International Journal of Hybrid Information Technology
Vol.8, No.7 (2015), pp.243-252
http://dx.doi.org/10.14257/ijhit.2015.8.7.23

Analysis and Proofs of Properties in Intransitive Noninterference

Congdong Lv[1], Xiaoyong Li[1], Fei Li[2], and Wei Ma[1]

(1. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
2. College of Computer Science and Technology, Zhejiang University, Zhejiang, China)
lvcongdonglv@163.com

Abstract

The goal of high assurance systems development by formal verification motivates the investigation of techniques whereby a systems design or implementation can be formally shown to satisfy a formal definition of security. The security of unwinding relations provides a proof method that has been applied to establish that a system satisfies noninterference properties, but it requires significant human ingenuity to define an unwinding relation that forms the basis for the proof, and typically also has involved manual driving (proof rule selection) of the theorem proving tool within which the proof is conducted. The property of purge-based definition proposed by Goguen and Meseguer, intransitive purge-based definition proposed by Haigh and Young, and some more definitions TA-secure, TO-secure, ITO-secure proposed by van der Meyden are considered in this paper. The property can be used in the proof of noninterference property without unwinding relations.

Keywords: *Intransitive Noninterference; Information Security; Noninterference Property*

1. Introduction

How to construct a secure system to high levels of assurance? A fundamental method is to decompose the system into trusted and un-trusted components. In architecture, constrains the possible causal effects and flows of information between these components. On the other hand, resource limitations and cost constraints my make it desirable for trusted and un-trusted components to share resources. For example, a data analyst can handle high security and low security information on different machines. But it is cheaper to store the information on one. In designs and implementations of complex systems, Flows of information between trusted and un-trusted components should be controlled because of sharing resources. In order to provide high levels of assurance of implementations, it is desirable to have a formal theory of systems architecture and information flow, so that a design or implementation may be formally verified to conform to an information flow policy.

The problems we consider in this paper address information flow and systems implementation attacks. A (passive) adversary may attack the system by attempting to make subtle deductions from his possible observations of the system, exploiting covert channels that may exist in the system, in order to learn secrets that he is not authorized to possess. The motivation of noninterference model is to provide assurance that the system has been designed in such a way that such attacks are not possible, or to discover such attacks when they exist.

ISSN: 1738-9968 IJHIT

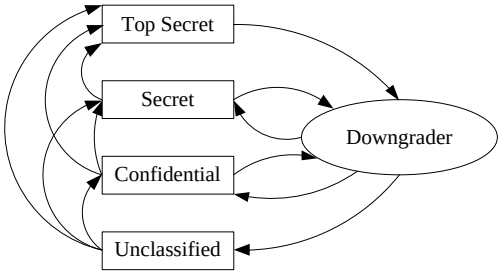

Figure 1. A Down-Grader Architecture

Noninterference Models are one approach to the formalization of information flow and causal relationships firstly introduced by Goguen and Meseguer [1] to deal with information flow in transitive architecture. Noninterference can be express as follow: two domains in an architecture, one domain running a sequence of actions, is non-interfering with another domain if what the fist domain does has no effect on what the second domain can see, that is information can't flow from the first domain to the second domain. For example, figure 1 [2] is a king of system classified architecture. There are four levels, Top Secret, Secret, Confidential and Unclassified. The edges on the left of the figure represent the conventional transitive architecture. The information flow between the classification levels is transitive. Information can flow "upward" in security level without restriction. Information can flow from low classification to higher classification.

Extensions to intransitive architecture, such as down-graders are proposed by Haigh and Young [3], and propounded by Rshby [2]. And more works has been down on the intransitive noninterference [4-10].On the right of figure 1 are edges to and from a special Downgrader domain that are intransitive. The information flow represented by these edges is intransitive because, although information can flow, for example, from the Top Secret to the confidential domain via the Downgrader, it can't flow directly from Top Secret to Confidential. Flow "downward" through the mediation of the presumably trusted Downgrader domain.

Goguen and Meseguer's definition of noninterference based on a $purge$ function. It does not yield the desired conclusions for intransitive architecture. Haigh and Young proposed a variant for intransitive architecture based on an $intransitive-purge$($ipurge$) function [3]. Rushby later refined their theory and developed connections to access control systems [2]. Var der Meyden has argued that the definitions of security for intransitive architecture in these works suffer from some subtle flaws, and proposed some improved definitions, *TA-secure*, *TO-secure* and *ITO-secure*.

The function *purge* and *ipurge* preserve not just certain actions from the original sequence, but also their order. This allows a domain to "know" this order in situations where an intuitive reading of the policy would suggest that it ought not to know this order. *TO-secure* states that a domain may not transmit information that it has not directly observed. *TA-secure* states that a domain's actions may transmit information about previous actions that it is permitted to have, even if it has not actually observed this information. *ITO-secure* allows for a faster transmission of information in that it permits an agent to transmit information by means of the same action by which it acquires that information.

The goal of high assurance systems development by formal verification motivates the investigation of techniques whereby a systems design or implementation can be formally shown to satisfy a formal definition of secure. The secure of unwinding relations [2, 11] provides a proof method that has been applied to establish that a

system satisfies noninterference properties, but it requires significant human ingenuity to define an unwinding relation that forms the basis for the proof, and typically also has involved manual driving (proof rule selection) of the theorem proving tool within which the proof is conducted.

In this paper, we recall noninterference model function *purge*, which is proposed by Goguen and Meseguer [1], $intransitive-purge$(*ipurge*), which is proposed by Haigh and Young [3], TA, TO and ITO which are proposed by Meyden [6]. The property and complexity of them are considered in this paper. The property can be used in the proof of noninterference property without unwinding relation. Contributions of this work can be concluded as follow:

(1) Prove that there exists the shortest action sequence in the function *purge*, *ipurge*, *ta*, *to* and *ito*;

(2) Prove that n-th *purge* and *ipurge* are equivalent to 1-th *purge* and *ipurge*.

The structure of this paper is as follows. In Section 2 we give some backgrounds to introduce the basis of our work. Some definitions are given in Section 3. Section 4 give the property of the function *purge*, *ipurge*, *ta*, *to*, and *ito*. In Section 5, we discuss the complexity of these functions. Next, in Section 6, we discuss the related literature and some concluding remarks are made in Section 7.

2. Backgrounds

Several different types of semantic models have been used in the literature on noninterference. (See [12] for a comparison and a discussion of their relationships.)We work here with the state-observed machine model used by Rushby [2], but similar results would be obtained for other models. This model consists of deterministic machines of the form $< S, s_0, A, step, obs, dom >$, where S is a set of states, $s_0 \hat{I} S$ is the initial state, A is a set of actions, $dom: A a D$ associates each action with an element of the set D of secure domains, $step: S ´ A a O$ is a deterministic transition function, and $obs: S ´ D a O$ maps states to an observation in some set O, for each security domain.

Noninterference is given a formal semantics in transitive case [1] using a definition based on a *purge* function. Given a set $E \hat{I} D$ of domains and a sequence $a \hat{I} A$.

$$purge_m(e) = e$$

$$purge_m(aa) = \begin{cases} a' \text{ if } purge_m(a); dom(a) \quad m \\ purge_m(a); otherwise \end{cases}$$

Definition 2.1(P-secure)

M is P-secure with respect to a policy ® if for all $m \hat{I} D$ and all sequences $a, a' \hat{I} A^*$ with $purge_m(a) = purge_m(a')$, we have $obs(s_0, a) = obs(s_0, a')$.

Haigh and Young [3] generalized the definition of the $purge$ function to intransitive architecture.

$$source(e, m) = \{m\};$$

$$source(aa, m) = \begin{cases} source(a, m) ? \{dom(a)\}; \quad n \text{ 冠 } source(a, m), dom(a) \quad n \\ source(a, m); otherwise \end{cases};$$

$$ipurge_m(e) = e;$$

$$ipurge_m(aa) = \begin{cases} a \,\|\, ipurge_m(a); dom(a) \quad source(aa, m) \\ ipurge_m(a); otherwise \end{cases}$$

Definition 2.2(IP-secure)

M is IP-secure with respect to a policy ® if for all $m \in D$ and all sequences $a, a' \in A^*$ with $ipurge_m(a) = ipurge_m(a')$, we have $obs(s_0, a) = obs(s_0, a')$.

Function *ta* has been noted by van der Meyden [6] that IP-secure classifies some systems as secure where there is, intuitively, an insecure flow of information that relates to a domain learning ordering information about the actions of other domains that it should not have.

$ta_m(e) = e$

$$ta_m(aa) = \begin{cases} (ta_m(a), ta_{dom(a)}(a), a); dom(a) \,\mathrm{®}\, m \\ ta_m(a); otherwise \end{cases}$$

Definition 2.3(TA-secure)

M is TA-secure with respect to a policy ® if for all $m \in D$ and all sequences $a, a' \in A^*$ with $ta_m(a) = ta_m(a')$, we have $obs(s_0, a) = obs(s_0, a')$.

In the definition of TA-secure, the operational model of information flow given by the function ta permits a domain to trnsmit information that it may have, even if it has never observed anything from which it could deduce that information. Arguably, this is too liberal. $view_m : A^* \to (A \cup O)^*$.

$$view_m(e) = obs_m(s_0);$$

$view_m(aa) = (view_m(a) ? b) \circ obs_m(s_0 a)$, where $b = a$ if $dom(a) = m$ or $b = e$ otherwise.

$$to_m : A^* \to T((A \cup O)^*, A)$$

$$to_m(e) = obs_m(s_0);$$

$$to_m(aa) = \begin{cases} (to_m(a), view_m(a), a); dom(a) \,\mathrm{®}\, m \\ to_m(a); otherwise \end{cases}.$$

Definition 2.4(TO-secure)

M is TO-secure with respect to a policy ® if for all $m \in D$ and all sequences $a, a' \in A^*$ with $to_m(a) = to_m(a')$, we have $obs(s_0, a) = obs(s_0, a')$.

A slight variant of this definition is *ito*.

$$ito_m : A^* \to T(O(A \cup O)^*, A)$$

$$ito_m(e) = obs_m(s_0)$$

$$ito_m(aa) = \begin{cases} (ito_m(a), view_m(a), a); dom(a) = m \\ (ito_m(a), view_m(aa), a); dom(a) ? m \\ ito_m(a); otherwise \end{cases}$$

This definition is just like that of to, with the difference that the information that may transmitted to by an action a such that $dom(a) \mathbb{R} n$ but $dom(a)$ ¹ m.

Definition 2.5(ITO-secure)

M is ITO-secure with respect to a policy ℝ if for all $m \hat{I} D$ and all sequences $a, a' \hat{I} A^*$ with $ito_m(a) = ito_m(a')$, we have $obs(s_0, a) = obs(s_0, a')$.

3. Some Definitions for the Property

To facilitate the presentation, we give the following **definitions. For an action sequence, any times of function $purge$ can be done on** the sequence. We define any times sequence as follow.

Definition 3.1(N-th *purge*)

$$purge_m^n(a) = purge_m(purge_m^{n-1}(a));$$

$$purge_m^1(a) = purge_m(a).$$

For an action sequence, any times of function *ipurge* can be done on the sequence. We define any times sequence as follow.

Definition 3.2(N-th *ipurge*)

$$ipurge_m^n(a) = ipurge_m(ipurge_m^{n-1}(a));$$

$$ipurge_m^n(a) = ipurge_m(ipurge_m^{n-1}(a)).$$

If we want to know the length of an action sequence, we should count actions in the sequence. The following function *len* is to get the length of the action sequence, $len: A^*$ a Z_n:

Definition 3.3(*len*)

$$len(a) = m, m ? Z_n.$$

Two action sequences are equivalent if the output of them are the same after doing function X.

Definition 3.4(X)

The action sequence a and the action sequence b are equivalent if

$$X_m(a) = X_m(b).$$

X can be $purge$, $ipurge$, ta, to and ito.

For any action sequences, if they are equivalent, they belong to an equivalence class.

Definition 3.5($[a]_X$)

$[a]_X$ is a equivalence class if it satisfied the following condition:

$$"a ? A^*, [a]_X \quad \{b \mid b \text{槡} A^* \quad X_m(a) = X_m(b)\}.$$

X can be *purge*, *ipurge*, *ta*, *to* and *ito*.

4. The Property of P, IP, TA, TO, ITO-Secure

Some interesting properties of function purge, intransitive purge, TA, TO and ITO are found and proved in this section. For the function purge, a equivalence class has the unique smallest action sequence and n-th *purge* is equivalent to 1-th purge. Function *ipurge* (shorted for intransitive purge) almost has the same properties. A equivalence class of *ipurge* has the unique smallest action sequence and n-th *ipurge* is equivalent to 1-th *ipurge*. *ta*, *to* and *ito* all have the smallest action sequence. We believe that n-th *ta* is also equivalent to 1-th *ta*, n-th *to* is equivalent is equivalent to 1-th *to*, and n-th *ito* is equivalent to 1-th *ito*. These properties can be used in the proof of *P-secure*, *IP-secure*, *TA-secure* and so on. The follow is the properties and the proof of them.

Theorem 4.1

For function *purge*, if $purge_m(a) = a$, then a is the unique smallest action sequence for $[a]_{purge}$.

Proof:

Suppose $a = a_1$ 秔a_n and $purge_m(a) = a$;

Assume $\$b', purge_m(b') = purge_m(a) ? len(b') \quad len(a)$;

Then $\$a_i ? \{a_1, ..., a_n\}$ and $dom(a_i)$ m

But $purge_m(a) = a = a_1...a_n$;

So $"a_i$ 冠$\{a_1, ..., a_n\}, dom(a_i) \quad m$;

This is inconsistent with the launch of the assumptions.

So not $\$b', purge_m(b') = purge_m(a) ? len(b') \quad len(a)$.

That is if $purge_m(a) = a$, then a is the unique smallest action sequence for $[a]_{purge}$.

Theorem 4.2

n-th *purge* is equivalent to 1-th *purge*, that is:

$purge_m^n(a) = purge_m^1(a) = purge_m(a)$.

Proof:

Suppose $purge_m(a) = a_1...a_n$;

Then $purge_m^1(a) = purge_m(a) = a_1...a_n$

Then $purge_m^2(a) = purge_m(purge_m(a)) = purge_m(a_1...a_n) = a_1...a_n$;

Assume $purge_m^n(a) = a_1...a_n$;

Then $purge_m^{n+1}(a) = purge_m(purge_m^n(a)) = purge_m(a_1...a_n) = a_1...a_n$;

So $purge_m^n(a) = purge_m^1(a) = purge_m(a)$.

Theorem 4.3

For function *ipurge*, if $ipurge_m(a) = a$, then a is the unique smallest action sequence for $[a]_{ipurge}$.

Proof:

Suppose $a = a_1 \cdots a_n$ and $ipurge_m(a) = a$;

Assume $\exists b', ipurge_m(b') = ipurge_m(a) \wedge len(b') < len(a)$;

Then $\exists a_i \in \{a_1,...,a_n\}$ and $dom(a_i) \in source_m(a_{i+1}...a_n)$

But $ipurge_m(a) = a = a_1...a_n$;

So $\forall a_i \in \{a_1,...,a_n\}, dom(a_i) \notin source_m(a_{i+1}...a_n)$;

This is inconsistent with the launch of the assumptions.

So not $\exists b', ipurge_m(b') = ipurge_m(a) \wedge len(b') < len(a)$.

That is if $ipurge_m(a) = a$, then a is the unique smallest action sequence for $[a]_{purge}$.

Theorem 4.4

n-th *purge* is equivalent to 1-th *purge*, that is:

$ipurge_m^n(a) = ipurge_m^1(a) = ipurge_m(a)$.

Proof:

Suppose $ipurge_m(a) = a_1...a_n$;

Then $ipurge_m^1(a) = ipurge_m(a) = a_1...a_n$

Then $ipurge_m^2(a) = ipurge_m(ipurge_m(a)) = ipurge_m(a_1...a_n) = a_1...a_n$;

Assume $ipurge_m^n(a) = a_1...a_n$;

Then $ipurge_m^{n+1}(a) = ipurge_m(ipurge_m^n(a)) = ipurge_m(a_1...a_n) = a_1...a_n$;

So $ipurge_m^n(a) = ipurge_m^1(a) = ipurge_m(a)$.

Theorem 4.5

For function *ta*, there exists the smallest action sequence a:

$\forall b \in A^*$, if $ta_m(b) = ta_m(a)$, then $len(b) \geq len(a)$.

Proof:

Suppose the smallest sequence is $a = a_1...a_n$;

Then $\forall a_i \in \{a_1,...,a_n\}$, $dom(a_i) \in dom(a_{i+1}) \vee dom(a_i) \in dom(a_{i+2}) \vee ... \vee dom(a_i) \notin m$;

Assume $\exists b \in A^*$, $ta_m(b) = ta_m(a)$, and $len(b) < len(a)$;

Then $\exists a_i \in \{a_1,...,a_n\}$, $dom(a_i) \rightarrow dom(a_{i+1}) \wedge ... \wedge dom(a_i) \rightarrow dom(a_n) \wedge dom(a_i) \rightarrow m$, which conflicts with the existing condition.

So exist the smallest sequence a for the equivalent class $[a]_{ta}$.

Theorem 4.6

For function to, there exists the smallest action sequence a:
$\forall b \in A^*$, if $to_m(b) = to_m(a)$, then $len(b) \geq len(a)$.

Proof:

Suppose the smallest sequence is $a = a_1...a_n$;
Then $\forall a_i \in \{a_1,...,a_n\}$, $dom(a_i) \in m$;
Assume $\exists b \in A^*$, $to_m(b) = to_m(a)$, and $len(b) < len(a)$;
Then $\exists a_i \in \{a_1,...,a_n\}$, $dom(a_i) \notin m$, which conflicts with the existing condition.
So exist the smallest sequence a for the equivalent class $[a]_{to}$.

Theorem 4.7

For function ito, there exists the smallest action sequence a:
$\forall b \in A^*$, if $ito_m(b) = ito_m(a)$, then $len(b) \geq len(a)$.

Proof:

Suppose the smallest sequence is $a = a_1...a_n$;
Then $\forall a_i \in \{a_1,...,a_n\}$, $dom(a_i) \in m$;
Assume $\exists b \in A^*$, $ito_m(b) = ito_m(a)$, and $len(b) < len(a)$;
Then $\exists a_i \in \{a_1,...,a_n\}$, $dom(a_i) \notin m$, which conflicts with the existing condition.
So exist the smallest sequence a for the equivalent class $[a]_{ito}$.

5. Related Work

Our work is mainly about the property in noninterference models. Noninterference was first proposed by Goguen and Meseguer [1] based on function *purge* which was used to deal with information flow in transitive architecture. But it couldn't deal with information flow in intransitive architecture, such as downgrader. Haigh and Young proposed a variant for intransitive architecture based on an "intransitive purge" function [3]. Rushby [2] later refined their theory and developed connections to access control. Van der Meyden [6] has argued that the definitions of security for intransitive architecture in these works suffer from some subtle flaws, and proposed some improved definitions, *TA-secure*, *TO-secure* and *ITO-secure*. The revised definition can be shown to avoid the subtle flaws in the intransitive purge-based definition and lead to a more satisfactory proof of theory and connection to access control systems than in Rushby's work.

The technique of unwinding relations [2, 11] provides a proof method that has been applied to establish that a system satisfied noninterference properties, but it requires significant human ingenuity to define an unwinding relation that forms the basis for the proof, and typically also has involved manual driving(proof rule selection) of theorem proving tool within which the proof is conducted. To verify the property by fully automatic techniques is more acceptable. There is a substantial body of work on automated verification techniques for transitive noninterference properties, but there has been significantly less work on automatic verification techniques for intransitive noninterference properties. Recently, more works has been down on the application of noninterference. Roscoe and Huang [8] applied noninterference on Timed CSP. Engelhardt *et al.* [13] detected noninterference in nondeterministic systems. Eggert et

al. [14] considered complexity and unwinding of noninterference but they also depend on unwinding relations. Meyden and Vanflect applied the noninterference in architecture refinement [15-16].

6. Conclusion

In this paper, properties of function *purge*, *ipurge*, *ta*, *to* and *ito* are found and proved. For the function *purge*, an equivalence class has the unique smallest action sequence and n-th *purge* is equivalent to 1-th *purge*. Function *ipurge* (shorted for intransitive purge) almost has the same properties. An equivalence class of *ipurge* has the unique smallest action sequence and n-th *ipurge* is equivalent to 1-th *ipurge*. *ta*, *to* and *ito* all have the smallest action sequence. These properties can be used in the proof of the secure.

In the future, more works can be done on the noninterference models, include complexity of them. We believe that n-th *ta* is also equivalent to 1-th *ta*, n-th *to* is equivalent is equivalent to 1-th *to*, and n-th *ito* is equivalent to 1-th *ito*. The three functions need to change for the proof. They are also part of our future work.

Acknowledgements

The project supported by the Higher Specialized Research Fund for the Doctoral Program ordered Ministry of Education of China (No. 20120009110007), 2012 MOR Scientific Research and Development Program ordered by Ministry of Railways of China (Now is China Railway Company) (No. 2012X010-B), and Program for Innovative Research Team in University of Ministry of Education of China(No. IRT 201206).

References

[1] J. A. Goguen and J. Mesegure, "Security policies and security models", In Proc.1982 Symposium on Security and Privacy, **(1982)** 11-20; Oakland, CA.
[2] J. Rushby, "Noninterference, transitivity, and channel-control security policies", SRI International, Computer Science Laboratory **(1992)**.
[3] J. T. Haigh and W. D. Young, "Extending the Noninterference Version of MLS for SAT", IEEE Trans. Software Eng., vol. 13, no. 2, **(1987),** pp. 141-150.
[4] M. Denford, J. Leaney and T. O'Neill, "Non-functional refinement of computer based systems architecture", Engineering of Computer-Based Systems, **(2004),** pp. 168-177.
[5] N. B. Hadj-Alouane, S. Lafrance and F. Lin, "On the verification of intransitive noninterference in multi-level security", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 35, no. 5, **(2005),** pp. 948-958.
[6] R. Van Der Meyden, "What, indeed, is intransitive noninterference?", Computer Security-ESORICS 2007. Springer Berlin Heidelberg, **(2007),** pp. 235-250.
[7] G. Barthe, D. Pichardie and T. Rezk, "A certified lightweight non-interference Java bytecode verifier", Mathematical Structures in Computer Science, vol. 23, no. 5, **(2013),** pp. 1032-1081.
[8] A. W. Roscoe and J. Huang, "Checking noninterference in Timed CSP", Formal Aspects of Computing, vol. 25, no. 1, **(2013),** pp. 3-35.
[9] M. Hennessy, "The Security Picalculus and Non-interference", Electronic Notes in Theoretical Computer Science, vol. 83, **(2013),** pp. 113-129.
[10] T. S. Hoang, A. K. McIver and L. Meinicke, "Abstractions of non-interference security: probabilistic versus possibilistic", Formal Aspects of Computing, vol. 26, no. 1, **(2014),** pp. 169-194.
[11] J. A. Goguen and J. Meseguer, "Unwinding and inference control", IEEE Symp. on Security and Privacy, **(1984),** pp. 75-75.
[12] R. Van Der Meyden and C. Zhang, "A comparison of semantic models for noninterference", Formal Aspects in Security and Trust. Springer Berlin Heidelberg, **(2007),** pp. 235-249.
[13] K. Engelhardt, R. Van der Meyden and C. Zhang, "Intransitive noninterference in nondeterministic systems", In: ACM Conference on Computer and Communications Security, **(2012),** pp. 869-880.
[14] S. Eggert, R. Van der Meyden, Schnoor H. Complexity and Unwinding for Intransitive Noninterference. arXiv preprint arXiv:1308.1204 **(2013)**.

[15] R. Van Der Meyden, "Architectural refinement and notions of intransitive noninterference", Formal Aspects of Computing, vol. 24, nos. 4-6, **(2012),** pp. 769-792.
[16] W. M. Vanfleet, R. W. Beckwith and B. Calloni, "MILS: architecture for high assurance embedded computing", CrossTalk, vol. 18, no. 8, **(2005),** pp. 12-16.

Author

Congdong Lv was born in 1987. He got his Bachelor Degree from Nanjing Normal University. Now, he is a PhD candidate in Beijing Jiaotong University. His research interests include information security, cloud security and formal method.

