CS 5430: Example of Credentials-Based Authorization

Fred B. Schneider
Samuel B Eckert Professor of Computer Science

Department of Computer Science
Cornell University
Ithaca, New York  14853
U.S.A.
Language:

\[ C ::= F \quad (F \text{ a formula of First-order Predicate Logic}) \]
\[ \quad \mid P \text{ says } C \]
\[ \quad \mid P' \text{ speaksfor } P \]
\[ \quad \mid P' \text{ speaks } x:C \text{ for } P \]
\[ \quad \mid C \land C' \]
\[ \quad \mid C \lor C' \]
\[ \quad \mid C \Rightarrow C' \]

N.b. \( \neg C: (C \Rightarrow \text{false}) \)
Models for CAL

\( \omega(P) \) is the set of beliefs principal P has.

- P says C iff \( C \in \omega(P) \)
- P' speaks for P iff \( \omega(P') \subseteq \omega(P) \)

\( \omega(P) \) called the **worldview** of P
CAL Inference Rules: says

\[
\frac{C}{P \text{ says } C} \quad \frac{P \text{ says } C}{P \text{ says } (P \text{ says } C)} \quad \frac{P \text{ says } (P \text{ says } C)}{P \text{ says } C}
\]

\[
P \text{ says } (C \Rightarrow C') \Rightarrow (P \text{ says } C) \Rightarrow (P \text{ says } C')
\]
CAL Inference Rules: speaksfor

\[ P \text{ says } (P' \text{ speaksfor } P) \]

\[ \therefore P' \text{ speaksfor } P \quad \text{hand-off} \]

\[ P' \text{ speaksfor } P \]

\[ (P' \text{ says } C) \Rightarrow (P \text{ says } C) \]

\[ P \text{ speaksfor } P', P' \text{ speaksfor } P'' \]

\[ \therefore P \text{ speaksfor } P'' \]
Unrestricted Delegation

\[ P' \text{ says } C, \quad \frac{P' \text{ speaks for } P}{(P' \text{ says } C) \Rightarrow (P \text{ says } C)} \quad P \text{ says } C \]

- **Warning:** \( P \) inherits beliefs from any principal that was delegated to.

- \( P \) trusting \( P' \) means
  - \( P \) adopts all beliefs of \( P' \)
  - \( P \) also adopts beliefs of any principal \( P' \) trusts (transitive).
Why Delegate?

Transitivity of delegation allows clients to be ignorant of the implementation details of services the clients invoke.

- Transitive delegations are made by implementation of service to lower-level services.
- Transitive delegations are hidden from clients.
Restricted Delegation

\[ P' \text{ speaks } x: C \text{ for } P \]
\[ \frac{(P' \text{ says } C[x := \tau]) \Rightarrow (P \text{ says } C[x := \tau])}{\text{(Restricted Delegation)}} \]

Example:

\textit{CS says Major(Alice)}

\textit{CS says }\neg\text{Major(Alice)}

\textit{CU says (CS speaks for }CU\text{)}

\textit{CU says (CS speaks }x: \text{Major}(x)\text{ for }CU\text{)}

... CU does not inherit \(\neg\text{Major}(x)\) from CS
Compound Principals

- Every principal $P$ has a worldview $\omega(P)$.

- Compound principals combine worldviews from multiple principals to obtain a worldview for the compound principal.

- Example:
  - $P \land Q$: $\omega(P \land Q) = \omega(P) \cap \omega(Q)$
Useful Compound Principals

- Subprincipals of $P$: $P \cdot x$
- Groups $G = \{ G_1, G_2, \ldots, G_n \}$
Subprincipals

For any term $\eta$:

\[
\begin{align*}
P \text{ speaksfor } P.\eta \\
\eta = \eta' \\
P.\eta \text{ speaksfor } P.\eta'
\end{align*}
\]
Use of Subprincipals

● Any belief of $P$ is attributed to $P \cdot x$ for any $x$.
  – **Hack**: Employ $P \cdot \epsilon$ for beliefs by $P$ that should not be attributed to other sub-principals of $P$.

● If $L$ implements $H$ then $H$ is a subprincipal of $L$.
  – **Example**: HW implements OS, so HW.OS is the principal that corresponds to the operating system.
$L$ implements $H$, so $H$ is a subprincipal of $L$.

- $L$ says $(H$ says $C)$
- $L$ speaksfor $H$

$L$ says $(H$ says $C)$, \[ L \text{ speaksfor } H \]
\[ (L \text{ says } (H \text{ says } C)) \Rightarrow (H \text{ says } (H \text{ says } C)) \]
$L$ implements $H$, so $H$ is a subprincipal of $L$.

- $L$ says ($H$ says $C$)
- $L$ speaks for $H$

$L$ says ($H$ says $C$), \[ L \text{ speaks for } H \]

\[
\frac{(L \text{ says } (H \text{ says } C)) \Rightarrow (H \text{ says } (H \text{ says } C))}{H \text{ says } (H \text{ says } C)}
\]

\[
\frac{H \text{ says } (H \text{ says } C)}{H \text{ says } C}
\]
Group Principals

A **group** is defined by a finite enumeration of its member principals. \( G = \{ P_1, P_2, \ldots P_N \} \)

- Conjunctive Groups

\[
P_i \text{ says } C, \text{ for every } P_i \in G
\]

\[
P_G \text{ says } C
\]

\[
P_G \text{ says } C
\]

\[
P \text{ says } C
\]

\[
P_G \text{ speaks for } P
\]

for \( P \in G \)
Group Principals

- Disjunctive Groups. Hold beliefs that any member principal holds plus deductive closure!

\[
\begin{align*}
& P \text{ says } C \\
\implies & P_G \text{ says } C \\
\quad & P \text{ speaks for } P_G \\
\quad & P \in G
\end{align*}
\]

\[
\begin{align*}
P_G \text{ says } C, \quad P_G \text{ says } (C \Rightarrow C') \\
\implies P_G \text{ says } C'
\end{align*}
\]
Credentials Can Convey Beliefs

\( k_S \text{-} \text{sign}( \ C \ ) : \ K_S \text{ says } C \)

- Public keys are principals.
- \( K_S \text{ speaksfor } S \) if principal S is the only agent with access to private key \( k_S \).

A principal S can be a hash of the running code and data that was read.
Access to a Joint Project

- A works for Intel and is known as A@Intel.
  - Public key $K_A$; private key $k_A$
  - Laptop
  - Member of Atom group

- MS has web page Spec
  - ACL allows access to Spec for members of Atom
  - CAL models as: **Atom speaksfor Spec**
    - Therefore: Atom **says** (access Spec) ⊢ Spec **says** (access Spec)

Suppose A requests access a Spec web page...
Application: Accessing a Joint Project

1. read page: Spec
2. challenge: r
3. $k_A \text{-sign}(r, A)$
4. A?
5. $k_{\text{intel}}(K_A, A@Intel)$
6. A@Intel in Atom?
7. $k_{\text{MS}}(A@Intel, Atom)$
8. MS web server authorizes access by Atom: Atom $\in$ Spec.ACL
CAL Model for Spec Access

1. \( K_{SSL} \text{ says } (A@Intel \text{ says (read page: Spec))} \)
2. \( K_{SSL} \text{ says } r \)
3. \( K_{SSL} \text{ says } (K_A \text{ says } (r,A)) \)
   - \( K_{SSL} \text{ speaks for } K_A \) since \( K_A \) is a subprincipal of \( K_{SSL} \)
   - Conclude: \( K_A \text{ says } (r,A) \)
5. \( K_{intel} \text{ says } K_A \text{ speaks for } A@Intel \)
   - \( K_{intel} \text{ speaks for } *@Intel, \) so: \( K_{intel} \text{ speaks for } A@Intel \)
   - Conclude: \( K_A \text{ speaks for } A@Intel \)
7. \( K_{MS} \text{ says } (A@Intel \text{ speaks for } Atom) \)
   - MS \text{ speaks for } Atom since Atom is a subprincipal of MS
   - \( K_{MS} \text{ speaks for } MS \) defn of \( K_{MS} \)
   - Conclude: \( A@Intel \text{ speaks for } Atom \)
CAL Model for Spec Access

1. $K_{SSL}$ says ($A@Intel$ says (read page: Spec))

2. $K_{SSL}$ says $r$

3. $K_{SSL}$ says ($K_A$ says ($r,A$))
   - $K_{SSL}$ speaksfor $K_A$ since $K_A$ is a subprincipal of $K_{SSL}$
   - Conclude: $K_A$ says ($r,A$)

5. $K_{intel}$ says $K_A$ speaksfor $A@Intel$
   - $K_{intel}$ speaksfor $*@Intel$, so: $K_{intel}$ speaksfor $A@Intel$
   - Conclude: $K_A$ speaksfor $A@Intel$

7. $K_{MS}$ says ( $A@Intel$ speaksfor Atom)
   - $MS$ speaksfor Atom since Atom is a subprincipal of $MS$
   - $K_{MS}$ speaksfor $MS$ defn of $K_{MS}$
   - Conclude: $A@Intel$ speaksfor Atom

----

$A@Intel$ says (read page: Spec)
CAL Model for Spec Access

1. \( K_{SSL} \text{ says } (A@Intel \text{ says (read page: Spec)}) \)
2. \( K_{SSL} \text{ says } r \)
3. \( K_{SSL} \text{ says } (K_A \text{ says } (r, A)) \)
   \[ \text{K}_{SSL} \text{ speaks for } K_A \text{ since } K_A \text{ is a subprincipal of } K_{SSL} \]
   Conclude: \( K_A \text{ says } (r, A) \)
5. \( K_{intel} \text{ says } K_A \text{ speaks for } A@Intel \)
   \[ K_{intel} \text{ speaks for } *@Intel, \text{ so: } K_{intel} \text{ speaks for } A@Intel \]
   Conclude: \( K_A \text{ speaks for } A@Intel \)
7. \( K_{MS} \text{ says } (A@Intel \text{ speaks for } Atom) \)
   MS speaks for Atom since Atom is a subprincipal of MS
   \[ K_{MS} \text{ speaks for } MS \text{ defn of } K_{MS} \]
   Conclude: \( A@Intel \text{ speaks for } Atom \)

****

A@Intel \text{ says (read page: Spec)}

A@Intel speaks for Atom
Access Authorization

A@Intel says (read page: Spec)

A@Intel speaksfor Atom

Atom speaksfor Spec due to Atom ∈ Spec.ACL

\[ \vdash \]

Spec says (read page: Spec)