Criptografía y Códigos

Colonia del Sacramento

11-14 agosto 2008

Acumuladores Criptográficos

Philippe Camacho

Universidad de Chile

Plan

- Conceptos básicos
- Motivación
- Construcciones
  - RSA, Funciones de Hash, Pairings
- Conclusión
  - Resumen
  - Preguntas abiertas

Noción de acumulador

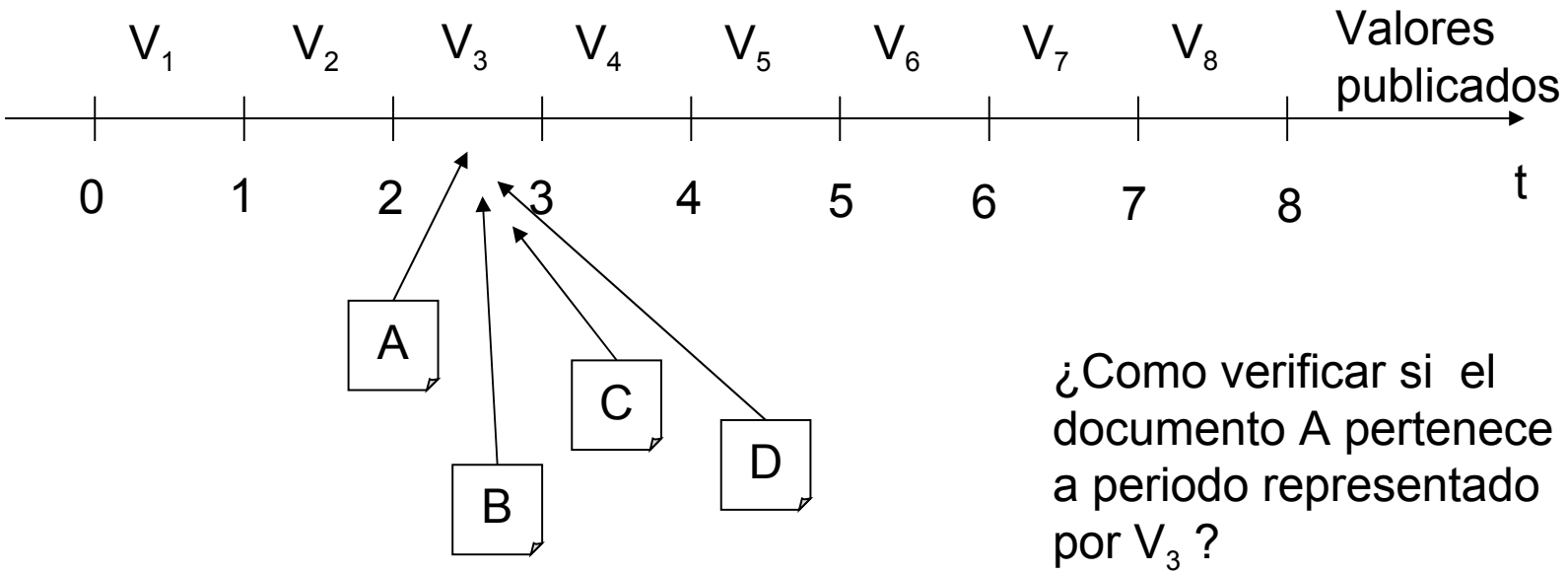
- Problema:
  - Un conjunto X
  - Mostrar que un cierto elemento x pertenece (o no) a X
- Tenemos un conjunto $X=\{x_1,x_2,\ldots,x_n\}$
  - Representado un valor corto Acc
  - Usando Acc y un valor testigo T, y un algoritmo de verificación, se puede asegurar que x, ha sido acumulado (o no) en X.

Tipos de acumuladores

- **Estáticos**
  - El conjunto no cambia.
- **Dinámicos**
  - Se permite insertar y borrar elementos del conjunto.
- **Débiles**
  - requieren que el participante que manipula (calcular valor acumulador, valores testigos, insertar, borrar) el acumulador (dueño) sea confiable.
- **Fuertes**
  - no requiere confiar en el dueño
- **Universales**:
  - Permiten demostrar que un elemento *pertenece o no* a un conjunto.

Motivación

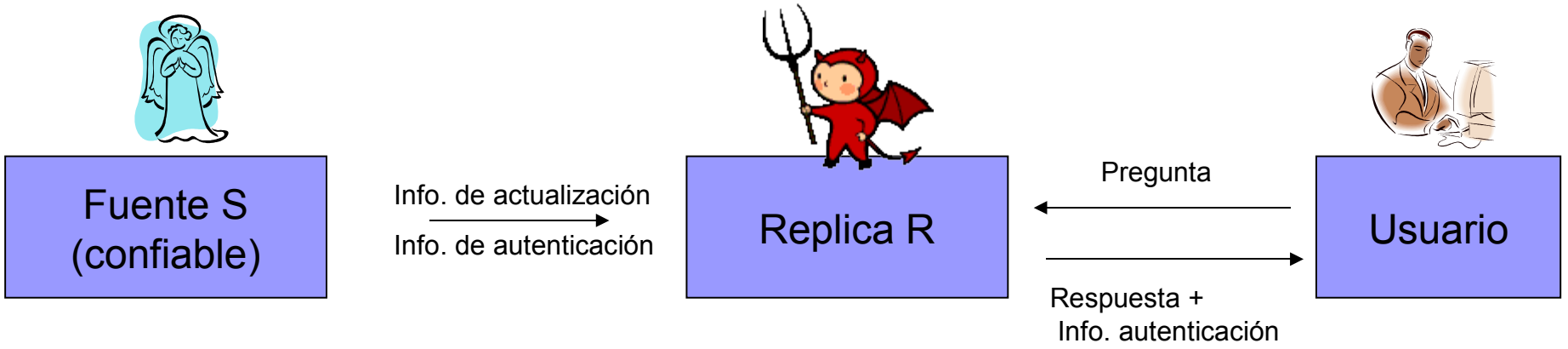
- Time-Stamping
  - Cada documento llega en un periodo de tiempo
  - La Autoridad de Time-Stamping publica un valor (corto) para cada periodo V_i



¿Como verificar si el documento A pertenece a periodo representado por V_3 ?

Motivación

- Diccionarios autenticables [GoTa01]



El usuario quiere asegurase que la información entregada por R es la misma que la entregada por S.

Motivación

- Listas negras
  - Gente que están en bancarrota
    - Ej: empresa evaluadora de morosidad de clientes
  - Lista de pacientes con enfermedades contagiosas
  - Lista de revocación de certificados
    - Uno quisiera demostrar que no pertenece a esa lista (de manera eficiente).

Herramientas

- Funciones de Hash Resistentes a Colisiones

$$H: \{0,1\}^* \rightarrow \{0,1\}^k$$

  - H es computable eficientemente
  - Resistencia a colisiones
    - Dado $y=H(x)$, es difícil encontrar x
    - Es difícil encontrar x,x' con $x \neq x'$ tal que $H(x)=H(x')$
    - Dado x, es difícil encontrar $x' \neq x$ tal que $H(x)=H(x')$
- OJO: noción intuitiva, la definición formal requiere hablar de familias de funciones.

Herramientas

- Random Oracle Model
  - Es una heurística que consiste en considerar que una función de hash resistente a colisiones se comporta como una *función escogida al azar.*
  - Usado en muchas construcciones, pero solamente una heurística.
    - Ver discusión en [CGH98]

Herramientas

- RSA
  - Inicialización
    - N=pq , p y q números primos “seguros” (safe)
    - e: exponente para cifrar, d: exponente para descifrar
    - $ed = 1 \bmod \Phi(N)$ $(\Phi(N) = (p-1)(q-1) = |Z_N^*|)$ (*Algoritmo de Euclides Extendido*)
    - x texto claro
      - Cifrar: $x^e \bmod N$
      - Descifrar: $y = x^e$, $y^d \bmod N = x^{ed} \bmod N = x \bmod N$

Herramientas

- RSA: Supuestos computacionales
  - Supuesto RSA (*RSA Assumption*):
    - Dados $y=x^e \bmod N$, y e (x aleatorio)
    - Difícil encontrar x
  - Supuesto fuerte RSA (*Strong RSA Assumption, [BarPfi97]*):
    - Dado y aleatorio
    - Difícil encontrar x,e con $1<e<\Phi(N)$ tal que:
      - $x^e = y \bmod N$
  - Strong RSA => RSA => Factoring

Construcciones

One-way Accumulators [BenMa94] (1)

- Primera noción de acumulador
  - Estático
  - No universal
  - "Fuerte"
    - [Sand99] Algoritmo probabilista para calcular N que sea de la forma N=pq con alta probabilidad (sin conocer p,q)
    - Computación Multipartita Segura
- Función *unidireccional pseudo-conmutativa*
  - Dados z e y aleatorios, es difícil calcular x tal que F(x,y)=z
  - F(F(x,y),z) = F(F(x,z),y)

Construcciones

One-way Accumulators [BenMa94] (2)

- Usar F para acumular valores
  - $X=\{x_1,x_2,\ldots,x_n\}$, u valor arbitrario inicial
  - $u, x_1 \rightarrow F(u,x_1) \rightarrow F(F(u,x_1),x_2) \rightarrow F(F(F(u,x_1),x_2),x_3) \rightarrow \ldots$
  - $Acc = F(F(\ldots F(F(u, x_1), x_2),\ldots), x_n)$
- Propiedades:
  - El valor acumulado no depende del orden de inserción de los elementos.
  - La verificación (x está en X) se puede hacer en tiempo constante
    - $Acc = F(F(\ldots F(F(u, x_1), x_2), \ldots, x, \ldots, x_{n-1}), x_n)$

      $= F(\underbrace{F(\ldots F(F(u, x_1), x_2), \ldots, x_n, \ldots, x_{n-1})}_{T}, x)$
    - T es el valor Testigo ssi F(T,x)=Acc

Construcciones

One-way Accumulators [BenMa94] (3)

Construcción usando RSA

- Inicialización
  - $N = pq$
  - $F(x,y) = x^y \bmod N$
- Conjunto:
  - $X=\{x_1,x_2,\ldots,x_n\}$
- Valor inicial aleatorio
  - $u \in Z_N^*$
- Valor acumulado
  - $Acc=u^{x_1.x_2\ldots x_n}$
- Testigo para x_i
  - $T= u^{x_1\ldots x_{i-1}.x_{i+1}\ldots x_n}$
- Verificación
  - $T^{x_i} = Acc$

Construcciones

One-way Accumulators [BenMa94] (4)

- Seguridad
  - El atacante trata de encontrar falsos testigos de pertenencia (colisión)
  - Problema
    - X = {5,6}
    - Es muy fácil encontrar una colisión
      - Acc = $u^{5.6}$ (6 ha sido acumulado, u^5 es un testigo)
        = $u^{5.3.2}$ (¡2 ha sido acumulado, u^{15} es un testigo!)
    - Solución (heurística)
      - acumular valores de Hash (*Random Oracle Model*)
      - Dados x,y es “poco probable” que H(xy) = H(x)H(y)

Construcciones

Collision-Free Accumulators [BarPfi97]

- Misma idea que [BenMa94] pero mejora de la definición y usa un nuevo supuesto de complejidad
  - Definición:
    - Formaliza noción de resistencia a colisiones.
    - El Adversario puede escoger los valores a acumular.
  - Construcción
    - basada en la “Supuesto fuerte RSA” (*Strong RSA Assumption*)
  - Idea de la construcción
    - Mapear elementos a números primos, y acumular solamente números primos, así se evitan las colisiones.

Construcciones

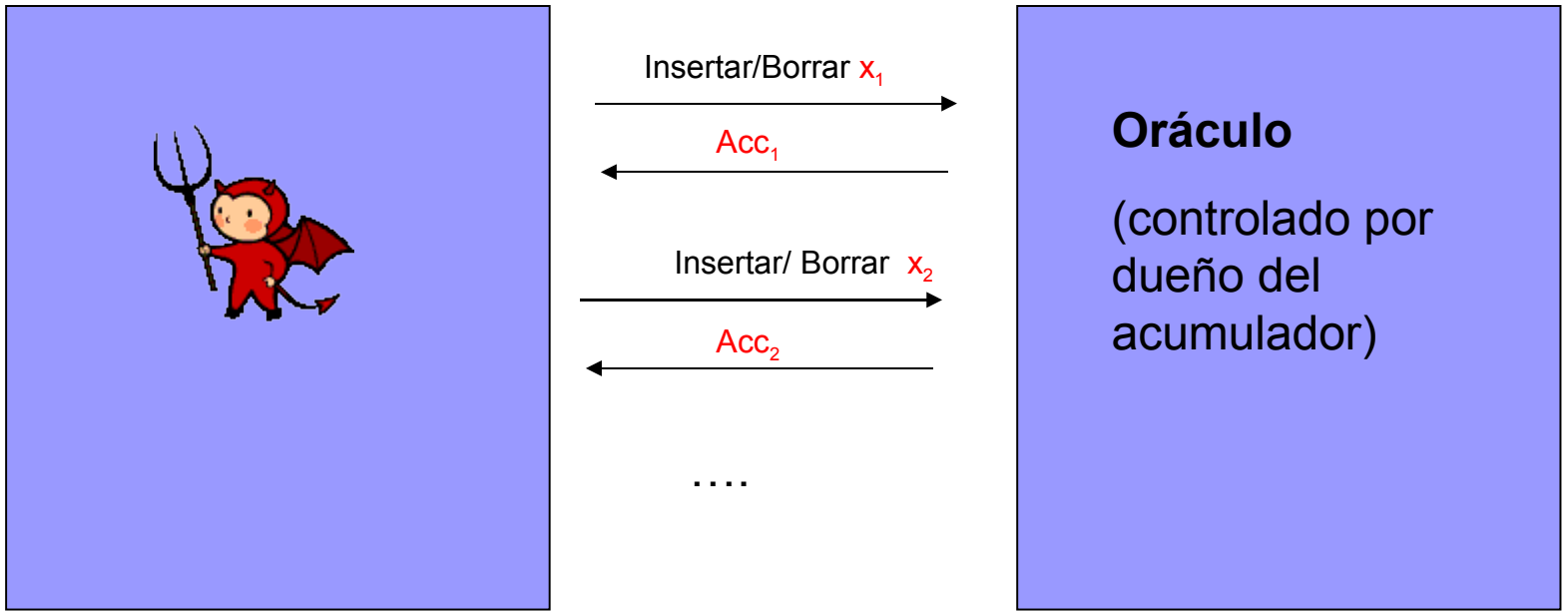
Dynamic Accumulators [CamLys02] (1)

- Acumuladores anteriores son estáticos
  - El conjunto X no puede disminuir
- Acumulador dinámico:
  - Agregar elemento
  - Borrar elemento
  - Recalcular valor acumulado y valor testigo para cada elemento
- Propiedades
  - Dinámico
  - Débil
  - No universal

Construcciones

Dynamic Accumulators [CamLys02] (2)

- Modelo de seguridad



Adversario gana si logra encontrar w,x tal que:

x no está en X y Verificar(x,w,Acc) = OK

Construcciones

Dynamic Accumulators [CamLys02] (3)

- Construcción
  - Similar a [BarPfi97]: basada en “Strong RSA Assumption”
    - Se mapean los elementos a acumular a números primos.
  - Para borrar elementos eficientemente
    - Dueño conoce $\Phi(N)$
    - $X=\{x_1,x_2,\ldots,x_n\}$ (x_i primos), se quiere borrar x:
      - $Acc = u^{x_1 \cdot x_2 \cdot \ldots \cdot x \ldots x_n} \bmod N$
      - Calcular $y=x^{-1} \bmod \Phi(N)$
      - $Acc_{new} = Acc^{1/x} = Acc^{y}$
    - El dueño del acumulador debe ser confiable sino puede calcular testigo (falsos) para cualquier elemento x:
      - $T=Acc^{1/x}$

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (1)

- Primer Acumulador Universal
  - Permite mostrar que un elemento pertenece *o no* al conjunto
- Basado en [CamLys02]
  - Dinámico
  - Débil

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (2)

Lema: Sea N un entero, dados $u,v \in Z_N^*$ y $a,b \in Z$ tal que $u^a = v^b \bmod N$ y $\gcd(a,b) = 1$, se puede calcular eficientemente $x \in Z_N^*$ tal que $x^a = v \bmod N$.

Demostración:

$\gcd(a,b)=1 \Rightarrow bd = 1 + ac$

$x = u^d v^{-c} \Rightarrow x^a = u^{da} v^{-ca} = (u^a)^d v^{-ca} = v^{bd} v^{-ca} = v$

Construcciones

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (3)

- Testigo de pertenencia (ídem [CamLys02])
  - Definición
    - $Acc = u^{x_1 \cdot x_2 \cdot \ldots \cdot x \ldots \cdot x_n} \bmod N = u^v \bmod N$
    - $T = u^{v/x} \bmod N$
  - Calculo de T
    - Usando Euclides Extendido => requiere conocer factorización de N para que sea eficiente.
  - Chequeo
    - $T^x \bmod N = Acc$

Construcciones

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (4)

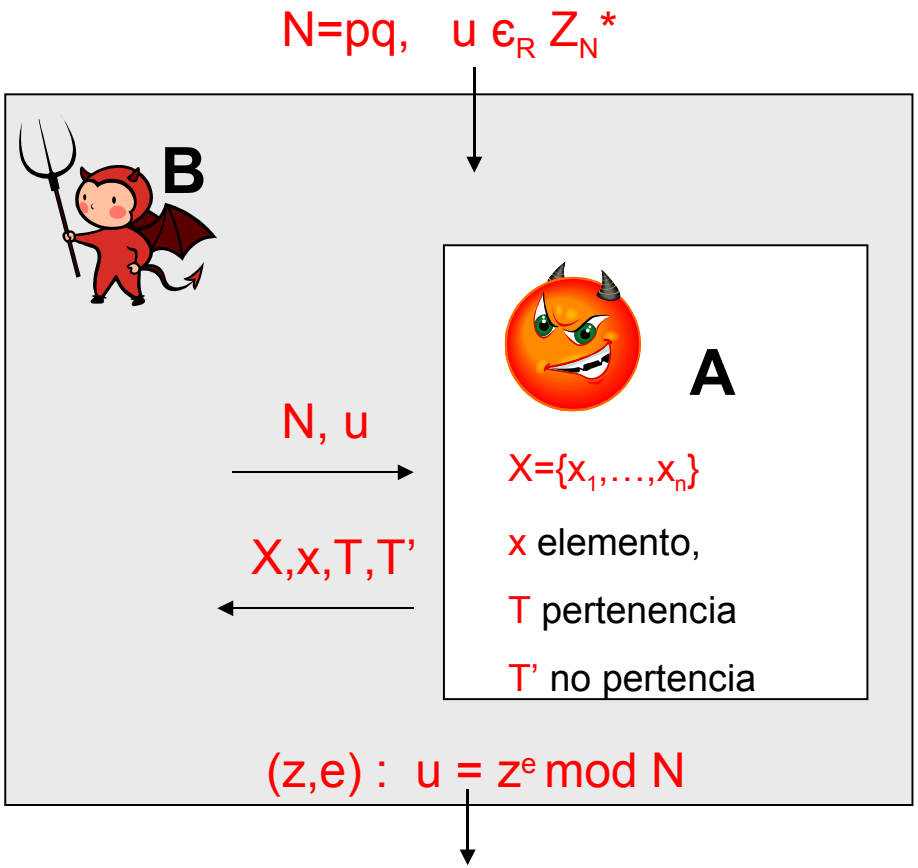
- Testigo de no pertenencia
  - Definición
    - $Acc = u^{x_1 \cdot x_2 \cdot \ldots x_n} \bmod N = u^v \bmod N$
    - $T = (a,d)$ tal que $Acc^a = d^x \cdot u \bmod N$
  - Calculo de T
    - x no está en X => $\gcd(v,x)=1$
      => Calcular (a,b): $av + bx = 1$
    - Calcular $b' = b \bmod \Phi(N)$
      - $T=(a, u^{-b'})$
  - Chequeo
    - $Acc^a = u^{va} \bmod N = u^{1-bx} \bmod N = (u^{-b'})^x.u$

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (5)

- **Teorema**
  - Bajo el Supuesto fuerte RSA, la construcción es segura.
- **Demostración**
  - Seguridad:
    - Dado $X=\{x_1,x_2,\ldots,x_n\}$ y x escogidos por el adversario
      - Si $x \in X$: difícil encontrar un testigo de no pertenencia.
      - Si $x \notin X$: difícil encontrar un testigo de pertenencia.
    - De manera equivalente
      - Encontrar un testigo de pertenencia y no pertenencia para un cierto x

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (6)

- Técnica: reducción



Si existe un adversario **A** capaz de quebrar nuestro esquema

Podemos construir un adversario **B** capaz de quebrar el Supuesto fuerte RSA.

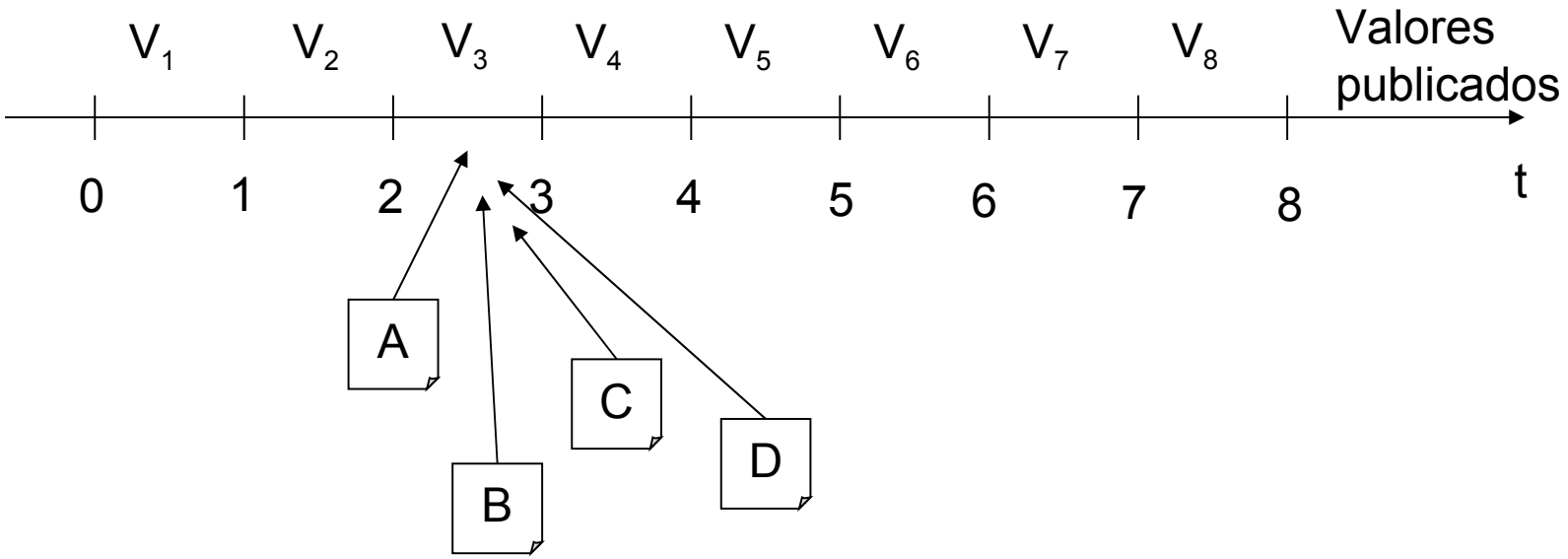
Construcciones

Universal Accumulators with Efficient Nonmembership Proofs [LLX07] (6)

- $X = \{x_1,\ldots,x_n\}$, x , $Acc = u^{x_1 \cdots x_n} = u^v$
- Testigo pertenencia T
  - $T^x = Acc$
- Testigo no pertenencia T'
  - $T' = (a,d)$, $Acc^a = d^x\, u \Leftrightarrow u^{va-1} = d^x$
- Si x está en X
  - $x \mid v$, $\gcd(av-1,x) = 1$, por el lema se puede encontrar z tal que $z^x = u$, $e=x$
- Si x no está X
  - $\gcd(v,x) = 1$ y $T^x=u^v$ => por el lema se puede encontrar z tal que $z^x = u$, $e=x$

Recuerdo

- Time-Stamping

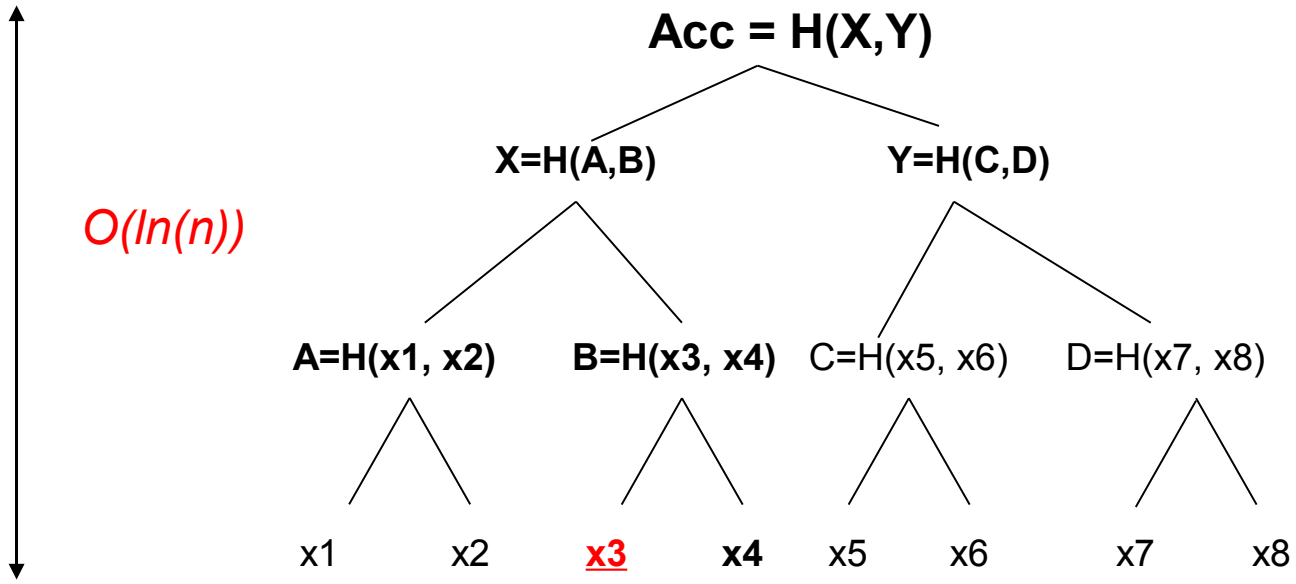


Efficient Broadcast Time-Stamping [BeMa92] (1)

- Primera solución (ineficiente)
  - Concatenar documentos en $V=x_1||x_2||\ldots||x_n$
  - $H(V)=Acc$ es el valor acumulado
  - Verificación
    - usando la lista $x_1,\ldots,x_n$ y recalculando Acc
- Problema
  - Tiempo de verificación $O(n)$ => ineficiente
- Nota
  - Todavía (1992) la noción de acumulador no ha sido planteada. Pero aparece la idea.

Efficient Broadcast Time-Stamping [BeMa92] (2)

- Solución más eficiente
  - Usando árboles de Merkle



Construcciones

Fast Accumulated Hashing [Nyb96]

- Construcción interesante de punto de vista teórico
  - No requiere el uso de un tercero confiable (Acumulador Fuerte)
  - Basada en funciones de Hash y “Random Oracle”
- Más eficiente que la solución ingenua
  - firmar cada elemento de una lista enlazada
- Pero no mucho más…
  - Espacio necesario (valor acumulado): nlog(n)
- **Interés práctico limitado.**

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (1)

- Primer acumulador dinámico, fuerte y universal
- Eficiencia no optimal pero razonable
  - Tamaño de testigos en O(ln(n))
- Idea construcción
  - Árboles de Merkle
  - Almacenar intervalos consecutivos de los elementos
    - X={1,2,4,5}
    - Almacenar (-∞,1), (1,2) , (2,4) , (4,5) , (5, ∞)

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)

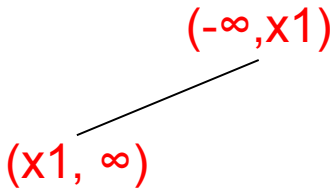
$(-\infty,\infty)$

$X=\emptyset$

Elemento a insertar: x_1

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)

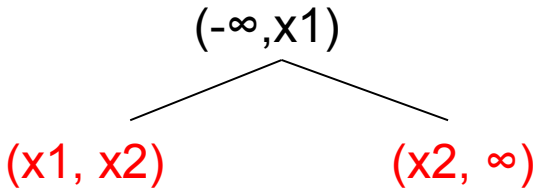


X={x1}

Elemento a insertar: x2

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)

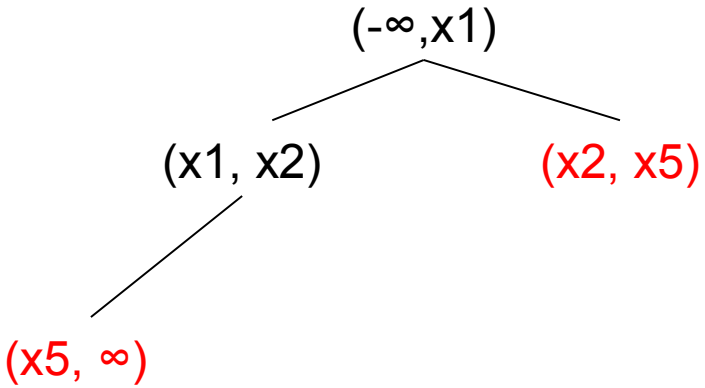


X={x1,x2}

Elemento a insertar: x5

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)

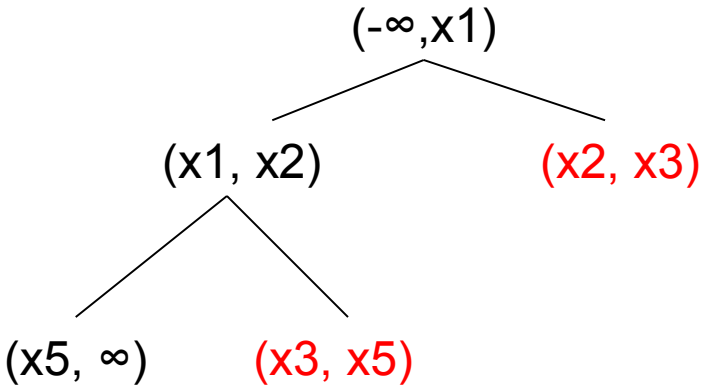


X={x1,x2,x5}

Elemento a insertar: x3

Construcciones

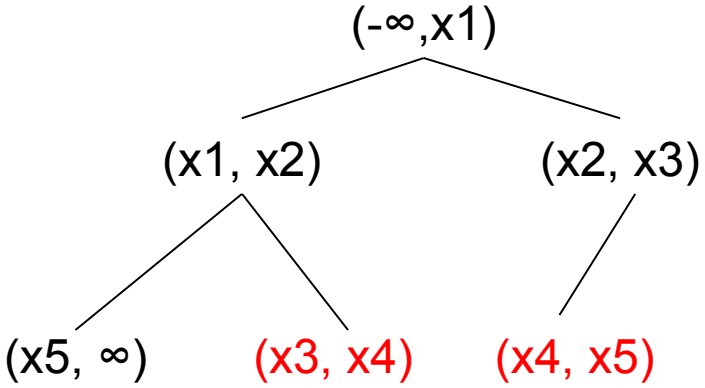
Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)



X={x1,x2,x3,x5},
Elemento a insertar: x4

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)

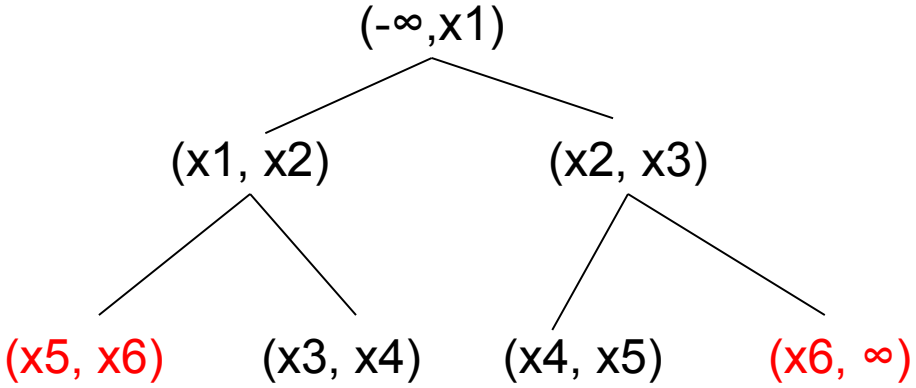


X={x1,x2,x3,x4,x5}

Elemento a insertar: x6

Construcciones

Strong Accumulators from Collision-Resistant Hashing [CKHO08] (2)

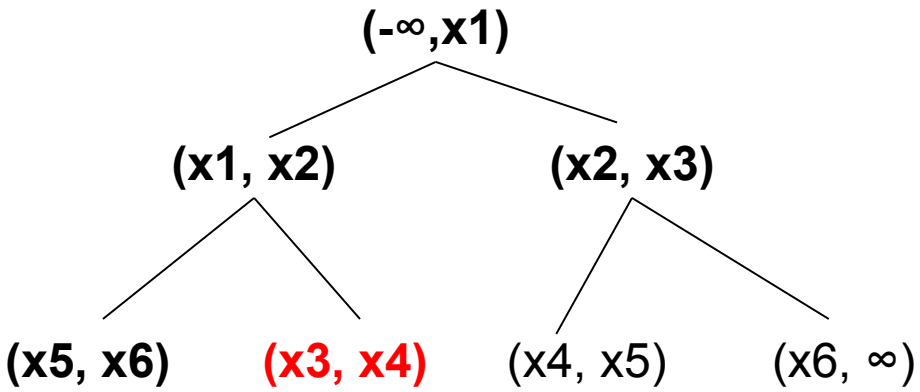


X={x1,x2,x3,x4,x5,x6}

Valor acumulado; H((a,b),izq,der)

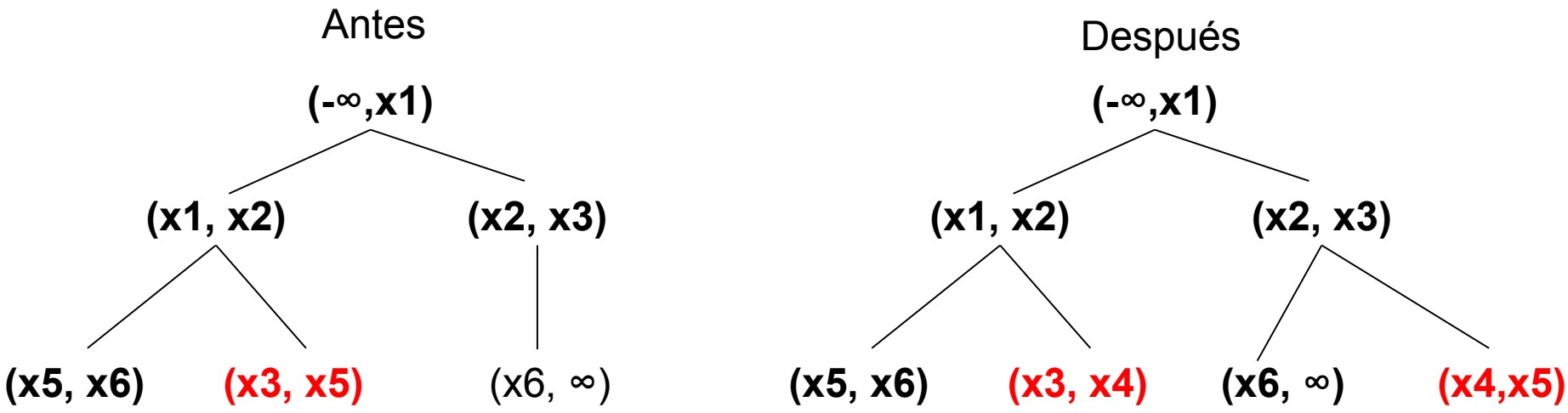
Strong Accumulators from Collision-Resistant Hashing [CKHO08] (3)

- Testigos



Strong Accumulators from Collision-Resistant Hashing [CKHO08] (4)

- Testigos de actualización (inserción)



Inserción de x4

Construcciones

Bounded Accumulators [AWSM07]

- Primera construcción en usar Pairings
  - Sirve para resolver un problema de Dinero Electrónico (E-Cash).
  - Se trata de acumuladores que pueden almacenar una cantidad limitada de elementos.

Dynamic Accumulator for Batch Updates [WWP08]

- Construcciones anteriores:
  - Cuando los usuarios quieren actualizar sus testigos
    - Preguntar al dueño del acumulador.
    - O recalcular los testigos después de cada operación de inserción / borrado.
  - Solución
    - Tiempo para actualizar los testigos, independientes de la cantidad de inserción / borrados => *Batch Update.*
- Solución sofisticada basada en el criptosistema de Paillier.

Resumen

	Dinámico	Fuerte	Universal	Seguridad	Eficiencia (tamaño testigo)	Nota
[BeMa92]	✗	✓	✗	**Collision-Resistant Hashing**	**O(ln(n))**	-
[BeMa94]	✗	✓*	✗	**RSA + RO**	**O(1)**	Primera definición
[BarPfi97]	✗	✓	✗	**Strong RSA**	**O(1)**	-
[CamLys02]	✓	✗	✗	**Strong RSA**	**O(1)**	Primer acumulador dinámico
[LLX07]	✓	✗	✓	**Strong RSA**	**O(1)**	Primer acumulador universal
[AWSM07]	✓	✗	✗	**Pairings**	**O(1)**	E-cash
[CKHO08]	✓	✓	✓	**Collision-Resistant Hashing**	**O(ln(n))**	Dinámico, universal, fuerte
[WWP08]	✓	✗	✗	**eStrong RSA Paillier**	**O(1)**	Batch Update

(*) Usando MPC o [San99].

Algunas preguntas abiertas

- Aspectos fundacionales
  - Función unidireccional pseudo-conmutativa => **?**
- Costo de la confianza
  - Construir un acumulador **fuerte**, **dinámico** y **universal** con testigos de tamaño inferior a O(ln(n))
- Búsquedas más sofisticadas
  - ¿Elementos de X que estan entre 20 y 40?
    - ¿Que pasa con el tamaño del testigo?
  - Búsqueda de patrones (expresiones regulares) …

¡Gracias!

http://www.dcc.uchile.cl/~pcamacho/presentations/uruguay08.ppt

Bibliografía

- **[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
- **[CGH98]** The random oracle methodoly 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
- **[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, Roberto Opazo* 2008