# 1 Initial Overview

Negotiation occurs between two parties: the appraiser and the target. The goal is to gather evidence about a target. The appraiser sends the target a **request**. The target then responds with a **proposal** which is a set of **protocols**.

# 2 Basics

To begin constructing the language, there are basic things we must implement.

- Place
  - Necessary to understand where the Negotiation is taking place.
  - Maybe place will be specialized for both the appraiser and the target but that is a question for later
  - For now, place is a natural number that represents where the Negotiation is occurring.
- Term
  - A term is a Copland term.
  - This is useful for the Privacy Policy to decide what terms can be shared.
  - A Proposal will be composed of a list of terms from the target to the appraiser.
  - I suppose when we go to define an ordering to establish a meet and join we will order the terms. This seems like a natural place to establish ordering.
- Request
  - A request asks the target for evidence.
  - It can ask for one thing, one thing or the other thing, or both things.
  - Request is an Inductive definition then so that the constructors can be ‘one‘ ‘prod‘ ‘sum‘.
- Proposal
  - Proposal is a definition (function) which takes a request and generates a list of terms
  - Somewhere somehow the privacy policy must be satisfied.
  - Dr. A suggest making a theorem that says ”forall protocols, the privacy policy is satisfied”
  - This leads me to ask ”How do we write the privacy policy in terms of code?”

 

Knowing these pieces are needed, an initial step by step procedure can be composed.

1. A Request is generated from the appraiser and sent to the target
   - Request now only composed of one thing.
2. The target looks at the proposal and

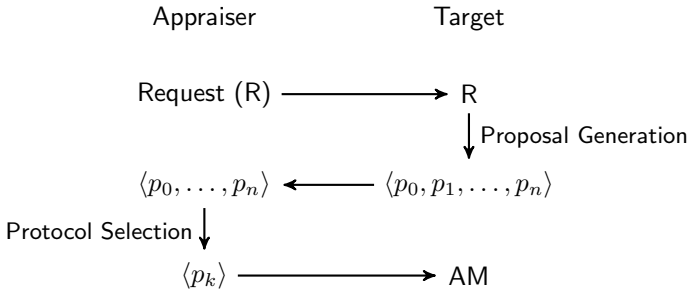


Figure 1: Processing sequence for Negotiation, Selection, Attestation and Appraisal during remote attestation.

# 3 Privacy Policy

At this point, the privacy policy is some large, overarching being that controls the data that is sent back and forth between the appraiser and the target. Both the appraiser and the target have their own, unique privacy policies. Also, in more general terms, each computer or cluster must have a privacy policy as a target and as an appraiser so that it can morph between the two roles.

# 4 Ideal trust establishment function

The ideal trust establishment $\delta_m : R \to (E, \preceq, \top, \bot)$ or delta$_m$.

$\delta_m$ relates a request to the set of all evidence packages that could result from that request. Those evidence packages are ordered by $\preceq$ that defines relative"quality" of evidence. If $e_1 \preceq e_2$ then evidence $e_2$ is of higher quality than $e_1$. Quality is subjective and this order reflects situational awareness. The relation $\preceq$ is by definition a partial order over evidence while $\bot$ and $\top$ are the worst and best evidence corresponding to no description and an exact description respectively. This defines a bounded lattice, however more work is needed to establish the correctness of the approach.

$\gamma_n$ produces a proposal $\langle p_0, p_1, \ldots, p_n \rangle$ from a request, $r$ based on target policy. $\delta_c$ transforms the proposal into evidence from each protocol, $\langle e_1, e_2, \ldots, e_n \rangle$. Thus $\delta_c$ is a functor over proposals—vectors of protocols—to vectors of evidence. $\alpha_n$ lifts the evidence vector into the evidence lattice.

# 5 Request

## 5.1 Certificate Authority

As part of ISAKMP, a certificate authority is needed for strong authentication of a communicating peer (ie the appraiser and the target).

## 5.2 Request Composition

Right now the exact understanding of the Request is not important. Only a general understanding in needed and useful.

A request is composed of some sort of evidence. It can be three things:

1. one evidence
2. sum of evidence (OR)
3. product of evidence (AND)

At this point, there really is not much else that is needed for a Request; all that is needed is evidence. In the future, some things that should be considered in the request are:

- place
- the appraiser privacy policy
- ISAKMP

# 6 Proposal

## 6.1 Producing a proposal

A proposal is a set of protocol generated by the target upon receiving the target's request. Therefore, the proposal takes in the appraiser's request and returns a list of terms. The coq definition for this may look something like an is considered an interpreter:

```
Definition propose : (request ev) -> list term
```

 

In this definition, the appraiser receives a list of terms. The list of terms must satisfy the privacy policy for the target before being sent to the appraiser. In the future, some things that should be considered in the proposal are:

- the target's privacy policy
- ISAKMP

The proposal can be ordered or unordered, it does not really matter. The appraiser decides the ordering so the appraiser orders the set before selection.

# 7 Selecting Proposal

The function, $(\gamma_n)$, selects a protocol $(\mathrm{P}_k) fromaproposal \langle p_0, p_1, \ldots, p_n \rangle$.

The selection of a Proposal will involve ensuring that the chosen protocol meets the initial negotiation condition. This can be represented in an Inductive definition as follows:

```
Inductive negotiationR : Request -> Place -> term -> Prop :=
| n1 : (ev1) -> n -> (USM 1) -> True
.
.
.
. where one can list various options for negotiation.
```

The selection of a proposal has been greatly discussed. The appraiser will always select the proposal, but how? There must be some type of ordering on the members of the set that established a notion of "best." This is how a lattice comes into play. Within a lattice, the is some sort of ordering. Therefore, each unique system will produce its own notion of ordering allowing for there to then be a best and worst protocol in each set.

The lattice definition is tricky because we cannot list out cases as seen in the example because each system is different. Instead, we must keep it more general.

# 8 Evaluating Proposal

Evaluating a proposal will occur in the Attestation Monad. The job of Negotiation is simply to choose a protocol that will be evaluated.

 

# 9 How ISAKMP fits

In general, ISAKMP is the protocol that establishes Security Associations (SA) and cryptographic keys.It allows an entity's initial communications to indicate which certificate authorities (CAs) it supports.

# 10 Questions:

1. What does the certificate authority get us? A secure channel but does it say anything about the appraiser or target's privacy policy?
2. How does the request generate a proposal?
   - Could think of a request as a condition like all even numbers.
   - Then Proposal consists of many different sets composed of even and odd numbers with each set having varying amounts of numbers.
   - Then what filters the set to include only even numbers? Is that $\delta_c$?
   - $\delta_c$ is a functor that transforms proposals into evidence. I don't really understand this at all.
3. Is the targets response, the proposal, an ordered list? I think we need a function to ensure ordering.

# 11 Examples

Throughout the process of understanding negotiation, there are many examples that have helped me get a better grasp on Coq and what Negotiation entails.

## 11.1 The Fruit Example: understanding constructing values

Let Fruit be a set such that Fruit = apple , orange , pear . Then an inductive data structure for Fruit could read:

```
Inductive request : Type :=
 | one n -> fruit
 | prod fruit -> fruit -> fruit
 | sum fruit -> fruit -> fruit.
```

Where prod is equivalent to the boolean condition AND and sum is equivalent to the boolean condition OR. Then creating examples of this would look like:

(one apple)

(prod (one apple) (one pear))

(sum (one apple) (one pear))

(prod ((prod (one apple) (one pear)) one apple)

Therefore, one apple is a constructing value. It creates a new element that is now part of the data structure. Overall, this Inductive definition of Request is a "little language."

Then, generalizing this to all data type we consider the problem of wanting to use the request data struct for a McDonald's order. If the structure was untyped, then one could just request an order. To do this, implement the following structure.

```
Inductive request (ev : Type) : Type :=
| one n -> ev
| prod ev -> ev -> ev
| sum ev -> ev -> ev
```