Formal verification of secure group communication protocols modelled in UML

P. de Saqui-Sannes · T. Villemur · B. Fontan · S. Mota · M. S. Bouassida · N. Chridi · I. Chrisment · L. Vigneron

Abstract The paper discusses an experience in using Unified Modelling Language and two complementary verification tools in the framework of SAFECAST, a project on secured group communication systems design. AVISPA enabled detecting and fixing security flaws. The TURTLE toolkit enabled saving development time by eliminating design solutions with inappropriate temporal parameters.

Keywords UML · Formal verification · Security · Real-time · Group protocols

1 Introduction

Secured group communication systems, or SGC for short, implement group management functions and communication services. The complexity level reached by SGCs has stimulated research work on dedicated modelling and formal verification techniques. The Unified Modelling Language (UML) enables system formalization and opens new avenues for formal verification of SGC models against security flaws and timing errors.

The paper shares an experience in joint application of UML and formal verification tools to SGC design. The SGC is modelled using an extended UML that contains sufficient information to derive two formal models in HLPSL (High Level Protocol Specification Language [1]) and TURTLE (Timed UML and RT-LOTOS Environment [2]), respectively. The AVISPA [1] tool uses the Dolev-Yao intruder model [3] to detect security flaws. The TURTLE toolkit, or TTool for short,[1] checks TURTLE models against temporal requirements. The SGC protocol designed in the framework of SAFECAST project [4] serves as running example throughout the paper.

The paper is organized as follows. Section 2 defines a UML method that captures requirements using SysML requirement diagrams, and extends UML to achieve use-case driven analysis and object-oriented design. Section 3 introduces requirement, analysis and design patterns that apply to a broad variety of SGCs. Section 4 presents the UML model of the SAFECAST SGC. The latter is hierarchically organized, and consequently members may be upgraded or downgraded. Section 5 addresses the *Upgrade* service and discusses the benefits of using two complementary verification tools (AVISPA and TURTLE). Section 6 surveys related work. Section 7 concludes the paper.

P. de Saqui-Sannes · T. Villemur ( )
CNRS; LAAS, 7 avenue du colonel Roche,
31077 Toulouse, France
e-mail: villemur@laas.fr

P. de Saqui-Sannes · T. Villemur
Université de Toulouse; UPS, INSA, INP, ISAE; LAAS,
31077 Toulouse, France

B. Fontan
THALES Research and Technology France,
Campus Polytechnique, 1 Avenue Augustin Fresnel,
91767 Palaiseau Cedex, France

S. Mota
ITESM, Campus Toluca, Eduardo Monroy Cárdenas #2000,
San Antonio Buenavista, Toluca, Mexico

M. S. Bouassida
CNRS, Laboratoire Heudiasyc UMR 6599, Compiègne, France

N. Chridi · I. Chrisment · L. Vigneron
LORIA, University of Nancy, Campus Scientifique, BP 239,
54506 Vandoeuvre-lés-Nancy Cedex, France

[1] http://labsoc.comelec.enst.fr/turtle/ttool.html.

2 UML method

The UML standard defines a notation, not a method. The paper promotes the use of verification-centric methods that enable early detection of design errors. The purpose is not to cover the entire design trajectory from requirement capture to maintenance, but to emphasize on the early stages of that trajectory.

2.1 Overview

The OMG-based UML does not provide any diagram to capture requirements. The method depicted in Fig. 1 imports SysML requirement diagrams. In SysML, requirements remain informal, which hampers formal checking of design models against user or system requirements. The solution proposed for SGC design is to include logic formulas and chronograms into requirement diagrams in order to formalize security and temporal requirements, respectively.

Use-case driven analysis enables to specify the system by its boundary, the set of actors it interacts with, and the function or services it is expected to provide. Use-cases are documented by scenarios expressed in terms of sequence diagrams. Analysis diagrams contain annotations that contribute to achieve security and temporal requirement traceability.

Object-oriented design enables to model the system's architecture. An active class has a behaviour described by a state-machine which contains security and real-time information. The UML model gives sufficient information to derive HLPSL and TURTLE codes, and therefore to cater the AVISPA and TTool, respectively.

2.2 Security-oriented verification with AVISPA

The AVISPA [1] tool checks Internet security-sensitive protocols against security flaws. AVISPA accepts a problem specification and a property specification. Both are expressed in

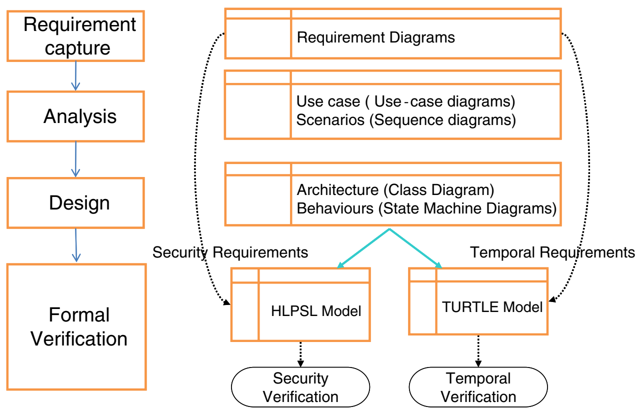


Fig. 1 A UML method including formal verification

HLPSL, which describes each participant by a basic role and composes roles to represent scenarios. A HLPSL specification is converted to an intermediate form that is accepted by all back-ends of AVISPA. The Constraint-Logic-based Attack Searcher (CL-AtSe) backend verifies security properties, such as secrecy, authentication, fairness, and non-repudiation. Security properties may be expressed as linear logical formulas or algebraic properties. Rewriting and constraint solving techniques enable attack detection.

2.3 Temporal verification with TTool

TURTLE belongs to the family of real-time UML profiles that bridge the gap between the UML and formal methods worlds. The TURTLE toolkit offers a user-friendly interface to formal verification tools and supports a verification-centric method for distributed real-time system design. Formal code generators for Real-Time LOTOS (RTL), Construction and Analysis of Distributed Processes (CADP) and UPPAAL allow one to access verification techniques such as timed reachability analysis, transition system minimization and model checking of logic formulas. A Java code generator enables rapid prototyping of systems whose model includes component and deployment diagrams in addition to the requirement capture, analysis and design ones.

3 Patterns for SGCs

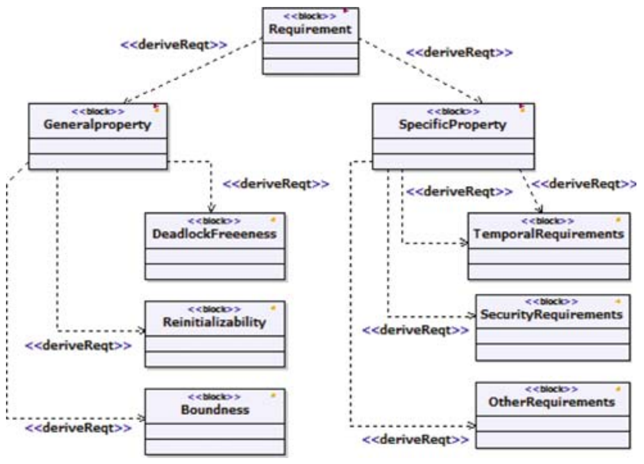
The benefits of using patterns have regularly been acknowledged in the literature. This section introduces patterns dedicated to a broad variety of SGCs, including situations where groups are hierarchically organized. The patterns nevertheless focus on two major functions: security algorithms that use keys and group management.

3.1 Requirement capture pattern

The requirement diagram pattern depicted in Fig. 2 categorizes requirements in two groups (see the two <<deriveReqt>> links in the upper part of the figure). The term "general properties" denotes a set of properties that almost of systems should satisfy. Other properties are specific to the studied system. Figure 2 focuses on security and temporal requirements. For space reasons, blocks whose name ends with an "s" (e.g. *SecurityRequiments*) are not refined.

3.2 Analysis pattern

Secured group communications commonly use security keys. The pattern in Fig. 3 identifies key creation, distribution and renewal services.



<<block>> Requirement
<<deriveReqt>>
<<block>> Generalproperty
<<block>> SpecificProperty
<<block>> DeadlockFreeeness
<<block>> Reinitializability
<<block>> Boundness
<<block>> TemporalRequirements
<<block>> SecurityRequirements
<<block>> OtherRequirements

Fig. 2 Requirement diagram pattern

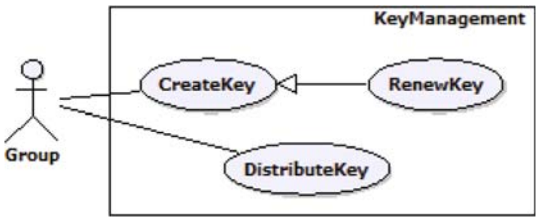


Fig. 3 Analysis pattern for key management

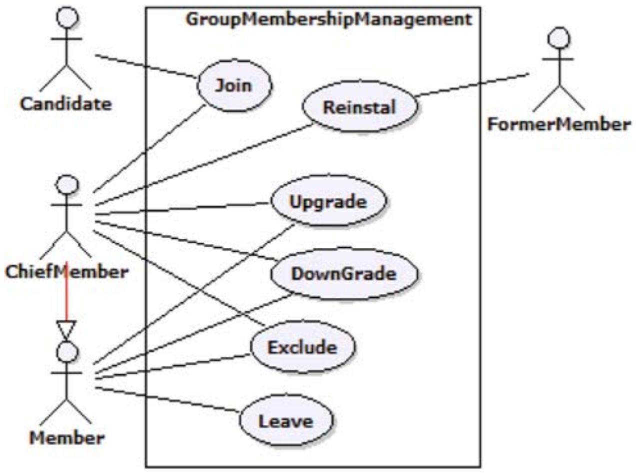
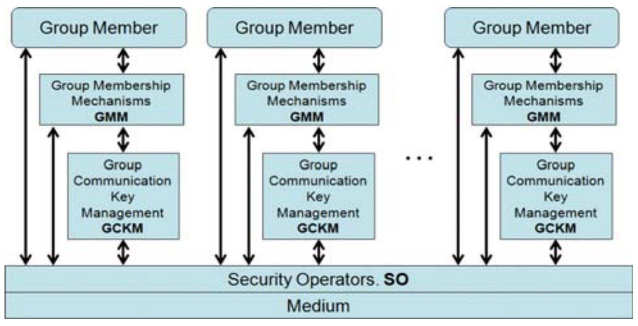


Fig. 4 Pattern for group management

SGCs also manage groups. The pattern in Fig. 4 presents services that allow one person to join a group (*Join*), to leave a group upon request (*Leave*), to leave a group after an exclusion (*Exclude*), and to reenter the group (*Reinstal*).

When the group is hierarchically organized, one member moves up and down in the hierarchy using the *Upgrade* and *Downgrade* services, respectively.



Group Member
Group Membership Mechanisms GMM
Group Communication Key Management GCKM
...
Security Operators. SO
Medium

Fig. 5 Architectural pattern

3.3 Design pattern

It is common practice in protocol design to define a three-layer architecture where the protocol entities in the central layer rely on some pre-existing communication service to render in turn a value-added service to their upper users. The pattern in Fig. 5 extends that principle to SGCs and splits the protocol layer into two sub-layers. *GMM* and *GCKM* respectively implement group management and group communication functions.

4 SAFECAST system modelling

4.1 The SAFECAST system

The secured group communication protocol designed in the framework of SAFECAST project manages hierarchically organized group of policemen, firemen and military men that work together on the same operation theatre. Each group member owns a mobile phone, and talks to others using the PMR technology (Private Mobile Radio). The protocol secures communication and manages newly created groups of Humans in a way that preserves the original hierarchies of the original groups of policemen, firemen and military men. The groups are not only hierarchical but also dynamic, since receivers may join or leave groups at any time. Last but not least, security requirements may not be dissociated from temporal ones.

4.2 Security and temporal requirements

Security requirements essentially address integrity and confidentiality issues (Table 1). On the other hand, temporal requirements mainly relate to maximum response delays.

4.3 Key management services

The use-case diagram in Fig. 6 identifies several functions that achieve communication key renewal. *RenewHierarchicalKey* applies to a group whose members are hierarchically

Table 1 Hierarchical group key management protocol requirements

	Security	Temporal
Group	Group member authentication	Group access delay
	Confidentiality before and after adhesion	
	Fighting against collusion	
	Node losses tolerance	
Interaction	Message confidentiality before/after member adhesion/departure	
	Traffic encryption keys confidentiality before/after member adhesion/departure	
Communication	Confidentiality and integrity of messages	Propagation delay
	Confidentiality and integrity of the traffic encryption keys	Throughput
	Message authentication	Jitter
	Replay avoidance	

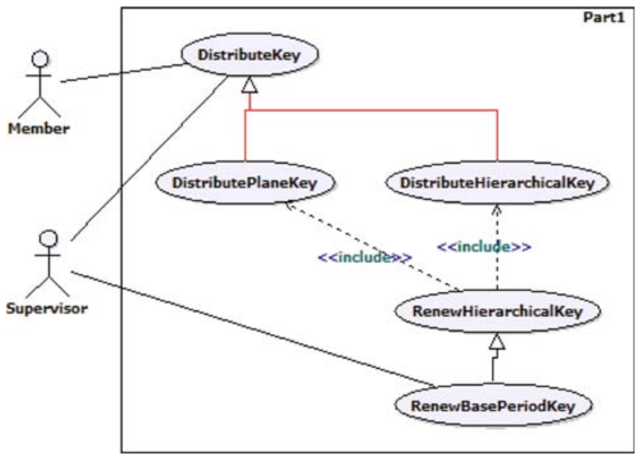


Fig. 6 Security mechanism management services

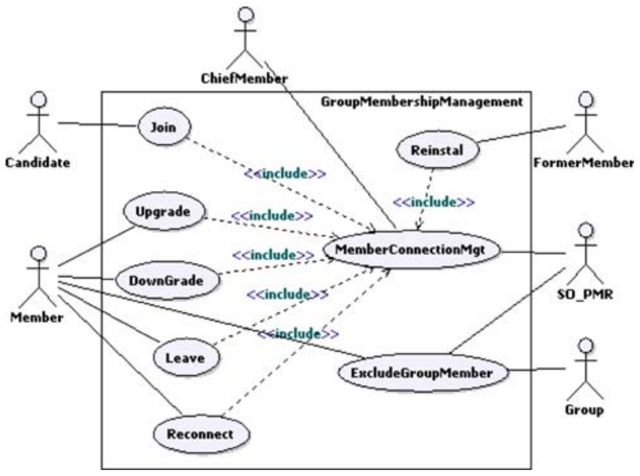


Fig. 7 Services for group dynamics

linked. The use-case includes two use-cases named *DistributeHierarchicalKey* (DHK) and *DistributePlaneKey* (DPK). DHK allows one chief to send his/her key to one group member that is responsible for his/her hierarchical level. DPK allows each responsible member to send a session key to all members belonging to its hierarchical level. DHK and DPK both use *DistributeKey*.

The keys have to be renewed on a regular basis. The diagram in Fig. 6 thus contains the *RenewBasePeriodKey* use-cases.

4.4 Group management services

The use-case diagram in Fig. 7 identifies a set of services that enable changes inside one group.

- *Join* and *Leave* allow one member to enter and exit a group, respectively.
- *Upgrade* and *Downgrade* services. A member has promotion when he/she moves from his/her current class i to an upper class j.
- *Reconnect* allows a member who formerly lost connection to connect again.
- *ExcludeGroupMember* manages member exclusion.
- *Reinstal* may be invoked to reinstall a previously excluded member.

All services but *ExcludeGroupMember* use the *MemberConnectionMgmt* subservice, which manages group member connections and disconnections.

The next section focuses on *Upgrade*, a service for which we identified security flaws and temporal violations.

5 Upgrade service

5.1 Overview

The SAFECAST system manages a set of dynamic and hierarchical groups. The group chief is the one who decides which actions may be performed by his/her group members.

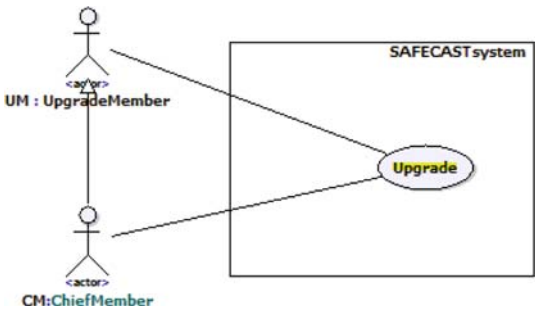


Fig. 8 Use-case diagram

Moving up in the hierarchy using the Upgrade service is an example of such actions.

5.2 Requirement capture

The *Upgrade* service enables a member endowed with responsibilities to leave his/her group (Fig. 8) and to be replaced by another member *um* who occupied a lower position in the hierarchy. The group's administrator (not necessarily the group's chief) asks *um* to move up in the hierarchy.

5.3 Use-case driven analysis

The *Upgrade* protocol works as follows. Member *um* issues a move-up request—including his/her Identity Certificate CI—to the chief *cm*. Depending on the validity of the attributes of *um*'s Identity certificate, *cm* decides to accept the *Upgrade* request or not. In case of acceptance, *um* receives the key of the leader's class (TEK) and a new group membership certificate (CAp), both encrypted under his or her public key pk_{um}. After successful completion, the *Upgrade* service brings the upgraded member up to an upper hierarchical level (not necessarily the closest upper level). If the upgrade request is refused, a message informs *um*, which therefore stays at the same hierarchical level. The sequence diagram in Fig. 9 depicts that scenario. For the sake of clarity, only the main attributes of the certificates are represented.

5.4 Design

The design model (Fig. 10) of the SAFECAST system relies on the architectural pattern that is depicted in Fig. 5.

Figure 11 depicts a fragment of the state machine used to implement the *Upgrade* service. Of particular interest are the *tempo* primitive and the security requirement starting with *secret*.

The above state machine is one among the design model elements from which formal code is derived to cater the AVISPA and TURTLE toolkits.

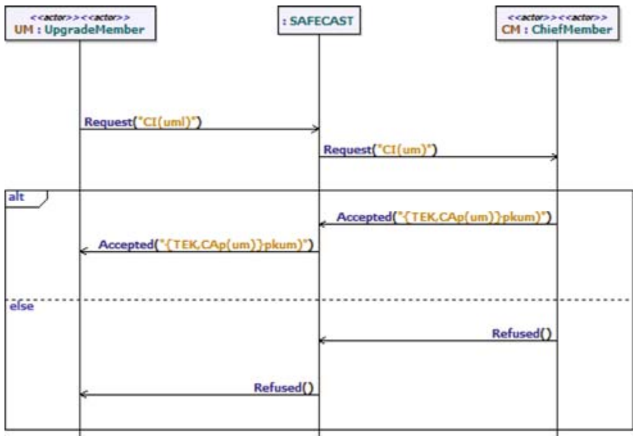


Fig. 9 Sequence diagram

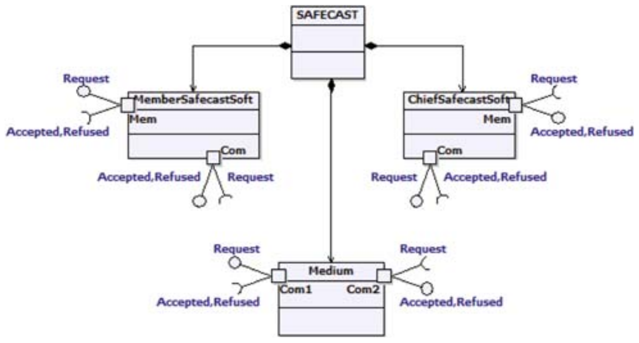


Fig. 10 SAFECAST system design model

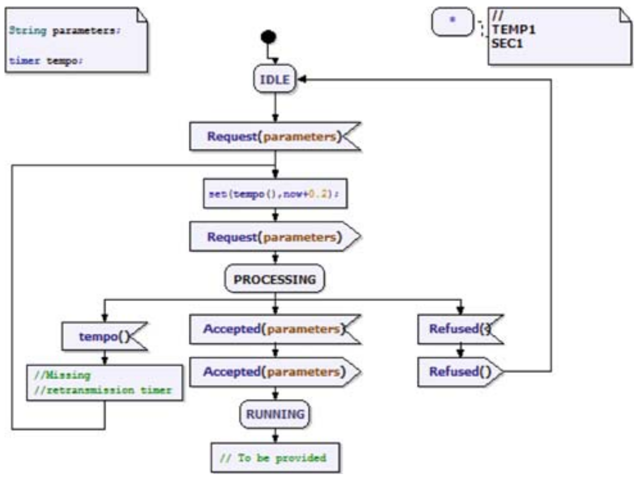
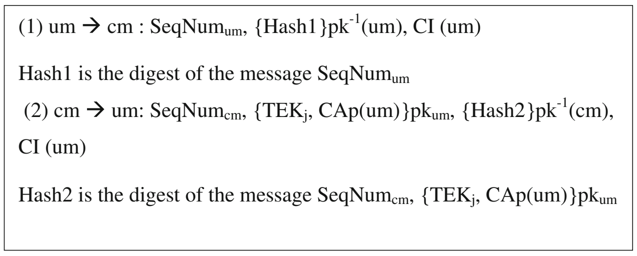


Fig. 11 UpgradeMemberCom behaviour

5.5 Security flaws detection using AVISPA

A security flaw was found in the scenario attack described in Fig. 12, step (1). It conforms to the "Man in the Middle" paradigm. It results from playing a single session between the *um* and *cm* members. The intruder *i* starts executing the protocol with *um*. *um* sends his/her identity certificate. The


(1) um $\rightarrow$ cm : SeqNum_{um}, $\{\text{Hash1}\}pk^{-1}(um)$, CI (um)

Hash1 is the digest of the message SeqNum_{um}

(2) cm $\rightarrow$ um: SeqNum_{cm}, $\{\text{TEK}_j, \text{CAp}(um)\}pk_{um}$, $\{\text{Hash2}\}pk^{-1}(cm)$, CI (um)

Hash2 is the digest of the message SeqNum_{cm}, $\{\text{TEK}_j, \text{CAp}(um)\}pk_{um}$

Fig. 12 AVISPA code derived from the model

certificate is divided into an encrypted part and a part in clear. Therefore, the intruder can intercept the message and get the public key of *um*. Then, the intruder sends the intercepted message to the chief *cm* who gets the public key *pk* and uses it to encrypt the group key TEK_j as well as the new group membership certificate *CAp*. Finally, the intruder uses the encrypting key to create a message similar to the one awaited by *um*. Also, it forces the participant *um* to take as group key any key not coming from *cm*, but composed by the intruder. None of the members is able to directly communicate with the other member, but the intruder is able to decrypt any message sent by any of them. Moreover, the intruder has the ability to become a communication relay between the two members.

The attack was fixed by adding a signature of the message (Fig. 12, step (2)), using the private key of *cm*. Thus, *um* can authenticate the source of the message and extract the valid class key TEK_j. A sequence number is introduced in each message in order to avoid replay attacks.

AVISPA enabled detection of other non trivial flaws, in particular confidentiality violation. Dealing with reinstallation of a former member, AVISPA demonstrates that an intruder was able to access to information private to the group. The Reinstallation sub-protocol was fixed in [5].

5.6 Temporal verification using TURTLE

The reader may ask himself or herself whether the fixed version of the *Upgrade* protocol verified by AVISPA meets its expected deadlines or not. Formal verification using the TURTLE toolkit enabled comparing two configurations with a low-rate PMR at 6 kb/s with a 3 km range and an average-rate PMR at 100 kb/s value with a 100 km range, respectively.

Formal verification identified four temporal requirements met for an average-rate PMR (Table 2), but unmet for a low-rate PMR. A concrete benefit of formal verification using the TURTLE toolkit was to save development time: it was decided to not develop the SAFECAST SGC over a low-rate PMR.

For the middle-rate PMR network, all the services verified using TURTLE meet the requirements of middle-rate PMR network, but the *Downgrade* and *Reinstall* services. Duration of 490 ms was computed for the two services, which exceeds the 350 ms limit required for the "accessing to multimedia groups." In order not to sacrifice the entire security procedure, it was decided to relax the "accessing to multimedia groups" requirement to 500 ms.

6 Related work

6.1 Formal automated security verification of group protocols

The benefits of applying formal verification to security protocols have first been acknowledged for two- and three-party protocols. Nowadays, group protocols raise much more complex security problems [6–9] since they involve an unbounded number of participants and consider some complicated security properties. Significant attacks on such protocols have been found using automated techniques.

Taghdir and Jackson [8] modelled the multicast group key management protocol proposed by Tanaka and Sato [10]. They exhibited several properties not satisfied by the protocol and proposed an "improved" protocol whose model did not include any active attacker. Steel and Bundy [11] identified serious attacks in the so-called "improved" protocol. They used CORAL, a tool also used to discover attacks on the Asokan-Ginzboorg and Iolus [12] protocols.

Work on group protocol verification systematically raises an infinite search space problem since even one legal execution of the protocol requires an unlimited number of steps. Meadows and Syverson [13] extended the NRL protocol analyzer in order to tackle the GDOI's protocols [7].

Several tools primarily designed for attack search have been extended to handle group protocols. Examples include algebraic primitives (e.g. XOR) and the exponentiation often

Table 2 Temporal requirements and computed time for the Upgrade service

Requirement	Limit duration (ms)	Upgrade protocol on average-rate network (execution time 331 ms)
Detecting an integrity violation	10,000	Widely validated
Detecting a replay	10,000	Widely validated
Accessing to a multimedia group	350	Shortly validated
Accessing to textual message groups	60,000	Very widely validated

encountered in extensions of key agreement based upon Diffie-Hellman. CL-AtSe, one of the four back-ends used in AVISPA [1], is an example addressed in this paper.

Tools for protocol falsification (searching for attacks) have been extended to handle group protocols and to cope with additional requirements, such as algebraic primitives and exponentiation (regularly encountered in extensions of Diffie-Hellman-based key agreement). These tools include CL-AtSe. Modular by its extensibility to new classes of protocols or requirements, and powerful by the number of protocol sessions that it can deal with, the tool has been applied to a large number of Internet security protocols.

Other tools extensions are due to the fact that most group protocols include algebraic properties (xor, exponentiation). To our knowledge, CL-AtSe is the only tool for protocol analysis that simultaneously offers complete unification algorithms for xor and exponentiation and does not limit either terms or intruder operations.

Apart from algebraic requirements, group protocols guarantee security properties that do not limit to secrecy or authentication properties. Unlike tools that exclusively verify these two properties, CL-AtSe can verify any state-based security property. Besides secrecy and authentication, it indeed verifies additional properties such as fairness and non-repudiation.

6.2 Temporal requirements and verification of SGC systems

Research papers that identified security flaws in SGC systems mostly address security functions without taking temporal constraints into account. Corin et al. [14] demonstrated that protocols with secret exchanges that had been proven robust and secure by time-independent analysis may be no longer robust as soon as time is explicitly taken into account.

With the exponential growth of wireless networks [15], ad-hoc networks [16,17] and peer to peer technology, SGC have become an extremely important and active research area. The complexity in these SGC stems from the addition of security and temporal requirement to the dynamic evolution of the groups.

Isis [18,19], RMP [20], Transis [21], Horus/Ensemble [19] and Totem [22] were the first communication systems developed with distributed group management in mind. They offer programmers a flexible group communication model and group protocols stacks. Auto-configuration was introduced in Renesse et al. [23]. Other improvements include auto-adaptation, integrated security, real-time and fault-tolerance mechanisms. Bimodal-Multicast (Gossip-based protocols) [18] and Springlass systems (Ensemble follow-up) include new reliability, authentication and delivery services to improve scalability and stability of secure group communication systems. Evaluation and failure identification were proposed for these approaches, based on formal analysis.

Group communication systems with security services were introduced at the network level by the Enclave project [24] and at the middleware level by the Cactus environment [25]. Another avenue was opened by policy-based systems. For instance, the Antigone system [26] uses policies to address membership capabilities (e.g. control access) and security requirements (e.g. data confidentiality, integrity and authentication).

Other recently published work simultaneously addresses temporal requirements and security constraints. Spread [22] is a group communication platform which offers an integrated and secure architecture for distributed client–server systems. Group communications are enhanced with security services without sacrificing the robustness and performance of the system. Spread's layered architecture is based on dedicated servers implementing security services. Almeida [27] proposes a set of group communication protocols to satisfy real-time and dependability requirements, despite of some difficulties introduced by the groups' dynamicity. Three different Quality of Service properties are guaranteed: timeliness, order and agreement. Gutierrez-Nolasco et al. [28] also explore two adaptability issues—namely security and synchrony of group communications systems (GCS)—to maintain a consistent view of dynamic groups.

6.3 UML modelling of SGC systems

Unified Modelling Language standards and extensions are of great help for proposing methodological approaches that embed temporal and security requirements during the system design processes.

In [29], Jürjens et al. apply a UML profile (UMLsec) to a mobile communication system. The authors use analysis, design and deployment diagrams. The system is verified against security flaws. In the paper, we propose a method centred on requirement capture, analysis and design. The deployment phase is not addressed. We map UML models into their corresponding formal representations for automated verification using TURTLE and AVISPA. The result is that we check the security and temporal properties of the model correspondingly. This joint use of two formal verification tools enabled eliminating design solutions that passed security tests but did not meet the deadlines.

Like Jürjens et al. [29], Morimoto et al. [30], Abie et al. [31] and Woodside et al. [32] extend UML with security-centric constructs. Morimoto et al. [30] promotes the use of patterns. In practice they detail an "authentication pattern" and its translation to Z, a formal language which enables formal verification. The patterns proposed in this paper are more general and take group management functions into account.

Woodside et al. [32] discusses performances issues and therefore opens a new avenue for security modelling in UML. To evaluate performance and scalability, the SAFECAST project used an approach based on simulation with the NS[2] tool.

7 Conclusions

Secure group communication systems capture complex design problems in terms of group management, security flaws and temporal violations detection. Some SGC, in particular the SAFECAST system discussed in the paper, further manage hierarchically organized groups. The design of such systems therefore deserves research investigations in rigorous development methodologies based on modelling techniques and formal verification tools.

The paper proposes a UML method which covers the requirement capture, analysis and design steps of the design trajectory of SGCs. The requirement, analysis and design steps use an annotated UML to take security and temporal requirements into account. Formal codes amenable to the AVISPA and TURTLE toolkits are derived from the design models in UML. They enable early checking of design models against security and temporal requirements. AVISPA and TURTLE remain separate and so each tool explores the system's state space separately. Work has still to be done for discovering problems where security and timing cannot be verified in sequence but in parallel.

The proposed method was applied to the SAFECAST system. The latter was checked against security flaws and temporal requirements. Several security flaws were detected with AVISPA. The problems have been fixed and the group communication protocol is now more secure. On the other hand, the system was investigated with two PMR radios, termed as 'low-rate' and 'medium-rate' PMR radios. The TURTLE toolkit proved that most temporal requirements are satisfied by the version based on the medium-rate PMR. Conversely, the configuration using a low-rate PMR violates important temporal requirements. It was decided to not develop it.

The approach proposed in the paper is not restricted to the SAFECAST SGC. Indeed, we plan to apply our approach to validate the applicability of others communication architectures, such as an audio–video multicast streaming application within ad hoc networks. This kind of applications requires a high level of security, in addition to the real-time requirements, to offer the best possible quality of service, within constrained environment.

Acknowledgments Support from the *Réseau National de la Recherche en Télécommunications* (RNRT) is greatly acknowledged. The authors are grateful to all the partners of SAFECAST project. AVISPA has been developed by LORIA. The TURTLE profile has been partly developed by LAAS-CNRS, ISAE and Telecom-Paris. The TURTLE toolkit (TTool) has been developed by Ludovic Apvrille.

[2] http://www.isi.edu/nsnam/ns/.

References

1. Armando A, Basin D, Boichut Y, Chevalier Y, Compagna L, Cuellar J, Hankes Drielsma P, Héam PC, Kouchnarenko O, Mantovani J, Mödersheim S, Von Oheimb D, Rusinowitch M, Santos Santiago J, Vigano L, Turuani M, Vigneron L (2005) The AVISPA tool for the automated validation of internet security protocols and applications. In: Proceedings of 17th international conference on computer aided verification (CAV'05). Springer-Verlag (LNCS 3576). Edinburgh, Scotland, pp. 281–285
2. Apvrille L, Courtiat J-P, Lohr C, de Saqui-Sannes P (2004) TURTLE: a real-time UML profile supported by a formal validation toolkit. IEEE Trans Softw Eng 30(7):473–487
3. Dolev D, Yao AC (1983) On the security of public-key protocols. IEEE Trans Inform Theory 29(2):198–208
4. Mota S (2008) Protocol modeling and verification for secured group communications. PhD thesis, University of Toulouse (in French)
5. Bouassida MS, Chrisment I, Festor O (2008) Group key management in MANETs. Int J Netw Secur IJNS 6(1):67–79
6. Steiner M, Tsudik G, Waidner M (1998) CLIQUES: a new approach to group key agreement. In: Proceedings of the 18th IEEE international conference on distributed computing system, Amsterdam, pp 380–387
7. Meadows C (2000) Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In: Proceedings of the first workshop on issues in the theory of security. Degano P, Geneva, Switzerland, pp 87–92
8. Taghdir M, Jackson D (2003) A lightweight formal analysis of a multicast key management scheme. In: Proceedings of 23rd IFIP international conference on formal techniques for networked and distributed systems, FORTE'03, pp 240–256
9. Steel G, Bundy A, Maidl M (2004) Attacking a protocol for group key agreement by refuting incorrect inductive conjectures. In: Proceedings of the international joint conference on automated reasoning. Springer-Verlag (LNAI 3097), pp 137–151
10. Tanaka S, Sato F (2001) A key distribution and rekeying framework with totally ordered multicast protocols. In: Proceedings of 15th IEEE international conference on information networking, ICOIN'01, pp 831–838
11. Steel G, Bundy A (2004) Attacking group multicast key management protocols using CORAL. In: Proceedings of the ARSPA workshop. ENTCS, vol. 125, no 1, pp 125–144
12. Mittra S (1997) Iolus: a framework for scalable secure multicasting. In: Proceedings of ACM, SIGCOMM'97, pp 277–288
13. Meadows C, Syverson P (2001) Formalizing GDOI group key management requirements in NPATRL. In: Proceedings of the 8th ACM conference on computer and communications security, pp 235–244
14. Corin R, Etalle S, Hartel PH, Mader A (2004) ADER, timed analysis of Security Protocols. In: Proceedings of ACM workshop on formal methods in security engineering, Washington DC, USA, pp 26–32
15. Chick T, Teo JCM (2006) Energy-efficient ID-based group key agreement protocols for wireless networks. In: Proceedings of 20th parallel and distributed processing symposium
16. Wu B, Wu J, Fernandez B, Ilyas M, Magliveras S (2007) Secure and efficient key management in mobile ad hoc networks. J Netw Comput Appl 30(3):937–954
17. Bohio M, Miri A (2004) Authenticated secure communications in mobile ad hoc networks. Proc IEEE Conf Electr Comput Eng 3:1689–1692
18. Birman KP, Hayden M, Ozkazap O, Xiao Z, Bidiu M, Minsky Y (1999) Bimodal multicast. ACM Trans Comput Syst 17(2): 41–88

19. Birman KP, Hayden M, Hickey J, Kreitz C, van Renesse R, Rodeh O, Vogels W (2000) The Horus and Ensemble projects: accomplishments and limitations. In: Information survivability conference and exposition DISCEX '00, vol 1, pp 149–161
20. Whetten B, Kaplan S, Montgomery T (1994) A high performance totally ordered multicast protocol. In: Proceedings of the workshop on theory and practice in distributed systems
21. Amir Y, Dolev D, Kramer S, Malki D (1992) Transis: a communication sub-system for high availability. In: Proceedings of 22nd annual international symposium on fault tolerant computing, pp 76–84
22. Amir Y, Moser LE, Melliar-Smith PM, Agarwal DA, Ciarfella P (1995) The totem single-ring ordering and membership protocol. ACM Trans Comput Syst 13(4):311–342
23. van Renesse R, Birman KP, Maffeis S (1996) HORUS: a flexible group communication system. Commun ACM 39(4):76–83
24. Gong L (1997) Enclaves: enabling secure collaboration over the internet. IEEE Trans Selected Areas Commun 15(3):567–575
25. Hiltunen MA, Schlichting RD (1996) Adaptative distributed and fault-tolerant systems. Comput Syst Sci Eng 11(5):125–133
26. Irrer J, Prakash A, McDaniel P (2003) Antigone: policy-based secure group communication system and AMirD: antigone-based secure file mirroring system. In: Proceedings of the DARPA information survivability conference and exposition DISCEX, pp 44–46
27. Almeida C (2004) Handling QoS in a dynamic real-time environment. In: Proceedings of the 8th IEEE international workshop on object-oriented real-time dependable systems, pp 217–224
28. Gutierrez-Nolasco S, Stehr M, Talcott C, Venkata N (2004) Exploring adaptability of secure group communication using formal prototyping techniques. In: Proceedings of 3rd workshop on adaptative and reflective middleware, Toronto, Canada
29. Jürjens J, Schreck J, Bartmann P (2008) Model-based security analysis for mobile communications. In: 30th international conference on software engineering (ICSE'08), Leipzig, Germany
30. Morimoto S, Cheng J (2005) Pattern protection profiles by UML for security specifications. In: International conference on computational intelligence for modelling control and automation (CIMCA-IAWTIC'05), Vienna, Austria
31. Abie H, Aredo DB, Kristoffersen T, Mazaber S, Raguin T (2005) Integrating a Security requirement Language with UML. In: 7th international conference on the unified modeling language (UML 2004), Lisbon, Portugal, LNCCS, vol 3273, pp 350–364
32. Woodside M et al (2009) Performance analysis of security aspects by weaving scenarios extracted from UML models. J Syst Softw 82(1):56–74