

# Strong Accumulators from Collision-Resistant Hashing

*ISC 2008  
Taipei - Taiwan*

Philippe Camacho (*University of Chile*)

Alejandro Hevia (*University of Chile*)

Marcos Kiwi (*University of Chile*)

Roberto Opazo (*CEO Acepta.com*)



# Outline

- Notion of accumulator
- Motivation
  - e-Invoice Factoring
- Our construction
- Conclusion

# Notion of accumulator

## ■ Problem

- A set  $X$ .
- Given an element  $x$  we wish to prove that this element belongs or not to  $X$ .

## ■ Let $X = \{x_1, x_2, \dots, x_n\}$ :

- $X$  will be represented by a short value  $Acc$ .
- $Belongs(Acc, x, w) = True \Leftrightarrow x$  belongs to  $X$ .

↑  
Witness



# Notion of accumulator

## ■ Accumulator Manager

- Computes setup values.
- Computes the accumulated value  $Acc$ .
- Computes the witness  $w_x$  for a given  $x$ .

## ■ Accumulator Users

- Check that an element belongs or not to the set, using  $Acc$ ,  $w_x$  and  $x$ .



# Applications

- Time-stamping [BeMa94]
- Certificate Revocation List [LLX07]
- Anonymous credentials [CamLys02]
- E-Cash [AWSM07]
- Broadcast Encryption [GeRa04]
- ...

Nothing to see with  
Number Theory!

# Factoring Industry in Chile

**Factoring  
Entity**

**Provider**  
(Milk seller)

**Client**  
(Supermarket)

Nothing to see with  
Number Theory!

# Factoring Industry in Chile

Factoring  
Entity

**Provider**  
(Milk seller)

1) I want (a lot of) milk now \*.

**Client**  
(Supermarket)

(\*) but I do not want to pay yet.

Nothing to see with  
Number Theory!

# Factoring Industry in Chile

Factoring  
Entity

**Provider**  
(Milk seller)

1) I want (a lot of) milk now \*.

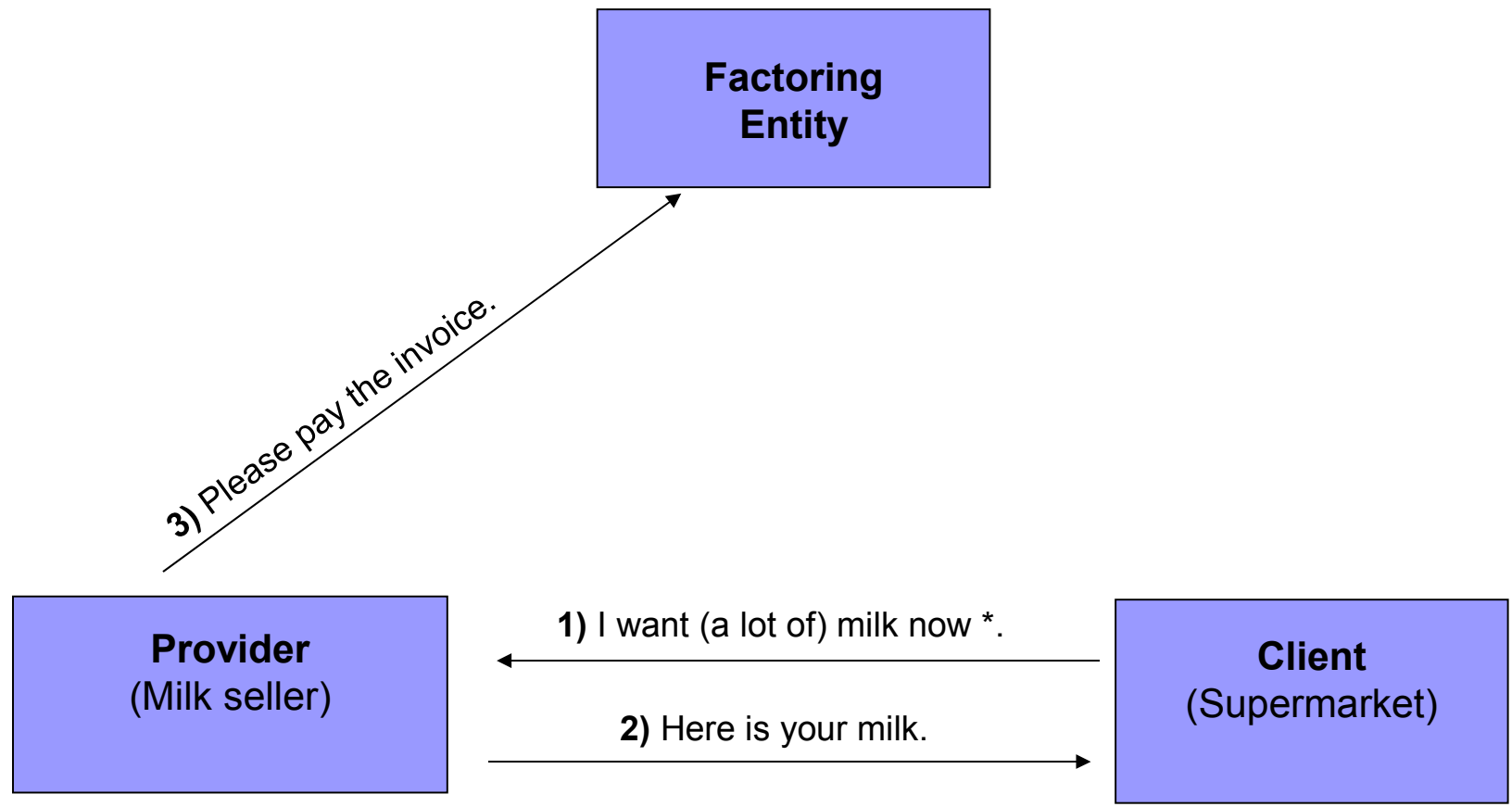
2) Here is your milk.

**Client**  
(Supermarket)

(\* ) but I do not want to pay yet.

Nothing to see with  
Number Theory!

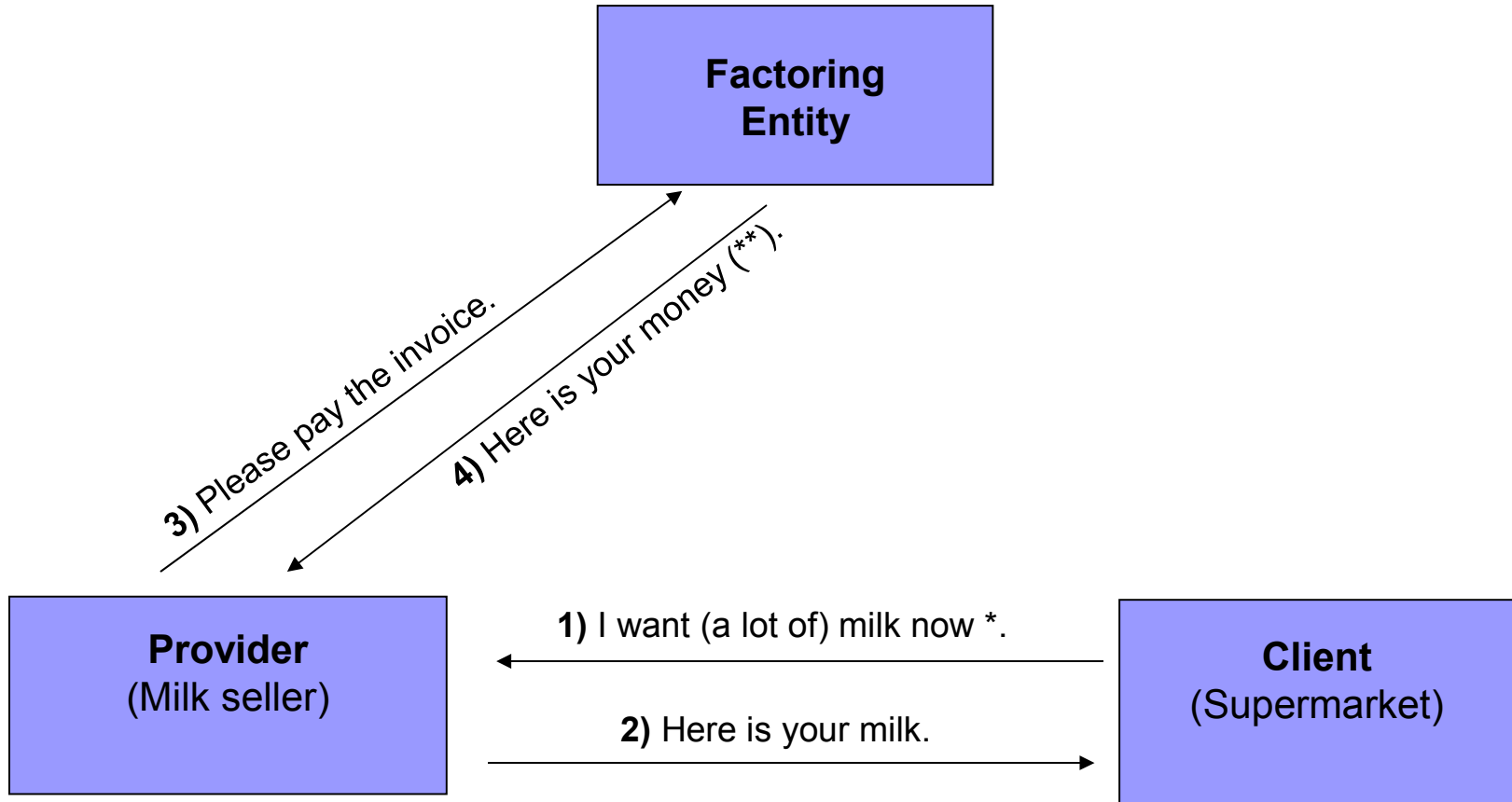
# Factoring Industry in Chile



(\* ) but I do not want to pay yet.

Nothing to see with  
Number Theory!

# Factoring Industry in Chile

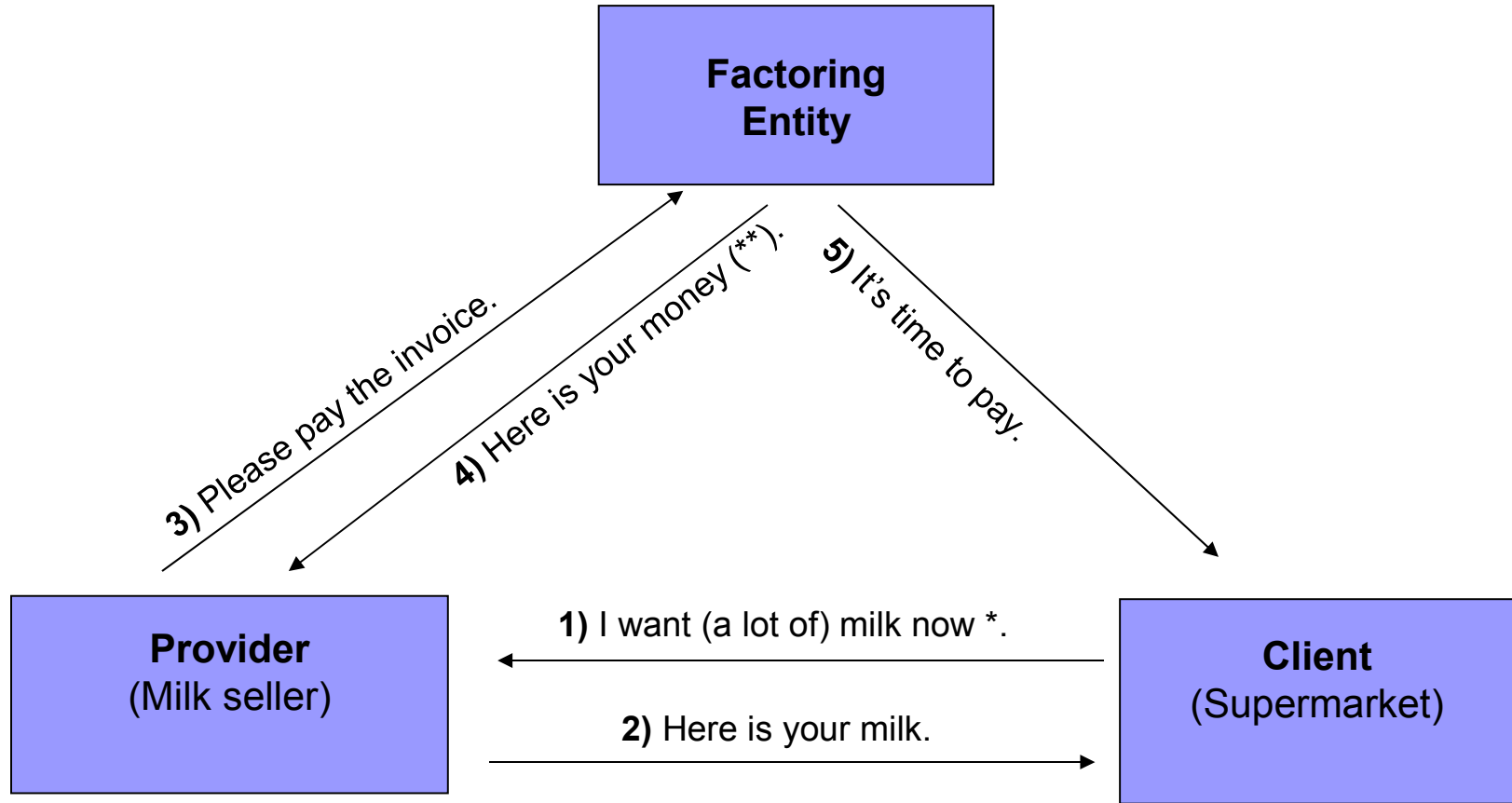


(\* ) but I do not want to pay yet.

(\*\* ) minus a fee.

Nothing to see with  
Number Theory!

# Factoring Industry in Chile

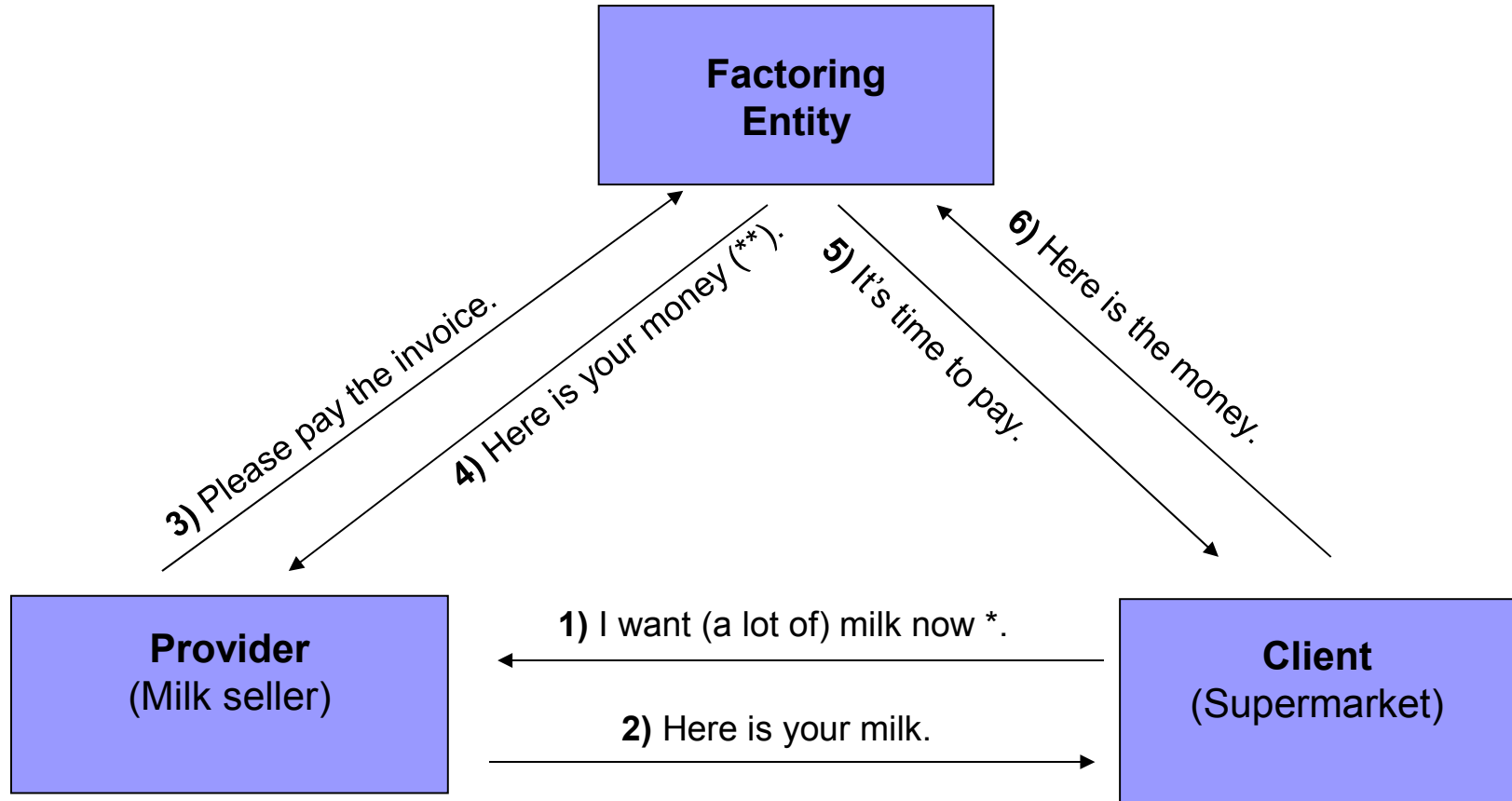


(\* ) but I do not want to pay yet.

(\*\* ) minus a fee.

Nothing to see with  
Number Theory!

# Factoring Industry in Chile



(\* ) but I do not want to pay yet.

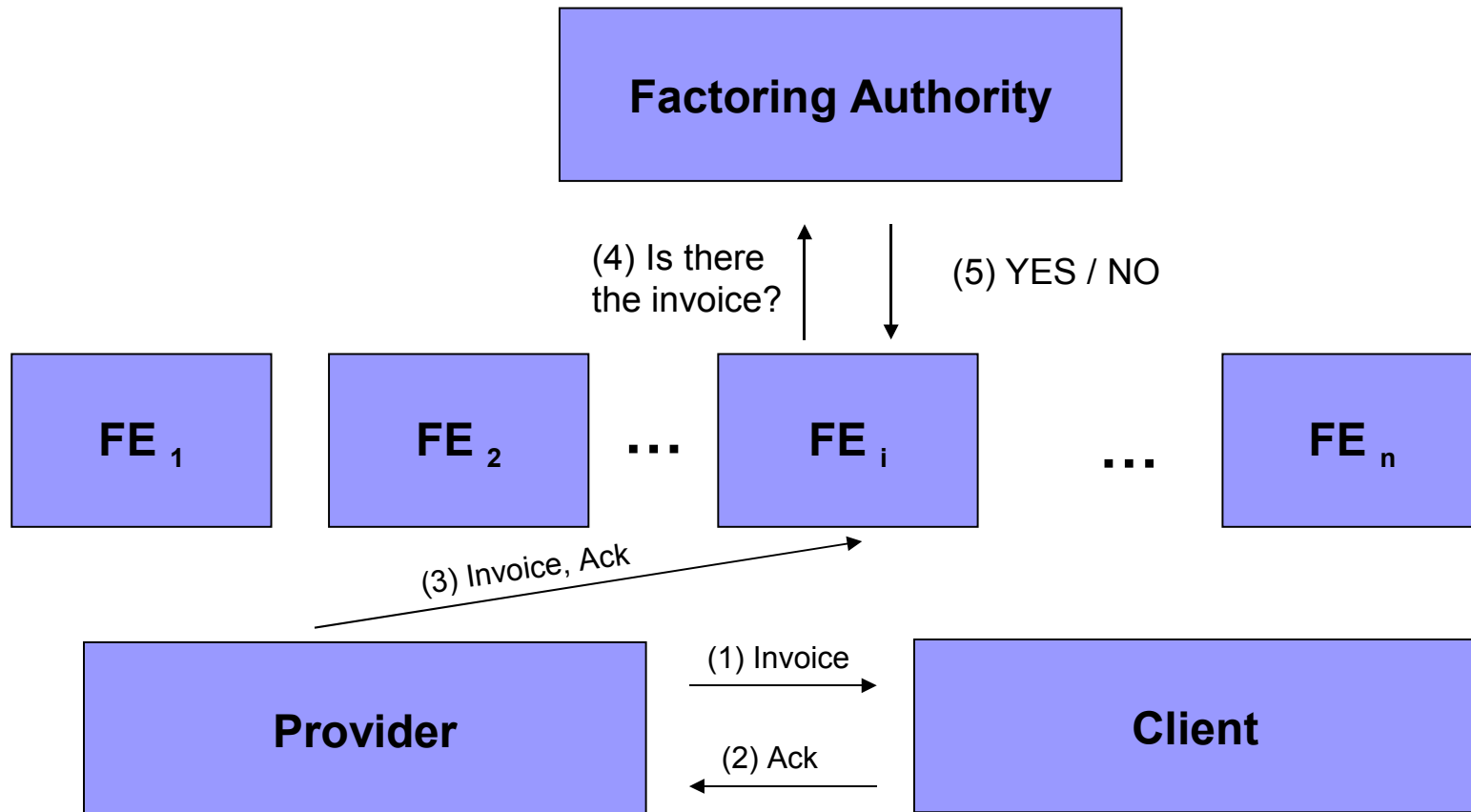
(\*\* ) minus a fee.

# The Problem

- A malicious provider could send the same invoice to various Factoring Entities.
- Then he leaves to a far away country with all the money.
- Later, several Factoring Entities will try to charge the invoice to the same client. Losses must be shared...



# Solution with Factoring Authority





# Caveat

- This solution is quite simple.
- **However**
  - Trusted Factoring Authority is needed.
- Can we remove this requirement?



# Properties

- **Dynamic**

- Allows insertion/deletion of elements.



















- **Universal**

- Allows proofs of membership and nonmembership.

- **Strong**

- No need to trust in the Accumulator Manager.

# Prior work

	Dynamic	Strong	Universal	Security	Efficiency (witness size)	Note
[BeMa94]				RSA + RO	O(1)	First definition
[BarPfi97]				Strong RSA	O(1)	-
[CamLys02]				Strong RSA	O(1)	First dynamic accumulator
[LLX07]				Strong RSA	O(1)	First universal accumulator
[AWSM07]				Pairings	O(1)	E-cash
[WWP08]				eStrong RSA Paillier	O(1)	Batch Update

# Prior work

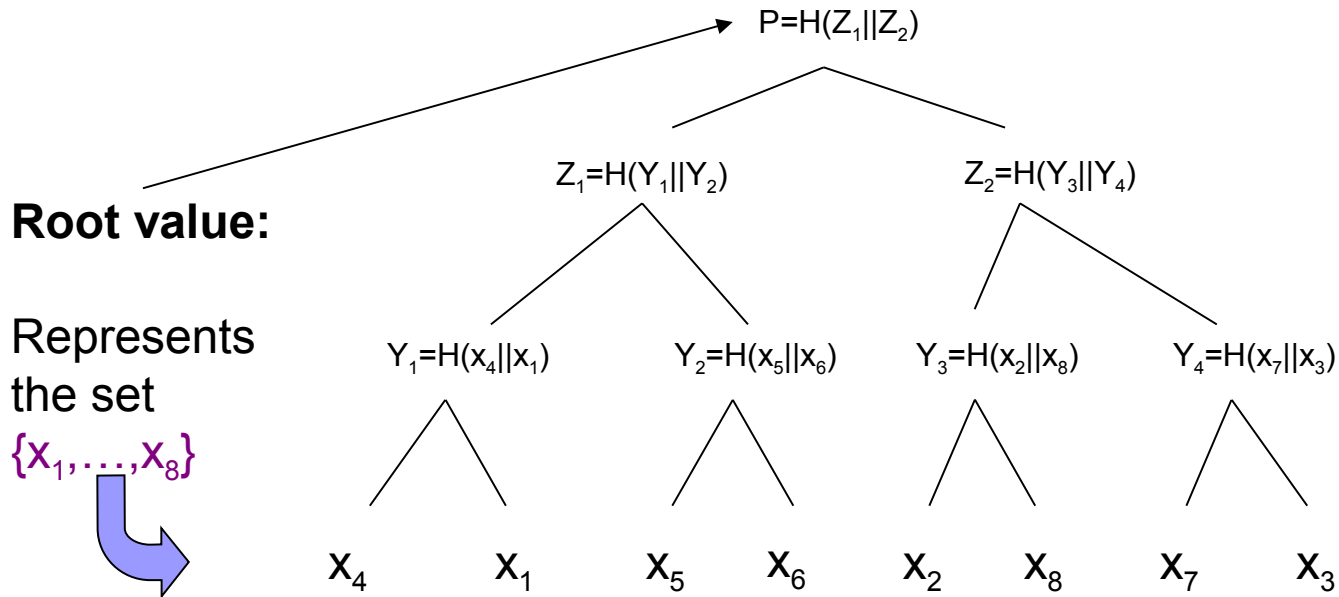
	Dynamic	Strong	Universal	Security	Efficiency (witness size)	Note
[BeMa94]				RSA + RO	O(1)	First definition
[BarPfi97]				Strong RSA	O(1)	-
[CamLys02]				Strong RSA	O(1)	First dynamic accumulator
[LLX07]				Strong RSA	O(1)	First universal accumulator
[AWSM07]				Pairings	O(1)	E-cash
[WWP08]				eStrong RSA Paillier	O(1)	Batch Update
[CHKO08]				Collision-Resistant Hashing	O(ln(n))	<b>Our work</b>

# Notation

- $H: \{0,1\}^* \rightarrow \{0,1\}^k$ 
  - randomly chosen function from a family of collision-resistant hash functions.
- $x_1, x_2, x_3, \dots \in \{0,1\}^k$ 
  - $x_1 < x_2 < x_3 < \dots$  where  $<$  is the lexicographic order on binary strings.
- $-\infty, \infty$ 
  - Special values such that
    - For all  $x \in \{0,1\}^k$ :  $-\infty < x < \infty$
- $\|$  denotes the concatenation operator.

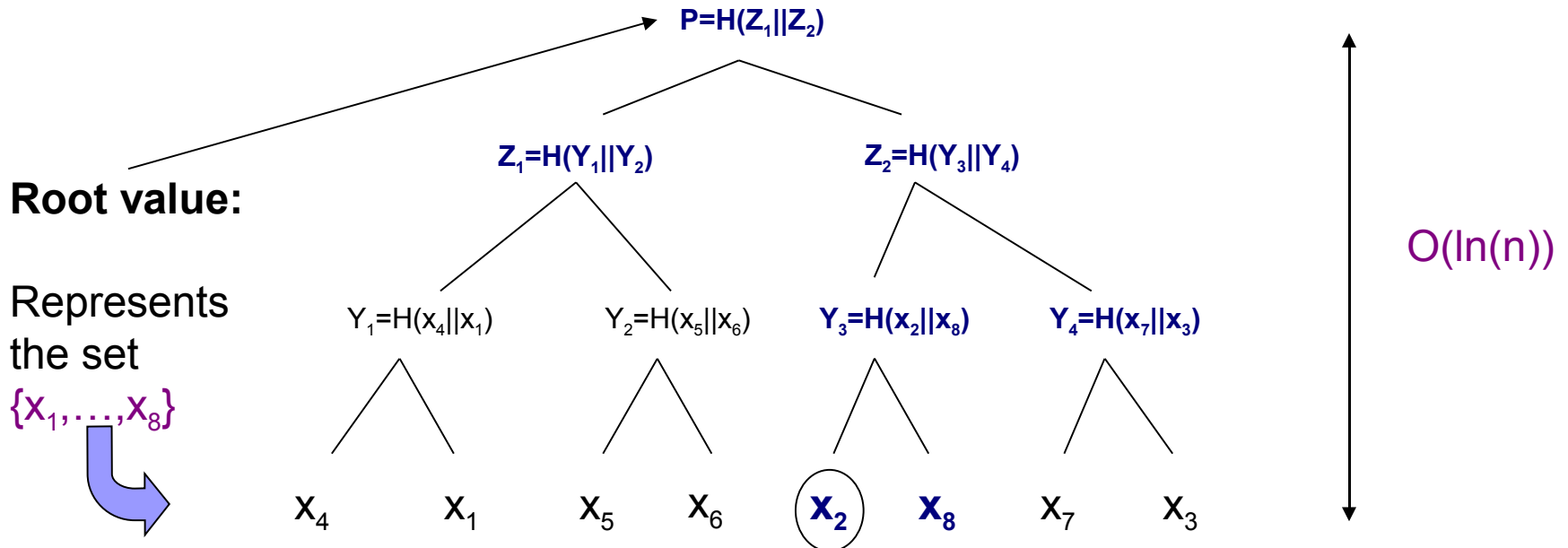
# Ideas

## ■ Merkle-trees



# Ideas

## ■ Merkle-trees



# Ideas

- How to prove non-membership?
  - Kocher's trick [Koch98]: store pair of consecutive values
    - $X = \{1, 3, 5, 6, 11\}$
    - $X' = \{(-\infty, 1), (1, 3), (3, 5), (5, 6), (6, 11), (11, \infty)\}$
    - $y=3$  belongs to  $X \Leftrightarrow (1, 3)$  or  $(-\infty, 1)$  belongs to  $X'$ .
    - $y=2$  does not belong to  $X \Leftrightarrow (1, 3)$  belongs to  $X'$ .



# Public Data Structure

- Called “Memory”.
- Compute efficiently the accumulated value and the witnesses.
- In our construction the Memory will be a binary tree.

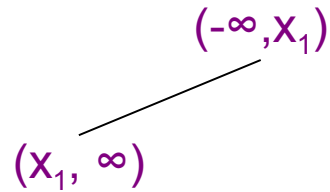


# How to insert elements?

$(-\infty, \infty)$

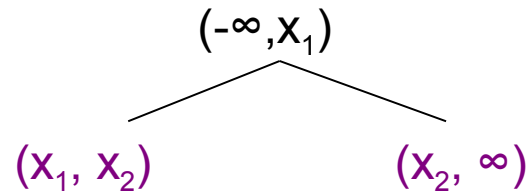
$X = \emptyset$ , next:  $x_1$

# How to insert elements?



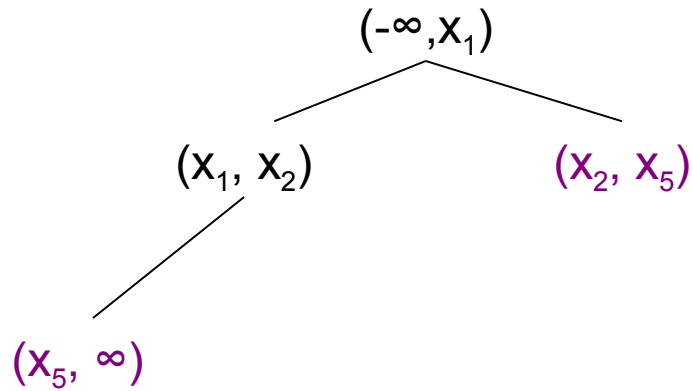
$X = \{x_1\}$ , next:  $x_2$

# How to insert elements?



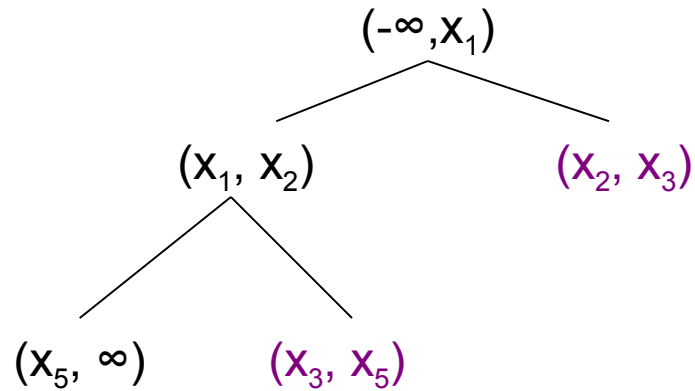
$X = \{x_1, x_2\}$ , next:  $x_5$

# How to insert elements?



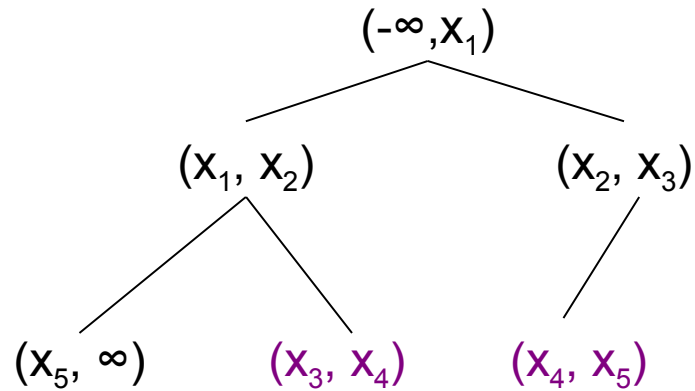
$X = \{x_1, x_2, x_5\}$ , next:  $x_3$

# How to insert elements?



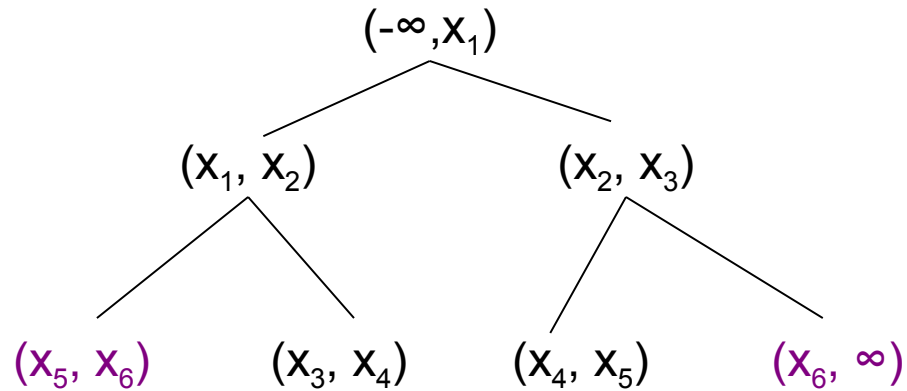
$X = \{x_1, x_2, x_3, x_5\}$ , next:  $x_4$

# How to insert elements?



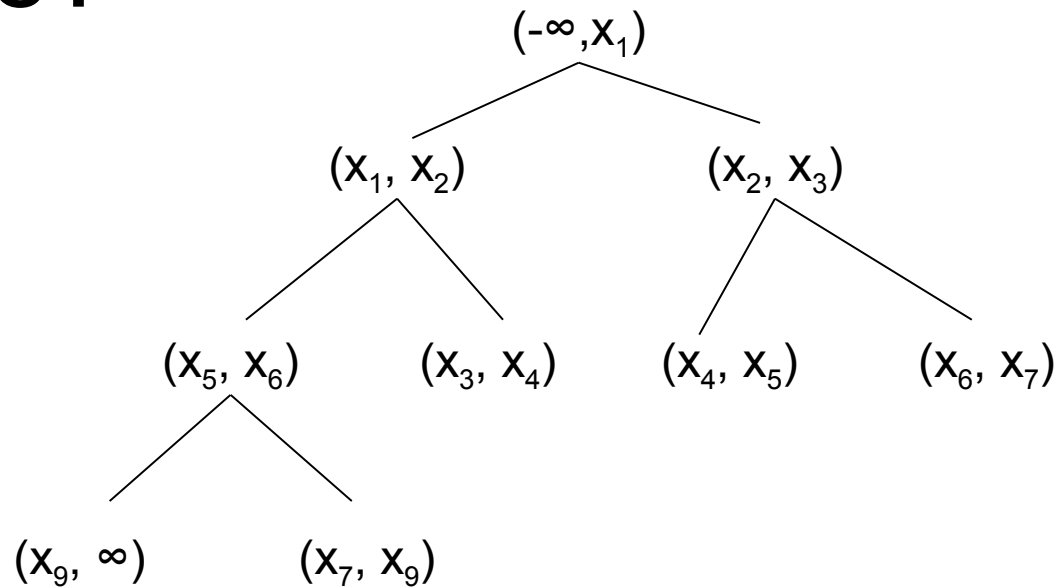
$X = \{x_1, x_2, x_3, x_4, x_5\}$ , next:  $x_6$

# How to insert elements?



$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

# How to compute the accumulated value?



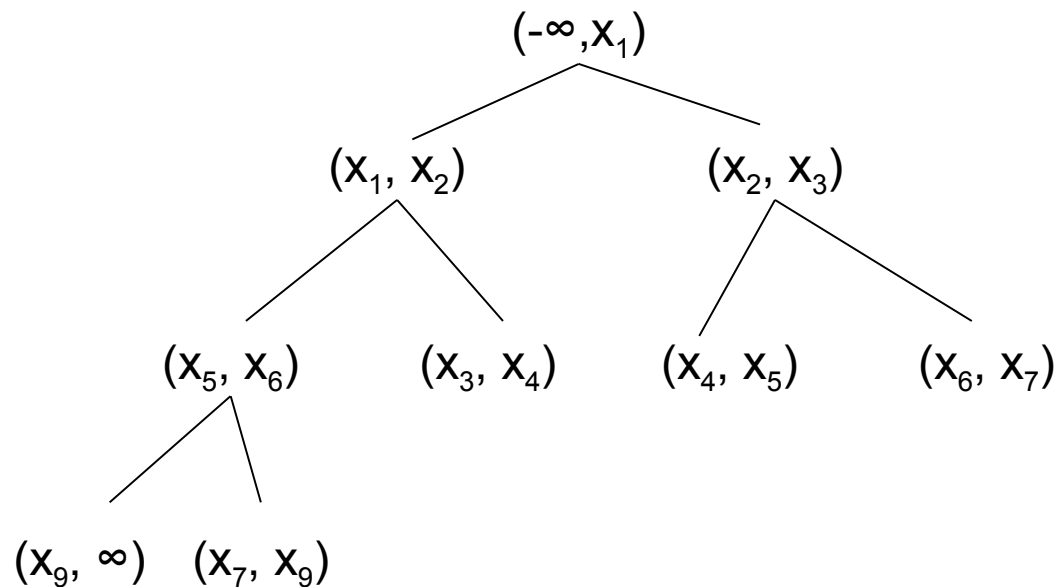
$$\text{Proof}_N = H(\text{Proof}_{\text{left}} || \text{Proof}_{\text{right}} || \text{value})$$

$$\text{Proof}_{\text{Nil}} = ""$$

$$\text{Acc} = \text{Proof}_{\text{Root}}$$

A pair  $(x_i, x_j)$

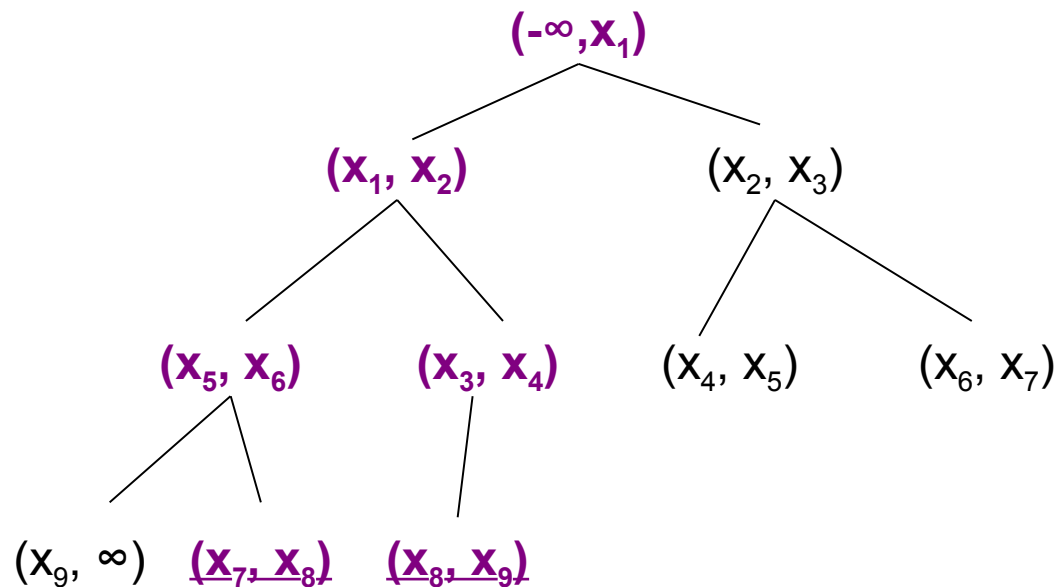
# How to update the accumulated value? (Insertion)



Next element to be inserted:  $x_8$

We will need to recompute proof node values.

# How to update the accumulated value? (Insertion)



New element:  $x_8$ .

$\text{Proof}_N$  stored in each node.

Dark nodes do not require recomputing  $\text{Proof}_N$ .

**Only a logarithmic number of values needs recomputation.**

# Security

## ■ Consistency

□ Difficult to find witnesses that allow to prove inconsistent statements.

■  $X=\{1,2\}$

■ Hard to compute a membership witness for 3.

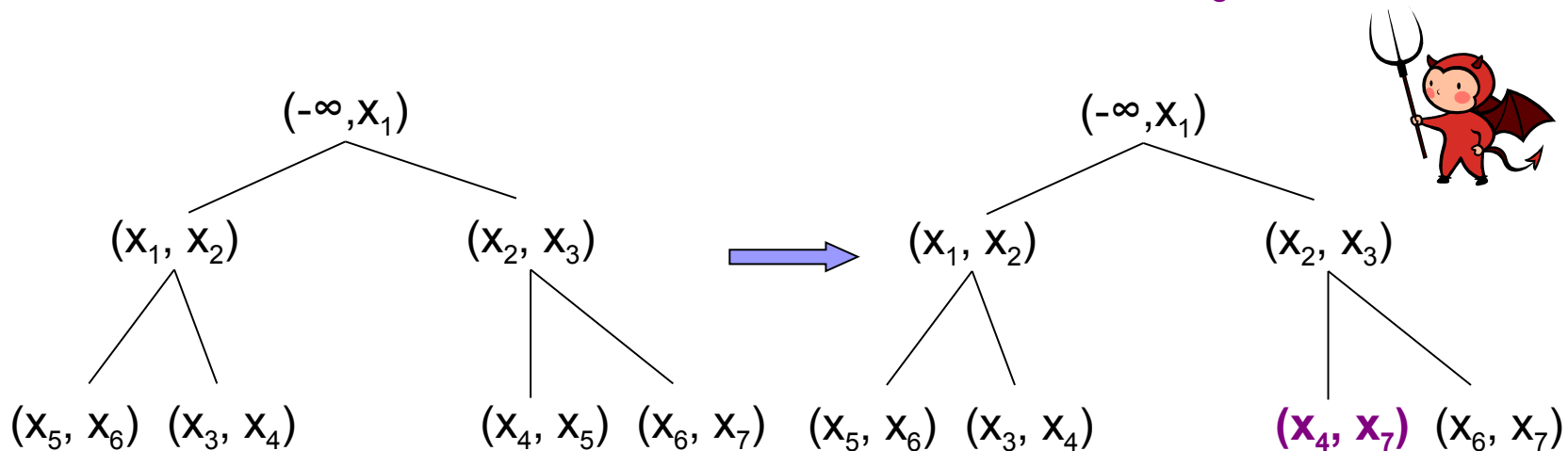
■ Hard to compute a nonmembership witness for 2.

## ■ Update

□ Guarantees that the accumulated value represents the set after insertion/deletion of  $x$ .

# Security

- **Lemma:** Given a tree  $T$  with accumulated value  $\text{Acc}_T$ , finding a tree  $T'$ ,  $T \neq T'$  such that  $\text{Acc}_T = \text{Acc}_{T'}$ , is difficult.
- *Proof (Sketch):*  $\text{Proof}_N = H(\text{Proof}_{\text{left}} \parallel \text{Proof}_{\text{right}} \parallel \text{value})$

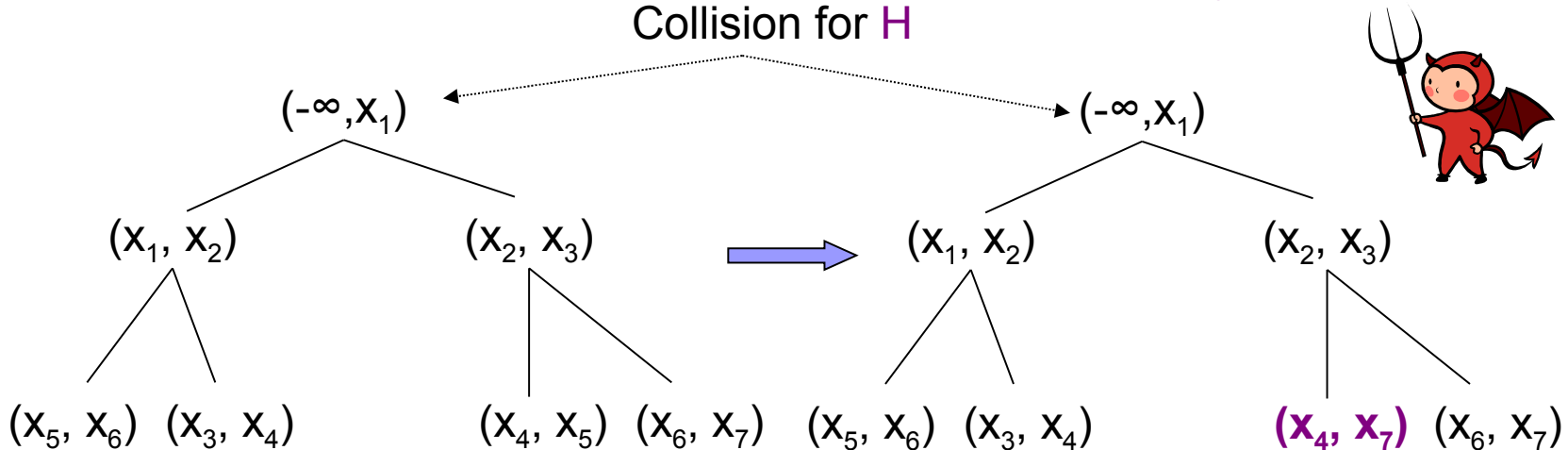


# Security

- **Lemma:** Given a tree  $T$  with accumulated value  $\text{Acc}_T$ , finding a tree  $T'$ ,  $T \neq T'$  such that  $\text{Acc}_T = \text{Acc}_{T'}$ , is difficult.

- *Proof (Sketch):*  $\text{Proof}_N = H(\text{Proof}_{\text{left}} \parallel \text{Proof}_{\text{right}} \parallel \text{value})$

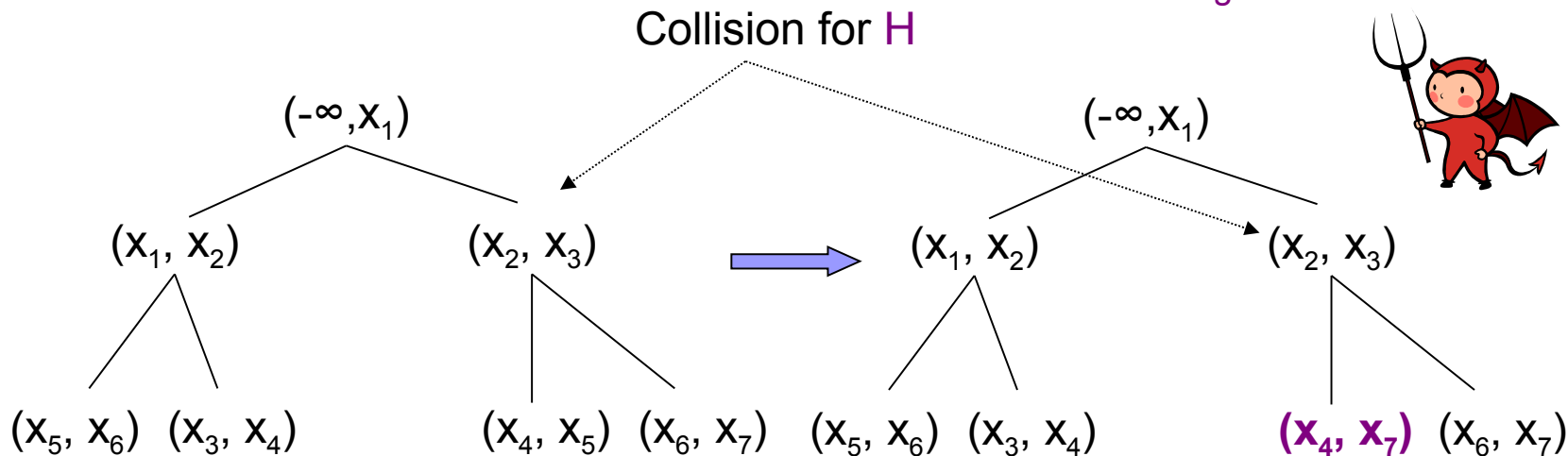
Collision for  $H$



# Security

- **Lemma:** Given a tree  $T$  with accumulated value  $\text{Acc}_T$ , finding a tree  $T'$ ,  $T \neq T'$  such that  $\text{Acc}_T = \text{Acc}_{T'}$ , is difficult.

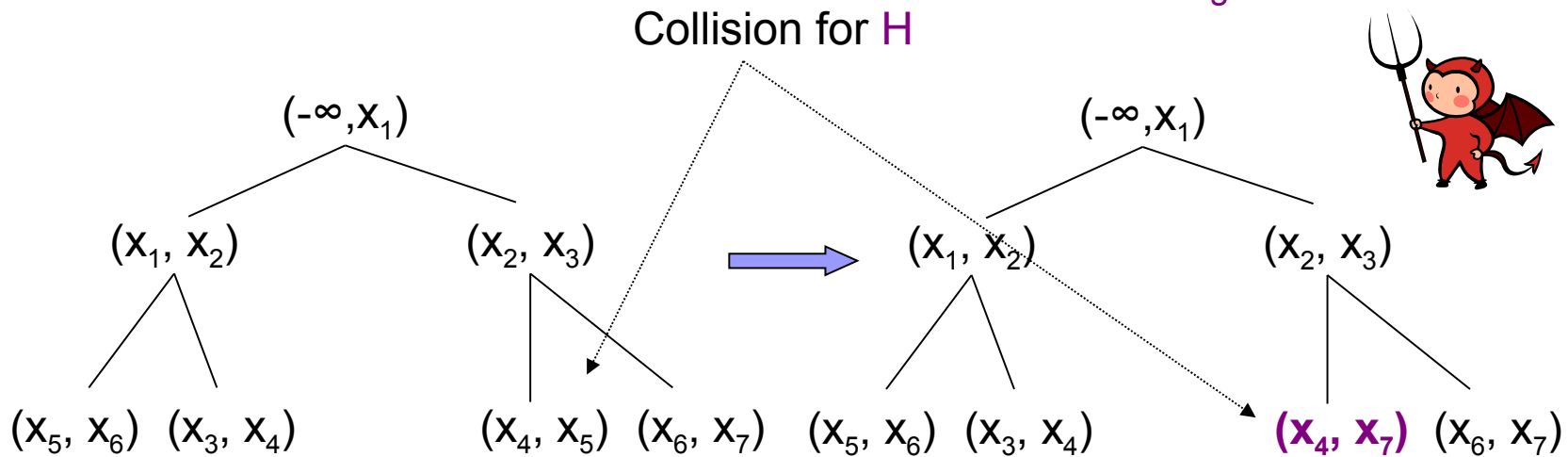
- *Proof (Sketch):*  $\text{Proof}_N = H(\text{Proof}_{\text{left}} \parallel \text{Proof}_{\text{right}} \parallel \text{value})$



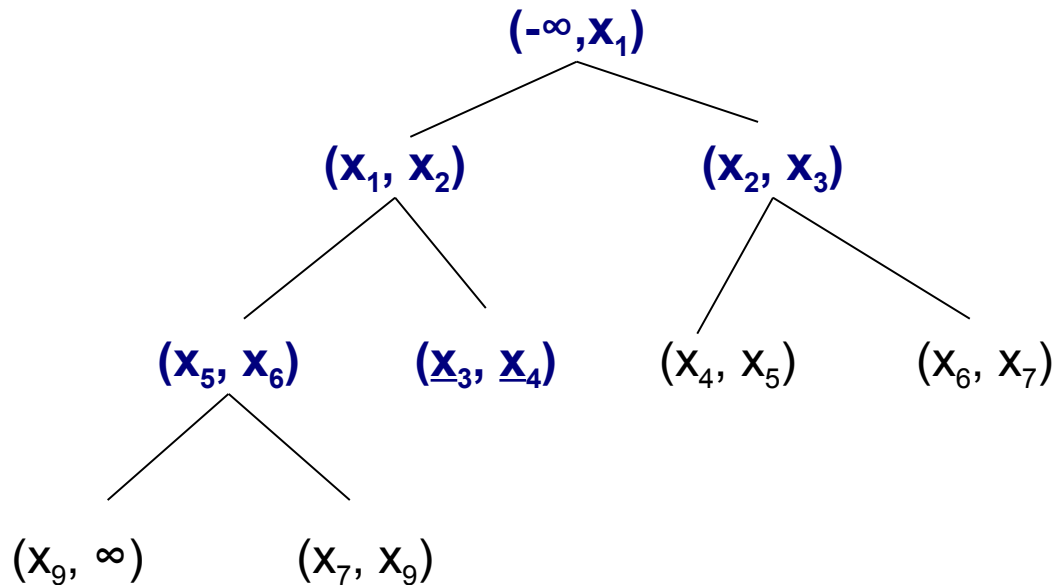
# Security

- **Lemma:** Given a tree  $T$  with accumulated value  $\text{Acc}_T$ , finding a tree  $T'$ ,  $T \neq T'$  such that  $\text{Acc}_T = \text{Acc}_{T'}$ , is difficult.

- *Proof (Sketch):*  $\text{Proof}_N = H(\text{Proof}_{\text{left}} \parallel \text{Proof}_{\text{right}} \parallel \text{value})$



# Security (Consistency)

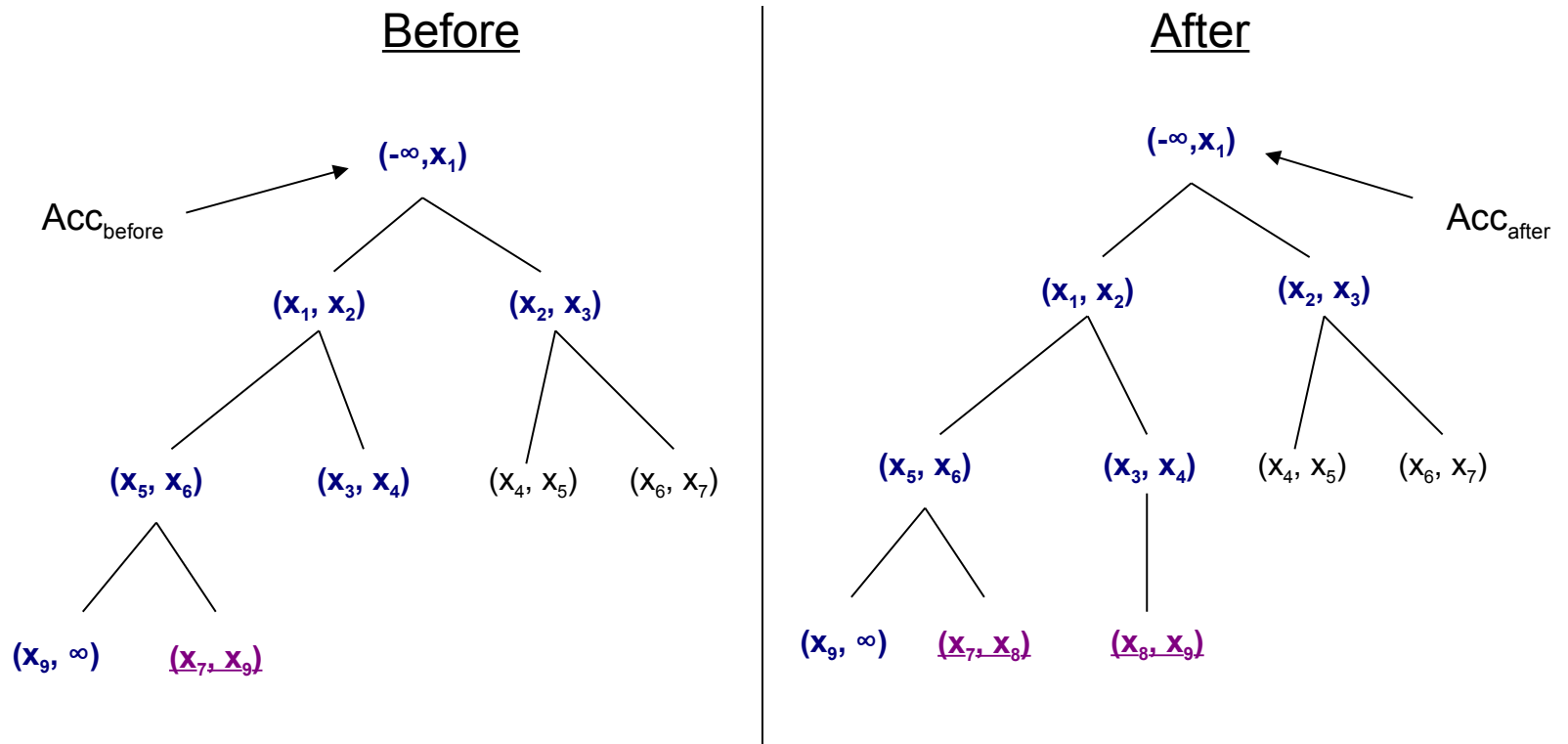


**Witness:** blue nodes and the  $(x_3, x_4)$  pair, size in  $O(\ln(n))$

**Checking that  $x$  belongs (or not) to  $X$ :**

- 1) compute recursively the proof  $P$  and verify that  $P=Acc$
- 2) check that:
  - $x=x_3$  or  $x=x_4$  (membership)
  - $x_3 < x < x_4$  (nonmembership)

# Security (Update)



Insertion of  $x_8$

# Conclusion & Open Problem

- First *dynamic, universal, strong* accumulator.
- Simple.
- Security
  - Existence of collision-resistant hash functions.
- Solves the e-Invoice Factoring Problem.
- Less efficient than other constructions
  - Size of witness in  $O(\ln(n))$ .
- Open Problem
  - “Is it possible to build a *strong, dynamic* and *universal* accumulator with witness size lower than  $O(\ln(n))$ ?”

Thank you!

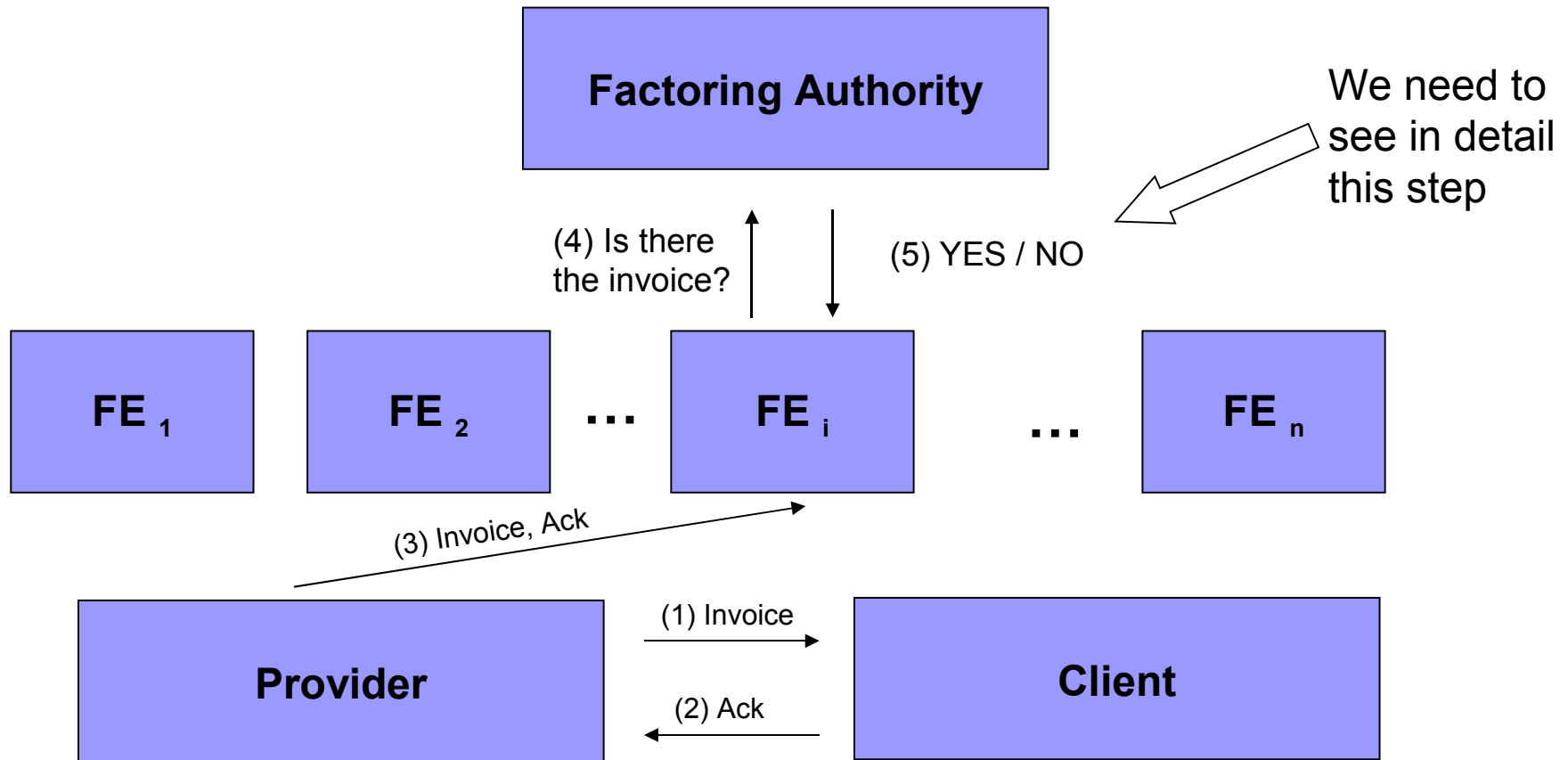




# Invoice Factoring using accumulator

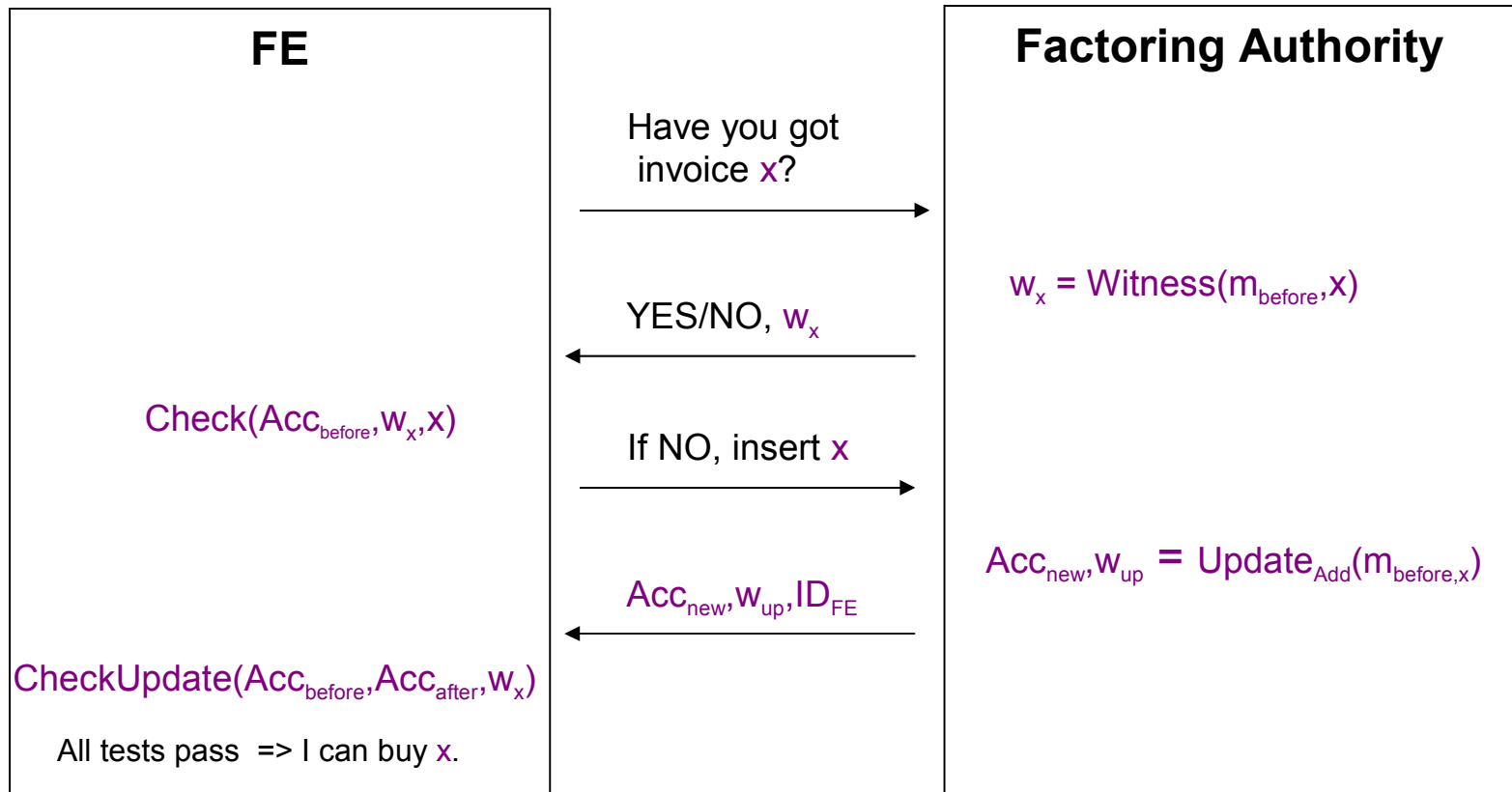
- We need a secure broadcast channel
  - If a message  $m$  is published, every participant sees the same  $m$ .
- Depending on the security level required
  - Trusted http or ftp server
  - Bulletin Board [CGS97]

# Invoice Factoring using accumulator



# Invoice Factoring using accumulator

## ■ Step 5 (Details)

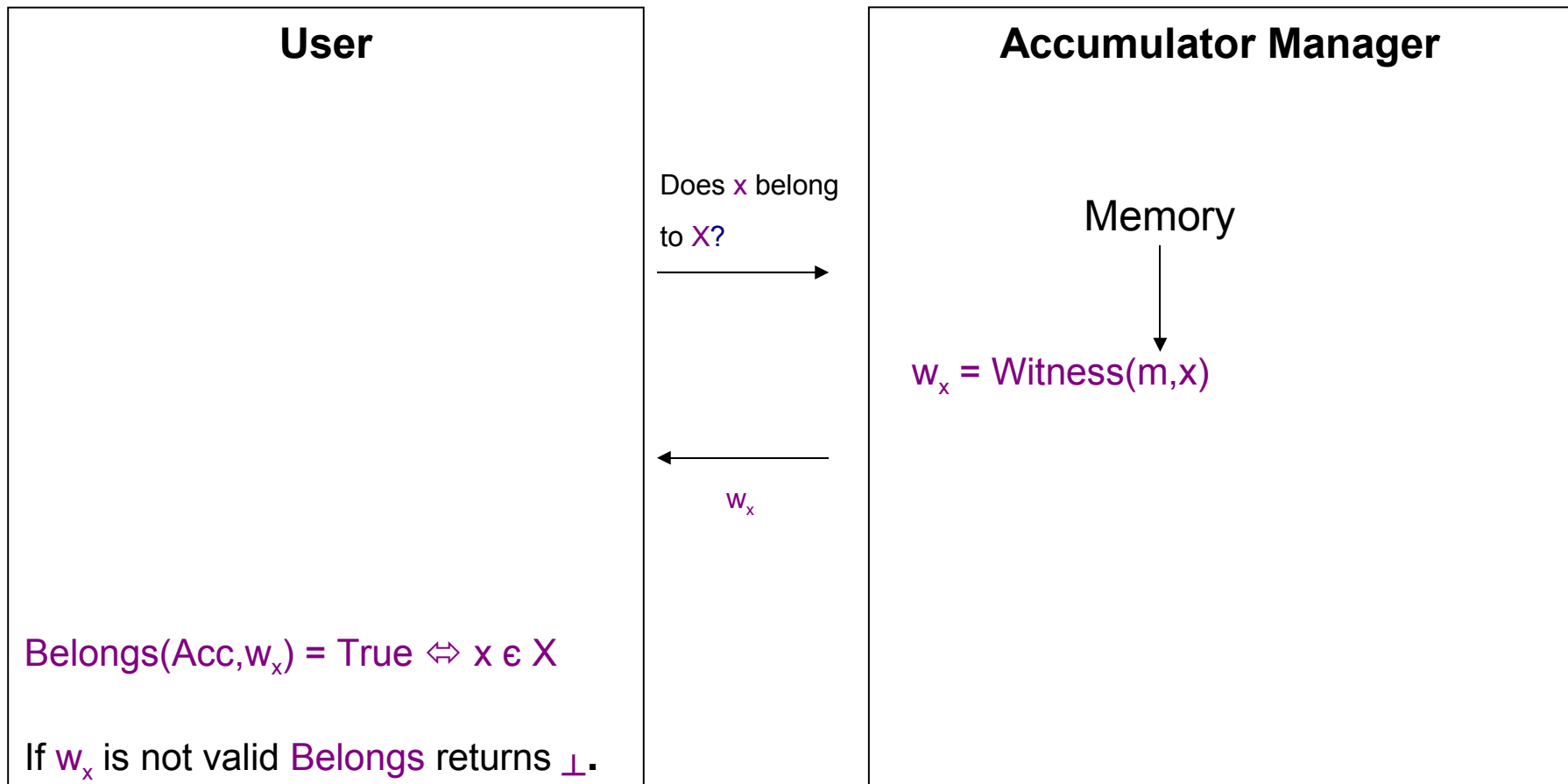




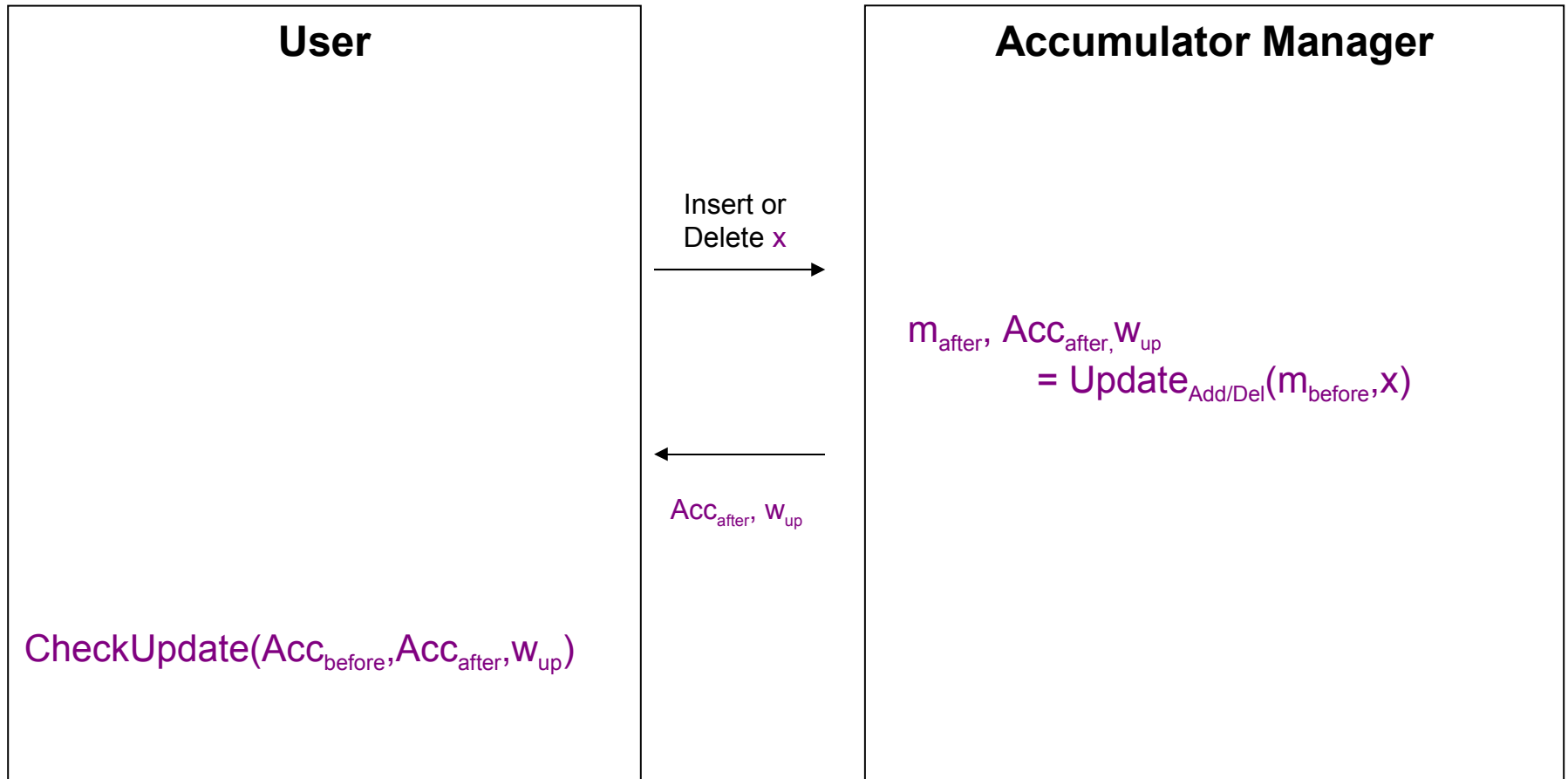
# Distributed solutions?

- Complex to implement
- Hard to make them robust
- High bandwidth communication
- Need to be online – synchronization problems
- **That's why we focus on a centralized solution.**

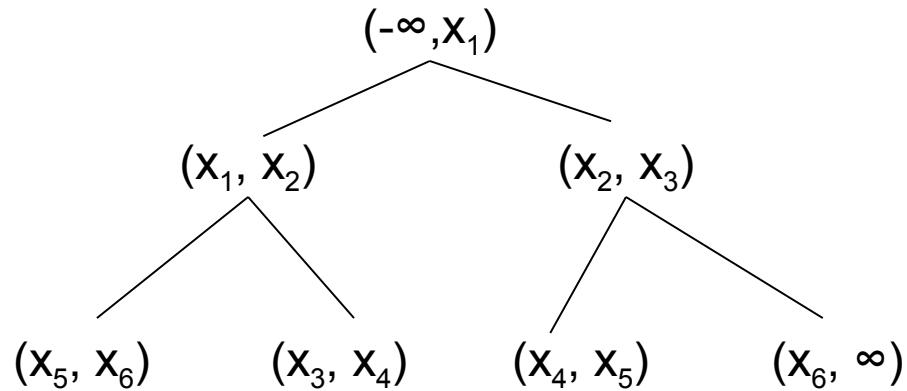
# Checking for (non-)membership



# Update of the accumulated value



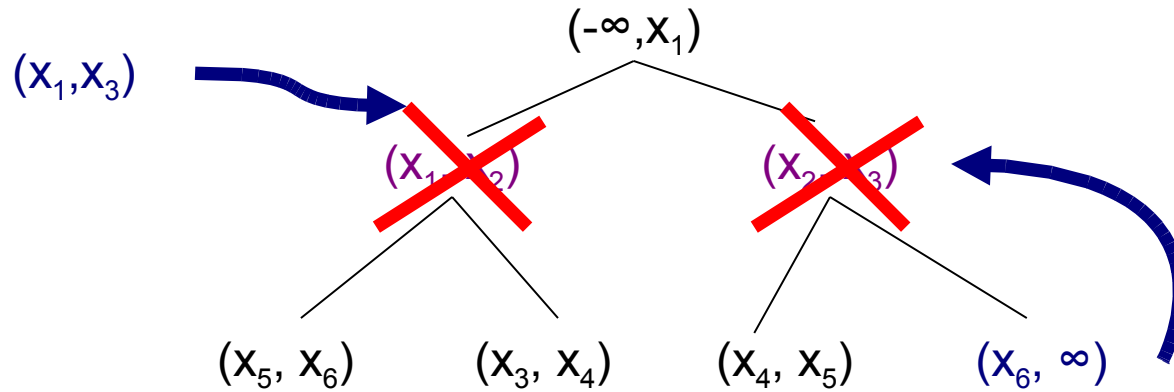
# How to delete elements?



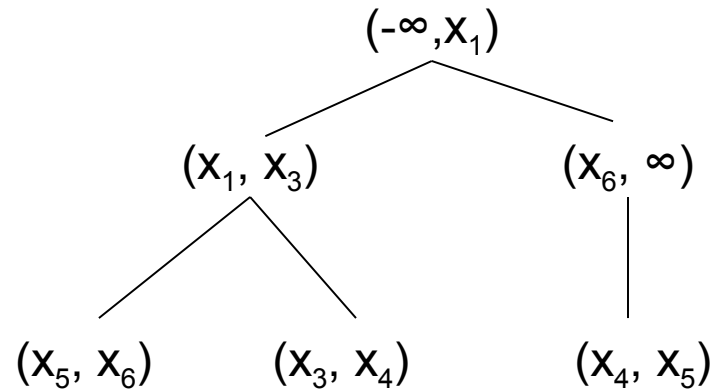
$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

element to be deleted:  $x_2$

# How to delete elements?



# How to delete elements?



# Bibliography

- **[BeMa92]** Efficient Broadcast Time-Stamping *Josh Benaloh and Michael de Mare* 1992
- **[BeMa94]** One-way Accumulators: A decentralized Alternative to Digital Signatures *Josh Benaloh and Michael de Mare* , 1994
- **[BarPfi97]** Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees *Niko Barić and Birgit Pfitzmann* 1997
- **[CGS97]** A secure and optimally efficient multi-authority election scheme *R. Cramer, R. Gennaro, and B. Schoenmakers* 1997
- **[Koch98]** On certificate revocation and validation *P.C. Kocher* 1998
- **[CGH98]** The random oracle methodology revisited *R. Canetti, O. Goldreich and S. Halevi* 1998
- **[Sand99]** Efficient Accumulators Without Trapdoor *Tomas Sanders* 1999
- **[GoTa01]** An efficient and Distributed Cryptographic Accumulator *Michael T. Goodrich and Roberto Tamassia* 2001
- **[CamLys02]** Dynamic Accumulators And Application to Efficient Revocation of Anonymous Credentials *Jan Camenisch Anna Lysyanskaya* 2002
- **[GeRa04]** RSA Accumulator Based Broadcast Encryption *Craig Gentry and Zulfikar Ramzan* 2004
- **[LLX07]** Universal Accumulators with Efficient Nonmembership Proofs *Jiangtao Li, Ninghui Li and Rui Xue* 2007
- **[AWSM07]** Compact E-Cash from Bounded Accumulator *Man Ho Au, Qianhong Wu, Willy Susilo and Yi Mu* 2007
- **[WWP08]** A new Dynamic Accumulator for Batch Updates *Peishun Wang, Huaxiong Wang and Josef Pieprzyk* 2008
- **[CKHO08]** Strong Accumulators from Collision-Resistant Hashing *Philippe Camacho, Alejandro Hevia, Marcos Kiwi, and Roberto Opazo* 2008