Temporal Verification in Secure Group Communication System Design

B. Fontan(*), S. Mota, P. de Saqui-Sannes(*), T. Villemur
LAAS-CNRS University of Toulouse
(*) ENSICA, France
{bfontan, desaqui}@ensica.fr, {smota, villemur}@laas.fr

Abstract

The paper discusses an experience in using a real-time UML/SysML profile and a formal verification toolkit to check a secure group communication system against temporal requirements. A generic framework is proposed and specialized for hierarchical groups.

1. Introduction

Secure Group Communication Systems, or SGCS for short, capture complex problems in terms of security, group management, and timeliness. Whether security protocol verification and group management have often been discussed – separately or not - in the literature, little work has been published on formal verification of SGCS against temporal requirements.

The paper proposes a formal modeling and verification framework for checking an SGCS against temporal requirements. The proposed modeling language is TURTLE (Timed UML and RT-LOTOS Environment [2]), based on the Unified Modeling Language [14] and a subset of the System Modeling Language (SysML [11]). TURTLE adds a formal semantics to UML and SysML. It improves UML with powerful temporal operators and extends SysML with a language dedicated to temporal requirement expression. Also, TURTLE is supported by a toolkit which enables verification of distributed systems against temporal requirements.

Without loss of generality, the paper discusses the use of TURTLE on a specific SGCS where group members are hierarchically organized. The running example raises usual security problems, such as "Who issues the cryptographic key and owns it?" Further, operations such as group merging and member reinsertion must be executed with hierarchical principles in mind. This makes our example original with respect to other systems published in the literature.

The paper is organized as follows. Section 2 surveys related work. Section 3 introduces the TURTLE modeling language and the formal verification tools. Section 4 identifies the main functions to be offered by an SGCS and depicts the results in terms of use-case diagrams. Section 5 proposes a design architecture to model the previously identified functions. Section 6 focuses on the SGCS investigated in the framework of SAFECAST project [8]. Section 7 concludes the paper.

2. Related work

2.1. Encryption keys

SGCS commonly achieve data protection by using encryption keys [17] that may be asymmetric or symmetric.

Asymmetric algorithms use a pair of public and private keys. Their application to SGCS is hampered by scalability problems (combinatory of keys) and by the complexity of the asymmetric encryption algorithms.

Therefore, much work on SGCS implements symmetric algorithms with one group secret key shared by the group's members (cf. the Diffie-Hellman's algorithm [5] and its adaptation to groups).

2.2. Examples of SGCS

The *Ensemble* system [9] adds one security layer on top of former ISIS and HORUS group communication systems. *Ensemble* efficiently computes group keys, offers several security policies at the application level, and supports multiple partitioning.

The *Secure Spread* [1] system implements five key generation protocols mostly based on Diffie-Hellman's group protocol algorithm. The user selects one protocol depending on the security compromise algorithm that is acceptable to him/her. The weak point is that Secure Spread works only for servers that never fail.

0-7695-2989-5/07 $25.00 © 2007 IEEE
DOI 10.1109/SECURWARE.2007.39

2.3. Formal verification of security protocols

So far, security protocols have essentially been verified using rewriting rule techniques. Examples of tools that use these techniques include *CASRUL* [4] and *AVISPA* [3]. Though powerful in detecting security flaws (in particular key management problems), these tools do not take time into account. On the opposite side, formal verification tools such as *UPPAAL* [15], *TINA* [12] and *TTool-RTL* [13] [10] use timed modeling techniques that enable formal verification of temporal properties.

The remainder of this paper addresses TURTLE, the UML/SysML language supported by TTool-RTL.

3. TURTLE

The TURTLE modeling language adds formality to the Unified Modeling Language (UML [14]) and borrows the concept of requirement diagrams from the System Modeling Language (SysML [12]). Beyond formality, the strength of the language stems from its support by TTool [13], which is interfaced with the formal verification tools RTL [10] and CADP [16].

3.1. Modeling in TURTLE

Modeling in TURTLE starts with a SysML-like requirement diagram where temporal requirements may be formally expressed and connected with verification results in order to achieve traceability.

The output of the analysis phase is a use-case diagram which defines the boundary and main functionalities of the system. The use-cases are documented by sequence diagrams structured by an Interaction Overview Diagram.

The design phase uses one class/object diagram to model the static architecture of the system and several activity diagrams to describe the inner workings of the objects. A class/objects diagram allows one to express synchronization, parallelism, sequencing and pre-emption between pairs of objects. Also, TURTLE extends activity diagram with three temporal operators: a fixed delay, a time interval, and a time-limited offer (TURTLE objects indeed communicate by means of rendezvous offers).

3.2 Use of formal verification tools

The use of TURTLE tools may be sketched as follows. The designer first draws the requirement, analysis and design diagrams using TTool [13]. The RT-LOTOS code generator implemented by TTool translates the TURTLE model into a RT-LOTOS specification that may be verified by RTL (RT-LOTOS Laboratory [10]). For bounded systems of "reasonable" size, in particular the secure group communication system discussed in this paper, RTL generates a reachability graph that may be in turn minimized using CADP [16]. The later outputs a quotient automaton which gives an abstract view of the system's behavior, focusing on the system's actions which are of interest for the set of requirements to be verified. Note that RTL and CADP are invoked from TTool's interface. Also, TTool indicates how the identifiers in the quotient automaton relate to the identifiers used in the TURTLE model.

4. Analysis

This section identifies the set of functions to be provided by a SGCS after an active session started, i.e. after each member in a group was attributed his/her rights. The procedures used to give members their rights and to set up groups are not discussed in this paper, since they capture weaker requirements in terms of interactivity and security.

The TURTLE analysis model of an active session distinguishes between security and intra-group functions, respectively. Fig.1 and Fig.2 depict the corresponding use-case diagrams. The latter are documented by scenarios expressed in terms of sequence diagrams (not shown for space reasons).

4.1. Structuring groups using roles

The use of roles to structure groups was suggested by the necessity to describe hierarchically organized groups of Humans with clearly separate roles, as well as groups where all members have the same role.

Each member participating to an active session has a *Member role* which grants him/her access to a set of predefined resources, rights and functions (Fig.2). One member plays a special role as he/she heads one or several groups. The so-called *ChiefMember* manages the group and knows its current status.

During the key distribution phase, the *Supervisor* role is hold by the person who is responsible for creating and distributing the key. Meanwhile, other members keep the *Member* role.

Groups' composition dynamically evolves and roles are introduced to handle that dynamicity. The *ConcernedMember* role is given to one member who is ready to enter a group, to exit from his/her group, or to move up (down) in the hierarchy.

The so-called *Administrator* owns the right to

exclude one member and to make one member move up or down in the hierarchy. The Administrator role may be held by any authorized person, in particular the group's chief.

Besides exclusion, a member may also leave contact with his/her group due to communication problems. Then he/she may ask permission to come back by playing the *ConcernedMember* role.

Finally, two actors named *SO_PMR* and *Group* respectively model the communication medium and a set of members belonging to the same group.

4.2. Group key management

This section addresses access control mechanisms, source authentication mechanisms, integrity and confidentiality in data exchanges inside the same session. Again, groups may evolve dynamically. Also, for the functions identified in figure 1, group keys are symmetric. Asymmetric keys are used in other parts of the system that are not addressed in this paper.

DistributeKey is the basic security function for distributing a previously generated symmetric key. The *DistributeKeyPlane* function specializes the distribution for groups with one hierarchy level. The *DistributeKeyHierarchical* function adapts the key distribution mechanisms for N-level hierarchical groups. Keys are generated in such a way lower-level (higher level) messages may (not) be understood. Lists of generated keys avoid the understanding of higher-levels messages, whereas lower-level messages remain understandable. The *RenewKeyHierarchical* function supervises the key renewal process.

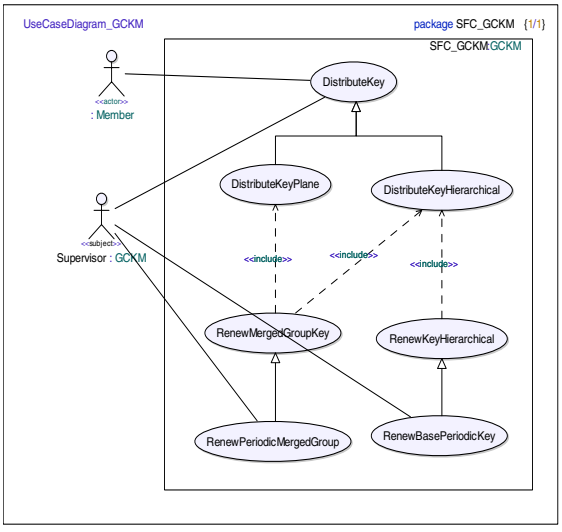

Fig.1. Use-case diagram for security functions

Security is improved by renewing keys on a periodic basis, and more precisely every N hours (this function is implemented by *RenewBasePeriodicKey*). Note that *RenewMergedGroupKey* and *Renew PeriodicMergedGroup* are left for further study.

4.3. Intra-group functions

The functions identified by the use-case diagram in Fig.2 manage one group from inside. Dynamicity is handled.

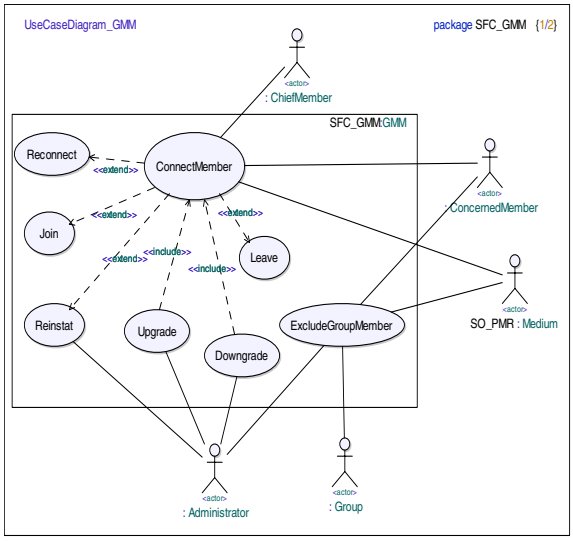

Fig.2. Use-case diagram for intra group functions

Basically, a member may join a group, leave it, and move up or down in the hierarchy. The *Join*, *Leave*, *Upgrade*, and *DownGrade* use-cases have been defined accordingly. *Reconnect* may be used in case of connection loss, and *Reinstat* further applies when the member had previously been excluded. The aforementioned functions use *ConnectionMgtMember* to make a member connect to one group. Finally, *ExcludeGroupMember* is used to exclude a member for ever.

5. Design

Previous section identified group key management and intra-group functions to be implemented by a hierarchically organized SGCS. That analysis is followed by a design step where the system's architecture and the objects' behaviors are defined.

To our knowledge, no design pattern has so far been published for security systems [1]. In this paper, we propose the 4-layer architecture depicted by Fig.3.

Layers 1 and 2 handle secure communication operations. Layers 3 and 4 respectively manage communication keys and groups. The members connected to the system are located at the application layer.

5.1. Secured broadcasting services

The *Medium* layer offers elementary multipoint broadcasting functions. It implements three basic services: a point to point communication service, a multipoint service "1 to N" and a "1 to all" broadcasting service.

The *Security Operators* layer, or SO for short, use hashing functions to guarantee integrity, encryption to achieve confidentiality, signature to guarantee authentication, and an index to prevent from replay and repudiation.

5.2. Management services

The *Group Communication Key Management* layer, or GCKM for short, manages session keys in order to make group communication secure. Key renewal includes key generation and distribution. One key is generated for one specific hierarchical level inside a communication group. Key renewal must always maintain the security properties of the system, even when the roles evolve or when one member enters (resp. exits) a group.

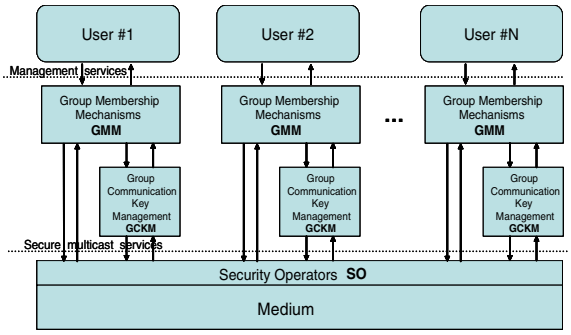

Fig.3. Generic architecture

The *Group Membership Mechanisms,* or *GMM for short,* manages groups. It controls their structure, their evolution and dynamicity. GMM includes the intra-group functions identified in section 4.3. One service is created per function. The resulting set of services works using the underlying session key management mechanisms and the roles which grant rights to members.

So far, discussion has not been targeted to a SGCS in particular. Next section discusses the use of TURTLE in the framework of SAFECAST project [8].

6. SAFECAST project

The SAFECAST project [8] addresses secure group communication systems in the context of operation theatres where first-aid services, firemen, and policemen cooperate to achieve a security mission in common. The challenge is to create, dynamically make evolve, and command coherent groups of Humans, despite of heterogeneous origins and specific hierarchical rules.

6.1. Architecture and mechanisms

The architecture as well as the security and group management protocols developed in the framework of SAFECAST project have been proposed under the assumption that people engaged on operation theatres are equipped with mobile terminals that securely transmit voice and data over a multicast radio network (Private Mobile Radiocommunication, or PMR for short). Groups evolve dynamically, which makes online update of security elements a necessity.

The most important mechanism implemented by the upper layer is session key management. The dynamic nature of the groups makes it necessary to manage keys using a cryptographic, symmetric, contributive and distributed algorithm. Diffie-Hellman's algorithm [5] has been extended to a group context. The algorithm is exclusively used by group chiefs to generate the session key. The latter is broadcasted to other members. The session key is used by the encryption and decryption operations implemented by the physical layer. The index used to prevent from replay contains each member's identity.

The project considers a radio medium with rate and range values that are common in ad-hoc PMR networks. The middle rate class has a 100 kb/s rate and a 100 km range.

5.2. Verification against temporal requirements

The SAFECAST system has been modeled in TURTLE relying on the architecture depicted by Fig.3. Groups include up to seven members. The TURTLE model includes eight security and group management functions (1. Key Generation and Distribution. 2. Join. 3. Leave. 4. Reconnection. 5. Reinstallation. 6.

Exclusion. 7. Upgrade. 8. Downgrade).

The prime objective of modeling the SAFECAST system in TURTLE was to check the system against a set of temporal requirements which constraint security requirements. As suggested by Tab.1, examples of temporal requirements include the amount of time taken to set up a function, the user's reaction time depending on the group's configuration, and the amount of time allowed to detect that messages are not exchanged in a normal way.

Each requirement is expressed in a SysML requirement diagram. A SysML requirement is a block [7] characterized by four attributes: (1) an identifier; (2) a text (an informal description of the requirement); (3) a type: "functional", "non-functional", or "performance"; (4) a risk level: "low" or "high".

Requirements are first expressed in an informal way. Formal requirements are then derived from the formal ones and expressed in a chronogram style, using a so-called "TRDD" (Timing Requirement Description Diagram). Fig.4 depicts an example for the following requirement: "Access Time for a Multimedia Group must remain below 350 ms". The TRDD contains two observations points "Begin_MA" and "End_MA" that define the borders of the valid temporal interval (350 ms). For a delay value ranging between 0 and 350 ms, the requirement is met (denoted by the OK interval). The delay values that exceed 350 ms correspond to a requirement violation (denoted by KO interval).

The TRDDs serve as starting point for automatic requirement verification using the TURTLE toolkit [7].

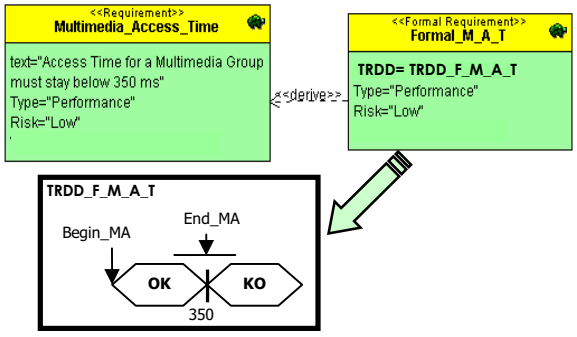

Fig.4. Requirement Diagrams for Access Time for a Multimedia Group

The multipoint broadcast radio PMR included in the SAFECAST system leads to work with time constraints ranging from milliseconds to hours. In Tab.1, duration is expressed in ms.

First column in Tab.1 lists the requirements to be verified by the SAFECAST system.

Second column in Tab.1 associates an upper temporal bound with each requirement. The eight subsequent columns list the functions investigated for the middle-rate radio network (100 kb/s).

Letters T and F respectively indicate whether a temporal requirement is satisfied or not. An empty box indicates that the requirement does not apply. For instance, on second line, the average delay for entering into an encrypted communication should remain lower than 1000 ms. The requirement applies to the Join, Reconnection and Reinstallation functions. We formally verified that the temporal requirement is met by the three functions.

For each function, the TURTLE environment computed a duration that is indicated by the *Duration obtained* line. For instance the *Key Generation and Distribution* function takes 121 ms when it is implemented over the middle-rate PMR network.

Requirements \ Functionnality	Limit Duration (ms)	Medium Rate (100 kb/s)							
		Gen&Dist	Join	Leave	Reconn	Upgrade	Downgrade	Reinstat	Exclude
Delay for **Setting** a an Encrypted Communication (< 350 ms)	350	T							
Average Delay for **Entering** in an Encrypted Communication (< 1s)	1 000		T		T			T	
Highest Delay for a **Late Arrival** in an Encrypted Communication (< 2s)	2 000		T		T			T	
Delay for **Detecting** an Integrity Violation (< 10 s)	10 000	T	T	T	T	T	T	T	T
Delay for **Detecting** a Replay (< 10 s)	10 000	T	T	T	T	T	T	T	T
Access Time for a Multimedia Group (< 350 ms)	350		T		T	T	F	F	
Access Time for a Text Group (< 1 m)	60 000		T		T	T	T	T	
Highest Delay for **Excluding** a Member by an Administrator (< 1 h)	3 600 000								T
Best Delay for **Excluding** a Member by an Authorized Member (< 1 m)	60 000								T
Duration obtained (ms)		121	296	222	323	323	482	482	145

F	Property NOT Satisfied
T	Property Satisfied

Tab.1. Temporal Requirement verification results

As shown by Tab.1, all the services but one meet their expected limit duration when the system is deployed over a middle-rate PMR network. The exception is *multimedia access*, for which *downgrade* and *reinstat* raise temporal violations. The reason is that the access time for a multimedia group must remain below 350 ms, whereas an upper bound of 60 000 ms is accepted for text communications.

Tab.1 indicates that both *reinstat* and *downgrade* have a total access time equal to 482 ms, which is not too far from 350 ms. The SAFECAST project partners agreed on relaxing temporal constraints without modifying the security protocols implemented to achieve security, authentication, confidentiality and no-repudiation.

7. Conclusions

Secure group communication systems, or SGCS for short, capture complex design problems in terms of security flaws, group management, and timeliness. The SGCS designed in the framework of SAFECAST project further introduces hierarchically organized

groups that must cooperate on the same operation theatre.

The level of complexity reached by the SAFECAST SGCS has convinced the project's partners to use formal modeling techniques and verification tools. The partners also agreed on the necessity to use two complementary verification tools. The AVISPA tool was selected to specifically address security issues. A security flaw was identified [3] and fixed. On the other hand, relying on former experience [6] in applying the TURTLE language and tools to security protocols, it was decided to use TURTLE to check the SAFECAST system against temporal requirements. It has been established that the SAFECAST SGCS does not work correctly over a low-rate PMR network. By contrast, the middle-rate PMR network is appropriate as soon as multimedia services are not implemented. This result has been of high importance in making design decisions for the final product.

The modeling framework proposed in the paper is not limited to the SAFECAST system. The functions identified in Fig.1 and Fig.2 are generic, and so is the architecture depicted by Fig.3. These diagrams will be the starting point for further study on group key management mechanisms. Novel group merging and splitting operations are also to be investigated.

Finally, the challenge in terms of verification tool is to cope with groups of hundreds - if not thousands - members. Recent work on translating TURTLE models into Time Petri Nets (instead of RT-LOTOS) will make it possible to interface TTool [13] and TINA [12] and thus to benefit from TINA's performances.

8. Acknowledgements

This work has been carried out in the framework of SAFECAST project. SAFECAST has been funded by the RNRT (Réseau National de Recherche en Télécommunications). Acknowledgements are due to all the project partners for fruitful discussions on the SAFECAST system and the use of verification tools.

The TURTLE models were verified using TTool and RTL. TTool has been developed by Ludovic Apvrille. RTL was developed by various people at LAAS-CNRS, in particular Jean-Pierre Courtiat and Christophe Lohr.

9. References

[1] Amir Y., Nita-Rotaru C, Stanton J, Tsudik G. Secure Spread: An integrated Architecture for Secure Group Communication, *IEEE Transactions on Dependable and Secure Computing*, Sept 2005.

[2] Apvrille L., Courtiat J.P., Lohr C., de Saqui-Sannes P., TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit, *IEEE Transactions on Software Engineering*, Vol. 30, No , pp 43-48, July 2004.

[3] Bouassisa M.S, Chridi N, Chrisment I, Festor O., Vigneron L., Automatic Verification of a Key Management Architecture for Hierarchical Group Protocols. *Proceedings of SAR'06*, 2006.

[4] http://www.loria.fr/equipes/cassis/softwares/casrul/

[5] Diffie W., Hellman M. E., New Directions in Cryptography. *IEEE Transactions on Information Theory*, Vol. 22, n. 6, p. 644-654 (196).

[6] Fontan B., Mota Gonzalez S., Villemur T., de Saqui-Sannes P., Courtiat J.P., UML-Based Modeling and Formal Verification of Authentication Protocols, *IEEE International. Symposium on Secure Software Engineering*, Washington DC, USA, March 2006.

[7] Fontan B., Apvrille L., de Saqui-Sannes P., Courtiat J.-P., Real-Time and Embedded System Verification Based on Formal Requirement, *IEEE Symposium on Indus. Embedded Systems (IES'06)*, Antibes (France), October 2006.

[8] RNRT project SAFECAST. http://rnrt-safecast.org/

[9] Rodeh O., Birman K.P., Dolev D., "The Architecture and Performance of Security Protocols in the Ensemble Group Communication System: Using Diamonds to Guard the Castle". *ACM Transaction on Information and System Security*, Vol. 4, No. 3, August 2001, pp. 289-319.

[10] RTL, http://www2.laas.fr/RT-LOTOS/index.html.en.

[11] SysML (System Modeling Language), Object Management Group, http://www.sysml.org/

[12] TINA, http://www.laas.fr/tina/

[13] Ttool, http://labsoc.comelec.enst.fr/turtle/ttool.html.

[14] UML (Unified Modeling Language), Object Management Group, http://www.uml.org/

[15] UPPAAL, http://www.uppaal.com/

[16] VASI, CADP, http://www.inrialpes.fr/vasy/cadp/

[17] Zou X., Ramamurthy B., Magliveras S.S., Secure Group Communications over Data Networks, Springer, 2005.