Accountable Certificate Management using Undeniable Attestations

Ahto Buldas
Küberneetika AS (Estonia) & U. of Tartu (Estonia)
ahto.buldas@cyber.ee

Peeter Laud
Universität des Saarlandes (Germany)
laud@cs.uni-sb.de

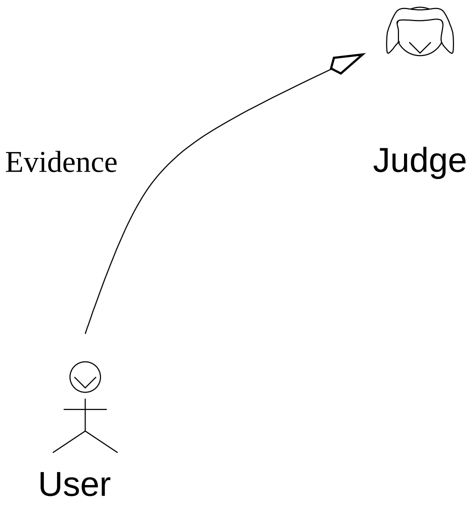
Helger Lipmaa
Helsinki U. of Technology (Finland) & U. of Tartu (Estonia)
helger@tml.hut.fi

Motivations

- Our main motivation: law and order[*] with help of digital signatures
  - For this one needs certificate management
- For law and order one needs the court
  - Court = our roots
- Let us look at what happens in court . . .

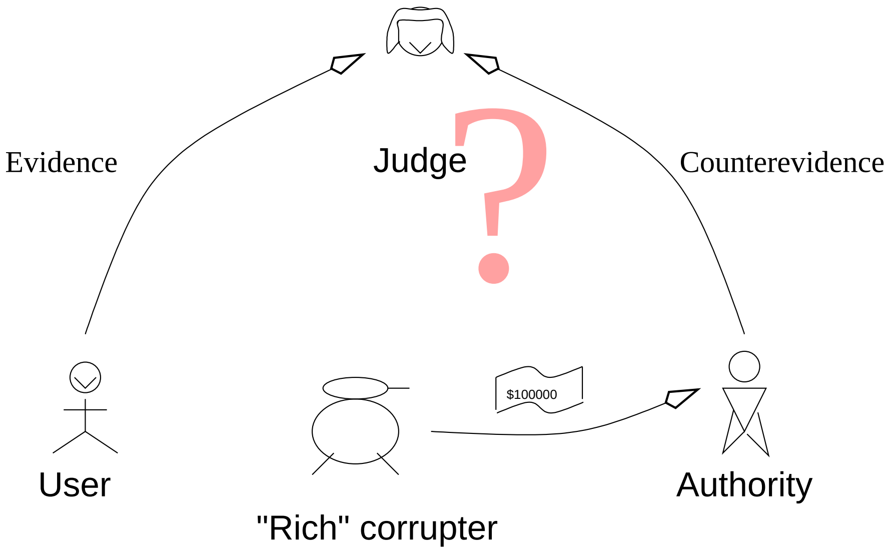
[*] Warning! This is continuation of work, originally financed by lawyers from Estonian government

We are now in court . . . 1



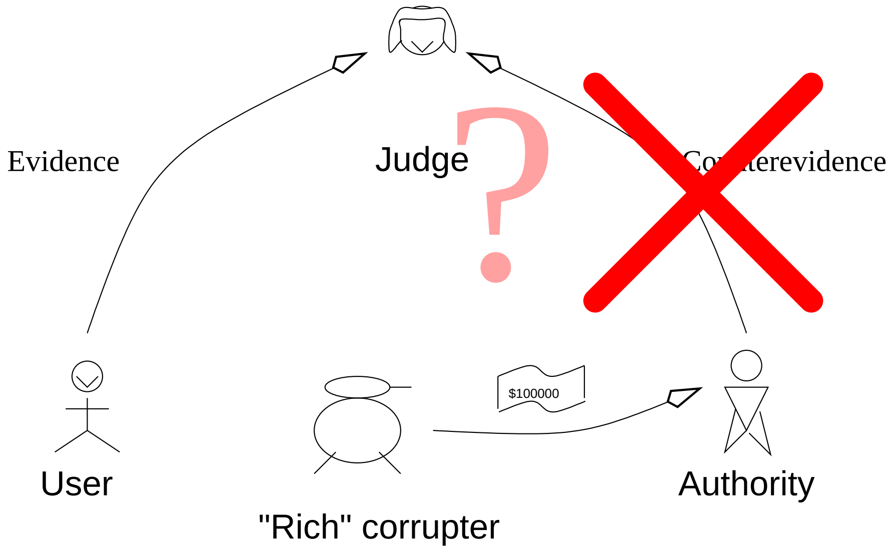
- Can the judge solve the case, given an evidence?

We are now in court . . . 2



- Mostly not! Somebody could create a counter-evidence

We are now in court . . . 3



- Solution: make creating of counter-evidence impossible!

Accountable Certificate Management (ACM)

- *Undeniability* = no possibility of “counter-evidence”:
  - ⋆ If a certificate was valid, nobody can “attest” it was invalid (no false negatives)
  - ⋆ If a certificate was invalid, nobody can “attest” it was valid (no false positives)
- In ACM, certificates are accompanied with *undeniable attestations*

Model of Accountable Certificate Management 1

- The CA maintains the database S of valid certificates

- Certificate issuing and removal procedures are observed by a notary

  - ⋆ Other operations should not be (nor are) audited!

- Certificate x is accompanied by undeniable attestation $P(x, S)$ of status of $x \overset{?}{\in} S$

- For their own sake, clients should store the attestations (“evidence”)

Model of Accountable Certificate Management 2

- After the end of current round, *digest* $D(S)$ of the database is published in “New York Times”
  - $\star$ In many ways, model is the same as in time-stamping!
  - $\star$ E.g., we do not use public-key cryptography

- Verifier obtains certificate x, digest d and attestation p.
  - $\star$ $V(x, d, p) \overset{?}{=}$ Accept.

Undeniable Attesters

- Attester = triple (P, D, V) of efficient algorithms.

- For "correct" inputs x, $D(S)$, $P(x, S)$:

$$V(x, D(S), P(x, S)) = \text{Accept} \iff x \in S$$

- Attester is *undeniable* if it is intractable to create a tuple $(x, d, p, \overline{p})$, s.t. $V(x, d, p) =$ Accept but $V(x, d, \overline{p}) =$ Reject.

- That is, in court, (x, d, p) is an evidence s.t. there does not exist counter-evidence.

Some examples

List Take $P(x, S) = S$, $D(S) = h(S)$:

$V(x, d, S) =$ Accept iff $x \in S$ and $d = h(S)$

Inefficient if $|S| \geq 10^3$. (Similar to CRLs!)

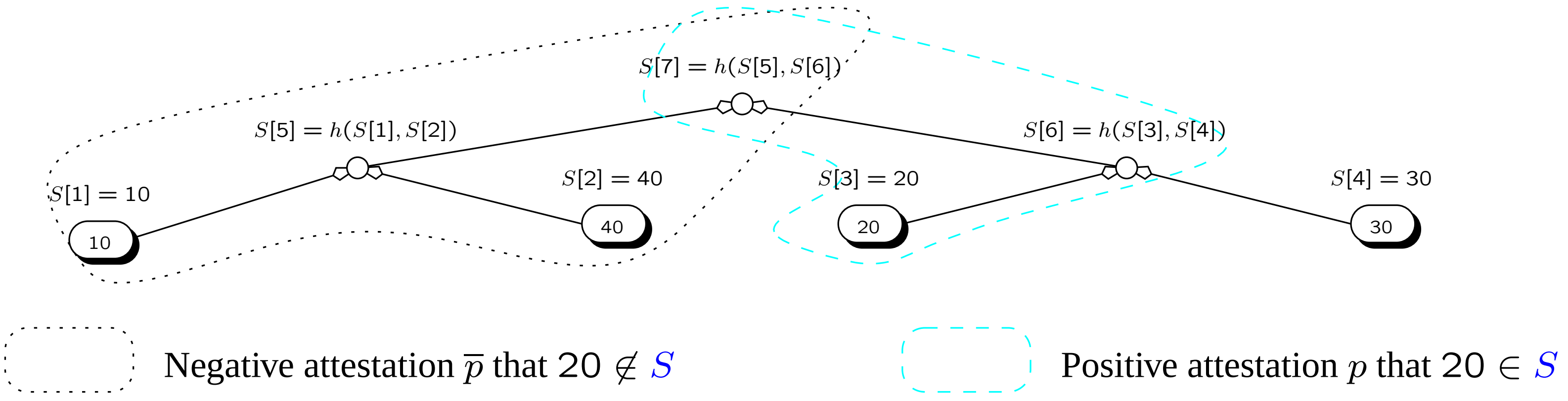
Hash Tree Can prove efficiently that $x \in S$, but not that $x \notin S$. (Similar to Merkle's hash trees)

Sorted Hash Tree (Similar to CRTs) Can do both efficiently . . .

- but it is tractable to create counter-evidence!

- Where does the sorted hash tree fail?

Sorted Hash Tree

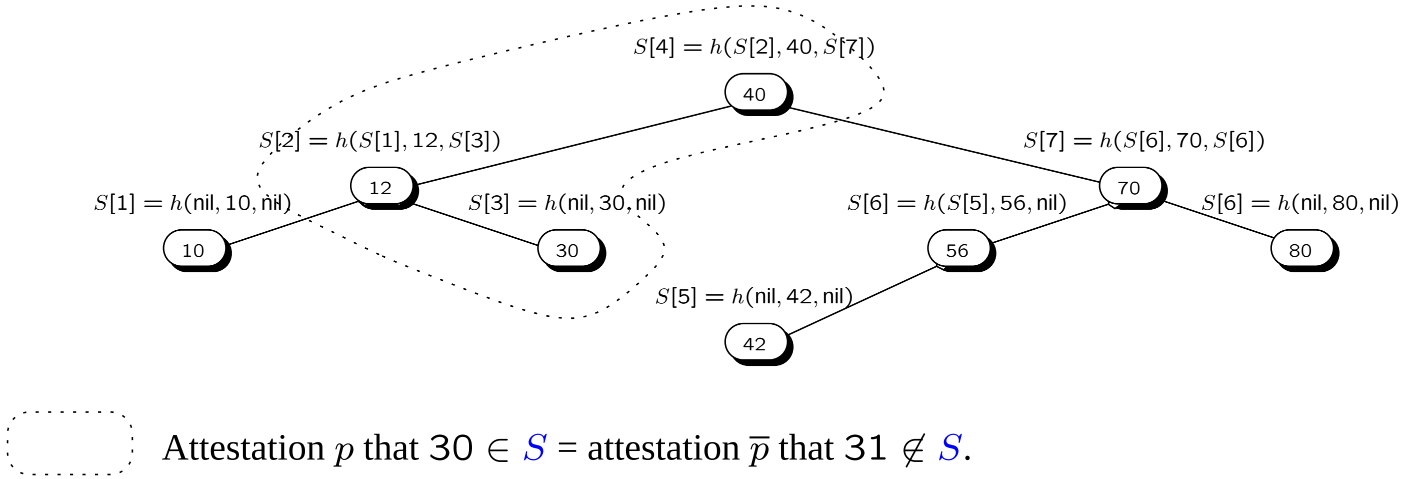


Negative attestation $\overline{p}$ that $20 \notin S$

Positive attestation p that $20 \in S$

- The CA can leave the tree unsorted!
- Tracing this would need access to whole S
- We need more efficient way of detecting the “non-sorting attack”

Our Solution: Authenticated Search Trees



$S[4] = h(S[2], 40, S[7])$
40
$S[2] = h(S[1], 12, S[3])$
12
$S[7] = h(S[6], 70, S[6])$
70
$S[1] = h(\text{nil}, 10, \text{nil})$
10
$S[3] = h(\text{nil}, 30, \text{nil})$
30
$S[6] = h(S[5], 56, \text{nil})$
56
$S[6] = h(\text{nil}, 80, \text{nil})$
80
$S[5] = h(\text{nil}, 42, \text{nil})$
42

Attestation p that $30 \in S$ = attestation $\overline{p}$ that $31 \notin S$.

- $\forall$ node v is associated with $K[v] \in S$; $S[v] = h(S[v_\ell], \underline{\underline{K[v],}}S[v_r])$
- If v' is in left subtree of v then $K[v'] < K[v]$
  If v' is in right subtree of v then $K[v'] > K[v]$

Security Analysis

Theorem If h is a CRHF then authenticated search tree attester is undeniable.

Proof Idea. Doing local verifications is sufficient!

Theorem If an undeniable attester exists then there exists also a CRHF.

Comparison

Method	Attestation length	$k = 160$, $\|S\| = 10^7$
List	$k\|S\|$	191 MB
Ours	$2k \log_2 \|S\|$	930 B
Gain:	$\frac{\|S\|}{2 \log_2 \|S\|}$	$> 200,000$ times

- Our solution is $200,000$ times more efficient than the list attester :-)

- The sorted hash tree attester has still twice shorter attestations :-(

More about Efficiency

Attestations can be compressed by *standard* compression methods, such that the worst case attestation length is $k(n+1)+\frac{n^2+n}{2}$, where in practice $n = \log_2 |S| \ll \sqrt{k}$.

Method	Attestation length	$k = 160$, $\lvert S\rvert = 10^7$
List	$k \cdot 2^n$	191 MB
Ours	$2kn$	930 B
Ours (compressed)	$k(n+1)+\frac{n^2+n}{2}$	520 B
SHT (insecure)	kn	465 B

Conclusions

- New model for accountable certificate management
  - ⋆ $\boxed{\text{It should be intractable to create counter-evidence!}}$
- Security of our model $\Leftarrow$ security of new primitive, *undeniable attester*
- We proposed an efficient construction of the latter
- New methods in cryptography:
  - ⋆ authenticated search trees
  - ⋆ standard compression methods

More information

- Webpage:
  - ⋆ `http://www.tml.hut.fi/˜helger/cuculus`

- Email me (`helger@tml.hut.fi`)

- Or ask here (now or later)!