A High Level Formal Language for Specifying Security Properties

Anders Moen, Thor Kristoffersen and Olaf Owe

NWPT 2002

Norwegian Computing Center 2002-20-11

Why define a precise languages for security?

- Concepts in security are unclear, and hence a topic for discussion
- Even security experts and formal method people doing security disagree and spend time defining the meaning of e.g. "authenticity"
- EU project CASENET: To take formal methods into industrial settings : verify security requirements of concrete applications
- joint work with Sharhzadhe Mazaher and Demisse Aredo

Abstraction Thesis

> There should be a precise formal language invariant to the various formal methods used to reason about security, that security experts could use to seek agreement and where non-experts in security could look up the definition of certain security properties

Requiremens to such a language

- Few primitives: (No argument needed!)
- Well understood primitives: Provide formal semantics
- expressibility: Security properties (requirements) should be easily definied within the language
- logically and computably simple: Feasible decidable logic that would interface gently to verification/modelchecking tools

How do we satisfy all the criterias?

Security properties

Security properties definable in SLSD cover the main concepts of security like

- *confidentiality*
- *authenticity*
- *integrity*
- *non-repudiation*
- *access control*

In the paper, the formal language is presented, and we show how the security properties mentioned above (except accept) can be represented in SLSD.

Trust

In Security FrameWork trust is defined by:

> Entity X is said to trust entity Y for a set of activities if and only if entity X relies upon entity Y behaving in a particular way with respect to the activities.

- Concept 'trust' unclear
- Defines trust in terms of 'behavior', 'relies upon'
- The definition does not give meaning to our intuition with respect to trust

Consider now agents X and Y:

Definition 1 *X trusts Y iff X believes everything that Y says*

Definition 2 *X trusts Y iff X believes everything that Y believes*

Problem with the last definition is that X requires access to Y beliefs

Formalization of trust

For X to trust Y, the formal counterpart of the previous informal definitions yields: $\mathsf{says}_Y(\varphi)$ reads "Y says the sentence φ", $Bel_X\varphi$ reads "Agent X believes the sentence φ":

Definition 3 $\mathrm{Trusted}(X,Y) \equiv \forall\varphi(\mathsf{says}_Y(\varphi) \rightarrow Bel_X\varphi)$

Definition 4 $\mathrm{Trusted}(X,Y) \equiv \forall\varphi(Bel_Y\varphi \rightarrow Bel_X\varphi)$

As observed in Abadi et al. [1993] trust is a second order consept, since the quantifier ranges over sentences.

Trusted Third Parties

Some agents (principals) are more trustworthy than others, they are called *Trusted Third Parties* (TTP). Trusted third parties might be reliable principals, that is, principals that say what they belive to be the case.

Definition 5 $\text{Reliable}(TTP) \equiv \forall\varphi(\mathsf{says}_{TTP}(\varphi) \rightarrow Bel_{TTP}\varphi)$

Given a group G, we might say that a principal T is a trusted third party for a group if every principal in G believes what T says, and T itself is reliable:

Definition 6 $Trusted(T, G) \equiv \forall\varphi(Bel_T\varphi \rightarrow \forall x \in G(Bel_x\varphi)) \wedge \text{Reliable}(T)$

Confidentiality

Confidentiality is "(...) the property that information is not made available or disclosed to unauthorized individuals, entities or processes" ITU:814. That a message m is confidential w.r.t. a group G, written GroupConfidential(G, m), means that m is known only to the members in G, that is:

Definition 7 $\mathrm{GroupConfidential}(G, m) \equiv \forall P \notin G(\neg \mathsf{knows}(\mathsf{P}, \mathsf{m}))$

That a message m is private to a principal a, and two principals a and b share secret m, yields respectively;

- $\mathrm{Private}(a, m) \equiv \mathrm{GroupConfidential}(\{A\}, m)$
- $\mathrm{Confidential}(a, b, m) \equiv \mathrm{GroupConfidential}(\{A, B\}, m)$

But what does it mean to know?

Being authentic

Authentication is explained as "The provision of assurance of the claimed identity of an entity". For a group of principals G_A to be authentic to a group of principal G_B means:

Definition 8

$$\mathrm{GroupAuth}(G_A, G_B, m) \equiv \forall a \in G_A \forall b \in G_B(\mathsf{Writes}(a, m) \rightarrow \mathsf{Bel}_b(\mathsf{Writes}(a, m))$$

Then from $\mathrm{GroupAuth}$ we can define

$$\mathrm{SymmetricAuthentic}(G_A, G_B, m) \equiv$$
$$\mathrm{GroupAuth}(G_A, G_B, m) \wedge \mathrm{GroupAuth}(G_B, G_A, m)$$

and that a principal b is authentic to a principal a:

$$\mathrm{Authentic}(a, b, m) \equiv \mathrm{GroupAuth}(\{a\}, \{b\}, m)$$

"Authentication is meaningful only in the context of a relationship between a principal and a verifier." (SF)

Non repudiation

"The goal of the Non-repudiation service is to collect, maintain, make available and validate irrefutable evidence concerning a claimed event or action in order to resolve disputes about the occurence or non-occurence of the event or action.(...) non-repudiation involves the generation of evidence that can be used to prove that some kind of event or action has taken place, so that this event or action cannot be repudiated later ." X.813

Non repudiation

Let a and b be principals, and m a message:

Non repudiation of receipt means that after a has sent message m, a can be sure that b have received the message.

Definition 9

$\mathrm{ReceiverNR}(a, b, m) \equiv \mathsf{Writes}(a, m) \wedge \mathsf{Reads}(b, m) \rightarrow \mathsf{canProve}_a(\mathsf{Reads}(b, m))$

Non-repudiation of origin means that b can prove that a actually wrote m

Definition 10

$\mathrm{OriginNR}(a, b, m) \equiv \mathsf{Writes}(a, m) \wedge \mathsf{Reads}(b, m) \rightarrow \mathsf{canProve}_b(\mathsf{Writes}(a, m))$

Non repudiation including Trusted Third Party

Consider the principal a sending message m to b:

Definition 11 $\mathrm{SubmitterNR}(a, t, m) \equiv \forall B(\mathsf{Writes}(a, m) \wedge \mathrm{Trusted}(t, a) \rightarrow \mathsf{canProve}_t(\mathsf{Writes}(a, m))$

Definition 12

$\mathrm{ReceiverNR}(b, t, m) \equiv \forall B(\mathsf{Reads}(b, m) \wedge \mathrm{Trusted}(t, b) \rightarrow \mathsf{canProve}_t(\mathsf{Reads}(b, m))$

Time

Note that the writing of message m should occur before the reading of m. This can be specified using the relation *Before*, written $\mathcal{B}$:

1. $\mathrm{ReceiverNR}(a, b, m) \equiv \mathsf{Writes}(a, m)\mathcal{B}\ \mathsf{Reads}\ (b, m) \rightarrow \mathsf{canProve}_a(\mathsf{Reads}(b, m))$
2. $\mathrm{OriginNR}(a, b, m) \equiv \mathsf{Writes}(a, m)\mathcal{B}\ \mathsf{Reads}\ (b, m) \rightarrow \mathsf{canProve}_b(\mathsf{Writes}(a, m))$

Confidentiality could be specialized to talk about future as well:

$\mathrm{GroupConfidential}(G, m) \equiv \forall P \notin G(\neg \Diamond \mathsf{knows}(\mathsf{P}, \mathsf{m}))$

canProve

'canProve' is a modal operator that carries the commitment of turning an "almost" proof into a real proof if additional information is available:

Definition 13 $\mathrm{canProve}_a(\varphi) \equiv \exists\psi(\mathrm{Reads}(a,\psi) \wedge \exists y \mathrm{Proof}(y(\psi),\varphi))$

Note that

- The proofs can be carried out locally by the agent a
- $\mathrm{Proof}(y(\psi),\varphi)$ is a proof predicate for a system that might be L_{SLSD} itself. ψ is a piece of information sufficient for completing the proof.

Fragments of a logic

$\vdash \mathsf{canProve}_a(\varphi_1 \rightarrow \varphi_2) \wedge \mathsf{canProve}_a(\varphi_1) \rightarrow \mathsf{canProve}_a(\varphi_2)$

$\vdash \mathsf{Bel}_a(\varphi_1 \rightarrow \varphi_2) \wedge \mathsf{Bel}_a(\varphi_1) \rightarrow \mathsf{Bel}_a(\varphi_2)$

$\vdash \mathsf{says}_a(\varphi_1 \rightarrow \varphi_2) \wedge \mathsf{says}_a(\varphi_1) \rightarrow \mathsf{says}_a(\varphi_2)$

$\vdash \mathsf{says}_{a \wedge b}(\varphi) \leftrightarrow \mathsf{says}_a(\varphi) \wedge \mathsf{says}_b(\varphi)$

Remark that the Necessitation rule in modal logic, is problematic for every modal operator

If $\vdash \varphi$ then $\vdash \mathsf{canProve}_a(\varphi)$

If $\vdash \varphi$ then $\vdash \mathsf{Bel}_a(\varphi)$

If $\vdash \varphi$ then $\vdash \mathsf{says}_a(\varphi)$

Semantics

We rely on Wooldriges temporal belief logic TBL + canProve:

Definition 14 *A model* M *is a tupple* $\langle \sigma, Agent, \mathrm{bel}, \mathrm{read}, \mathrm{write}, I \rangle$

Definition 15 *Truth in a world* x*, in model* M *is defined by*

1. $M \models_x \top$
2. $M \models_x \phi_1 \vee \phi_2$ iff $M \models_x \phi_1$ or $M \models_x \phi_2$
3. $M \models_x \mathsf{Reads}(a, m)$ iff $\langle I(a), m \rangle \in \mathrm{read}(\sigma, x)$
4. $M \models_x \mathsf{Writes}(a, m)$ iff $\langle I(a), m \rangle \in \mathrm{write}(\sigma, x)$
5. $M \models_x \mathsf{Bel}_a(\phi)$ iff $\phi \in \mathrm{bel}(\sigma, I(a), x)$
6. $M \models_x \bigcirc \phi$ iff $M \models_{x+1} \phi$
7. $M \models_x \phi \mathcal{U} \psi$ iff $\exists y (y \geq x \wedge M \models_x \psi \wedge \forall z (y < z < x \rightarrow M \models_x \phi))$
8. $M \models_x \phi \mathcal{B} \psi$ iff $\forall y (y \leq x \wedge M \models_x \phi \rightarrow \exists z (y < z < x \wedge M \models_x \psi))$

Further work

- Stabilize the language, what primitives should be in the language
- Apply the language on real examples
- Investigate refinements of the security concepts systematically
- Provide axioms for the modal operators involved
- Formalize the two roles to be played in an authentication process, claimant and verifier

References

M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.

ITU:814. Security frameworks for open systems: Confidentiality framework, November 1995. ITU-T Recommendation X.814.