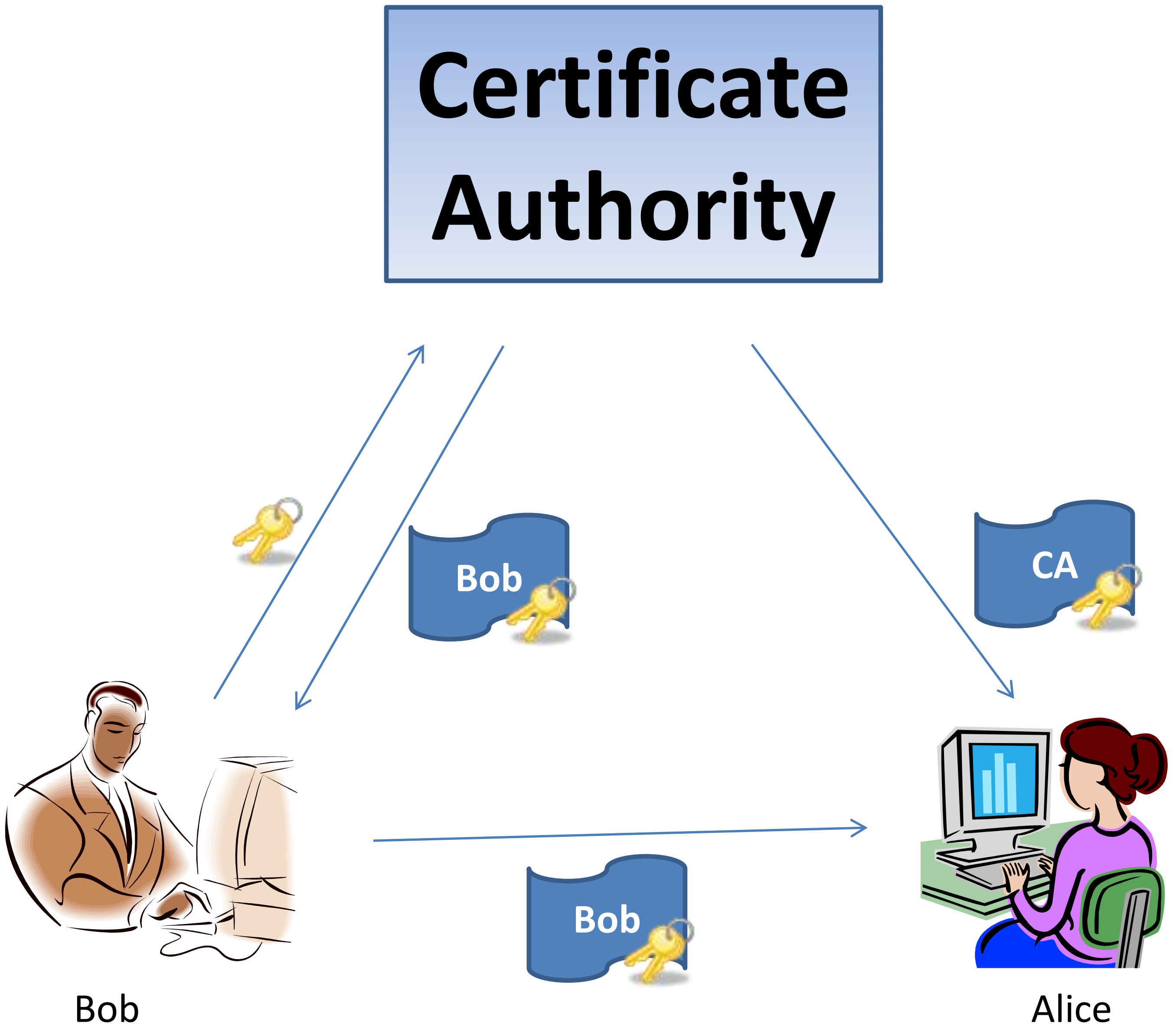
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

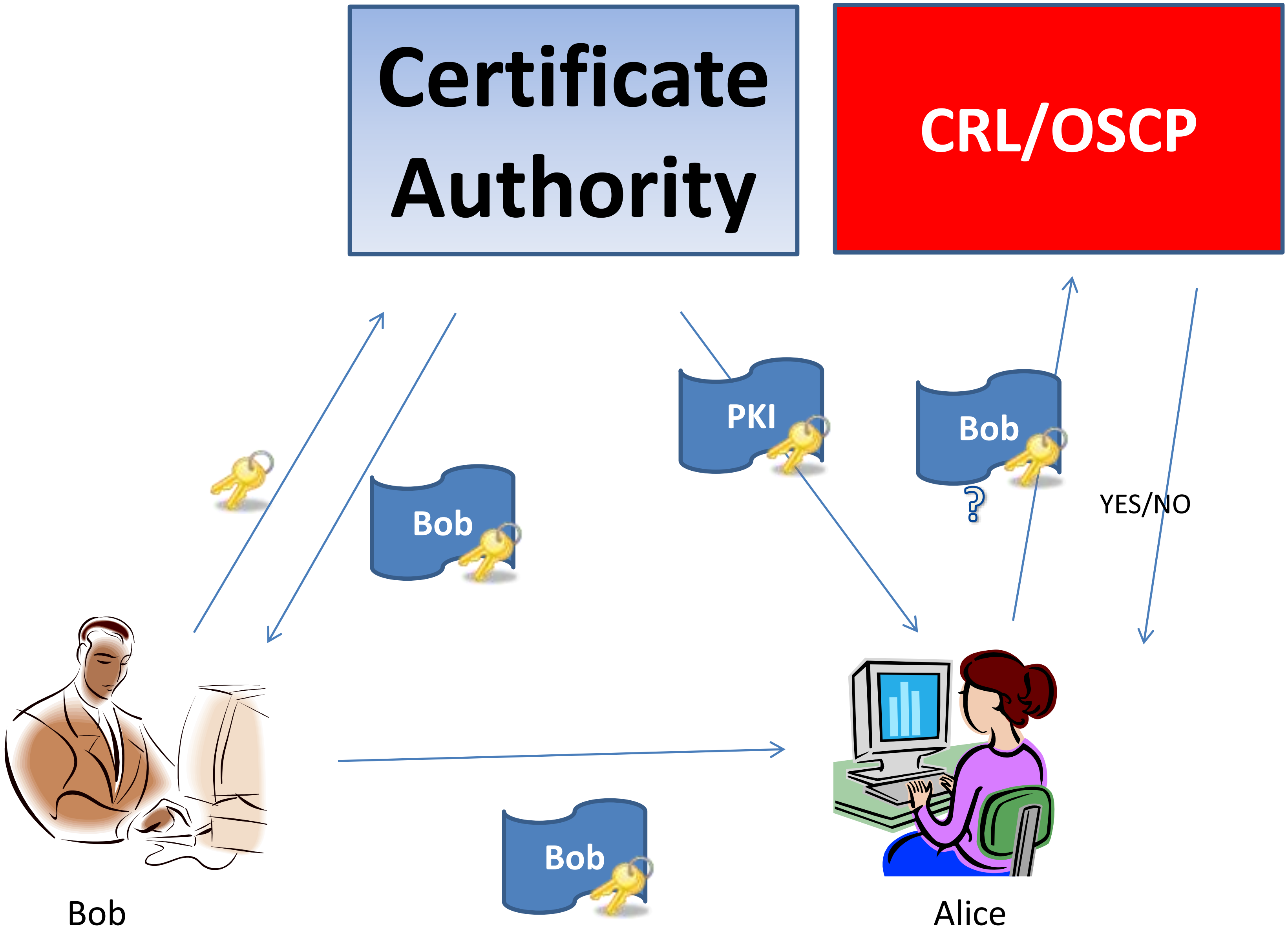
latin crypt
PUEBLA 2010

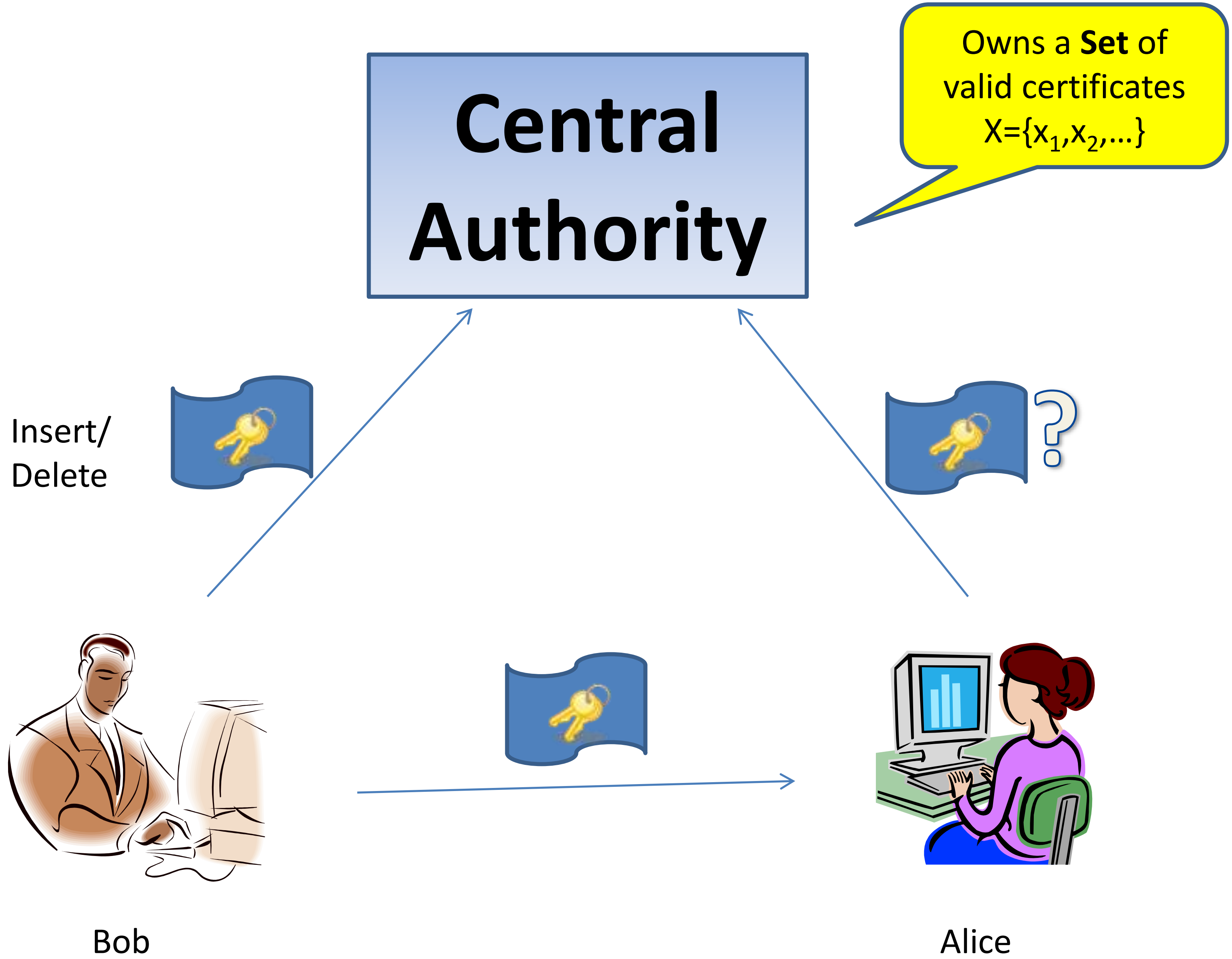
On the Impossibility of Batch Update for Cryptographic Accumulators

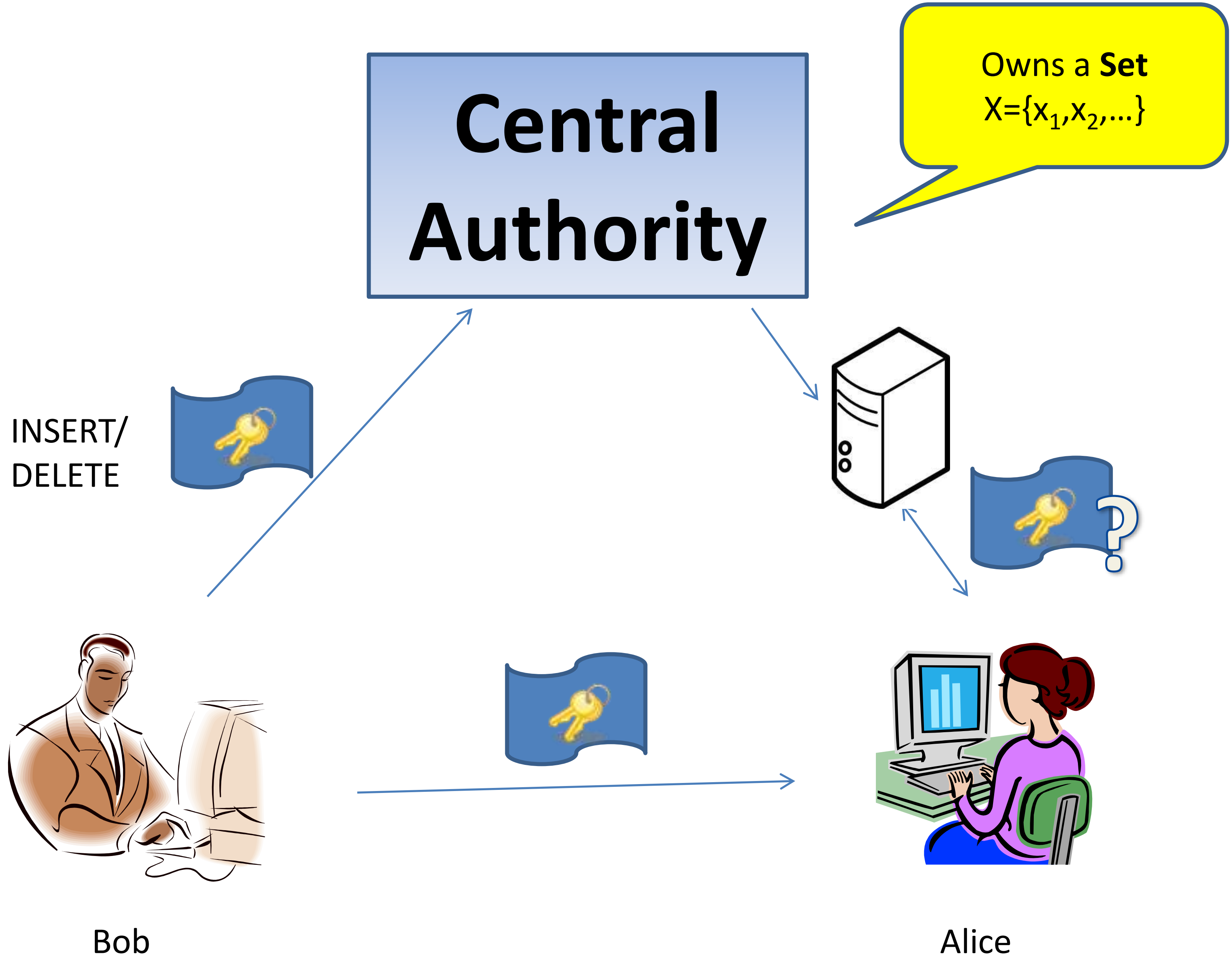
Philippe Camacho and Alejandro Hevia

University of Chile

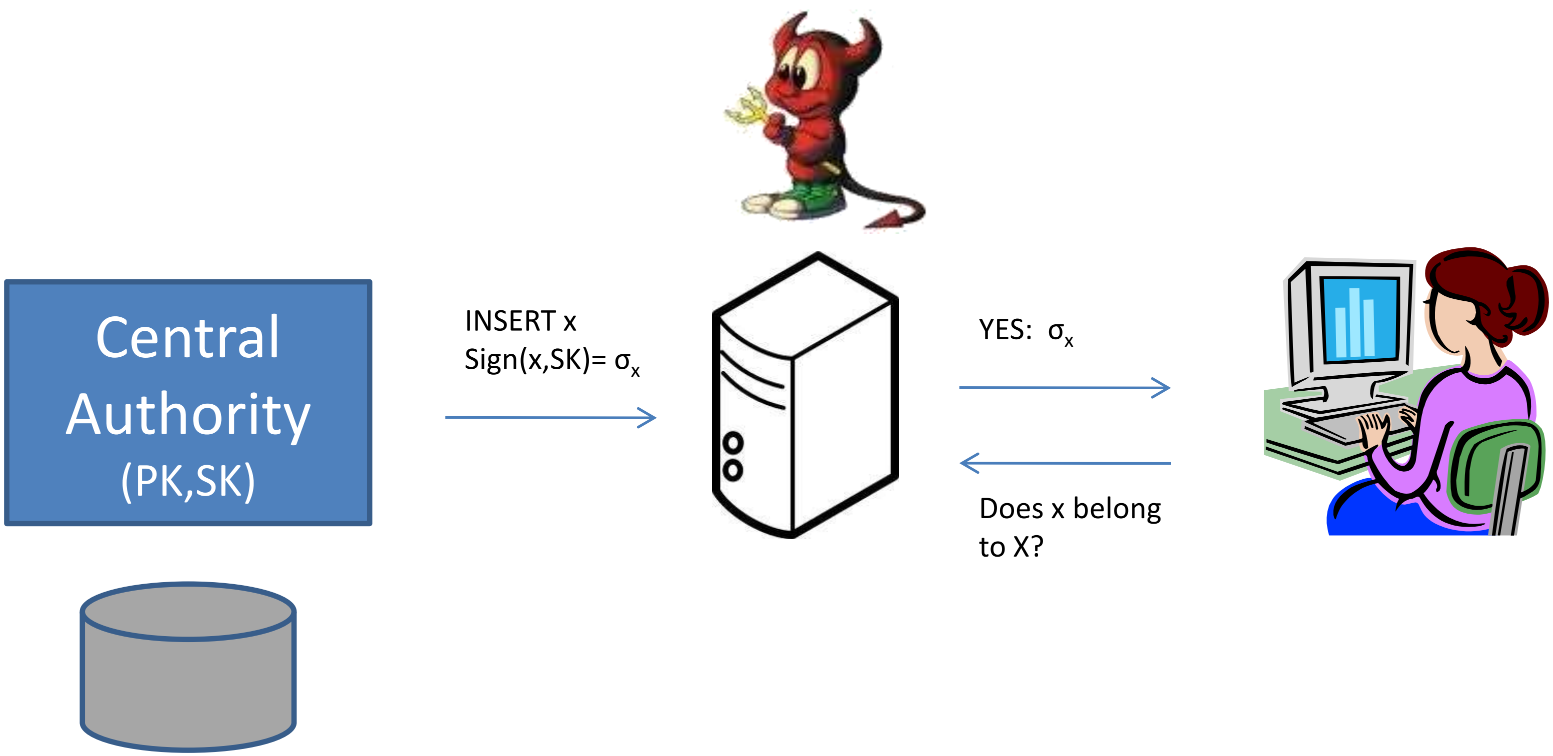

Certificate Authority
Bob
CA
Bob
Bob
Alice


Certificate Authority
CRL/OSCP
PKI
Bob
Bob
?
YES/NO
Bob
Bob
Alice

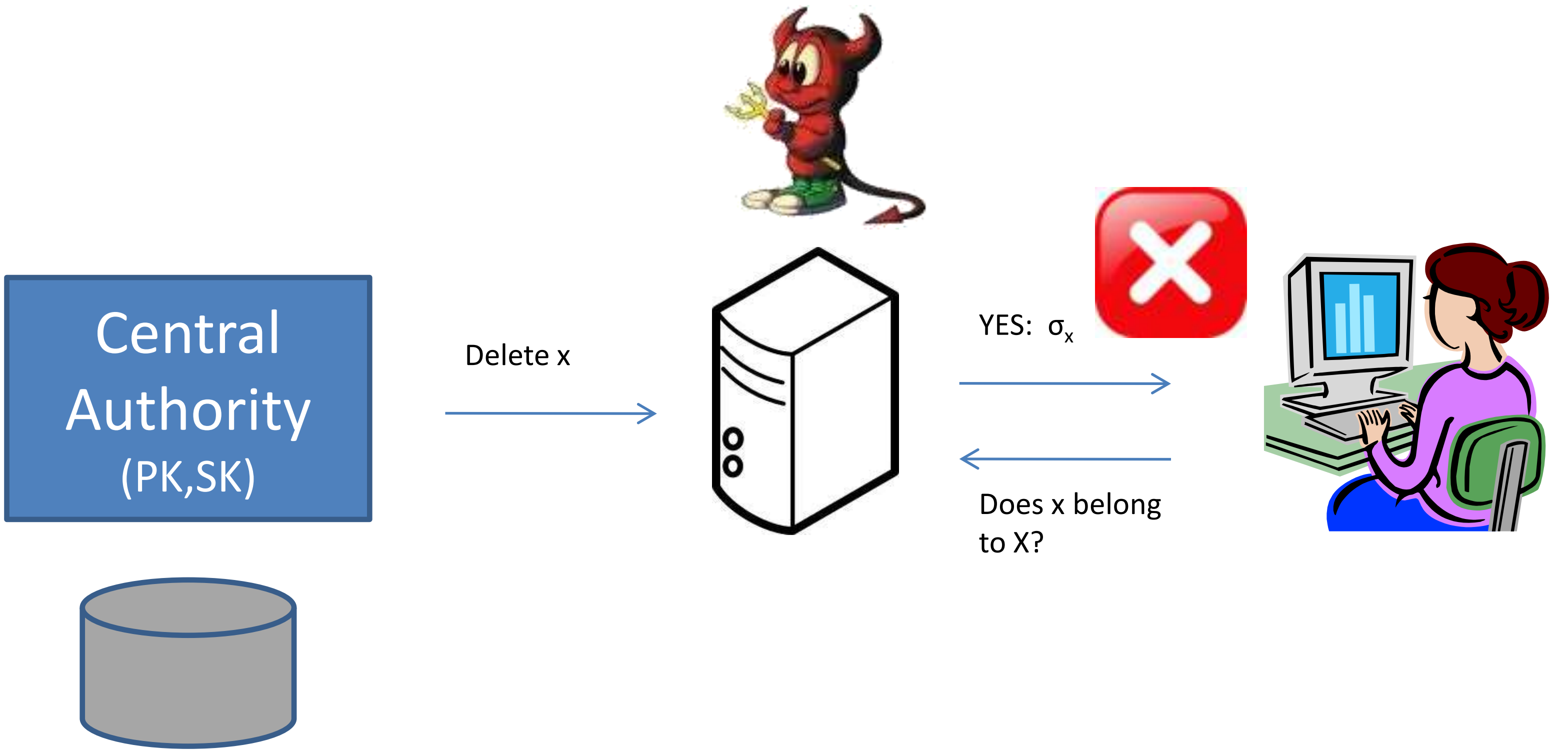

Central Authority
Owns a **Set** of valid certificates $X=\{x_1,x_2,\ldots\}$
Insert/ Delete
?
Bob
Alice

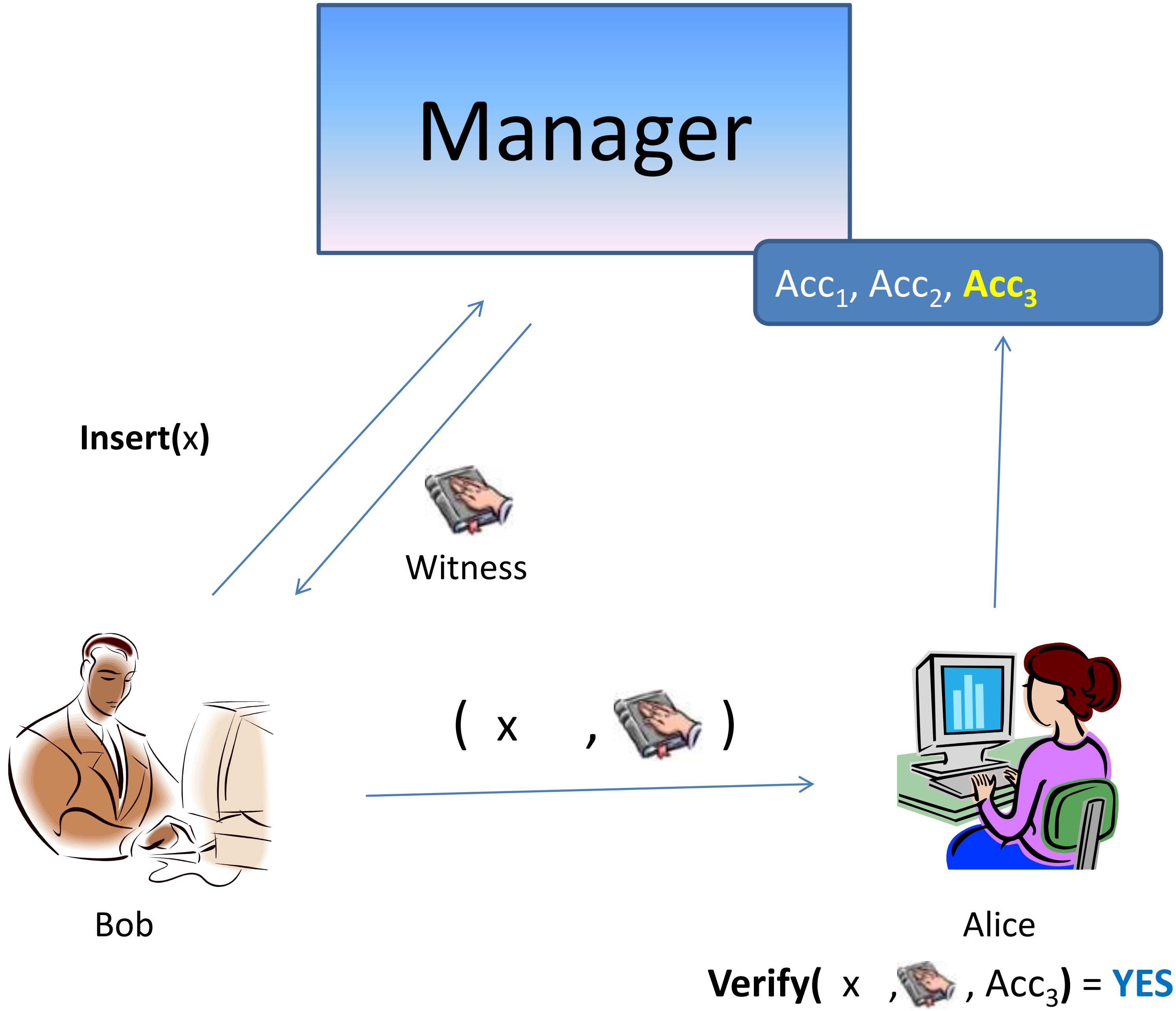

Central Authority
Owns a **Set** $X=\{x_1,x_2,\ldots\}$
INSERT/
DELETE
Bob
Alice

Replay Attack

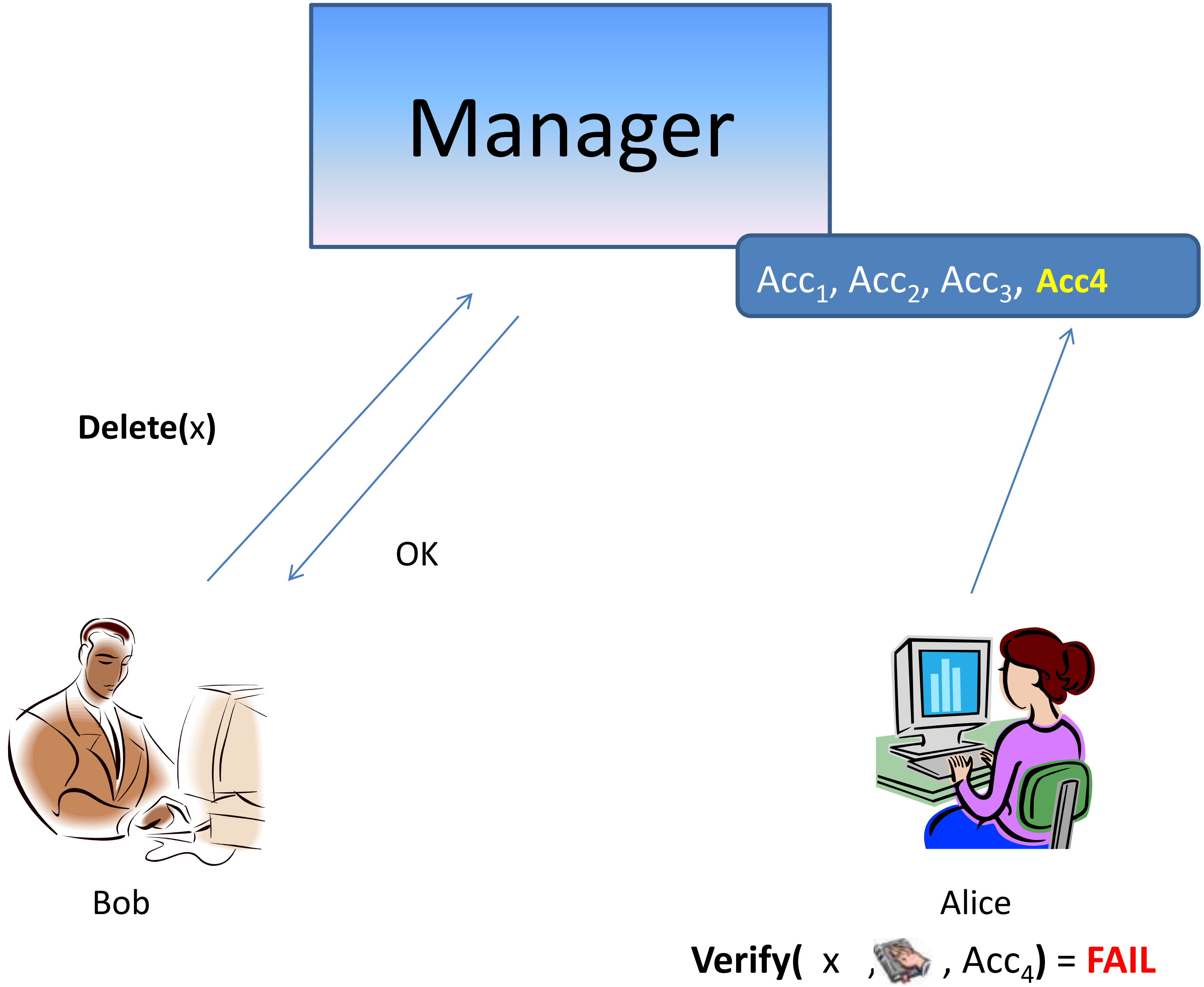


Replay Attack



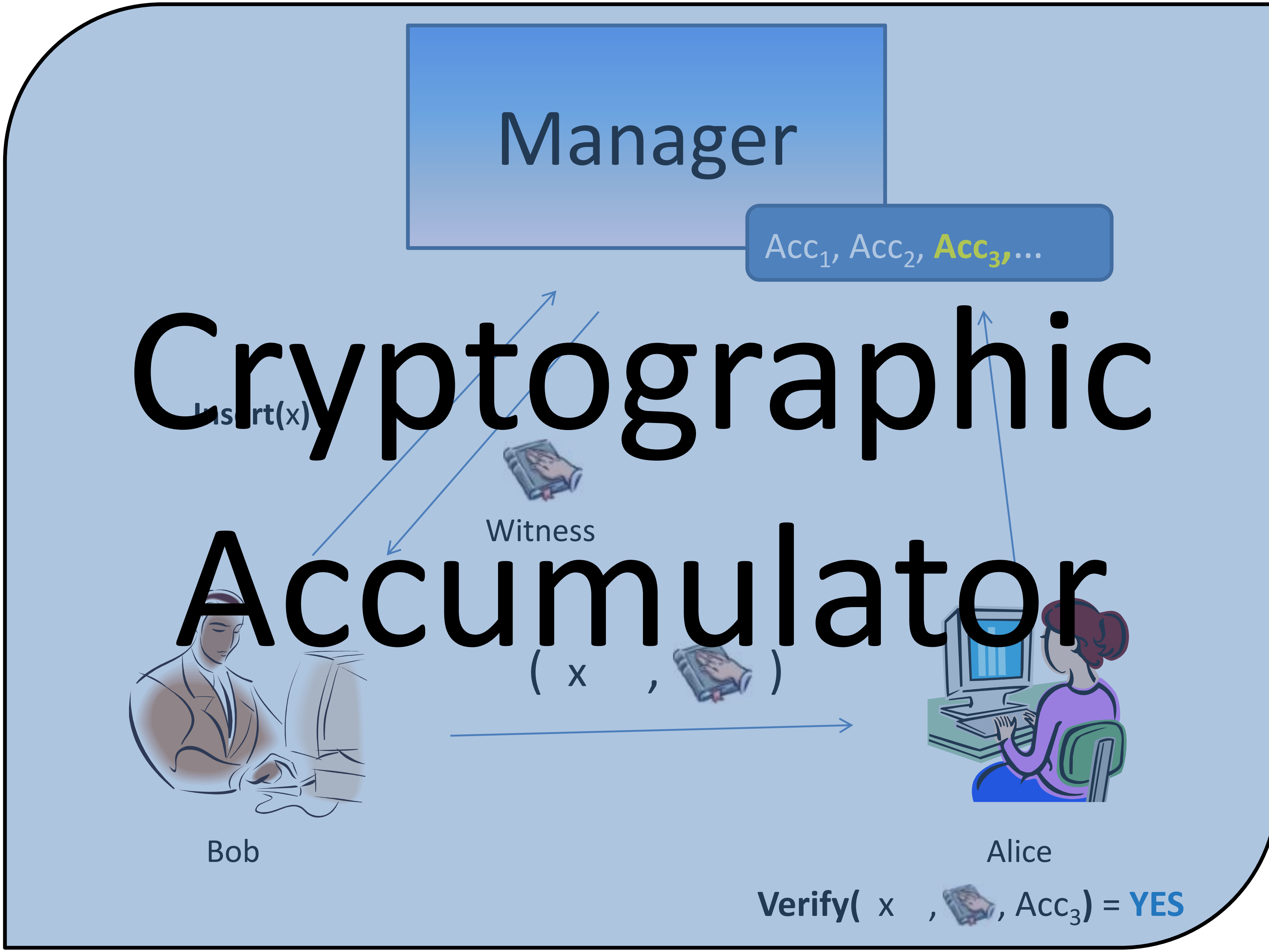


Manager
Acc_1, Acc_2, Acc_3
Insert(x)
Witness
(x ,)
Bob
Alice
Verify(x , , Acc_3) = YES



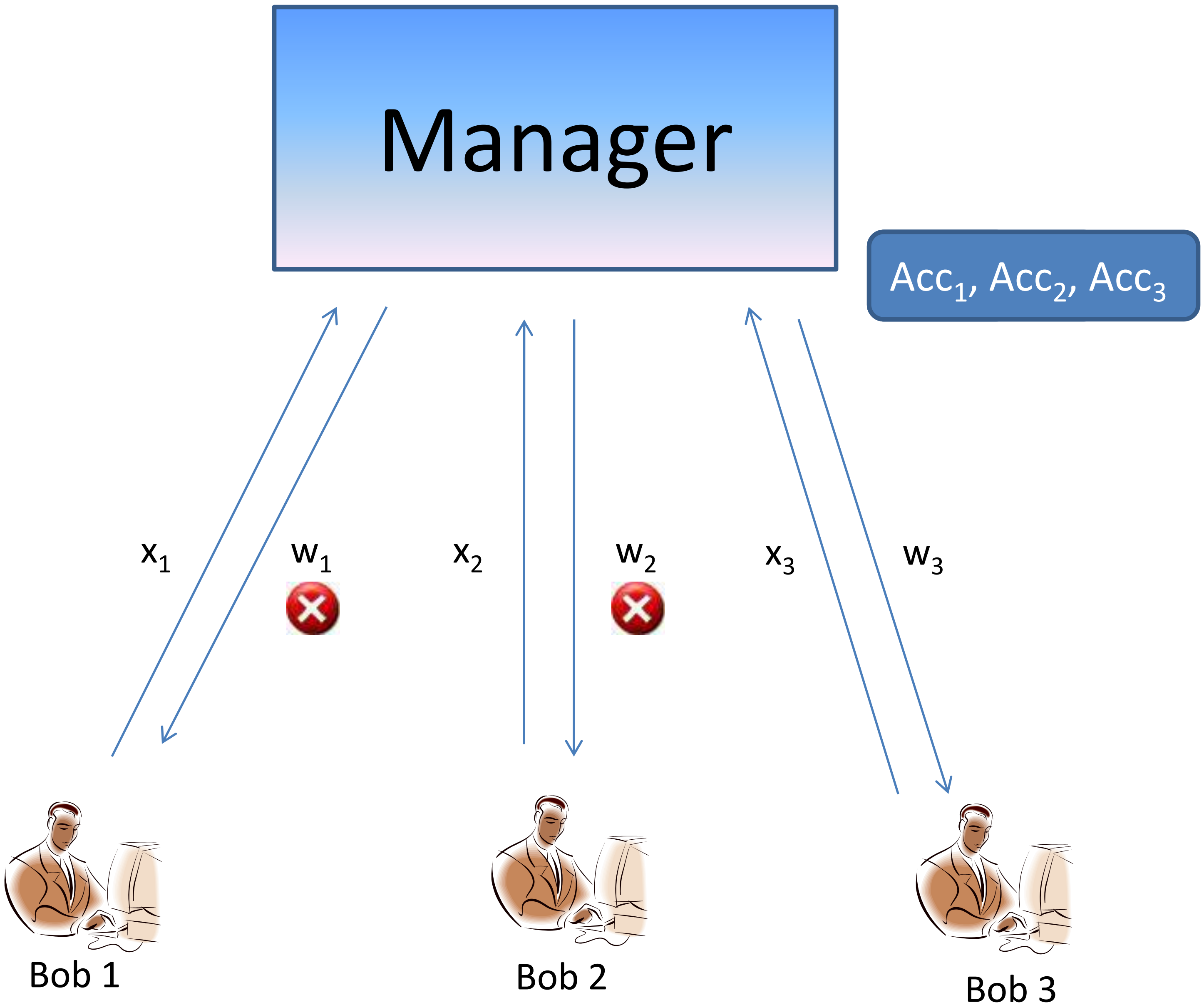
Manager
Acc1, Acc2, Acc3, Acc4
Delete(x)
OK
Bob
Alice
Verify(x , , Acc4) = FAIL

Cryptographic Accumulator



Main constructions

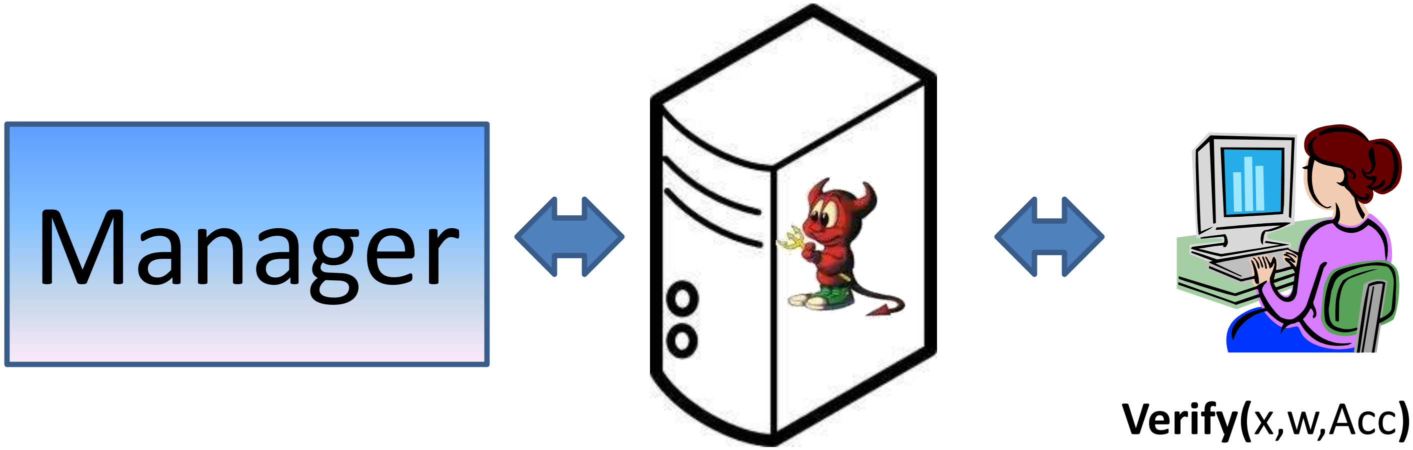
	Security	Note
[BeMa94]	RSA + RO	First definition
[BarPfi97]	Strong RSA	-
[CamLys02]	Strong RSA	First dynamic accumulator
[LLX07]	Strong RSA	First universal accumultor
[Ngu05]	Pairings	E-cash, ZK-Sets,…
[WWP08]	eStrong RSA Paillier	Batch Update
[CHKO08]	Collision-Resistant Hashing	Untrusted Manager
[CKS09]	Pairings	Group multiplication



Manager
Acc1, Acc2, Acc3
x1
w1
x2
w2
x3
w3
Bob 1
Bob 2
Bob 3

Problem: after each update of the accumulated value it is necesarry to recompute all the witnesses.

Delegate Witness Computation?



Constructions	Replica (Compute a single witness)	User (Verify)
[CL02]	O(|X|)	O(1)
[GTT09]	$O(\|X\|^{1/\varepsilon})$	O(ε)
[CHK08]	O(log |X|)	O(log |X|)

Batch Update [FN02]

Manager

…, Acc_{99}, $\mathbf{Acc_{100}}$, Acc_{101}, …, $\mathbf{Acc_{200}}$, …

$Upd_{100,200}$

Bob 1

(x_1, w_1, Acc_{100})
(x_2, w_2, Acc_{100})
(x_6, w_6, Acc_{100})

Bob 2

$(x_{36}, w_{36}, Acc_{100})$
$(x_{87}, w_{87}, Acc_{100})$

Bob 29

(x_1, w_1, Acc_{100})
$(x_{20}, w_{20}, Acc_{100})$
$(x_{69}, w_{68}, Acc_{100})$
$(x_{64}, w_{64}, Acc_{100})$
…

Bob 42

(x_1, w_1, Acc_{100})
(x_2, w_2, Acc_{100})
(x_6, w_6, Acc_{100})
….

Batch Update [FN02]

Manager

$\ldots, Acc_{99}$, $\mathbf{Acc_{100}}$, $Acc_{101}, \ldots$, $\mathbf{Acc_{200}}, \ldots$

Bob 1

$(x_1, \mathbf{w_1', Acc_{200}})$
$(x_2, \mathbf{w_2', Acc_{200}})$
$(x_6, \mathbf{w_6', Acc_{200}})$

Bob 2

$(x_{36}, \mathbf{w_{36}', Acc_{200}})$
$(x_{87}, \mathbf{w_{87}', Acc_{200}})$

Bob 29

$(x_1, \mathbf{w_1', Acc_{200}})$
$(x_{20}, \mathbf{w_{20}', Acc_{200}})$
$(x_{69}, \mathbf{w_{68}', Acc_{200}})$
$(x_{64}, \mathbf{w_{64}', Acc_{200}})$
…

Bob 42

$(x_1, \mathbf{w_1', Acc_{200}})$
$(x_2, \mathbf{w_2', Acc_{200}})$
$(x_6, \mathbf{w_6', Acc_{200}})$
….

Batch Update [FN02]

Trivial solution:

Upd_{X_i,X_j} = {list of all witnesses for X_j}

More interesting:

$|Upd_{X_i,X_j}| = O(1)$

What happens with [CL02]?

- PK=(n,g) with n=pq and g ϵ $\mathbf{Z}_n^*$
- $Acc_{\emptyset}$:= g mod n
- **Insert(**x,Acc**)** := Acc^x mod n /* x prime */
- **Delete(**x,Acc**)** := $Acc^{1/x}$ mod n
- **WitGen(**x,Acc**)** $\stackrel{?}{:=}$ $Acc^{1/x}$ mod n
- **Verify(**x,w,Acc**)**: $w^x = Acc$
- $|\mathbf{Upd}_{x_i,x_j}|$ = **O(|{list of insertions / deletions}|)**

Syntax of B.U. Accumulators

Algorithm	Returns	Who runs it
KeyGen(1^k**)**	**PK,SK,**$\mathbf{Acc}_{\emptyset}$	Manager
AddEle(x,Acc_X,SK**)**	$\mathbf{Acc}_{X \cup \{x\}}$	Manager
DelEle(x,Acc_X,SK**)**	$\mathbf{Acc}_{X \setminus \{x\}}$	Manager
WitGen(x,Acc_X,SK**)**	Witness **w** relative to $\mathbf{Acc}_X$	Manager
Verify(x,w,Acc_X,PK**)**	Returns **Yes** whether $\mathbf{x \in X}$	User
UpdWitGen(X,X',SK**)**	$\mathbf{Upd}_{X,X'}$ for elements $\mathbf{x \in X \cap X'}$	Manager
UpdWit(w,Acc_X,$Acc_{X'}$,$Upd_{X,X'}$,PK**)**	New witness **w'** for $\mathbf{x \in X'}$	User

Correctness

- **Definition**

The scheme is correct iff:

w := **WitGen(**x, Acc_X, SK**)** $\Rightarrow$ **Verify(**x, w, Acc_X, PK**)** = Yes

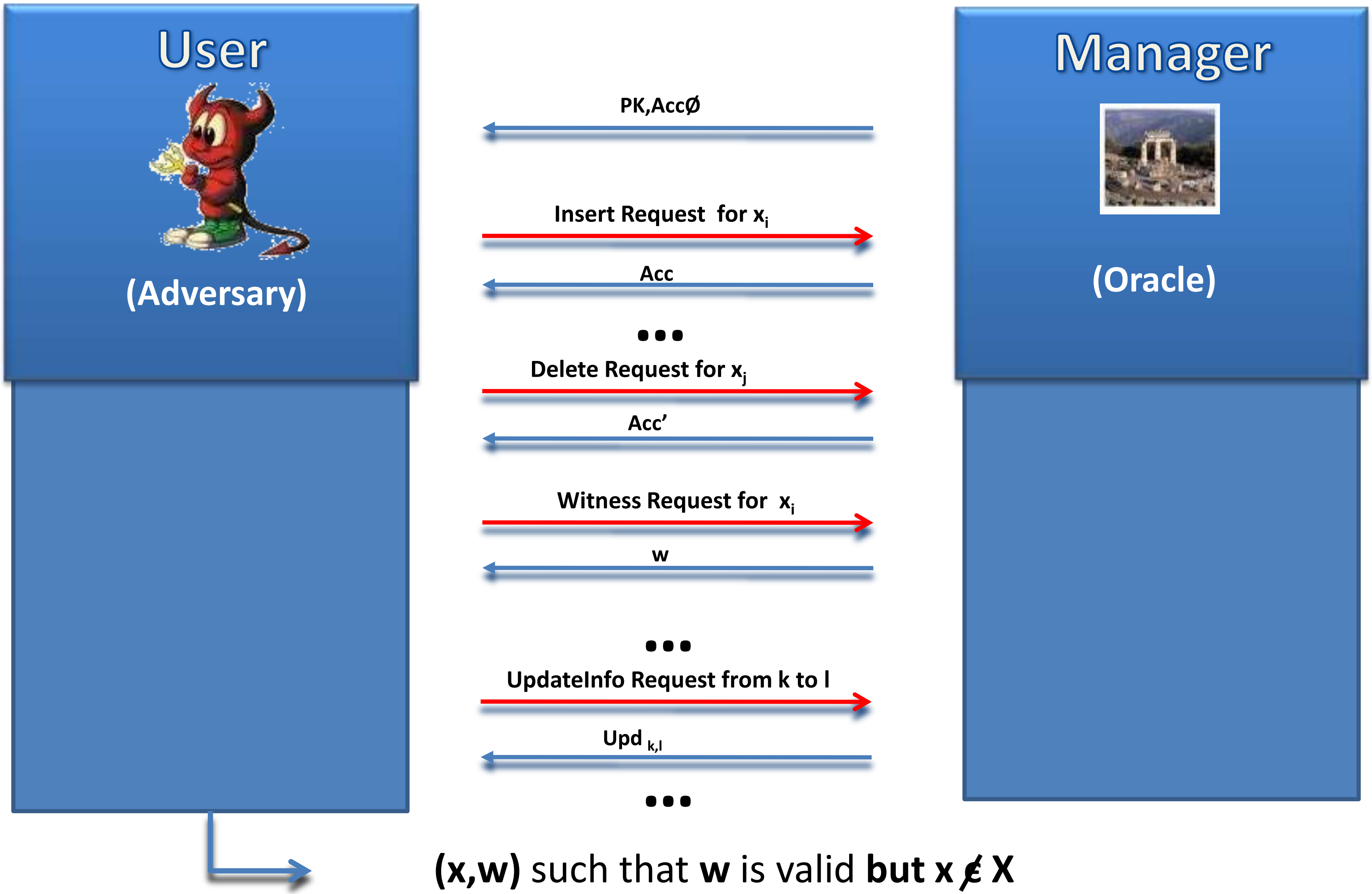
w := **WitGen(**x, Acc_X, SK**)**

$Upd_{X,X'}$:= **UpdWitGen(**X, X', SK**)**

w' := **WitGen(**$w, Acc_X, Acc_{X'}, Upd_{X,X'}, PK$**)**

$\Rightarrow$ **Verify(**$x, w', Acc_{X'}, PK$**)** = Yes

Security Model [CL02,WWP08]



Batch Update Construction [WWP08]

CONSTRUCTION. Wang et al.'s accumulator relies on the Paillier cryptosystem [8] which we recall in Appendix A.2. In the following, λ will denote the value $lcm(p-1, q-1)$ where $n = pq$ is a product of large-enough safe primes p, q, and $F : u \to \frac{u-1}{n}$ is Paillier's L function [8].

- KeyGen(1^k): given the security parameter k in unary, compute a safe-prime product $n = pq$ that is k-bits long and create an empty set V. Let $\mathcal{C} = \mathbb{Z}^*_{n^2} \setminus \{1\}$ and $T' = \{3, ..., n^2\}$. Let $\beta \xleftarrow{R} \mathbb{Z}^*_{\varphi(n^2)}$ and $\sigma \xleftarrow{R} \mathbb{Z}^+$ be two random numbers. The public key PK is set to (n, β) and the private key SK to (σ, λ). The output is the parameter $\mathcal{P} = (PK, SK)$.
- AccVal($X, \mathcal{P}$): given a set $X = \{c_1, ..., c_m\}$ with $X \subset \mathcal{C}$, and the parameter $\mathcal{P}$, take $c_{m+1} \xleftarrow{R} \mathcal{C}$ and compute

$$\begin{aligned} x_i &= F(c_i^\lambda \bmod n^2) \bmod n \ \ (\text{for } i = 1, ..., m+1) \\ Acc_X &= \sigma \textstyle\sum_{i=1}^{m+1} x_i \bmod n \\ y_i &= c_i^{\lambda\sigma\beta^{-1}} \bmod n^2 \ \ (\text{for } i = 1, ..., m+1) \\ a_c &= \Pi_{i=1}^{m+1} y_i \bmod n^2 \end{aligned}$$

  Output the accumulated value Acc_X and the auxiliary information a_c.
- WitGen($a_c, X, \mathcal{P}$): given the auxiliary information a_c, a set $X = \{c_1, ..., c_m\}$, and the parameter $\mathcal{P}$, choose uniformly at random a set of m numbers $T = \{t_1, ..., t_m\} \subset T' \setminus \{\beta\}$ (for $i = 1, ..., m$) and compute

$$w_i = a_c c_i^{-t_i\beta^{-1}} \bmod n^2 \ \ (\text{for } i = 1, ..., m)$$

  Output the witness $W_i = (w_i, t_i)$ for c_i (for $i = 1, ..., m$).
- AddEle($X^\oplus, a_c, Acc_X, \mathcal{P}$): given a set $X^\oplus = \{c_1^\oplus, ..., c_l^\oplus\} (X^\oplus \subseteq \mathcal{C} \setminus X)$, to be inserted, the auxiliary information a_c, the accumulated value Acc_X, and the parameter $\mathcal{P}$, choose $c_{l+1}^\oplus \xleftarrow{R} \mathcal{C}$ and a set of l numbers $T^\oplus = \{t_1^\oplus, ..., t_l^\oplus\} \xleftarrow{R} T' \setminus (T \cup \{\beta\})$, and compute

$$\begin{aligned} x_i^\oplus &= F((c_i^\oplus)^\lambda \bmod n^2) \bmod n \ \ (\text{for } i = 1, ..., l+1) \\ Acc_{X \cup X^\oplus} &= Acc_X + \sigma \textstyle\sum_{i=1}^{l+1} x_i^\oplus \bmod n \\ y_i^\oplus &= (c_i^\oplus)^{\lambda\sigma\beta^{-1}} \bmod n^2 \ \ (\text{for } i = 1, ..., l+1) \\ a_u &= \Pi_{i=1}^{l+1} y_i^\oplus \bmod n^2 \\ w_i^\oplus &= a_c a_u (c_i^\oplus)^{-t_i^\oplus \beta^{-1}} \bmod n^2 \ \ (\text{for } i = 1, ..., l) \end{aligned}$$

  Set $a_c = a_c a_u \bmod n^2, T = T \cup T^\oplus$, and $V = V \cup \{a_u\}$. Then output the new accumulated value $Acc_{X \cup X^\oplus}$ corresponding to the set $X \cup X^\oplus$, the witness $W_i^\oplus = (w_i^\oplus, t_i^\oplus)$ for the new added elements $c_i^\oplus$ (for $i = 1, ..., l$), and the auxiliary information a_u and a_c.
- DelEle($X^\ominus, a_c, Acc_X, \mathcal{P}$): given a set $X^\ominus = \{c_1^\ominus, ..., c_l^\ominus\} (X^\ominus \subset X)$ to be deleted, the auxiliary information a_c, the accumulated value Acc_X, and the parameter $\mathcal{P}$, choose $c_{l+1}^\ominus \xleftarrow{R} \mathcal{C}$ and compute

$$\begin{aligned} x_i^\ominus &= F((c_i^\ominus)^\lambda \bmod n^2) \bmod n \ \ (\text{for } i = 1, ..., l+1) \\ Acc_{X \setminus X^\ominus} &= Acc_X - \sigma \textstyle\sum_{i=1}^{l} x_i^\ominus + \sigma x_{l+1}^\ominus \bmod n \\ y_i^\ominus &= (c_i^\ominus)^{\lambda\sigma\beta^{-1}} \bmod n^2 \ \ (\text{for } i = 1, ..., l+1) \\ a_u &= y_{l+1}^\ominus \Pi_{j=1}^{l} (y_j^\ominus)^{-1} \bmod n^2 \end{aligned}$$

  Set $a_c = a_c a_u \bmod n^2$ and $V = V \cup \{a_u\}$. Then output the new accumulated value $Acc_{X \setminus X^\ominus}$ corresponding to the set $X \setminus X^\ominus$ and the auxiliary information a_u and a_c.
- Verify(c, W, Acc_X, PK): given an element c, its witness $W = (w, t)$, the accumulated value Acc_X, and the public key PK, test whether $\{c, w\} \subset \mathcal{C}, t \in T'$ and $F(w^\beta c^t \bmod n^2) \equiv Acc_X(\bmod n)$. If so, output Yes, otherwise output $\perp$.
- UpdWit(W_i, a_u, PK) : given the witness W_i, the auxiliary information a_u and the public key PK, compute $w_i' = w_i a_u \bmod n^2$ then output the new witness $W_i' = (w_i', t_i)$ for the element c_i.

In the following section we show that Wang et al.'s construction is not secure.

Attack on [WWP08]

 **User**		**Manager**
		$X_0 := \emptyset$
	Insert $\mathbf{x_1}$ →	
	Delete $\mathbf{x_1}$ →	$X_1 := \{x_1\}$
	Please send $\mathbf{Upd}_{x_1,x_2}$ →	$X_2 := \emptyset$
	← $\mathbf{Upd}_{x_1,x_2}$	
With $\mathbf{Upd}_{x_1,x_2}$ **I can** update my witness $\mathbf{w}_{x_1}$		

But x_1 does not belong to X_2!

Batch Update is Impossible

- **Theorem:**

Let **Acc** be a secure accumulator scheme with deterministic **UpdWit** and **Verify** algorithms.

For an update involving **m** delete operations in a set of **N** elements, the size of the update information $\mathbf{Upd}_{X,X'}$ required by the algorithm **UpdWit** is Ω**(m log(N/m))**.

In particular if **m=N/2** we have $|\mathbf{Upd}_{X,X'}| = \Omega(m) = \Omega(N)$

Proof 1/3

 User		Manager
$X=\{x_1,x_2,\ldots,x_N\}$		$X=\{x_1,x_2,\ldots,x_N\}$
	Acc_X , $\{w_1,w_2,\ldots,w_N\}$ ←	**Compute** $Acc_X, \{w_1,w_2,\ldots,w_N\}$
		Delete $X_d:=\{x_{i_1},x_{i_2},\ldots,x_{i_m}\}$ $X' := X\setminus X_d$
	$Acc_{X'}$, $Upd_{X,X'}$ ←	**Compute** $Acc_{X'}, Upd_{X,X'}$

Proof 2/3

User

$X=\{x_1,x_2,\dots,x_N\}$
$\{w_1,\dots,w_N\}$
Acc_X, $Acc_{X'}$, $Upd_{X,X'}$

CASE 1	CASE 2
If **x** is not in **X'** => **Scheme insecure** **x** still in **X'**	If **x** is in **X'** => **Scheme incorrect** **x** not in **X'** anymore

For each element **x∈X** → **x** → **w' := UpdWit(w,Acc_X,$Acc_{X'}$,$Upd_{X,X'}$)** → **w'** valid?

YES → **CASE 1**

NO → **CASE 2**

User can reconstruct the set $\mathbf{X_d}$

Proof 3/3

- There are $\binom{\mathbf{N}}{\mathbf{m}}$ subsets of ***m*** elements in a set of ***N*** elements

- We need $\mathbf{\log\binom{N}{m} \geq m \log(N/m)}$ bits to encode $\mathbf{X_d}$

(See updated version at eprint soon for a detailed proof)

Conclusion

- Batch Update is **impossible**.
- Batch Update for accumulators with ***few*** delete operations?
- Improve the lower bound in a factor of ***k***.

Thank you!

Correction

- With negligible probability
  Bob could obtain a fake witness
  (and the scheme would still be secure)

=> The number of “good”subsets X_d is less than $\binom{N}{m}$

A more careful analysis

- Pr[X_d leads to a fake witness] $\leq$ ε(k)

  => #"Good X_d sets" $\geq \binom{N}{m}$ (1- ε(k))

  => $|Upd_{X,X'}| \geq m \log(M/m) + \log(1- \varepsilon(k))$

  => $|Upd_{X,X'}| \geq m \log(M/m) -1$

  => $|Upd_{X,X'}| = \Omega(m \log(M/m))$