\boldsymbol{\kappa} \log \boldsymbol{n})$

Security [MR02]



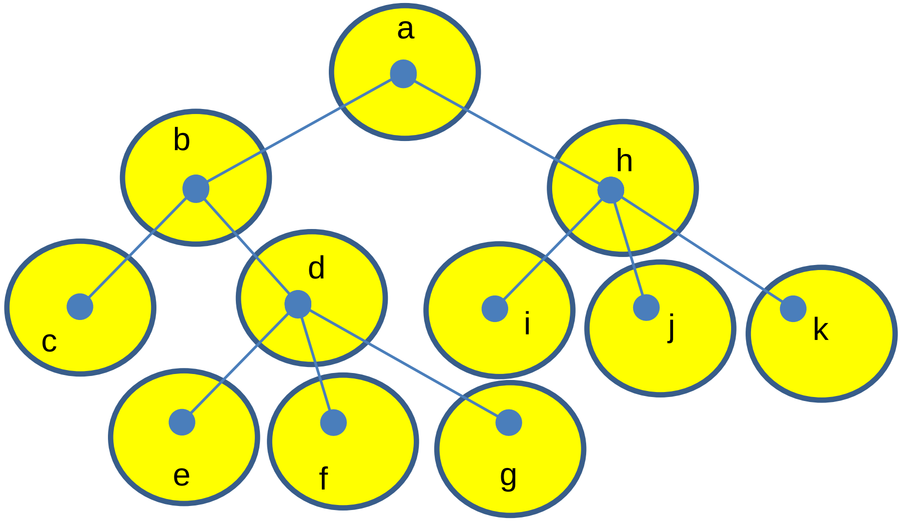
(σ^*, B, E):

$\leftarrow TVerify(B, E, \sigma^*, \ )$ and

There is **no path** from ***B*** to ***E***

BASIC CONSTRUCTION

Pre/Post Order Tree Traversal



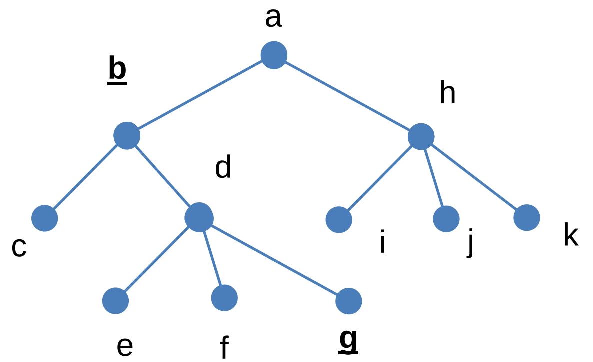
Pre order: a b c d e f g h i j k

Post order: c e f g d b i j k h a

Property of Pre/Post order Traversal

- **Proposition** [Dietz82]

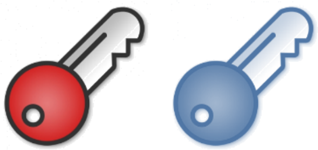
There is a path from $\boldsymbol{x}$ to $\boldsymbol{y}$ $\Longleftrightarrow$ $\boldsymbol{pos}(\boldsymbol{x}) < \boldsymbol{pos}(\boldsymbol{y})$ in $\boldsymbol{Pre}$, $\boldsymbol{pos}(\boldsymbol{y}) < \boldsymbol{pos}(\boldsymbol{x})$ in $\boldsymbol{Post}$



Pre order: a **b** c d e f **g** h i j k

Post order: c e f **g** d **b** i j k h a

Idea



Is there a path from a to e?

- Compute $\boldsymbol{pos(g)}$ in $\boldsymbol{Pre}$ and $\boldsymbol{Post}$
- Sign $\boldsymbol{g||7||4}$ and resign **values** that have changed

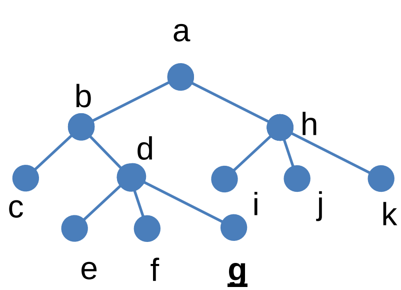
Signature of path $(\boldsymbol{a}, \boldsymbol{e})$:

- Signature of $\boldsymbol{a||1||11}$
- Signature of $\boldsymbol{e||5||2}$

- Check signatures
- Check

$$\mathbf{1} < \mathbf{5}$$
$$\mathbf{11} > \mathbf{2}$$

$\boldsymbol{G}$



Position	1	2	3	4	5	6	7	8	9	10	11
Pre	a	b	c	d	e	f	h	i	j	k	k
Post	c	e	f	d	b	i	j	k	h	a	a

How do we avoid recomputing a lot of signatures when an element is inserted?

Order Data Structure

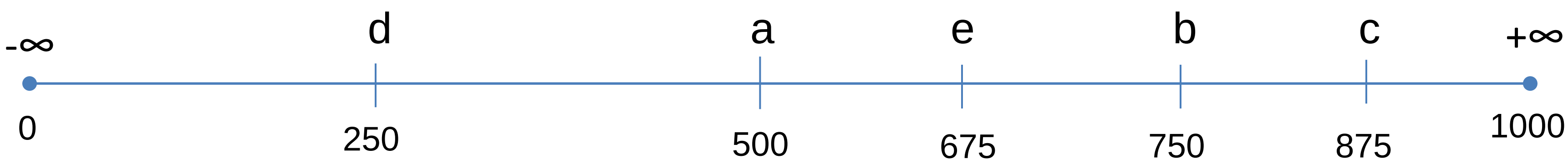
- Enables to
  - Insert elements **dynamically**
  - Compare them efficiently

- **Definition [Dietz82, MR+02]**
  - $ODInsert(X, Y)$
  - $ODCompare(X, Y)$

Trivial Order Data Structure

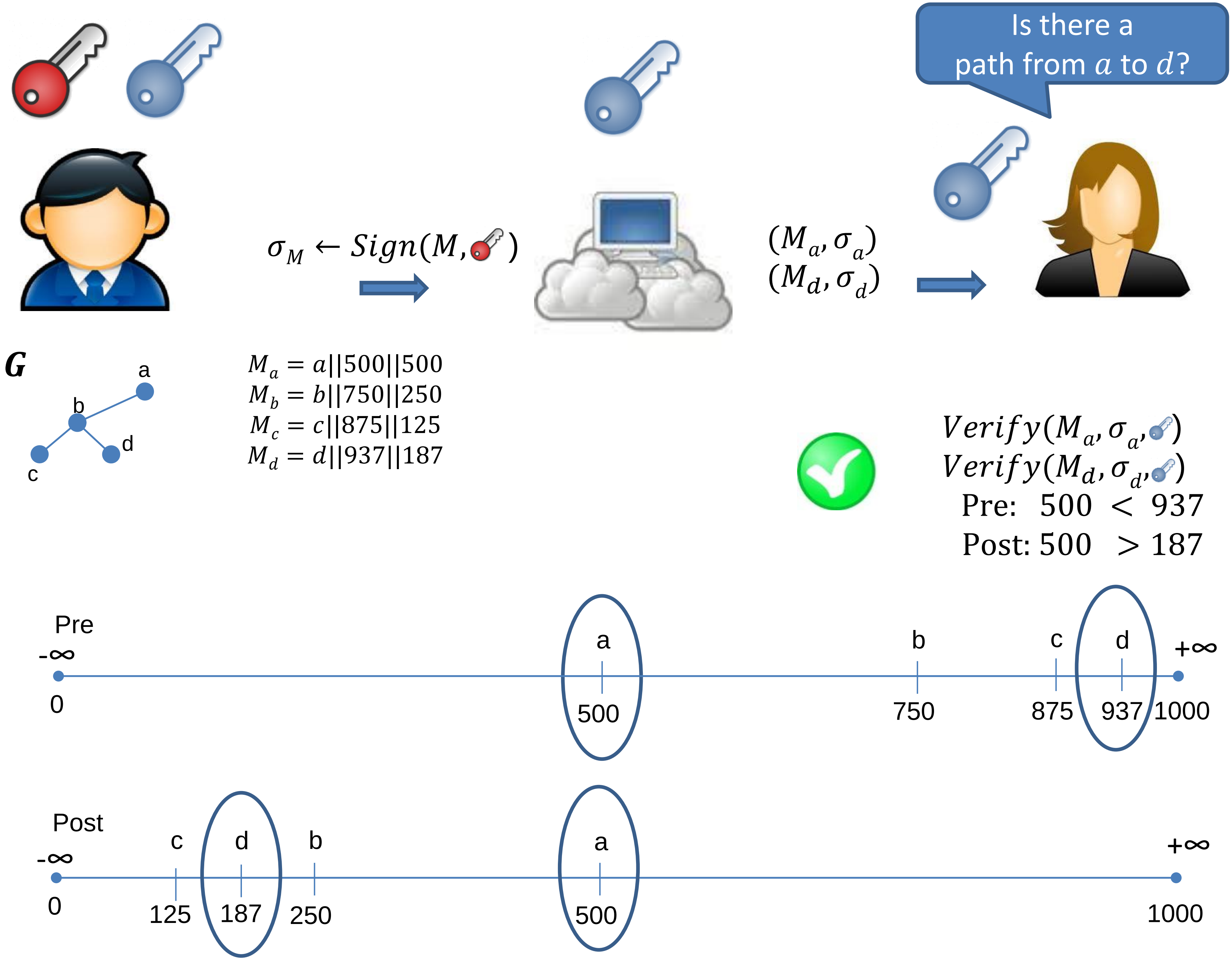
A Toy Example

Elements



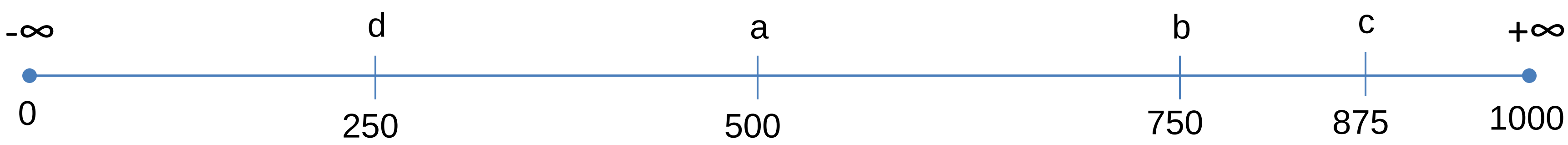
Labels

For n insertions we need to handle n bits



Is there a path from a to d?
$\sigma_M \leftarrow Sign(M, \cdot)$
(M_a, σ_a)
(M_d, σ_d)
G
a
b
c
d
$M_a = a||500||500$
$M_b = b||750||250$
$M_c = c||875||125$
$M_d = d||937||187$
$Verify(M_a, \sigma_a, \cdot)$
$Verify(M_d, \sigma_d, \cdot)$
Pre: 500 < 937
Post: 500 > 187
Pre
-∞
0
a
500
b
750
c
875
d
937
1000
+∞
Post
-∞
0
c
125
d
187
b
250
a
500
+∞
1000

Trivial Order Data Structure

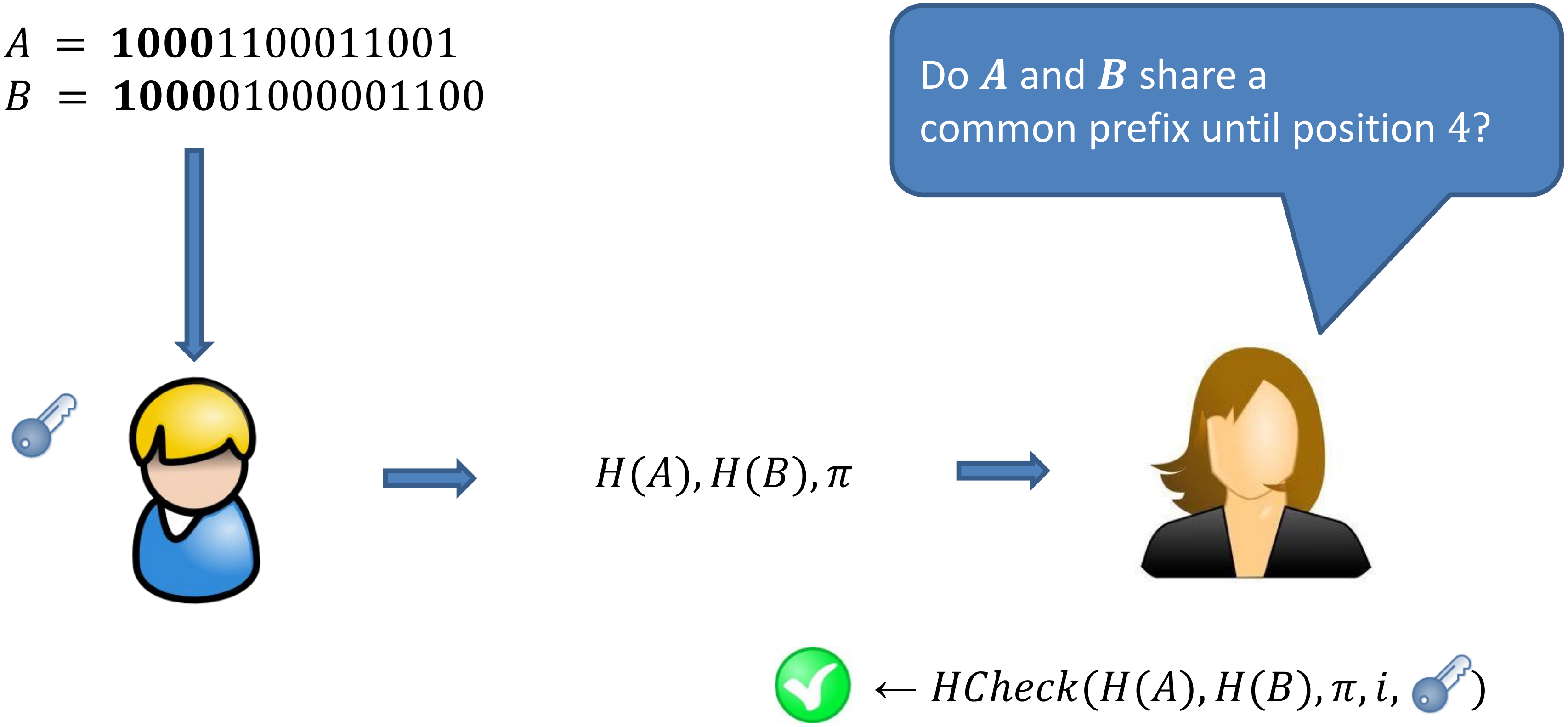


- Signature of size $O(n)$
- Better than $O(n \log n)$ **[Neven08]**, but still room for improvement.

New CRHF! It allows to:
- compress the strings
- efficiently compare them from their hashes

HASHING WITH COMMON PREFIX PROOFS

The Idea



We want:

$\boldsymbol{H}$ collision resistant hash function + proofs

Security

$\boldsymbol{HGen(1^{\kappa}, n) \rightarrow PK}$

$\boldsymbol{(A, B, i, \pi)}$

$$\boldsymbol{Adv(A) = \Pr\left[\begin{array}{c} HCheck(H(A), H(B), \pi, i, PK) = True \\ \wedge \\ A[1..i] \neq B[1..i] \end{array}\right]}$$

n-BDHI assumption [BB04]

$\boldsymbol{e}: \boldsymbol{G} \times \boldsymbol{G} \rightarrow \boldsymbol{G_T}$
$\boldsymbol{s} \leftarrow \boldsymbol{Z_p}$
$\boldsymbol{g}$ generator of $\boldsymbol{G}$
$(\boldsymbol{g^s}, \boldsymbol{g^{s^2}}, \ldots, \boldsymbol{g^{s^n}})$

$\boldsymbol{e(g, g)^{1/s}}$

The hash function

- $\boldsymbol{HGen(1^{\kappa}, n)}$

  $(\boldsymbol{p}, \boldsymbol{G}, \boldsymbol{G_T}, \boldsymbol{e}, \boldsymbol{g}) \leftarrow BMGen(1^{\kappa})$

  $\boldsymbol{s} \leftarrow \boldsymbol{Z_p}$
  $\boldsymbol{T} := (\boldsymbol{g^{s}}, \boldsymbol{g^{s^2}}, \ldots, \boldsymbol{g^{s^n}})$

  return $\boldsymbol{PK} := (\boldsymbol{p}, \boldsymbol{G}, \boldsymbol{G_T}, \boldsymbol{e}, \boldsymbol{g}, \boldsymbol{T})$

- $\boldsymbol{HEval(M, PK)}$

$$\boldsymbol{H(M)} := \prod_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{g^{M[i]s^i}}$$

Toy example**:** $\boldsymbol{M = 1001 \Rightarrow H(M) = g^{s} . g^{s^4}}$

Generating & Verifying Proofs

- $A = A[1..n] = 1000111001$
- $B = B[1..n] = 1000101100$

- $\Delta := \frac{H(A)}{H(B)} = \frac{g^{s} g^{s^5} g^{s^6} g^{s^7} g^{s^{10}}}{g^{s} g^{s^5} g^{s^7} g^{s^8}} = g^{s^6}\, g^{-s^8}\, g^{s^{10}}$

- $\Delta = \prod_{j=1}^{n} g^{C[j] s^j}$ with $C = [0, 0, 0, 0, 0, 1, 0, -1, 0, 1]$

Generating & Verifying Proofs

- $\boldsymbol{\Delta} = \prod_{j=1}^{n} g^{C[j]s^j}$ with $\boldsymbol{C} = [0, 0, 0, 0, 0, 1, 0, -1, 0, 1]$

- "Remove" factor $\mathbf{s}^{\mathbf{i+1}}$ in the exponent **without knowing** $\mathbf{s}$

$$\boldsymbol{\pi} := \Delta^{\frac{1}{s^{i+1}}} = \prod_{j=i+1}^{n} g^{C[j]s^{j-i-1}} = g\, g^{-s^2} g^{s^4}$$

- Check the proof : $e(\pi, g^{s^{i+1}}) = e(\Delta, g)$

Security

- **Proposition:**
  If the n-BDHI assumption holds then the previous construction is a secure HCPP family.

- Proof (idea)

$$\mathbf{A} = 100010$$
$$\mathbf{B} = 101001$$
$$\mathbf{i} = 3$$

$$\mathrm{H}(\mathrm{A}) = \mathrm{g}^{s}\,\mathrm{g}^{s^5}$$
$$\mathrm{H}(\mathrm{B}) = \mathrm{g}^{s}\,\mathrm{g}^{s^3}\,\mathrm{g}^{s^6}$$
$$\Delta = \frac{\mathrm{H}(\mathrm{A})}{\mathrm{H}(\mathrm{B})} = \mathrm{g}^{-s^3}\,\mathrm{g}^{s^5}\,g^{-s^6}$$
$$\pi = \Delta^{\frac{1}{s^4}} = \mathrm{g}^{-1/s}\,\mathrm{g}^{s}\,\mathrm{g}^{s^2}$$

CRHF is incremental

$A = \mathbf{1000}$

$B = \mathbf{1000}1$

$\boldsymbol{H}(\boldsymbol{B}) = \boldsymbol{H}(\boldsymbol{A})\, \boldsymbol{g}^{\boldsymbol{s}^{\mathbf{5}}}$

It’s fast to compute $\boldsymbol{H}(\boldsymbol{B})$ from $\boldsymbol{H}(\boldsymbol{A})$
(we don’t need the preimage $\boldsymbol{A}$)

Comparing strings

- $\boldsymbol{A < B \Leftrightarrow CommonPrefix(A, B, i) \wedge A[i+1] < B[i+1]}$

E.g: $A = \mathbf{100}01$, $B = \mathbf{100}10$ $\Big\}$ $C = 100$

- Check:

$e(H(A)/H(C), g) = e(\pi_1, g^{s^4})$ // C is a prefix of A

$e(H(B)/H(C), g) = e(\pi_2, g^{s^4})$ // C is a prefix of B

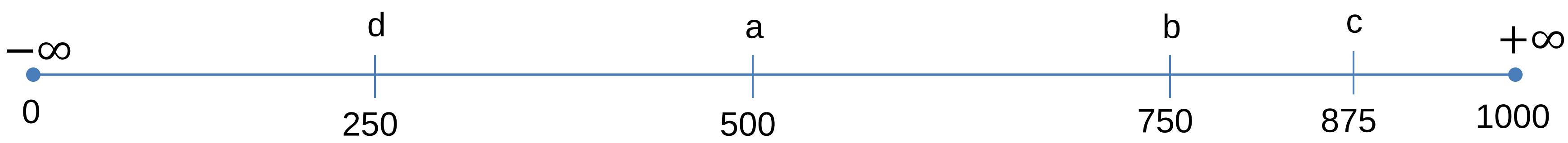
$e(H(C)H(0^3||0)/H(A), g) = e(\pi_3, g^{s^5})$ // $C||0$ is a prefix of A

$e(H(C)H(0^3||1)/H(B), g) = e(\pi_4, g^{s^5})$ // $C||1$ is a prefix of B

$0 < 1$

FULL CONSTRUCTION

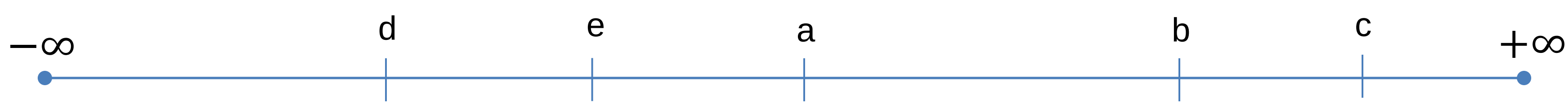
Trivial Order Data Structure



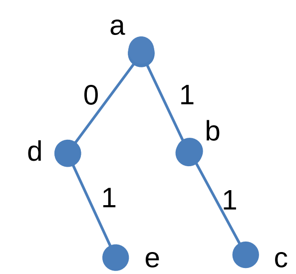
Signer has to compute new labels before hashing them
$\Rightarrow$ Time to sign an edge still $\boldsymbol{O(n)}$.

New Order Data Structure:
$\boldsymbol{ODInsert(X, Y)}$ s.t. new label $\boldsymbol{Z}$
shares every bit except one with $\boldsymbol{X}$ or $\boldsymbol{Y}$

New Order Data Structure

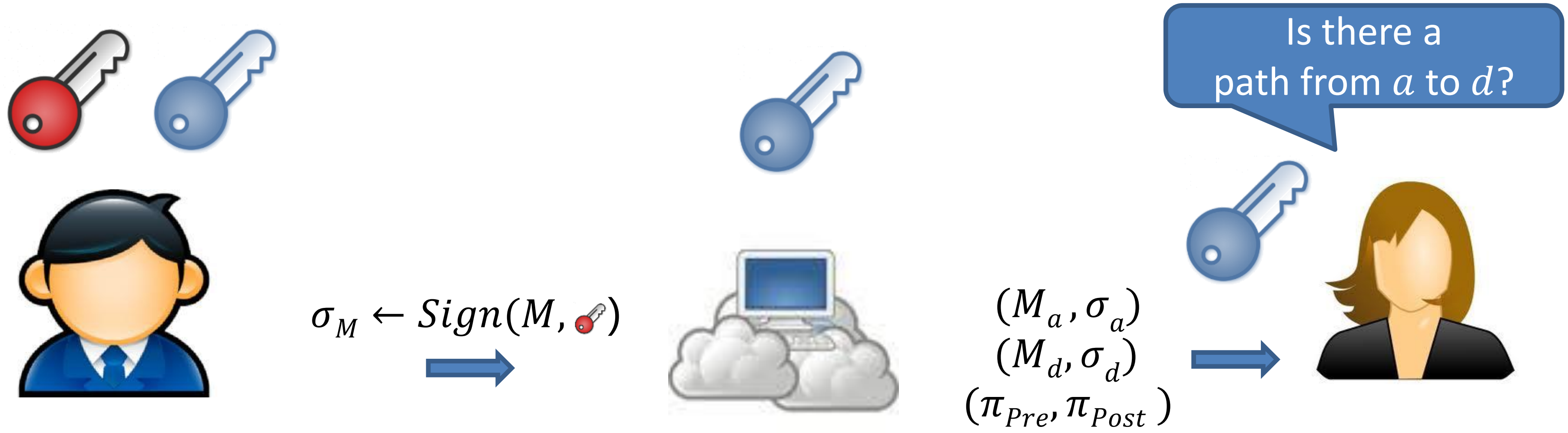


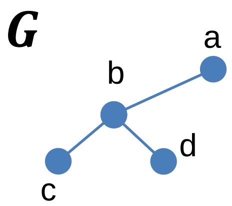
Use a binary tree to obtain
an «incremental» order data structure



$$L(a) = \varepsilon$$
$$L(b) = 1$$
$$L(c) = 11$$
$$L(d) = 0$$
$$L(e) = 01$$

$$0 < \$ < 1$$
$$L(d) = 0\$ < L(a) = \varepsilon\$$$
$$L(d) = 0\$ < L(b) = 1\$$$
$$L(b) = 1\$ < L(c) = 11\$$$
$$L(e) = 01\$ < L(a) = \varepsilon\$$$





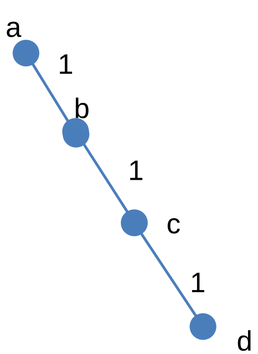
$$M_a = a||H(\varepsilon)||H(\varepsilon)$$
$$M_b = b||H(1)||H(0)$$
$$M_c = c||H(11)||H(00)$$
$$M_d = d||H(111)||H(001)$$

- $Verify(M_a, \sigma_a,)$
- $Verify(M_d, \sigma_d,)$

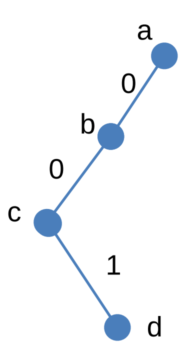
- Use HCheck with π_{Pre} and π_{Post}

$$LPre(a) = \varepsilon\$ < 111\$ \doteq LPre(d)$$
$$LPost(d) = 001\$ < \varepsilon\$ = LPost(a)$$

ODPre

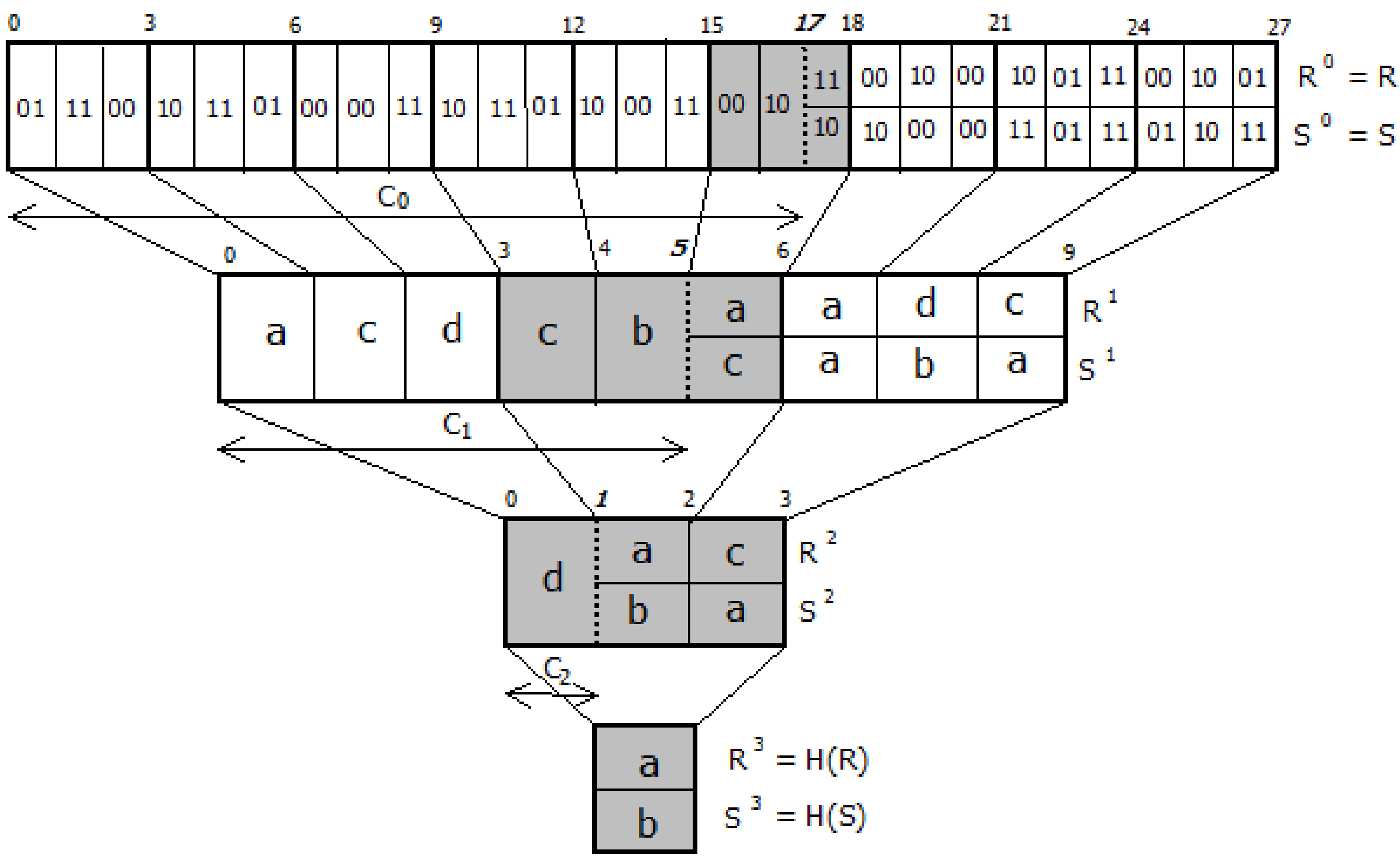


ODPost



Trade off

$$n = 54, \quad \kappa = 2, \quad \Sigma = \{a, b, c, d\}$$
$$n/\kappa = 54/2 = 27$$
$$\lambda = 3 \Rightarrow (n/\kappa)^{1/\lambda} = 3$$



Conclusion and Open Problems

- Efficient transitive signature scheme for directed trees
- Possible to balance the time to compute and to verify the proof
- Based on a general new primitive HCPP
- New constructions / applications for HCPP
- Can we improve the trade off?
- **Stateless** transitive signatures for directed trees

Thank you!