Sindh Univ. Res. Jour. (Sci. Ser.) Vol.44 (2) 333- 338 (2012)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Unified Modelling Language as an Architectural Description Language for Federated Identity Management

H. A. NIZAMANI, Q.U.A NIZAMANI, F. H. CHANDIO, M. S. CHANDIO, N. H. ARIJO*, I. A. KOREJO

Institute of Maths and Computer Science, University of Sindh, Jamshoro, Pakistan
Email: q_agro@hotmail.com : chandiofida@gmail.com: mschandio@hotmail.com: niaz_arijo@hotmail.com: cskorejo@gmail.com

Corresponding author: H. A. NIZAMANI, Email: hanizamani@gmail.com Ph. No. +92-229213177

Received 28th May2012 and Revised 10th June 2012

Abstract We use UML to model architectural aspects of federated identity management systems (FIMs). Specifically, we apply two modelling approaches: the UML "as-is" approach and a "profile-based" approach recently proposed in the literature. Adopting FIMs as a realistic case study, we analyse the suitability of the Unified Modelling Language (UML) as an architectural description language (ADL) in both approaches.

Keywords: UML, Federated Identity Management, Software Architecture, and Architectural Description Language

1. INTRODUCTION

In this paper we study how UML could be used to model architectural aspects of federated identity management systems (FIMs). To this purpose, we follow two different approaches. The first approach is the so called UML "as-is" approach which exploits only the basic linguistic features of UML to express architectural aspects of systems. The second approach is based on the recent proposal in (Bruni *et al.,* 2009) that introduces profile (inspired by the formal framework introduced in (Bruni *et al.,* 2009a,2009b)) specifically designed to support architectural modelling.

Notably, FIMs are becoming more and more appealing as they enable organisations to smoothly join up and share distributed services or resources. More precisely, FIMs make users' authentication information available in a global context so that organisations can easily combine their services. For instance, a university may federate with a digital library to grant students access to some electronic publications.

Instead of considering just general aspects of architectural description languages (ADL), we focus also on peculiar architectural issues arising in the context of FIMs. We contend that the architectural modelling of FIMs offers a relatively complete range of challenging issues common to many other realistic scenarios. For instance, architectural styles and reconfigurations are paramount for FIMs due to the increasing quest for dynamically configurable FIMs. Such architectural aspects of FIMs impact on the security threats they are exposed to (cf. § 2); therefore, it is crucial to precisely describe architectural views of FIMs.

Besides the two actual UML models we give in § 3, the main contributions of our study is an analysis, given in § 6, of the two models according to the criteria discussed in § 5 where we distinguish between "general" and "pattern specific" criteria. The former pertain to aspects of software architectures common to all sufficiently complex systems. The latter can be considered as relevant to FIMs. Nonetheless, we argue that pattern-specific criteria could be germane to other classes of systems as they mainly concern generation and identification of actual systems' configurations required to satisfy architectural scheme.

2. FIM PATTERNS

Federated Identity Management (FIM) -like (Gupta, *et al.*, 2008, McKenzie, *et al.*, 2008, Chadwick, *et al.*, 2009a, 2009b, Avetisyan, *et al.* , 2010) to mention but a few- enables organisations to "federate" and share distributed services or resources. The notion of Circle of Trust (CoT) is key to FIMs and permits to establish common access policies. Precisely, a CoT in FIMs can be described as a federation of identity providers (IDPs) and service providers (SPs). To access resources of SPs, users provide identities verified by the IDPs. In this way, users of different organisations can be authenticated when accessing remote resources in their CoT (Oltsik, 2006).

*Institute of Information and Communication Technology, University of Sindh, Jamshoro, Pakistan

In (Nizamani and Tuosto, 2010) we formalised a few FIM patterns: Bilateral Federation (BF), Multiple IDPs Federation (MIF), Multiple SPs Federation (MSF), Arbitrary Federation (AF), and chain of CoTs. All but last patterns are taken from (Kylau, *et al.,* 2009). Such patterns are distinguished according to trust management and the security threats they are under. We now give a brief account of each pattern and its threats.

In BF, a single IDP and a single SP federate; the access to the SP services is mediated by the IDP. The IDP may acquire knowledge on users' behaviour and this is regarded as a threat to users' privacy. Also, the SP might autonomously decide to disclose (i.e., to other SPs) such information.

In MIF, a single SP is federated to multiple IDPs; users register at several IDPs notify the SP about which IDPs will be used for authentication. The additional threat with respect to the BF pattern is that some IDPs might decide to crosscheck the information about the accesses to the SP.

In MSF, a single IDP is federated to multiple SPs; in such case, delegation of user authentication is typically necessary because an invoked SP may need to provide users' credentials upon invocation of other services in the federation. The additional threats with respect to the MIF pattern include "collusion" of SPs to accumulate identity information and unauthorised delegation of authentication. Collusion is harmful as it would allow SPs to correlate their information and accumulate data on users.

Pattern AF is the most vulnerable as it allows the free combination of the patterns BF, MIF, and MSF and is exposed to all their threats. In a chain of CoTs two or more CoTs are connected to form a chain of FIMs where users from an IDP of one CoT may access SPs in other CoTs in the chain. This pattern is exposed to the same threats of AF.

3. DESCRIBING FIMS STYLE

In general, a software architecture (SA) yields a high-level description of a system given in terms of components, connectors, and their composition. Architectural styles (Kim and Garlan, 2010) are one of the pillars of SA. A style defines a family of systems - like the FIMs patterns in § 2- by providing a common design vocabulary together with suitable constraints. The vocabulary of a style enumerates the types of architectural elements used to build architectural configurations which may be used to analyse the system. For instance, in our context, we would like to validate the configuration of a CoT that adheres a FIMs pattern. For this, one has to check conformance of the FIMs configurations against these patterns. Constraints specify "legal" configurations. For example, a valid FIMs configuration requires at least one IDP and one SP. Style checking validates a configuration with respect to its style. Therefore, suitable mechanisms to support styles and style checking are needed.

We model structural aspects of FIMs patterns described in § 2 by describing their style. Hereafter, we consider two approaches: the UML "as-is" (cf. § 3.1) uses the standard UML notations; the other approach (cf. § 3.2) is based on the UML profile given in (Bruni, *et al.,* 2009).

3.1 USING UML "as-is"

The class diagram in **(Fig. 1a) (Fig. 1b)** models FIM components, their ports, and their associations. The components of FIM (cf. § 2) are captured by the classes

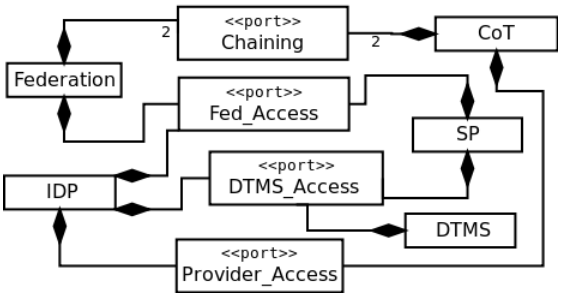

Fig. 1a FIM architectural elements

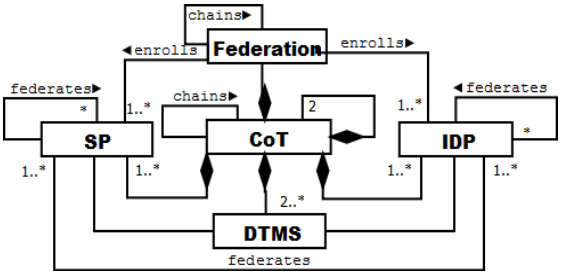

Fig. 1b Metamodel of FIM systems

CoT, Federation, SP, IDP, and DTMS. (The latter class deals with dynamic security and trust management and is not relevant in this paper.) The stereotype «port» singles out classes representing ports, namely Chaining, Fed_Access, DTMS_Access, and Provider_Access. For instance, an IDP may interact with SPs using a Provider_Access port. Likewise, IDPs and SPs communicate with a Federation through Fed_Access ports. Also, a Federation and a CoT may respectively interact with other Federations and CoTs using Chaining ports.

Remarkably, SA components have a composition association with the attached ports and the constraint[1] (**GC0**) is needed on **(Fig. 1a)**.

GC0: *a unique instance of a port is attached to a single component.*

Also, associations may partially define constraints of FIMs styles like in **(Fig. 1b)** which describes[2] two different architectural views of a CoT. Precisely, a view represents a CoT that models a federation of providers by considering patterns BF, MIF, MSF, and AF while the other describes a CoT as a chain of CoTs. Note that, it is difficult to clearly describe the constraint between two mutually exclusive sets of associations for modelling such a CoT in **(Fig. 1b)**. This can be accommodated by the constraint:

GC1: *a CoT may consist of either a Federation together with its associated IDPs and SPs, or CoTs attached to each other in a chain.*

In **(Fig. 1b)**, a Federation is associated with at least one IDP and SP, and several DTMSs. An IDP is associated to one DTMS, zero or more other IDPs (and at least one SP). Similarly, an SP is associated to one DTMS, zero or more other SPs (and at least one IDP). Besides those multiplicity constraints common to FIMs, one may need to specify the constraints given below to restrict valid models of interest:

GC2: *each instance of type IDP and SP associated to a Federation should be within the CoT.*

GC3: *each instance of type IDP and SP should be associated to the rest of instances of type IDP and SP associated to the Federation.*

GC4: *all instances of type DTMS attached to IDPs and SPs should be within the CoT.*

GC5: *a unique instance of a DTMS is attached to a single IDP (and SP).*

Finally, observe that **(Fig. 1b)** uses two different kinds of associations over a CoT. In fact, a composition association is used to describe that a CoT may consist of two CoTs while a self-association chains is used to describe that CoTs are attached to each other within the CoT to form a chain. (A similar association is defined over a Federation.) As a result, **(Fig. 1b)** yields a model that enables one to create a complex CoT.

[1] Constraints can suitably be given in OCL. However, a textual description of constraints suffices for our purposes.
[2] For simplicity, ports are not represented in **(Fig. 1b)**.

3.2 USING UML PROFILE

We use the profile in (Bruni, et al., 2009) to model FIMs. Consider **(Fig. 2)** where the top-left class diagram has two «production» components Chain and Fed associated to a «refineable» component CoT. A CoT can be replaced by configurations corresponding to the productions Chain and Fed. A production in the profile represents a composeable pattern in the corresponding structure diagram (which gives the internal structure of the CoT). More precisely, a structure diagram describes the legal connections between the components of a «refineable» component. In our case, the configurations of a CoT can be generated using the productions Chain and Fed given in **(Fig. 2)**. To replace a CoT in production Chain with the configuration, one may freely use production Chain to generate a complex chain of CoTs, or production Fed to generate the other configurations.

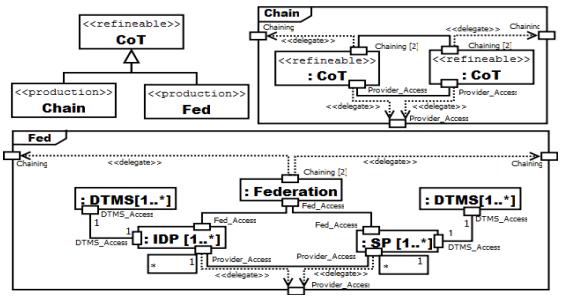

Fig. 2 Architectural style productions for FIMs

The former production is the top-right structure diagram in **(Fig. 2)**; it states that each CoT component has two Chaining ports and a Provider_Access one. The Chaining ports enable CoTs to chain up while the latter ports connect providers. The production Fed is the bottom diagram in **(Fig. 2)**; it describes a CoT as a federation of providers in a general way capturing patterns BF, MIF, MSF, and AF.

As described earlier, a legal FIMs configuration is subject to some constraints. This is rendered by decorating (parts of) structure diagrams with multiplicities. For instance, in production Fed of **(Fig. 2)**, a Federation is attached to at least one IDP and one SP through a Fed_Access port and each provider uses a different DTMS via a DTMS_Access port. Each IDP connects to a (possibly empty) set of IDPs through a Provider_Access port by a connector rendering a self-association on IDP. Similarly, each SP is attached to zero or more SPs. Each IDP is attached to at least one SP and each SP is attached to at least one IDP through Provider_Access ports.

We remark that for FIM architectures, it is not always convenient to infer the multiplicity of connectors from the multiplicity of the types of instances (as typical

in UML designs). The reason being that the same structure diagram is used to specify constraints pertaining to different architectural levels. In fact, the multiplicities on connectors refer to instances of actual configurations while those in classes pertain to architectural constraints of enclosing classes.

Example 1 *Production Fed in **(Fig. 2)** imposes a 1-to-1 association between DTMSs and IDPs components. Accordingly, CoTs may be composed by many DTMSs and many IDPs.*

In Example 1 the constraints of DTMS and IDP refer to the architectural aspects of the enclosing class CoT, while the connector between the two classes specifies that there is a 1-to-1 association among DTMS and IDP components in a legal configuration. Notice that this implies that UML forces the designer to explicitly consider the multiplicities of connectors as they may 'conflict' with those of the classes when the associations need special conditions.

An advantage of structure diagrams to describe the style is that they may reduce the use of OCL constraints. In fact, it may not be required to give OCL constraints over composition associations (e.g., GC4) and the association between a structured classifier (e.g., CoT) and its parts (e.g., DTMS, SP, Federation, and IDP) may be implicitly represented. OCL constraints may still be required on other associations; for instance, the structure diagram Fed in **(Fig. 2)** requires GC3 and the additional constraint

GC5: *each instance of type IDP and SP should be associated to a Federation.*
that is very similar to GC2 but eliminates CoT.

4. CREATING FIM CONFIGURATIONS

We discuss, for both the UML "as-is" and the profile-based approaches, how FIM configurations can be obtained.

UML "as-is" We create a few FIM configurations using object diagrams where objects model components and links between objects specify potential communication capabilities of components. For simplicity, ports of components are not considered in these diagrams. **(Fig. 3)** represents FIM configurations of the model given in § 3.1. Specifically, **(Fig. 3a)** shows a configuration of pattern BF where a CoT consists of a Federation having a single IDP and a single SP attached together with their DTMSs. Similarly, **(Fig.3b)** describes a configuration of pattern MIF.

UML Profile Object diagrams are typically used to model simple scenarios (e.g., pattern BF) and are less suitable for complex configurations (e.g., chain of CoTs). To model such a configuration, we use a structure diagram (at instance level) that may impose a hierarchy over a complex system. These diagrams are essentially object diagrams for the classes (e.g., CoT) with internal structure (Miles and Hamilton, 2006).

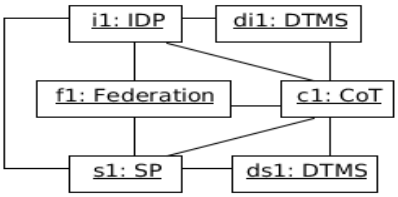

Fig. 3a BF configuration

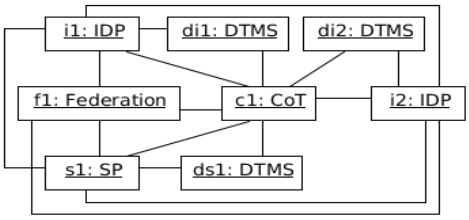

Fig. 3b MIF configuration

The structure diagram in **(Fig. 4)** (names are omitted for readability) represents a configuration of a chain of two CoTs of pattern BF generated by using the corresponding productions in **(Fig. 2)**.

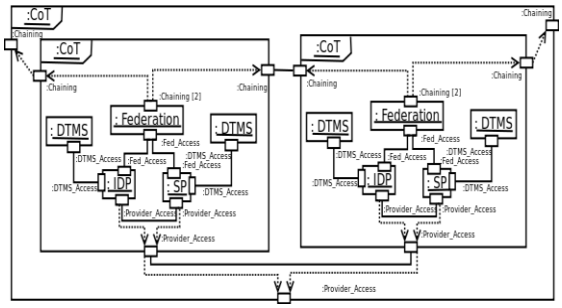

Fig. 4 Structure diagram showing a configuration

In fact, a «refineable» component CoT is replaced with two CoTs attached to each other in a chain. Each CoT in this chain is further refined with configuration of pattern BF by applying the production Fed. Observe that this refinement allows us to deal with the design (at type level) of a complex system (e.g., chain of CoTs). However, designers have to use rules of thumb to generate UML configurations (at instance level) of the FIM patterns.

5. EVALUATION CRITERIA

We choose to evaluate the suitability as ADLs of the UML "as-is" and the profile-based approaches according to general and pattern-specific criteria. This section motivates our choice illustrating the relevance of the criteria to FIM.

General criteria Besides the obvious criterion for any ADL to provide mechanisms supporting core architectural concepts, the general criteria of interest concern *(i) architectural styles, (ii) style checking, (iii), and refinement.* We comment on their relevance to FIMs.

(i) As said, styles are quite important to deal with complex SA. In particular, for FIM patterns styles are paramount as they allow the designer to precisely characterise such patterns so as to control the security threats of a configuration.

(ii) To fully exploit the feature of styles, the designer should be supported with features allowing him/her to validate a configuration against a style. For FIM patterns this corresponds to be able to classify configurations as well as to decide which productions or reconfigurations to apply.

(iii) Architectural refinement is a convenient way to handle complex SAs. The intuition is the designer considers high-level descriptions of architectural elements and refines abstract SAs into actual configurations in several steps. In FIM, refinement allows concepts like circle of trust, federation, etc., to be clearly separated and related with each other. For instance, an abstract CoT may represent a complex configuration of a FIMs or possibly a chain of CoTs. Style-based refinement (Garlan, 1996) requires precise refinement rules yielding valid refinement relations among abstract designs and concrete ones. Style-based refinement is germane to FIM as it allows to e.g. relate abstract components like CoT to actual configurations forming circles of trust.

Pattern-specific criteria The most relevant evaluation criteria specific to FIM patterns concern *(i) pattern generation and (ii) pattern identification.* Regarding (i), for FIMs it is paramount to have precise mechanisms for generating configurations because FIM patterns impact on security requirements (Kylau, *et al.*, 2009). In fact, one may need to enforce specific security measures for configurations according to the underlying pattern. When such mechanisms are available, specific guidelines about the composition of architectural configurations can be given in order to respect a given style. This allows designers to consider the required security mechanism beforehand for FIM configurations. Therefore, an ADL suitable for FIM should provide mechanisms to specify how to instantiate specific instances (e.g., configurations of pattern MIF) of the styles.

Regarding (ii), a crucial aspect of FIM systems is their high degree of dynamicity. More precisely, CoTs can add/remove new IDPs or SPs at run-time and this would require a reconfiguration of the system, but also at design-time it might be necessary to reconfigure architectural views of a FIM system. In the latter case, one needs to identify the actual pattern of the system of interest in order to (a) assess its security threats and (b) to decide when a given reconfiguration could be applied. For instance, adding an SP to a system of pattern BF will reconfigure it into one of pattern MSF. Notice that such a change in the configuration has a direct effect on the security requirements. Hence, pattern identification techniques are paramount for FIM and calls for suitable mechanisms to support designers.

6. UML AS AN ADL (FOR FIM)

We now evaluate how suitable is UML as an ADL for FIM. We mainly follow the approaches to UML-as-ADL in the literature (notably (Medvidovic *et al.*, 2002, Ivers, *et al.*, 2004, Bruni, *et al.*, 2009)).

In brief, our study shows that components and connector can suitably represented as UML classes or components object or structure diagrams can conveniently represent configurations the combination of multiplicities and OCL constraints to represent architectural styles makes UML cumbersome to use; in fact, style checking is problematic in presence of OCL constraints and requires to define style checking support according to UML profiles. In addition to this, UML is not suitable to express architectural refinements as UML is designed to model object orientation UML does not provide any mechanism to generate actual configurations abiding by a given architectural style identifying patterns in complex UML configurations can be difficult.

7. CONCLUSION

We have applied and compared two different methodologies to use UML as an ADL in the concrete scenarios of FIMs. Our investigation spots a few merits and drawbacks of each methodology. It is worth to remark that ADLs are generally capable of dealing with run-time aspects of systems which we have not considered here.

In (Nizamani and Tuosto, 2010) architectural aspects of FIMs are modelled in a formal language based on graph transformations. Such formal language has inspired the UML profile in (Bruni, *et al.*, 2009); a precise comparison between the formal model in (Nizamani and Tuosto, 2010) and the profile-based model presented here is under development. Also, some work on modelling reconfiguration in UML for FIM is under consideration. We remark that the model in (Nizamani and Tuosto, 2010) does not suffer the drawbacks of the profile-based approach highlighted

here. It would be desirable to refine the profile-based model in (Bruni, *et al.,* 2009) to tackle such issues.

As mentioned throughout the paper, UML has been promoted as an ADL and used in specific contexts. We point out that, as far as we are aware of, there are no work that consider a realistic case study as FIMs.

Different approaches such as (Pérez-Martínez, 2003) and (Kacem, *et al.,* 2006) change the UML metamodel to directly support ADL features by introducing new kind of diagrams. As observed in (Medvidovic, *et al.,* 2002), such an approach uses the notations which will not conform to the UML standard. Therefore, albeit those approaches provide interesting research directions, they significantly deviate from the UML specification.

REFERENCES:

Avetisyan, A.I. (2010) 'Open Cirrus: A Global Cloud Computing Testbed' Computer, 43 **(4):** 35-43

Bruni, R., M. Hölzl, N. Koch, A. L. Lafuente, P. Mayer, U. Montanari, A. Schroeder, and M. Wirsing (2009) 'A Service-Oriented UML Profile with Formal Support' In Proceedings of the 7th International Joint Conference on Service-Oriented Computing, Springer-Verlag, Berlin, Heidelberg, 455-469.

Bruni R., A. L. Lafuente, U. Montanari, E. Tuosto (2008a) 'Service oriented architectural design' In Proceedings of the 3rd International Symposium on Trustworthy Global Computing (TGC'07), LNCS **(4912):** 186–203.

Bruni, R., A. L. Lafuente, U. Montanari, and E. Tuosto, (2008b) 'Style-Based Architectural Reconfigurations' In EATCS Bull. **(94):** 161-180.

Chadwick, D. (2009a) 'Federated identity management'. In Foundations of Security Analysis and Design V, LNCS **(5705):** 96-120, Springer-Verlag, Berlin, Heidelberg.

Chadwick, D. and G. Inman (2009b) 'Attribute aggregation in federated identity management' Computer 24 **(5):** 33-40.

Garlan, D. (1996) 'Style-based refinement for software architecture' In the proceedings of ISAW-2, ACM, NY, USA, 72-75.

Gogolla, M., F. Büttner, and M. Richters (2007) 'Use: A uml-based specification environment for validating UML and OCL' Sci. Comput. Program. 69 **(1-3):** 27-34.

Gupta, M., and R. Sharman (2008) Dimensions of Identity Federation: A Case Study in Financial Services. JIAS **(3):** 244-256.

Ivers, J., P. Clements, D. Garlan, R. Nord, R. Schmerl, J. Silva (2004) Documenting Component and Connector Views with UML 2.0. Technical Report CMU/SEI-2004-TR-008, SEI, CMU.

Kacem, M., A. Kacem, M. Jmaiel, and K. Drira (2006) 'Describing dynamic software architectures using an extended UML model' ACM SIGAPP, France, 1245-1249.

Kim, J., and D. Garlan (2010) 'Analyzing architectural styles' Journal of Systems and Software 83 **(7):** 1216-1235.

Kylau, U., I. Thomas, M. Menzel, and C. Meinel (2009) 'Trust Requirements in Identity Federation Topologies' In the proceedings of AINA, IEEE Computer Society, USA, 137-145.

McKenzie, R., M. Crompton, and C. Wallis (2008) 'Use cases for identity management in e-government' IEEE Security and Privacy 6 **(2):** 51-57.

Medvidovic, N., D. Rosenblum, D. Redmiles, and J. Robbins (2002) 'Modeling software architectures in the Unified Modeling Language' ACM Trans. Softw. Eng. Method., **(11):** 2-57

Miles, R. and K. Hamilton (2006) 'Learning UML 2.0'. O'Reilly Media, Inc.

Nizamani, H. and E. Tuosto (2010) 'Federated Identity Management System Patterns as Architectural Reconfigurations' ECEASST (31).

Oltsik, J. (2006) 'Services-Oriented Architecture (SOA) and Federated Identity Management (FIM)' White paper, ESG.

OMG (2009) 'Unified modeling language (version 2.2)' viewed on 03/08/2011 www.omg.org/spec/UML/2.2/Superstructure/PDF/.

Pérez-Martínez, J. (2003) 'Heavyweight extensions to the UML metamodel to describe the C3 architectural style' SIGSOFT Softw. Eng. Notes, **(28):**5Pp.

Roh, S., K. Kim, and T. Jeon (2004) 'Architecture Modeling Language based on UML2.0' In the proceedings of APSEC'04 **(7):** 667-669, IEEE Comp. Society.