A Note On SPKI’s Authorisation Syntax

Olav Bandmann*
Industrilogik L4i AB

Mads Dam
KTH/IMIT

*Work done while at SICS, Swedish Institute of Computer Science. Supported by Microsoft Research, Cambridge
SPKI

- SPKI: Approach based on binding names and authorisations to keys
- SPKI authorisation certificate

\[(\text{issuer, subject, propagate, tag, validity})\]

- Tags (= authorisation expressions) given in special S-expression (LISP-like) syntax
Tuple Reduction

- Decisions resolved by *tuple reduction*
  - Cert vs cert
  - Request vs cert
- For tags, compute
  \[
  Z = A\text{Intersect}(X,Y),
  \]
  the most inclusive authorisation granted by both \(X\) and \(Y\)
Contributions

• Problem: AIIntersect is not suitable for space/time critical applications

• In this paper: Restricted syntax which
  – Conforms with SPKI ”practice” (we think)
  – Has $n \log n$ procedure

• Other contributions:
  – Authorisation preorder $\leq$
  – Sound and complete axiomatisation for
    • Standard syntax
    • Restricted syntax
Objections

• But:
  – SPKI authorisation expressions (tags) are small
  – Requests do not involve the set construction

• It depends...
  – For hand-crafted certs and standard usage, maybe so
  – For e.g. macros, precomputation, richer delegation logics, maybe not

• More on this later
Authorisation Trees

Rivest S-expressions – example:

\[ x = (\text{object person} (\text{conds} (\text{group } "\text{admin}"") (\text{unit } "\text{finance}")) (\text{op income read})) \]

Authorisation and request in same syntax:

\[ y = (\text{object person} (\text{conds} (\text{group } "\text{admin}")) (\text{op income read})) \]
\[ z = (\text{object person} (\text{conds} (\text{group } "\text{admin}")) (\text{unit } "\text{finance}")) (\text{op income})) \]

Both \( y \) and \( z \) would grant \( x \)

- "Being authorised by" = lists are extended to right
Authorisation Order

Authorisation trees:

\[ t ::= a \mid (a \ t_1 \ldots \ t_n) \]

where \( a \) is an atom, \( n \geq 1 \)

Authorisation order \( t_1 \leq t_2 \), \( t_1 \) authorised by \( t_2 \):

- \( a \leq t \) iff \( t \leq a \) iff \( t = a \)
- \( (x_1 \ldots x_n) \leq (y_1 \ldots y_m) \) iff \( n \geq m \) and \( x_i \leq y_i \) \((1 \leq i \leq m)\)

Can show that:

\[ t_1 = \text{AIntersection}(t_1, t_2) \iff t_1 \leq t_2 \]
Star Forms

* forms abbreviate sets of S-expressions:
  – (*) : The wildcard
  – (* set $X_1 ... X_n$): Union of $X_1,...,X_n$
  – (* range $<order> <lower> <upper>$)
  – (* prefix $<string>$)

Example:

$$t = (\text{object person (conds (group "admin")
  (* set (unit "finance") (type "Managers")))
  (op income (* set read write)))}$$
S-Expressions

Set constants $b$, $Val(b)$ nonempty set of atoms

$$X ::= (*) | a | b | (a \ X_1 \ldots \ X_n) | (* \text{ set } X_1 \ldots X_m)$$

Semantics:

- $\| b \| = Val(b)$
- $\| (X_1 \ldots X_n) \| = \{(t_1 \ldots t_k) | k \geq n, \forall i:1 \leq i \leq n \ t_i \in \| X_i \|\}$
- $\| (* \text{ set } X_1 \ldots X_m) \| = \| X_1 \| \cup \ldots \cup \| X_m \|$

Obs: $\| X \| \text{ is lower closed}: \ t_1 \leq t_2 \in \| X \| \Rightarrow t_1 \in \| X \|$
S-Expression Preorder

Def. $X \leq Y$ iff $\|| X || \subseteq || Y ||$

$\|| . ||$ not suitable for algorithm
Computes all paths through a tree
Set constants (range, prefix) may give infinite sets

Task: Decide - without computing $\|| . ||$:
• Given $t$ and $X$ is $t \in X$?
• Given $X$ and $Y$, is $X \leq Y$?
Weak Preorder

Let $X \leq_w Y$ iff one of:

1 - 4. ...

5. $X=b_1$, $Y=b_2$ and $Val(b_1) \subseteq Val(b_2)$

6. $X = (X_1 \ldots X_m)$, $Y = (Y_1 \ldots Y_n)$, $m \geq n$, and $X_i \leq_w Y_i$, $1 \leq i \leq m$

7. $X = (\ast \text{ set } X_1 \ldots X_m)$, $X_i \leq_w Y$ for all $1 \leq i \leq m$

8. $X = b$, $Y = (\ast \text{ set } \ldots)$ and ...

9. $X$ not $b$ nor (\ast \text{ set } \ldots) form, $Y = (\ast \text{ set } Y_1 \ldots Y_n)$ and $X \leq_w Y_i$ for some $i$: $1 \leq i \leq n$
Weak Preorder

Let $X \leq_w Y$ iff one of:

1 - 4. ...

5. $X=b_1$, $Y=b_2$ and $\text{Val}(b_1) \subseteq \text{Val}(b_2)$

6. $X = (X_1 \ldots X_m)$, $Y = (Y_1 \ldots Y_n)$, $m \geq n$, and $X_i \leq_w Y_i$, $1 \leq i \leq m$

7. $X = (* \text{ set } X_1 \ldots X_m)$, $X_i \leq_w Y$ for all $1 \leq i \leq m$

8. $X = b$, $Y = (* \text{ set } \ldots)$ and ...

9. $X$ not $b$ nor ($* \text{ set } \ldots$) form, $Y = (* \text{ set } Y_1 \ldots Y_n)$ and $X \leq_w Y_i$ for some $i$: $1 \leq i \leq n$
Basic Properties

Results:
• $\leq_w$ is a preorder
• $\leq_w$ is sound, i.e. $X \leq_w Y$ implies $X \leq Y$
• $\leq_w$ is incomplete

Example:

$\left(a\ (*\ set\ b\ c)\right) \leq \left(*\ set\ (a\ b)\ (a\ c)\right)$ but neither
$\left(a\ (*\ set\ b\ c)\right) \leq (a\ b)$ nor $\left(a\ (*\ set\ b\ c)\right) \leq (a\ c)$
(9) Is Problem Case

Replace

9. $X$ not $b$ nor (* set ...) form, $Y = (*$ set $Y_1 \ldots Y_n$) and $X \leq_w Y_i$ for some $i$: $1 \leq i \leq n$

By

ix. $X$ not $b$ nor (* set ...) form, $Y = (*$ set $Y_1 \ldots Y_n$) and $\| X \| \subseteq \| Y \|$ 

Theorem: Preorder with $ix.$ in place of $9.$ is sound and complete w.r.t. $\leq$
Restricted S-Expressions

Non-atomic members of *-set expressions should have unique "tag"

- $r ::= (*) | a | b | (a \ r_1 \ldots \ r_n) | (* \ set \ r^{a_1} \ldots \ r^{a_m})$
- $r^{a} ::= a' | b | (a \ r_1 \ldots \ r_n)$

All $a_i$ must be distinct

Idea: Push "conflicts" to the leaves:

$$(a \ (* \ set \ (b \ c) \ (b \ d) \ b)) \rightarrow (a \ (* \ set \ (b \ (* \ set \ c \ d)) \ b))$$
Restricted S-Expressions, 2

Result:
For the restricted syntax $\leq_w$ is sound and complete:

$$r_1 \leq r_2 \text{ iff } r_1 \leq_w r_2$$

Result:
Any S-expression can be rewritten into equivalent restricted S-expression

By: Eliminating ”tag conflicts” and nested *-sets
AIntersect

Assume the Val(b) closed under intersections

Exploit ”tags” when computing AIntersect (details in paper)

Possible $n \log n$ algorithm: Sort *-set expressions according to tag, then use binary search

Obtain: $\| \text{AIntersect}(r,r') \| = \| r \|$ iff $r \leq r'$
Summary

• Characterisation of SPKI authorisation relation as partial order $\leq$
• Weak version $\leq_w$ of $\leq$
  – Sound, incomplete, $x \leq_w y$ computable in time $O(|x||y|)$
• Restricted S-expression syntax
  – $\leq_w$ complete
  – Appears to reflect SPKI practice
• AIntersect is glb with respect to $\leq$
  – Running time $n \log n$
The Objections

1. Certificates are small
2. Requests do not involve \(*\)-set expressions

Does this hold water?
The Objections, 2

1. What is SPKI practice?
2. SPKI cert-cert reductions can involve *-sets in both arg’s – are they sometimes time critical?
3. Cannot application program-generated certs become quite complex/large?
   - Ex: Request precomputation
   - Ex: Use *-set’s as macros, e.g.
     MidWestLocs =
     (* set ... (loc Nebraska Lincoln) (loc Kansas Topeka Centre) (loc Kansas Topeka North) ... )
Related Work: Delegation

A to delegate authority to B to administer A’s security policies
Delegation, 2

Richer models of delegation
Constrained delegation:
• Explicit issuance of new privileges
• Delegation tree constraints \((a \ b^* \ c)\)
• Stepwise refinement of constraints (requests)
Forthcoming: SPKI + Kleene star

Papers: Sadighi, Sergot, Bandmann - Security protocols 01
Bandmann, Dam, Sadighi - S&P 02