

Trust but Verify: Authorization for Web Services

Christian Skalka X. Sean Wang

The University of Vermont

Trust but Verify (TbV)

Reliable, practical authorization for web service invocation.

- Securing complex interactions of (cooperating?) services
- *Ideal* security may take many details and relations into account
- *Practical* security may require abstractions or simplifications to achieve efficiency

Simplification during authorization introduces elements of *trust* into *online* security decisions.

Essence of TbV: Online trust should be *verifiable offline*, to ensure that it is warranted.

Trust but Verify: Motivations

Example: online credit card transactions.

- Credit card companies should ideally verify that cardholders delegate purchasing rights to vendors
- In North American practice*, companies trust vendors, without extensive proof of cardholder delegation
 - Online transaction efficiency
 - “Good citizenship” onus on vendors

TbV approach allows *auditing* of trusted transactions, in case of disputes or as a matter of course, to verify trust content.

*SET architecture more sophisticated; popular in Europe.

A Formal Framework

TbV is a framework for augmenting trust management systems.

- *Logically well-founded*: precise *meaning* of “trust” and “verification”, provable guarantees
- *General*: framework can be applied to variety of systems

A TbV system is characterized by high-level formal conditions.

TbV implementations must adhere to these conditions.

ABLP: High-Level Trust Management

ABLP logic* is a high-level trust management logic.

- Comprises syntax of compound principals and statements, proof theory for logical deductions (undecidable)
- Defined at abstract level: e.g. perfect crypto assumed, no “implementation” details
- Sufficiently general to encode other trust management systems, e.g. SDSI/SPKI

A convenient and expressive formal setting for conceptual development of TbV.

*M. Abadi, M. Burrows, B. Lampson and G. Plotkin, *A Calculus for Access Control in Distributed Systems*, TOPLAS 1993

ABLP: Trust Management

ABLP comprises a calculus of *principals*:

P *atomic principals* $A|B$ *A “quoting” B* $A \text{ as } B$ *A in the role of B*

$A \wedge B$ *conjunction* $A \vee B$ *disjunction*

Statements are either terms in propositional logics, or assertions ascribed to principals:

$K_P \text{ says } s$ *statement s signed with key K_P*

A partial ordering \Rightarrow defines authorization relation between principals; systems parameterized by “speaks for” assumptions:

$K_P \Rightarrow P$ *K_P speaks for P*

ABLP: Trust Management

Logical axioms of ABLP allow deductions; e.g. :

$$\frac{s \vdash A \text{ says } s' \quad s \vdash A \Rightarrow B}{s \vdash B \text{ says } s'}$$

For example, we may deduce:

$$K_P \text{ says } s \wedge (K_P \Rightarrow P) \vdash P \text{ says } s$$

By associating privileges with statements s , authorization is entailed by provability of s from access requests and *local assumptions*.

In this presentation, we restrict assumptions to *access control lists*:

$$P \text{ controls } s \triangleq (P \text{ says } s) \supset s \quad \text{ACL entry}$$

Components of TbV Framework

ABLP provides a formal trust management setting in which to develop the components of TbV:

- *Authorization contexts and decisions*
 - Resource access predicated on automated theorem proving
- *Trust transformations*
 - A function from formulae to formulae
- *Auditing* (automated verification)
 - Searches for the preimage of trust-transformed formulae

Authorization Contexts and Decisions

Authorization contexts are compound formulae, conjoining:

- Access request
- Other request components, e.g. certificates
- Local beliefs, e.g. ACL

Resource privileges are represented by atomic formulae priv ; access is granted iff priv is derivable from a request context.

Condition 1 *Let s be an authorization context; then $s \vdash \text{priv}$ is decidable.*

Authorization: Example

Suppose a webservice WS requires priv to be accessed, and a trusted principal D makes an access request:

$\mathcal{A} \triangleq D \text{ controls } \text{priv} \wedge \mathcal{A}'$ *ACL defined by WS*

$D \text{ says } \text{priv}$ *access request* $s \triangleq D \text{ says } \text{priv} \wedge \mathcal{A}$ *authorization context*

$$\frac{s \vdash D \text{ says } \text{priv} \quad s \vdash D \text{ says } \text{priv} \supset \text{priv}}{s \vdash \text{priv}}$$

Since priv is provable in this context, authorization succeeds.

Trust Transformation

Online trust is specified via trust transformations: trust is codified through function definition.

- Trust transformations simplify authorization contexts; given *extrapolated* context s , transformed formula $\llbracket s \rrbracket$ is a *trusted* context.
- Trust transformation must be decidable, made explicit by TbV implementation.
- Function definition *formalizes trust* as transformational rules.

Trust Transformation: Example

Suppose that a web service WS requires `priv` to be accessed, and:

- Any access request must be accompanied by a signed certificate authenticating the request.
- ACL \mathcal{A} defined by WS comprises entries $(P \wedge K_P)$ *controls* `priv`.
- For the sake of efficiency, signed certificates are not actively included (*suppressed*) in online authorization.

$$\begin{aligned} \llbracket P \text{ says } s \wedge K_P \text{ says } s \wedge \mathcal{A} \rrbracket &= P \text{ says } s \wedge \llbracket \mathcal{A} \rrbracket \\ \llbracket (P \wedge K_B) \text{ controls } \text{priv} \wedge \mathcal{A} \rrbracket &= P \text{ controls } \text{priv} \wedge \llbracket \mathcal{A} \rrbracket \end{aligned}$$

Auditing: Automated Verification

Formal definition of trust transformation determines meaning of verification. *Auditing* implements offline verification.

Condition 2 *Let s be a trusted context; then if $\text{audit}(s)$ succeeds, it is the case that $\text{audit}(s) \vdash s'$ such that $\llbracket s' \rrbracket = s$.*

Condition 3 *Let s be a trusted context and priv be a privilege. If $s \vdash \text{priv}$ and $\text{audit}(s)$ succeeds, then $\text{audit}(s) \vdash \text{priv}$.*

Auditing: Example (continued)

Suppose that a principal P on machine M_P makes an access request to webservice WS on M_{WS} :

$$P \text{ says } \mathbf{priv} \wedge K_P \text{ says } \mathbf{priv}$$

Since \mathcal{A} is defined by WS , the relevant trust transformed context is:

$$s \triangleq P \text{ says } \mathbf{priv} \wedge [\mathcal{A}]$$

Assume that authorization for WS in this context succeeds:

$$s \vdash \mathbf{priv}$$

Auditing: Example (continued)

Goal of auditing on M_{WS} : $(K_P \text{ says priv} \wedge P \text{ says priv} \wedge \mathcal{A})$. Two possible strategies can be imagined:

- 1 Trust transformation and logging performed by requester on M_P
 - M_P communicates $(P \text{ says priv})$
 - $(K_P \text{ says priv})$ logged on M_P
 - M_{WS} demands $(K_P \text{ says priv})$ from M_P during auditing
- 2 Trust transformation and logging performed by web service on M_{WS}
 - M_P communicates $(P \text{ says priv} \wedge K_P \text{ says priv})$
 - $(K_P \text{ says priv})$ logged on M_{WS}
 - M_{WS} searches locally for $(K_P \text{ says priv})$ during auditing

TbV for Web Services

TbV is especially suited for web service authorization:

- Cooperating web services produce distributed “stacks” of web service activations
- Multiple intermediaries, machines

Trust transformation promote online efficiency, conditions on auditing provide high-level coherence of offline verification.

TbV for Web Services

Consider a medical diagnostic service, invoked on behalf of Dr. Bob, that in turn invokes an independent medical db:

Bob as Doctor → *Diagnostic Tool as Trusted Service|Doctor* → *Medical Database*

Tool interface, diagnostic tool, database all reside on separate machines; each invocation extends chain of intermediaries, certificates

TbV for Web Services

Trust transformation eliminates levels of complexity by simplifying authorization contexts:

Doctor → Diagnostic Tool as Doctor → Medical Database

Auditing allows flexibility for storage and retrieval of “trust suppressed” knowledge in verification.

Future Work

Implementation of TbV is most immediate target for ongoing research:

- Extension of SOAP messaging format to accommodate trust management scheme
- Logging and auditing strategies for trust suppressed knowledge
- Policy enforcement, communication

Topics of *theoretical* interest also exist:

- Formal verification of security in threat model (related to *Gordon et al., Tulufale: A security tool for web services, FMCO03*)

Conclusion

Trust but Verify for web services authorization:

- Refinement of trust management for web services security
- Conceptual distinction between efficient *online* and highly secure *offline* authorization components
- Formalization of online trust provides foundation for meaning and rigor of offline verification

`http://www.cs.uvm.edu/~skalka`