Synthesis of Trustworthy Protocol Specifications from Service Specifications

KASSEM SALEH
Department of Computer Science
American University of Sharjah
Box 26666 Sharjah
UNITED ARAB EMIRATES

Abstract: - A trustworthy protocol specification is a specification of a system of communicating entities that meets the security and privacy requirements of the services to be provided. In this paper, we extend an existing synthesis technique introduced earlier by the author [1] to include the synthesis of trustworthiness requirements starting from the specification of these requirements at the service level. Both the required service and the synthesized protocol specifications are specified using the communicating finite state machine model. The application of the proposed synthesis technique to a given service is also presented.

Keywords:- privacy, protocol, security, service, synthesis, trustworthiness.

1 Introduction

A trustworthy distributed system relies on the trustworthiness of its various components, namely, software, hardware and the information needed to provide the requested services. Among the software components that are needed to run a distributed system, communications protocols are considered to be the most critical for the proper exchange of information between the distributed users. A trustworthy distributed system is a system that provides secure, reliable and privacy-sensitive services to its distributed service users. This definition of trustworthy systems is an adaptation of the definition of trustworthy computing introduced in a Microsoft white paper [2].

A protocol specification consists basically of the specification of each of the communicating entities which cooperate to provide a set of services to the service users. Two categories of approaches for the development of protocol specifications are used, namely, analysis-based and synthesis-based. In the analysis approaches, the protocol designer starts with a preliminary version of the protocol and then applies protocol validation techniques, like reachability analysis, to ensure the satisfiability of both liveness and safety properties, and the provision of the intended services. The protocol specification version will be iteratively refined until the desirable properties are met. On the other hand, in the synthesis approaches, complete or partially complete protocol specifications are constructed systematically starting from the service specification. The construction technique must guarantee that the synthesized protocol specification is live and safe and meets the service specifications. For surveys of synthesis techniques, the reader can refer to [3, 4]. In addition to the synthesis of basic protocol entity specifications, additional features can be embedded in the synthesis process, including error-recoverability [5] and testability [6]. In this paper, we embed trustworthiness requirements in the synthesis process.

The paper is organized as follows. After this introduction, Section 2 provides some preliminary background related to our problem. Section 3 describes the proposed protocol synthesis method. Section 4 provides an illustrative example. Finally, Section 5 concludes the paper.

2 Preliminary Background

In this section, we provide some preliminary background related to our work. First, we start by introducing the synthesis problem. Then, we

introduce the basic trustworthiness infrastructures needed to support the delivery of trustworthy distributed services. Then, we describe the basic security and privacy requirements that will be considered, later in Section 3, during the synthesis process. Finally, we provide a formal description of the model used to specify both service and protocol specifications, and operations on them, while embedding the security and privacy requirements.

2.1 The synthesis problem

Before defining the problem we are solving, it is important to understand the concepts of services and protocols. A trustworthy communications service is a high level abstraction, as shown in Figure 1, in which distributed service users initiate and/or receive services, using service primitives (SPs) in an orderly, timely and secure manner at designated service access points (SAPs).

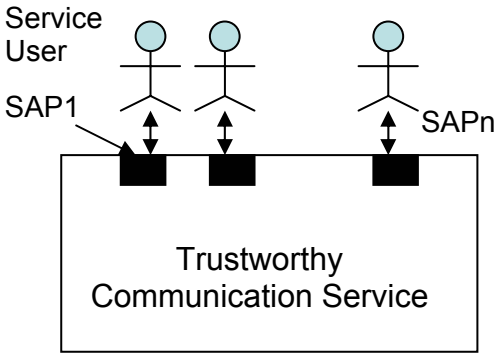

Fig. 1. An abstract view of a distributed service.

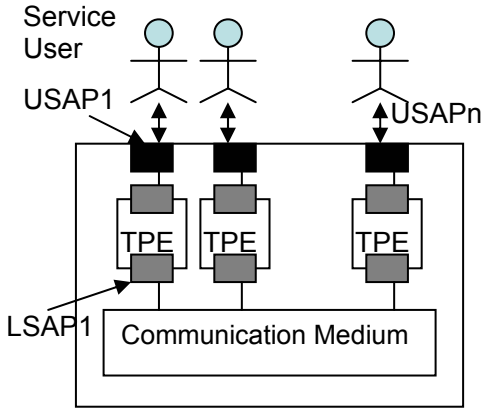

Fig. 2. Trustworthy protocol entities.

The trustworthy service is refined at a lower level of abstraction. This refinement includes the trustworthy protocol entities (TPEs) which interact with the service users at the upper SAPs (USAPs) and with the communication medium at the lower SAPs (LSAPs).

We define the synthesis problem as the "systematic generation of the trustworthy protocol entities (TPEs) starting from the trustworthy service specification (TSS)".

2.2 Trustworthiness infrastructure

Our proposed synthesis technique assumes that an infrastructure supporting the service security and privacy requirements is available. We assume that a public key infrastructure (PKI) providing a public and private key pair, (e, d), for each service user, an authentic digital certificate including the service user public key, and a cryptographic application programming interface (API) available at each SAP to be used by each of the TPEs [7, 8] to generate symmetric private session keys. In addition, the PKI would provide services for validating, revoking, suspending and activating the provided digital certificates.

2.3 Communications security requirements

The trustworthiness requirements that will be expressed at the service specification level include mainly security and privacy requirements. Firesmith [9] provides a complete list of security requirements categories. The requirements we are addressing in this work include the following.

2.3.1 Confidentiality requirement

A service requiring confidentiality will impose on the involved communicating protocol entities the necessity to encrypt all exchanged messages. An entity that sends a protocol message (m) should encrypt the message with the public key of the receiving protocol entity, that is, $e_R(m)$. The receiving protocol entity will decrypt the encrypted message it receives using its private key, that is $m = d_R(e_R(m))$. In this case, only the proper receiving protocol entity will be able to read (understand) the communicated message, hence achieving confidentiality.

2.3.2 Integrity requirement

A service requiring the integrity of any exchanged message will impose on the involved protocol entities to attach, using the cryptographic API, a hashed value, i.e., m+H(m) is sent, so that the receiving entity can ensure that no intruder has

maliciously modified the transmitted message, hence achieving integrity.

2.3.3 Authenticity

A service requiring authenticity imposes on the sending entity the use of its private key for the encryption of the transmitted message. The receiver, by using the sender's public key ($d_S(m)$) to recover the message will ensure that the sender is the true entity that sent the message, hence achieving authenticity.

2.3.4 Non-repudiation

A service requiring non-repudiation imposes on the sending entity the transmission of the message m along with the digital signature of the sender. The receiver would guarantee both the authenticity of the sender and enforces the non-repudiation requirement. The receiver can still verify the validity of the signature with the PKI.

2.3.5 Privacy

Other than guaranteeing the confidentiality of the exchanged protocol data, the sender may wish to hide its identity as part of privacy concerns. In this case, an approach similar to that used in email security, in which the sending protocol entity would camouflage its identity by providing an identity and private session key SK, both encrypted with the public key of the receiver and a message encrypted with the private session key. The receiving entity would use its private key to obtain the real identity of the sender and the session key to be able to decrypt the content of the message.

2.3.6 Hybrid requirements

The service may require a combination of two or more of the above trustworthiness requirements. For example, confidentiality and authenticity can be dealt with by the sending entity by sending $M=e_R(d_S(m))$, i.e., first encrypt the message with the private key of the sender, then encrypt the intermediate message with the public key of the receiver. Other interesting combinations include privacy and integrity, confidentiality and integrity, or privacy and confidentiality.

2.4 Specification model

In our approach, we use the finite state machine (FSM) specification model for describing the sequencing of control needed in a protocol entity specification, in addition, to the formal description of the abstract behavior of the desired service. In addition to the model, we describe some useful operations that can be performed on the model.

2.4.1 Service model and operations

We describe a trustworthy service model using an FSM specifying the legal sequences of SP occurrences that should be observed at the distributed SAPs. Due to the sequential nature of FSMs, the occurrence of concurrent SPs at the SAPs cannot be captured in the specification.

Definition 1. A trustworthy service specification TSS is denoted by a tuple (Ss, Σs, Ts, s0) where: Ss is a non-empty finite set of service states, Σs is a finite set of SPs, Ts is a partial transition function between service states (a subset of the cartesian product Ss × Σs × Ss), and s0 is the initial service state.

Definition 2. For every service state Ss ə s, OUT(s) denotes the set of SAPs associated with the SPs of its outgoing transitions.

Definition 3. A service primitive SPi is of type '↑', SPi↑, if the SP is directed upward from PESi to SAPi, i.e., it is a service reply. Similarly, SPj↓ means that the SP is directed downward from the service user at SAPj to PESj, i.e., it is a service request. A service primitive SP will be used by service users to initiate a service request. Protocol entities will also use service primitives as a vehicle to deliver requested services. An SP will be parameterized with the trustworthiness requirement or combination of requirements.

Definition 4. The projection onto a set X of SAPs (Π_X) is a unary function which can be applied to an FSM (S, Σ, T, s0) yielding another FSM (S, Σ', T', s0) in which Σ's is a subset of Σs ∪ {ε}, and T' = T with a relabeling to ε of events in Σ not contributing to the SAPs onto which the FSM is projected. To note here that ε is not an internal event.

Definition 5. A projected trustworthy service specification (PTSSi) is the projection of the FSM TSS onto SAPi (PTSSi = Π_{SAPi} TSS). PTSSi is represented by (Ss, Σ's, T's, s0), where Σ's is a subset of Σs and T's is a subset of the cartesian product Ss × (Σ's ∪ {ε}) × Ss.

2.4.2 Protocol model and operations

The protocol model is described by a tuple of specifications of a number of cooperating protocol entities. The interactions among these entities using the underlying communication medium (service) must yield the service specification (TSS). Protocol entities are also modeled using FSMs.

Definition 6. A trustworthy protocol entity specification (TPES) is denoted by a tuple (Sp, Σp, Tp, s0), where Sp is a finite set of protocol entity states, Σp is a set of protocol messages, Σp = Σ's ∪ IPE, where Σ's is a subset of Σs, and IPE is the set of internal protocol events, Tp is a partial transition function between protocol entity states, and finally s0 is the initial protocol entity state.

Definition 7. A trustworthy protocol specification (TPS) consists of the specification of several TPESs. We assume there is a one-to-one correspondence between a SAPi and a TPESi.

3 The Synthesis Technique

Starting from the trustworthy service specification (TSS), our proposed synthesis technique uses a sequence of transformations leading to the trustworthy protocol specification (TPS). The first transformation consists of the projections on each SAP as stated in Definition 4. In this section, we first introduce the trustworthy function for transforming a message according to the security and privacy requirements. Then, we introduce the second and third transformations used in our technique: the transition synthesis rules, and the optimization rules.

3.1 Trustworthiness function

This function, denoted by *TS*, is applied to a protocol message to transform it according to the requirements described as parameters to the related SP. The possible parameters are: C for confidentiality, I for integrity, A for authenticity, P for privacy, N for non-repudiation, in addition to combinations of them. This function uses the various keys and certificates provided by the PKI, in addition to the cryptographic API, to ensure the trustworthiness of the communicated message according to the specification provided in Section 2.3. For example, if we have a transition labeled with SP(C), then according to one of the rules below, we have to receive(SP) and then send (e_{Ri}(m), e_{Rj}(m), …) to other protocol entities. Upon reception, each of these receiving entities must apply the appropriate *TS*, in this case deciphering using the entity's private key, i.e., $TS(e_{Ri}(m))= d_{Ri}(e_{Ri}(m))$ in order to obtain m.

3.2 Transition synthesis rules (TSR)

These rules apply to each of the SP-labeled and ε-labeled transitions existing in each of the projected service specification (PTSSi). The rules and the intuition behind them are shown below.

3.2.1 SP-labeled transitions from si to sj

Rule 1: if sj is not a final state and OUT(sj) = SAPi, then there is no need to send any control message out of this machine.

Rule 2: if sj is an initial state and the labeling SP is of type '↓', then transform this transition into SP/send(*TS*(sp)) to all other SAPs to allow them to synchronize again at their respective initial states.

Rule 3: if sj is an initial state and the labeling SP is of type '↑', then there is no need to transmit any protocol message, and the SP is delivered to the service user using *TS*(SP).

Rule 4: if sj is not an initial state and the labeling SP is of type '↓', then transform this transition into SP/send(*TS*(sp)) to all other SAPs that are included in OUT(sj) to allow the appropriate protocol entities to take control.

Rule 5: if sj is not an initial state, OUT(sj)≠SAPi and the labeling SP is of type '↑', then there is no need to transmit any protocol message, and the SP is delivered to the service user using *TS*(SP).

3.2.2 ε-labeled transitions from si to sj

Rule 6: Corresponding to Rules 1, 3 and 5, this transition must remain unchanged since no protocol message was transmitted.

Rule 7: Corresponding to Rules 2 and 4, message reception transitions must be synthesized and the appropriate trustworthiness function is applied to be able to decipher the received message.

Note that the synthesis rules described in [1] apply when all SPs are not parameterized, i.e., no security and privacy requirements, hence all messages are exchanged in the clear.

3.3 Optimization rules

After applying the projection and synthesis rules, we first apply two algorithms for the removal of ε-cycles and ε-transitions. These algorithms are available in [10]. In addition, every uninterruptible sequence of message reception and message transmission can be made atomic, hence reducing the number of states and transitions.

3.4 The synthesis algorithm

To obtain a trustworthy protocol specification (TPS), the following steps must be performed:

a. Specify the trustworthiness requirement at the service level by parameterizing the SPs in the service specification (TSS).
b. Project TSS on each of the SAPs to obtain the PTSSs.
c. Apply the transition synthesis rules (TSRs) and the trustworthiness function (TS) appropriately.
d. Optimize the synthesized PTSSs to obtain the various TPESs that correspond to the TPS.

The proofs of correctness of the synthesis technique and algorithm are similar to those provided in [1].

4 Application

In this section, we apply our technique for the synthesis of a trustworthy protocol specification starting from a simple trustworthy service specification given in Figure 3 below and involving two SAPs.

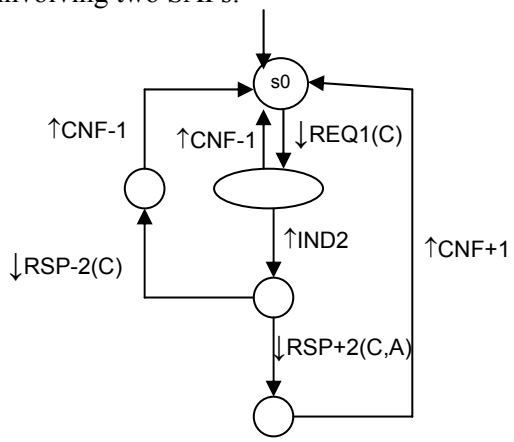

Fig. 3. Trustworthy service specification.

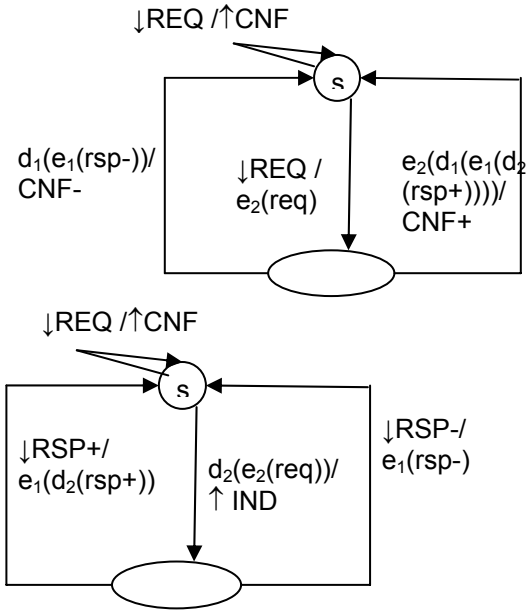

Fig. 4. The two synthesized TPESs.

The service shows that the request initiated at SAP1 and the negative response given at SAP2 should be treated with confidentiality. However, the positive response provided at SAP2 to the user at SAP1 should be handled with confidentiality while guaranteeing the authenticity of the sender.

The synthesized trustworthy protocol specification involving two protocol entities, each supporting one SAP, obtained by applying the synthesis algorithm is shown in Figure 4 below. The confidential and authenticated positive response message ($e_1(d_2(rsp+))$ sent by TPES2 is received by TPES1, and first the receiver's private key is applied (confidentiality) followed by the application of the sender's public key (authenticity of sender). This is shown by the receiving transition label $e_2(d_1(e_1(d_2(rsp+))))$.

5 Conclusions and Future Work

The main contribution of this research is the introduction of an automatic synthesis technique for the generation of communication protocol specifications that provide trustworthy distributed services. Security and privacy requirements are embedded in finite state machine-based service specifications. A synthesis technique starts from the given trustworthy service specification and generates the protocol entities specifications that meet the security and privacy requirements

included in the service specification. In the future, we intend to apply our technique to real-life services with specific security and privacy requirements. In addition, we plan to investigate possible extensions to the finite state machine model that will allow more expressive ways to capture and specify security and privacy concerns, and trustworthiness requirements in general.

References:

[1] K. Saleh and R. Probert, Automatic synthesis of protocol specifications from service specifications, Proc. of the 10th IEEE Intern. Phoenix Conf. on Computers and Communications, Scottsdale, Arizona, March 1991, pp. 615-621.

[2] C. Mundie, P. deVries, P. Haynes, and M. Corwine, 2002. Trustworthy computing, Microsoft White Paper, 10 pages.

[3] R. Probert and K. Saleh, Synthesis of communication protocols: survey and assessment, IEEE Transactions on Computers, Special issue on Protocol Engineering, Vol. 40, No. 4, April 1991, pp. 468-476.

[4] K. Saleh, Synthesis of communications protocols: an annotated bibliography, ACM SIGCOMM Computer Communications Review, Vol. 26, No. 5, October 1996, pp. 40-59.

[5] K. Saleh and R. Probert, Synthesis of error-recoverable protocol specifications, Proc. of the 2nd Intern. Conf. on Computing and Information (ICCI'90), Niagara Falls, May 1990, pp. 428-433.

[6] K. Saleh, Testability-directed service definitions and their synthesis, Proc. of the 11th IEEE Intern. Phoenix Conf. on Computers and Communications (IPCCC-92), March 1992, pp. 674-678.

[7] C. Pfleeger and S.L. Pfleeger, Security in Computing, 3rd edition, Prentice Hall, 2004.

[8] W. Whitman, and H. Mattord, Principles of Information Security, Thomson Course Technology, 2004.

[9] D. Firesmith, Engineering security requirements, Journal of Object Technology 2(1), 53-68, 2003.

[10] W.A. Barrett and J.D. Couch, Compiler Construction: Theory and Practice, Chapter 3, Research Associates, 1979.