An Iterative Approach for the Satisfaction of Security Using the Intransitive Non-Interference Property

Achraf Ben Said [(1)], Nejib Ben Hadj-Alouane[(1)], Moez Yeddes[(1)], Feng Lin [(2)]
[(1)]National School of Computer Sciences, University of Manouba, Tunisia
[(2)]Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202 USA, and
School of Electronics and Information Engineering, Tongji University, Shanghai, China

bensaidachraf@gmail.com, nejib_bha@yahoo.com, yeddes@yahoo.fr, flin@ece.eng.wayne.edu

In this paper, we derive an iterative approach for the design of systems and protocols that are secure, with respect to intransitive non-interference (INI), a basic security property, assuring the non-devulgation of information through covert channels.
Obviously, a system that does not satisfy INI, is, in some ways, not secure. Our iterative approach is composed of three steps : system design and modeling (1), INI verification (2), and INI satisfaction (3). The basic idea, is that a system's designer during the first step of design, is focused on developing the core and necessary functionality, perhaps, while paying little attention to INI. Then, security of the system is verified, and if INI is not satisfied, the system must be modified in order it to make it secure. These modifications must be carried out, in a minimal way, as to preserve, in as much as possible, its core behavior.
Our approach uses formulas for computing the minimal superlangage satisfying the INI property. We also investigate the implementation of these formulas based on automata models. Our approach can be applied to all systems and protocols with a three level security lattice, typically, sufficient for systems using cryptographic protocols. We also give a case study that illustrates our approach applicability to to cryptographic protocols and systems.

iterative approach, observability, intransitive non-interference, INI, security, purge function, infimal.

1. INTRODUCTION

As we depend more and more on computers, the issue of computer systems and networks security is becoming become crucial for the design of such systems. Indeed the literature dealing with security issues in computer systems and networks is extensive. Namely, it is important to guarantee that no information, about confidential/private data, can be seen by an external viewer. From a purely security perspective, several information flow security properties have already been proposed; namely, noninterference (Rushby 1992), non-inference (Halloran 1990), non-deducibility (Sutherland 1986), and opacity (Mazare 2004).

In fact, Intransitive non-interference (INI), has been proposed in the literature (Goguen 1982), as a more practical extension to non-interference. INI has been formally defined in (Rushby 1992). The widely used paradigm for INI is a three level security system (Nejib Ben Hadj-Alouane 2005), modeling a typical cryptosystem, with a secret or high level, a public or low level, and an in between cipher channel, the so-called downgrading level. The basic idea behind INI is that any information flow from the high level to the low level can only occur after being explicitly downgraded, that is, after transiting through the downgrading level (for example, to encrypt the information).

In addition to its use in the control of discrete-event systems, observability, the fundamental property in supervisory control of discrete event systems, has been used to characterize information flows between different parts of a security system. However, this characterization is not captured by a simple projection, as in the case of supervisory control, but by a purge function iP, since security is more complex. Therefore, the standard definition of observability is needed to be extended in order to encompass the security of systems and protocols.

In (Nejib Ben Hadj-Alouane 2005), we propose to characterize INI using an extended version of observability called iP-observability. This approach translates the problem of checking INI, to the problem of checking iP-observability for systems with three security domains, where the domains typically designate users, channels or the different parts of the system, among which the control of information flow is needed. These results are generalized in (Nejib Ben Hadj-Alouane 2005) for checking INI in systems with an arbitrary number of security domains governed by a security lattice, characterizing all possible information flows.

If a system does not satisfy INI, however, then we propose to secure it, by modifying it in a "minimal" way. This is done by either enlarging the system behavior or reducing it (Yeddes 2008). For systems with three security domains, the problem is solved in (Yeddes 2008) by translating an iP-observability problem into a set of P-observability problems.

This paper is an extension of our developed approach for the satisfaction of the INI property in systems and security protocols having three or fewer security domains (Ben Said 2011). In this work, we derive a new iterative approach for system and protocol design. Our approach is composed of three main steps: the design of the system (1), security verification (2) and security satisfaction based on the INI property (3). Our aim is to derive formulas to cover the step of INI satisfaction. During this step, the system's designer is looking to modify the behavior of the system, in order to make it satisfy the INI property.

Our approach works iteratively by modifying, at each step, the system's behavior, in a "minimal" way; thereby resulting in a system satisfying INI. The modifications are performed via an algorithm that enlarges the system's behavior. Our approach is practical in the sense that it does not require that the initial design of the system satisfies INI. It rather enables an iterative design process, where the system's designer starts by capturing the core and necessary functionality, perhaps while paying little attention to INI, and then uses our method to complement the system's behavior, as to make it secure.

In this paper, we propose and prove a new formula for computing the infimal observable super-language, based on a generalized definition of observability. With this new formula, we develop an algorithm for constructing an automaton generating the infimal observable super-language, from the automaton of the original, given language. Next, we apply the formula and the algorithm, for computing the infimal iP-observable super-language. This enables us to concretely secure systems by modifying their behavior to satisfy INI.

This paper is organized as follows: In Section 2, we briefly review INI and some related results. In Section 3, we give the generalized formula characterizing the infimal observable super-language and the computation algorithm. In Section 4, we apply the results of Section 3 to INI, and compute the infimal iP-observable super-language. Finally, in section 5, we present our approach and we apply our results to study an INI satisfaction problem for an example of electronic paiement protocol.

Due to space limitation, all the proofs are omitted. They can be found, however, in (Ben Said 2011).

2. INTANSITIVE NON-INTERFERENCE

In this section, we briefly review intransitive non-interference in multi-level systems as well as its characterization through $iP - observability$ as has been shown in (Nejib Ben Hadj-Alouane 2005). The property of non-interference, introduced by Goguen and Meseguer (Goguen 1982), can be simply stated as follows: The behavior of a given entity is said *not to* interfere with the behavior of a given second entity, if no action performed by the first can influence subsequent outputs seen by the second. The property of intransitive non-interference introduced by Rushby (Rushby 1992) extends basic non-interference, which requires that there is no unintended flow of information. It enables the specification of a more practical class of security policies that deal with channel control mechanisms, in-line with the following paradigm that promotes downgrading and controlled information flow: Given a system with three channels A, B, and C, information is allowed to flow from A to C only after passing trough B, and never directly (intransitivity); B is seen here as some kind of a downgrading channel. In terms of non-interference, the event stream generated by A is allowed to interfere with the event stream generated by C, indirectly, only through B events. This property have been used as a basis for specifying and analyzing several practical and modern security issues in many systems and protocols.

To formally define INI, we first describe a multi-level security system as follows: Given a set $\mathbb{D}$ of security domains and a set of events Σ partitioned over these domains, the operator $dom : \Sigma \to \mathbb{D}$ is used to capture this partition: to every domain $U \in \mathbb{D}$, the set $\Sigma_U = \{\sigma \in \Sigma \mid dom(\sigma) = U\}$ specifies the events associated with U. The domains are interpreted to represent the security channels, users, or system components, for which we shall define

non-interference requirements. We also consider an interference relation $\rightsquigarrow \subset \mathbb{D} \times \mathbb{D}$ defined over $\mathbb{D}$: given domains U, U', the domain U is allowed to interfere with the domain U' whenever $U \rightsquigarrow U'$.

We assume that the combined system behavior associated with all the domains is modeled by a language $K \subseteq \Sigma^*$. K is generated by a finite deterministic automaton $G = (\Sigma, X, \delta, x_0)$; i.e., $K = L(G)$, the language generalized or recognized by the automaton G. In G, Σ is a finite set of events as defined above; X is a finite set of states; $\delta : \Sigma \times X \rightarrow X$ is the (deterministic) transition function; and x_0 is the initial state. Since $K = Ł(G)$, K is prefix-closed.

To formally define INI, a string reduction function, iP, operating on strings $s \in \Sigma^*$, called $intransitive\ purge$, is introduced in (Rushby 1992). It removes, from a string, every event from domains that are not allowed to interfere with the given domain. The iP function is defined using the function $sources : \Sigma^* \times \mathbb{D} \rightarrow \mathbb{D}$ which is given as follows (ε is the empty string.):

$sources(\epsilon, U) = \{U\}$, and,

$$sources(\sigma s, U) = \begin{cases} sources(s,U) \cup \{dom(\sigma)\} \\ \quad \text{if } (\exists V \in sources(s,U))\mathrm{dom}(\sigma) \rightsquigarrow V, \\ sources(s,U) \quad \text{otherwise.} \end{cases}$$

Intuitively, $sources$ captures the set of domains which are allowed to interfere throughout every step of the execution of a string. This set of domains is determined backwards (i.e., starting from the end of the string). Moreover, the fact that a given domain V is in $sources(s, U)$, means either that $V = U$ or that there is a subsequence $\sigma_1, \sigma_2, \sigma_3 ... \sigma_n$ of the string s, such that $dom(\sigma_1) \rightsquigarrow dom(\sigma_2) \rightsquigarrow dom(\sigma_3) ... dom(\sigma_n)$ with $V = dom(\sigma_1)$ and $dom(\sigma_n) \rightsquigarrow U$.

Using the function $sources$, we define $iP : \Sigma^* \times \mathbb{D} \rightarrow \Sigma^*$ as follows:

$iP(\epsilon, U) = \epsilon$, and,

$$iP(\sigma s, U) = \begin{cases} \sigma iP(s,U) & \text{if } dom(\sigma) \in sources(\sigma s, U), \\ iP(s,U) & \text{otherwise.} \end{cases}$$

Based on the above, INI is defined as follows (Rushby 1992):

Definition 1 *A language K satisfies the property of* **intransitive non-interference** *if,*

$$(\forall_{U \in \mathbb{D}})(\forall_{s \in K})(\forall_{\sigma \in \Sigma_U})\ s\sigma \in K \Leftrightarrow iP(s,U)\sigma \in K.$$

To present an observability characterization of INI, we first generalize the notion of observability introduced by Lin and Wonham (Lin 1996). Let Σ_o be the set of observable events. We define the usual string projection, $P : \Sigma^* \rightarrow \Sigma_o^*$, as follows:

$$P(\varepsilon) = \varepsilon, \quad \text{and,} P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o, \\ P(s) & \text{otherwise.} \end{cases}$$

Throughout the rest of the paper, we shall use M to describe the largest possible behavior for a system; and, we shall assume that M is prefix-closed. Obviously, in many cases, $M = \Sigma^*$.

Generalized observability is defined as follows:

Definition 2 *Let $M \subseteq \Sigma^*$ be prefix closed, and $\Sigma_U \subseteq \Sigma$. A prefix-closed language $K \subseteq M$ is P-**observable** with respect to (M, Σ_U) if,*

$(\forall_{s,s' \in K})(\forall_{\sigma \in \Sigma_U})\ (P(s) = P(s')$
$\wedge\ s\sigma \in K\ \wedge\ s'\sigma \in M)\ \Rightarrow\ s'\sigma \in K.$

Intuitively, P-observability declares that if two traces are perceived in the same way, then for any event $\sigma \in \Sigma_U$, when it is permitted by M to follow one of the two strings, it must be permitted to follow the other. It should be noted that K is said to be observable in the sense of (Lin 1996), if it is P-observable with respect to (M, Σ).

Now we extend P-Observability to iP-Observability, by replacing the projection P with the purge function iP as follows.

Definition 3 *Let $M \subseteq \Sigma^*$ be prefix closed and $\Sigma_U \subseteq \Sigma$. A prefix-closed language $K \subseteq M$ is iP-**observable** with respect to (M, Σ_U) if,*

$(\forall_{s,s' \in K})(\forall_{\sigma \in \Sigma_U})\ (iP(s,U) = iP(s',U)$
$\wedge\ s\sigma \in K\ \wedge\ s'\sigma \in M)\ \Rightarrow s'\sigma \in K.$

The following theorem, proved in (Nejib Ben Hadj-Alouane 2005), states that the problem of verifying INI in a multi-level security system, can be reduced to the problem of checking iP-observability.

Theorem 1 *A language K satisfies INI if and only if K is iP-observable with respect to (Σ^*, Σ_U), for all $U \in \mathbb{D}$.*

The above theorem translates the INI verification problem into an iP-observability verification problem. For systems with three security domains, the iP-observability verification problem is further reduced to a P-observability verification problem (Nejib Ben Hadj-Alouane 2005,Y).

We shall further investigate P-observability in the next section, and in Section 4, and we shall come back to iP-observability.

3. A FORMULA FOR CHARACTERIZING THE INFIMAL OBSERVABLE SUPERLANGUAGE

In this section, observability refers to generalized observability, i.e., P-observability.

As noted in the previous section, the observability given in (Lin 1996) is a special case of P-observability. It coincides with P-observability with respect to (M, Σ). A formula characterizing the infimal P-observable superlanguage, with respect to (M, Σ), is given in (Rudie 1990). This formula uses a modified projection $\tilde{P} : \Sigma^* \to \Sigma^*$ defined as follows:

$$\tilde{P}(\varepsilon) = \varepsilon, \tilde{P}(s\sigma) = P(s)\sigma.$$

We extend $\tilde{P}$ to languages and define the inverse mapping $\tilde{P}^{-1}$ in the usual way.

Theorem 2 *The infimal, prefix-closed and P-observable superlanguage, with respect to (M, Σ), of a prefix-closed language $K \subseteq M$, is given by,*

$$H = M - (\Sigma^+ - \tilde{P}^{-1}(\tilde{P}(K)))\Sigma^*.$$

The above theorem is given in (Rudie 1990). The formula, however, does not work for an arbitrary $\Sigma_U \neq \Sigma$. Therefore, we extend it for generalized observability.

Theorem 3 *The infimal, prefix-closed and P-observable superlanguage with respect to (M, Σ_U), of a prefix-closed language $K \subseteq M$, is given by,*

$$H' = H \cap K\Sigma_U^*. \quad (1)$$

For the INI problem to be discussed next, $M = \Sigma^*$. In this case, the formula given by Equation1, can be simplified as follows:

Corollary 1 *If $M = \Sigma^*$, then the infimal, prefix-closed and P-observable superlanguage, with respect to (M, Σ_U), of a prefix-closed language $K \subseteq \Sigma^*$ is given by,*

$$H' = ((\tilde{P}^{-1}(\tilde{P}(K)))^c\Sigma^*)^c \cap K\Sigma_U^*, \quad (2)$$

where $(.)^c$ denotes the complement operator.

Let us now implement Equation 2 by constructing the corresponding automata from a given automaton G :

$$G = (\Sigma, X, \delta, x_0)$$

marking the langage k, such us $K = L(G)$. Please note that no implementation of Equation 1 is given in (Rudie 1990).

In the following, we give our algorithm for the computation of the infimal superlangage. Our algorithm consists of eight steps, presented as follows:

Algorithm : Computation of the infimal $P - observable$ superlanguage

***Step 1*: Computation of $\tilde{P}(L)$.**

We construct $G_{\tilde{P}}$ as follows:

$$\begin{aligned} G_{\tilde{P}} &= \{\Sigma, Y, \delta_{\tilde{P}}, y_0\} \\ &= Ac(\{\Sigma, X \times S, \delta_{\tilde{P}}, y_0\}). \end{aligned}$$

where $S = \{u, o\}$ such that :

- u : is a label to describe an unobservable event in Σ.
- o : is a label to describe an observable event in Σ .

The states space Y is the reachable part of $X * S$ and the initial state is $y_0 = (x_0, o)$. The transition function

$$\begin{aligned} \delta_{\tilde{P}} &= \{((x_i, \alpha), P(\sigma), (x_j, o)) \mid (x_i, \sigma, x_j) \in \delta \wedge \alpha = o\} \\ &\cup \{((x_i, \alpha), \sigma, (x_j, u)) \mid (x_i, \sigma, x_j) \in \delta \wedge \alpha = o \wedge \sigma \notin \Sigma_o\} \end{aligned}$$

Step 2 : Construction of the automaton $G^{nd}_{\tilde{P}^{-1}}$, marking the language $\tilde{P}^{-1}(\tilde{P}(K))$:

$$\begin{aligned} G^{nd}_{\tilde{P}^{-1}} &= \{\Sigma \cup \{\varepsilon\}, U, \delta^{nd}_{\tilde{P}^{-1}}, u_0, U_m\} \\ &= Ac(\{\Sigma \cup \{\varepsilon\}, Y \times H, \delta^{nd}_{\tilde{P}^{-1}}, u_0, U_m\}) \end{aligned}$$

such us $H = \{m, n\}$:

- m : is a label to describe a marked state,
- n : is a label to describe a unmarked state.

$U_m = \{(y_i, m) | y_i \in Y\}$ is the set of marked states and $u_0 = (y_0, m)$. The transition function $\delta^{nd}_{\tilde{P}^{-1}}$ is computed using the algorithm 1.

Step 3 : Computation of the automaton $G_{\tilde{P}^{-1}}$, the deterministic version of $G^{nd}_{\tilde{P}^{-1}}$, by applying the customary procedures.

$$G_{\tilde{P}^{-1}} = \{\Sigma \cup \{\varepsilon\}, U, \delta_{\tilde{P}^{-1}}, u_0, U_m\}.$$

Algorithm 1: Transition function $\delta^{nd}_{\tilde{P}^{-1}}$

begin
  foreach *state* $(y_i, m) \in U$ **do**
    foreach *transition* $(y_i, \sigma, y_j) \in \delta_{\tilde{P}}$ **do**
      if $\sigma = \varepsilon$ **then**
        $((y_i, m), \varepsilon, (y_j, m)) \in \delta^{nd}_{\tilde{P}^{-1}}$
      end
      if $\sigma \in \Sigma_o$ **then**
        $((y_i, m), h_k, (y_i, n)) \in \delta^{nd}_{\tilde{P}^{-1}},\ \forall h_k \in \Sigma_U$
        $((y_i, n), h_k, (y_i, n)) \in \delta^{nd}_{\tilde{P}^{-1}},\ \forall h_k \in \Sigma_U$
        $((y_i, n), \sigma, (y_j, m)) \in \delta^{nd}_{\tilde{P}^{-1}}$
        $((y_i, m), \sigma, (y_j, m)) \in \delta^{nd}_{\tilde{P}^{-1}}$
      end
      if $\sigma \in \Sigma_U$ **then**
        $((y_i, m), h_k, (y_i, n)) \in \delta^{nd}_{\tilde{P}^{-1}},\ \forall h_k \in \Sigma\{\Sigma_o \cup \sigma\}$
        $((y_i, n), h_k, (y_i, n)) \in \delta^{nd}_{\tilde{P}^{-1}},\ \forall h_k \in \Sigma\{\Sigma_o \cup \sigma\}$
        $((y_i, n), \sigma, (y_j, m)) \in \delta^{nd}_{\tilde{P}^{-1}}$
        $((y_i, m), \sigma, (y_j, m)) \in \delta^{nd}_{\tilde{P}^{-1}}$
        $((y_j, m), h_k, (y_i, n)) \in \delta^{nd}_{\tilde{P}^{-1}},\ \forall h_k \in \Sigma\{\Sigma_o \cup \sigma\}$
        $((y_j, m), \sigma, (y_j, m)) \in \delta^{nd}_{\tilde{P}^{-1}}$
      end
    end
  end
  return $\delta_{\tilde{P}^{-1}}$;
end

Step 4 : Computation of the automaton G_c, describing the complement language of $L(G_{\tilde{P}^{-1}})$.

Step 5 : Computation of the automaton $G^{nd}_c omc$, concatenating $L(G_c)$ with Σ^*.

Step 6 : Computation of the automaton G_{comc}, the deterministic version of $G^{nd}comc$, by applying the customary procedures.

Step 7 : Computation of the automaton G_{concat} by concatenating $L(G)$ with Σ^*_o.

Step 8 : Computation the automaton $G_{H'}$ by taking the intersection between G_{comc} and G_{concat}.
We note that the intersection, concatenation and complement, used in the above algorithm, are standard operations of automaton theory.

A Complexity Note : The algorithm presented in this section is of exponential complexity. The major contributions to this complexity are from steps 3 and 6, where the automaton is converted to a deterministic version, with a complexity of $O(2^{|X|})$. It should be noted, however, that, in practice, there are speedy techniques for performing this standard determinisation operation.

4. INFIMAL SUPERLANGUAGE SATISFYING INTANSITIVE NON-INTERFERENCE

We now apply the results in the previous section to the INI problem. In particular, we compute infimal superlanguage satisfying INI. We consider systems with three security domains: $\mathbb{D} = \{H, D, L\}$, where H is a high security domain (classified), L is a low security domain (unclassified), and D is the downgrading domain. The non-interference relation is such that:

$$\rightsquigarrow = \{(H, D), (D, L), (D, H), (L, D), (L, H)\},$$

that is, only $H \rightsquigarrow L$ is not allowed ($H \not\rightsquigarrow L$). For this three-domain system, we only need to check iP-observable of $K \subseteq \Sigma^*$ with respect to (Σ^*, Σ_L), that is, whether,

$$\forall_{w,w' \in K} \forall_{\sigma \in \Sigma_L} (iP(w, L) = iP(w', L) \wedge w\sigma \in K) \Rightarrow w'\sigma \in K.$$

To calculate the infimal iP-observable superlanguage of K, denoted by $\inf O^{iP}_{super}(K)$, we first consider languages without downgrading events: $J \subseteq (\Sigma_H \cup \Sigma_L)^*$. Define the projection $P : (\Sigma_H \cup \Sigma_L)^* \rightarrow \Sigma^*_L$ that erases the high-level events. Then the P-observability of J with respect to $((\Sigma_H \cup \Sigma_L)^*, \Sigma_L)$ is reduced to the following.

$$\forall_{u,u' \in J} \forall_{\sigma \in \Sigma_L} (P(u) = P(u') \wedge u\sigma \in J) \Rightarrow u'\sigma \in J.$$

From now on, P-observability will always be with respect to $((\Sigma_H \cup \Sigma_L)^*, \Sigma_L)$. The infimal P-observable superlanguage of J, denoted by $\inf O^{P}_{super}(J)$, is given by Equation 2 as:

$$\inf O^{P}_{super}(J) = ((\tilde{P}^{-1}(\tilde{P}(J)))^c \Sigma^*)^c \cap J\Sigma^*_U.$$

We now consider downgrading events. We would like to separate high-level and low-level behavior between two downgrading events. Therefore, we define the set of strings in K that either is the empty string or ends with a downgrading event as:

$$D(K) = \{\epsilon\} \cup (K \cap \Sigma^* \Sigma_D).$$

For each $s \in D(K)$, we define its continuation in $(\Sigma_H \cup \Sigma_L)^*$ as follows:

$$C_{HL}(s, K) = (\Sigma_H \cup \Sigma_L)^* \cap (K/\{s\}),$$

where $K/\{s\} = \{t \in \Sigma^* : st \in K\}$ is the set of all continuations of s in K. As shown in (Yeddes 2008),

language K can be decomposed as follows:

$$K = \bigcup_{s \in D(K)} \{s\} C_{HL}(s, K).$$

The following formula is derived in (Yeddes 2008) to compute the infimal iP-observable superlanguage of K:

$$\inf O^{iP}_{super}(K) = \bigcup_{s \in D(K)} \{s\} \inf O^{P}_{super}(C_{HL}(s, K)).$$

To implement the above formula, we note that the set $D(K)$ may be infinite if the automaton G for K has loops. However, we do not need to compute infinitely many of $\inf O^{P}_{super}(C_{HL}(s, K))$ because finite of them are distinct. To formalize our approach, for each $s \in D(K)$, let us look it path in G. Let us remove all loops in the path. Denote the resulting string $\underline{s}$. In other words, $\delta(x_0, s) = \delta(x_0, \underline{s})$ and $\underline{s}$ does not visit any state more than once. Clearly,
$C_{HL}(s, K) = C_{HL}(\underline{s}, K)$. There is only finite distinct $\underline{s}$ and hence finite distinct $C_{HL}(\underline{s}, K)$. They can all be calculated using Equation 3. Finally,

$$\begin{aligned} \inf O^{iP}_{super}(K) &= \bigcup_{s \in D(K)} \{s\} \inf O^{P}_{super}(C_{HL}(\underline{s}, K)) \\ &= \bigcup_{s \in D(K)} \{s\}((\tilde{P}^{-1}(\tilde{P}(C_{HL}(\underline{s}, K))))^c \Sigma^*)^c \\ &\quad \cap C_{HL}(\underline{s}, K)\Sigma^*_L. \end{aligned}$$

Let us now implement the above equation by constructing the corresponding automata from the automaton G describing a three domains systems : $G = (\Sigma, X, \delta, x_0)$

Algorithm : Computation of the Infimal $iP - observable$ **Superlangage**

Step 1 : Computation the set $\tilde{D}(K) \subseteq D(K)$ containing strings from K which was either is the empty string or ends with a downgrading event.

$$\tilde{D}(K) = \{\varepsilon\} \bigcup \{L \cap \Sigma^* \Sigma_D\}$$

The set of tarces $D(K)$ could be infinit, if the automaton G marking the language K contains some loops. So, we should compute a subset $D(\tilde{K}) \subset D(K)$, based on the minimal traces of K. The notion of minimal traces is detailed in (Nejib Ben Hadj-Alouane 2005).

To find the set of all minimal subtraces of K with respect to G, we construct an acyclic automaton G', that is the depth-first expansion of G as follows: Let K^n be the set of traces in K of length at most n, where n is the cardinality of the state space of G (that is, $n = |X|$). Then, G' is defined as follows:

$$G' = (\Sigma, X', \delta', x'_0) =^{def} Ac(\Sigma, X \times K^n \times 2^X, \delta', (x_0, \varepsilon, \{x_0\}))$$

where the operator Ac gives the accessible (reachable) part of the G; the transition function δ' is defined, recursively, starting from the initial state, as follows:

For the initial state $(x_0, \varepsilon, \{x_0\})$ and $\sigma \in \Sigma$,

$$\delta'((x_0, \varepsilon, \{x_0\}), \sigma) = \begin{cases} \delta(x_0, \sigma), \sigma, \{x_0, \delta(x_0, \sigma)\} \\ \quad \text{if } \delta(x_0, \sigma)\ is\ defined\ and \\ x_0 \neq \delta(x_0, \sigma), \\ \text{undefined} \quad \text{otherwise.} \end{cases}$$

For a state (x, s, Y) already in the rang of δ' and $\sigma \in \Sigma$,

$$\delta'((x, s, Y), \sigma) = \begin{cases} \delta(x, \sigma), \sigma, Y \cup \{x, \delta(x, \sigma)\} \\ \quad \text{if } \delta(x, \sigma)\ is\ defined\ and \\ x \notin Y, \\ \text{undefined} \quad \text{otherwise.} \end{cases}$$

So, the event set $\tilde{D}(K)$ is then given as follows, based on the previous notion of minimal subtraces :

$$\tilde{D}(K) = \{s'\sigma | s' = \underline{s} \ \wedge \ s\sigma \in K \ \wedge \ \sigma \in \Sigma_D\} \ \cup \ \{\varepsilon\}.$$

Step 2 : Construction of $C_{HL}(s, K)$ for all $s \in \tilde{D}(K)$: $C_{HL}(s, K)$ the continuation of $s \in D(\tilde{K})$ in $(\Sigma_H \cup \Sigma_L)^*$.

Given $s \in \tilde{D}(K)$, so, we define $G_{C_{HL}(s,K)} = \{\Sigma', X', \delta', x'_0\}$ the automaton that marks the language $C_{HL}(s, K)$ as follows:

$$G_{C_{HL}(s,K)} = Ac(\{\Sigma_H \cup \Sigma_L, X, \delta_{h,l}, x'_0\})$$

where $x'_0 = \delta(x_0, s), X' \subset X$ and $\delta' \subset \delta$ and the transition function $\delta_{h,l}$ is defined recursively as follows:

for a transition $(x', \sigma, x'') \in \delta$:

$$\delta_{h,l}(x', \sigma) = \begin{cases} \delta(x', \sigma) & if\ \sigma \notin \Sigma_D \\ undefined & otherwise. \end{cases}$$

Step 3 : Computation of $infO^{P}_{super}(G_{C_{HL}(s,K)})$ for all $s \in \tilde{D}(K)$, by applying the algorithm presented in the previous section

We will have as result the automaton $G_{inf}(s) = \{\Sigma', Q, \delta_{inf}, q_0\}$

Step 4 : Computation of $G"(s)$ from $G_{inf}(s)$ for all $s \in \tilde{D}(K)$, on projecting the set Q onto X'

Step 5 : Construction of G^{iP-O}_{sup} that marks the infimal and $iP - Observable$ superlanguage of G :

In this final step, we will make the union of all the enlarged languages described by the automatons $G'(s)$ for all $s \in \tilde{D}(K)$ of the given language $K = L(G)$.

5. AN ITERATIVE APPROACH AND AN EXAMPLE

To better illustrate the execution of the previous algorithm, we shall apply it to the problem of INI satisfaction for an electronic payment protocol. This protocol implements a credit card based transaction between a customer A and a merchant B, using the existing financial network for clearance and authorization. After the customer and merchant agree upon the transaction, B uses the information obtained from A in order to request an authorization and clearance from an acquired ACQ (for example a bank), for the payment, by forwarding the encrypted information obtained from A.

msg1 : $A \rightarrow_{\{n\}_K} B$
msg2 : $B \rightarrow_{A,\{n\}_K} ACQ$
msg3 : $ACQ \rightarrow_{resp} B$

where n is A's credit card number, k is the acquired public encryption key, and $resp$ is the acquired response, hence $resp = ok$ or $resp = notok$, and $\{n\}_k$ stands for n encrypted by k.

The process executed by the customer, the merchant and the ACQ, can be modeled using the process algebra model, as follows: $\alpha A = \{msg1(\{n\}_K)\}$.
$\alpha B = \{msg1(\{n\}_K), msg2(A, \{n\}_K), msg3(OK), msg3(NotOK)\}$.

Figure 1: *Electronic Payment Protocol.*

$\alpha ACQ = \{clearing(A, n), msg2(A, \{n\}_K), msg3(OK), msg3(NotOK)\}$.
$A = \mu X.\{msg1(\{n\}_K\} : (msg1(\{n\}_K) \rightarrow X)$
$B = \mu X.\{msg1(\{n\}_K), msg2(A, \{n\}_K), msg3(OK), msg3(NotOK)\} : (msg1(\{n\}_K) \rightarrow msg2(A, \{n\}_K \rightarrow (msg3(OK) \rightarrow X \mid msg3(NotOK) \rightarrow X))$
$ACQ = \mu X.\{clearing(A, n), msg2(A, \{n\}_K), msg3(OK), msg3(NotOK)\} : (msg2(A, \{n\}_K \rightarrow clearing(A, n) \rightarrow (msg3(OK) \rightarrow X \mid msg3(NotOK) \rightarrow X))$

The total protocol is specified as product of the concurrent principals: $P = A \mid\mid B \mid\mid ACQ$.

Based on the equivalence between process algebra and state machine models, our system can be easily converted into a deterministic generator G, as described in Figure 2. We note that in this model, $message1(\{n\}_k)$, and $message2(A, \{n\}_k)$, are downgrading events, since the secret value, n, is encrypted. Furthermore, $clearing(A, n)$, is a high-level event, since n appears within its parameters. Finally, $message3(ok)$, and $message3(notok)$, are low-level events, since these messages are typically exchanged over public channels, and can be intercepted by any intruder. Hence, wehave,

$\Sigma_D = \{msg1(\{n\}_K), msg2(A, \{n\}_K)\}$;
$\Sigma_H = \{clearing(A, n)\}$;
$\Sigma_L = \{msg3(OK), msg3(NotOK)\}$.

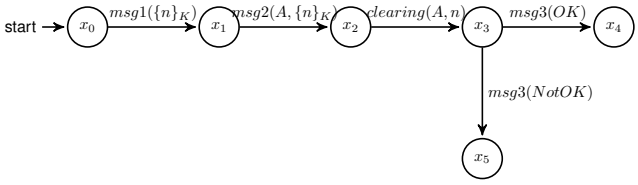

Figure 2: *Generator, G Modeling the Process P.*

It is clear that our protocol is invalid in terms of INI. Indeed, we take the two traces $s = msg1(\{n\}_K)msg2(A, \{n\}_K)Clearing(A, n)$ and $s' = msg1(\{n\}_K)msg2(A, \{n\}_K)$. We have $P(s) = P(s')$ and $smsg3(OK) \in L(G)$ and $s' = msg3(OK)\ notinL(G)$. This system has a leak of information that can be exploited by an intruder (or even a seller B) that can send messages $(A\{n\}_K)$ with different values of n , and possibly deduct if n is a number of electronic payment card valid or not, through the reception of messages $msg3(OK)$

and $msg3(NotOK)$.To overcome this problem we will calculate the minimum super-language verify the intransitive noninterference.

At first, let us compute the set of all minimal traces of G by constructing G'. This automaton is not shown since it can be easily deduced from G. Based on G', we will compute the set $\tilde{D}(K)$. In our example, the generator G is not containing any cycle so that we have $\tilde{D}(K) = DK$. So, we obtain :

$\tilde{D}(K) =$
$\{\varepsilon, msg1(\{n\}_K), msg1(\{n\}_K)msg2(A,\{n\}_K)\}.$

After that, for all $s \in \tilde{D}(K)$, we compute $C_{HL}(s,K)$, the continuation of $s \in D(\tilde{K})$ in $(\Sigma_H \cup \Sigma_L)^*)$. Here, we have just to compute $C_{HL}(msg1(\{n\}_K)msg2(A,\{n\}_K), L(G))$. This automaton is escribed in Figure 3 :

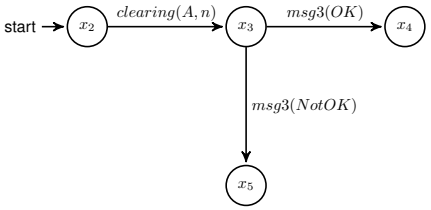

Figure 3: *Generator,* $C_{HL}(msg1(\{n\}_K)msg2(A,\{n\}_K), L(G))$

The results of applying the algorithm 1 on the automaton $C_{HL}(msg1(\{n\}_K)msg2(A,\{n\}_K), L(G))$ is the automaton $infO^P_{super}(msg1(\{n\}_K)msg2(A,\{n\}_K), L(G))$ given in Figure 4:

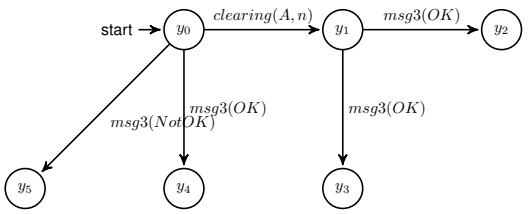

Figure 4: *Generator,* $infO^P_{super}(msg1(\{n\}_K)msg2(A,\{n\}_K), L(G))$

On the fourth step, we construct the automaton $G"(msg1(\{n\}_K)msg2(A,\{n\}_K))$ given in Figure 5 :

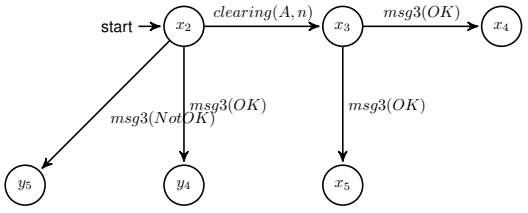

Figure 5: *Generator,* $G"(msg1(\{n\}_K)msg2(A,\{n\}_K), L(G))$

Finally, the automaton G^{iP-O}_{sup} marking the minimal superlangage of $L(G)$ is given in figure 6 :

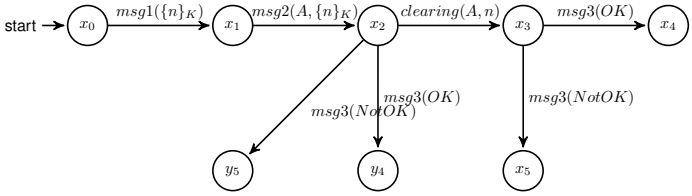

Figure 6: *Generator,* G^{iP-O}_{sup}

Our system is now secure as the intruder will not be able to perform his attacks to deduce information about the validity of the card number submitted. In fact, the two messages $msg3(OK)$ and $msg3(NotOK)$, have no longer any interest because they no longer describe the response of the validation unit about the number n introduced. So the resulting system meets both the technical requirements imposed by the manufacturer and also preserving the validity of information flows imposed by the property of Intransitive Non-interference.

6. CONCLUSION

In this paper, we derive a new iterative approach for system construction composed of three steps: conception, INI verification and INI validation. We First derived formulas for computing the infimal superlanguage satisfying INI property, under a general definition of observability. We also show how these formulas can be implemented on automata. This have never been done before in INI literature. Our approach is algorithmic: we developed an algorithm based on automata representing the system/protocol behavior that can be used to compute the best system's behavior not only, having necessary functionalities, but also satisfying INI property.

7. REFERENCES

D. Bell and L. LaPadula (1976). "Secure computer system: Unified exposition and multics interpretation", Tech. Rep. MTR-2997, Mitre Corp., Bedford, Mass., USA.

N. Ben Hadj-Alouane, S. Lafrance, F. Lin, J. Mullins, and M. Yeddes (2005). "Characterizing Intransitive Non-Interference in Security Policies with Observability", *IEEE Transactions on Automatic Control*, vol. 50, no. 6, pp. 920-925.

N. Ben Hadj-Alouane, S. Lafrance, F. Lin, J. Mullins, and M. Yeddes, "On the verification of intransitive noninterference in multilevel security", *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 35, no. 5, pp. 948-958, 2005.

F. Cassez, J. Mullins, and O. H. Roux, "Synthesis of non-interferent systems", In the 4th International

Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'07), Communications in Computer and Inform. Science, Vol. 1, Springer, pp. 307-321, September 2007.

H. Cho and S.I. Marcus, "Supremal and maximal sublanguages arising in supervisor synthesis problem with partial observation", *Mathematical Systems Theory*, vol. 22, pp. 177-211, 1989.

R. Focardi, A. Ghelli and R. Gorrieri, "Using non interference for the analysis of security protocol", *Proc. of DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.

R. Focardi and R. Gorrieri, "A classification of security properties for process algebras", *Journal of Computer Security*, vol. 3, no. 1, pp. 5-33, 1995.

G. Gardey, J. Mullins, and O. H. . Roux, " Non-interference control synthesis for security timed automata", In 3rd International Workshop on Security Issues in Concurrency (SecCo'05), Electronic Notes in Theoretical Computer Science 180, Vol. 1, Elsevier, San Francisco, USA, August 2005.

J. Goguen and J. Meseguer, "Security policies and security models", in *Proceedings 1982 IEEE Symposium on Research in Security and Privacy*, pp. 11-20, April 1982.

J. T. Haigh, R. A. Kemmerer, J. McHugh, W. D. Young, "An Experience Using Two Covert Channel Analysis Techniques on a Real System Design", *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 157-168, 1987.

F. Lin and W. M. Wonham, "On observability of discrete-event systems", *Information Sciences*, vol. 44, pp. 173-198, 1988.

J. McLean, "A general theory of composition for a class of possibilistic properties", *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 53-66, 1996.

J. Mullins, "Nondeterministic admissible interference", *Journal of Universal Computer Science*, vol. 6, no. 11, pp. 1054-1070, 2000.

J. Mullins and S. Lafrance, "Bisimulation-based nondeterministic admissible interference with applications to the analysis of cryptographic protocols", *International Journal in Information and Software Technology*, pp. 1-25, 2002.

R.J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, vol. 25 no. 1, pp. 206-230, 1987.

K. Rudie and W. M. Wonham, "The infimal prefix-closed and observable superlanguage of a given language", *Systems & Control Letters*, vol. 15, pp. 361–371, 1990.

A.W. Roscoe, "CSP and determinism in security modelling", *1995 IEEE Symp. on Research in Security and Privacy*, pp. 0114, 1995

A. W. Roscoe and M. H. Goldsmith, "What is intransitive noninterference", in *12th IEEE Computer Security Foundations Workshop*, 1999.

J. Rushby, "Noninterference, transitivity and channel-control security policies", Tech. Rep. CSL-92-02, SRI International, Menlo Park CA, USA, December 1992.

P. Ryan and S. Schneider, "Process algebra and non-interference", in *Proceedings of CSFW-12*, (Mordano, Italy), IEEE, June 1999.

R. Van Der Meyden, "What, Indeed, is Intransitive Noninterference?", in *Computer Security – ESORICS 2007*, vol. 4734 of LNCS, pp. 235-250, 2008.

D. Von Oheimb, "Information Flow Control Revisited: Noninfluence=Noninterference+Nonleakage", in *Computer Security – ESORICS 2004*, vol. 3193 of LNCS, pp. 225-243, 2004.

M. Yeddes, F. Lin, and N. Ben Hadj-Alouane, "Modifying Security Policies for the Satisfaction of Intransitive Non-Interference", *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1961-1966, 2008.

A. Ben Said al. Computing the Infimal Superlanguage for Generalized Observability. 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC 2011).

C. O'Halloran. A calculus of information flow properties. In European Symposium on Research in Computer Security, 1990.

D. Sutherland. A model of information. In 9th National Security Conference, 1986.

Mazar, L.: Using unification for opacity properties. In: Proceedings of the 4th IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS 2004), Barcelona, Spain, pp. 165-176 (2004)