Int. J. Communications, Network and System Sciences, 2012, 5, 165-177
doi:10.4236/ijcns.2012.53021 Published Online March 2012 (http://www.SciRP.org/journal/ijcns)

Formal Verification of Secrecy in Group Key Protocols Using Event-B

Amjad Gawanmeh[1], Sofiène Tahar[2], Leila Jemni Ben Ayed[3]
[1]Department of Electrical and Computer Engineering,
Khalifa University of Science, Technology and Research, Sharjah, UAE
[2]Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada
[3]Ecole Nationale des Sciences del'Informatique, University of La Manouba, Tunis, Tunisia
Email: amjad.gawanmeh@kustar.ac.ae, tahar@ece.concordia.ca, leila.jemni@fsegt.rnu.tn

Received December 20, 2011; revised January 19, 2012; accepted February 28, 2012

ABSTRACT

Group key security protocols play an important role in today's communication systems. Their verification, however, remains a great challenge because of the dynamic characteristics of group key construction and distribution protocols. Security properties that are well defined in normal two-party protocols have different meanings and different interpretations in group key distribution protocols, specifically, secrecy properties, such as group secrecy, forward secrecy, backward secrecy, and key independence. In this paper, we present a method to verify forward secrecy properties for group-oriented protocols. The method is based on a correct semantical link between group key protocols and event-B models and also uses the refinement process in the B method to model and verify group and forward secrecy. We use an event-B first-order theorem proving system to provide invariant checking for these secrecy properties. We illustrate our approach on the Tree based Group Diffie-Hellman protocol as case study.

Keywords: Group Key Protocols; Formal Verification; Forward Secrecy; Secrecy; Event-B

1. Introduction

Security protocols are used to establish secure channels between communicating systems. These protocols need great care in their development and their implementation. The complexity of security-protocol interactions can hide security weaknesses that normal analysis methods cannot reveal. Security properties that are well defined in normal two-party protocols have different meanings and different interpretations in group key distribution protocols. Therefore, they require a more precise definition before we look at how to verify them. For *group key distribution protocols*, secrecy property has a further dimension since there are long term secret keys, short-term secret keys, in addition to present, future, and past keys; where a principal who just joined the group and learned the present key should not be able to have enough information to deduce any previous keys, or similarly a principal who just left the group should not have enough information to deduce any future keys.

In group key protocols, there are generally four types of security properties [1]: *group key secrecy*, which guarantees that it is computationally infeasible for a passive adversary to discover any group key, intuitively, that the attacker should not be able to obtain a key that honest users think to be safe; *forward secrecy*, which guarantees that a passive adversary who knows a contiguous subset of old group keys cannot discover any subsequent group key; *backward secrecy*, which guarantees that a passive adversary who knows a contiguous subset group keys cannot discover preceding group keys, and finally, *key independence*, which guarantees that a passive adversary who knows a proper subset of group keys cannot discover any other group key.

Event-B [2] was introduced by extending B [3] without changing it to model operations that could be guarded in the process algebraic sense. The event-B method uses the set-theoretical and logical notations of the B method and provides new notations for expressing abstract models based on events. It provides invariants proofs based on a state-based system that is updated by guarded events. The refinement capability offered by event-B allows incremental development moving from an abstract level to a more concrete one. Refinement technique allows the preservation of proved properties and therefore it is not necessary to prove them again in the refined transition system. Moreover, in the refinement, it is not needed to re-prove these properties again while the model complexity increases. This advantage is important compared to classical model checking where the transition system describing the model is refined and enriched.

Group key protocols have special features, such as the concept of group secrecy, forward secrecy, and dynamic group events. These feature were rarely considered or addressed thoroughly when formal verification techniques were applied. In this paper, we provide an event-B based invariant checking for verification of group key protocols. We model group key protocols and verify their required properties, in particular secrecy and forward secrecy properties, using the event-B method. Event-B deals with tools allowing invariant checking, and can be used to verify group key secrecy properties. In order to model a group key protocol in event-B first order logic, a formal relation between the semantics of the event-B language [4] and the protocol model should be defined. This mapping relation should present the semantics of group key protocol model based on event-B, hence, allowing the verification of secrecy properties. This allows us to avoid user interaction with the theorem proving tool, and reduce the time required to verify these properties. This paper extends the work in [5] to verify forward secrecy using event-B refinement.

We apply our approach on the tree based Group Diffie-Hellman (TGDH) protocol [6] and provide invariant checking for secrecy under the static and the dynamic case by applying a single event (join/leave). We use the event-B first order prover platform Rodin [7,8] to perform invariant checking under the assumption that basic Diffie-Hellman key is correct. The dynamic case is also considered by applying events such as join and leave and verify the correctness of key construction for bounded tree size and bounded number of events. We assume perfect cryptography conditions in our approach. In addition the group key protocol is analyzed in the presence of passive adversaries.

The rest of the paper is organized as follows. Section II discusses related work to ours. In Section III, we present our methodology to verify forward secrecy using event-B refinement. In Section 4, we apply our approach on TGDH protocol. Finally, Section V concludes the paper with future work hints.

2. Related Work

The recent years have seen the emergence of successful applications of formal approaches to reasoning about security protocols. Earlier methods were concerned with reasoning about the events that a security protocol can perform, and make use of a causal dependency that exists between protocol events. Methods like strand spaces [9] and the inductive method of Paulson [10] have been designed to support an intensional, event-based, style of reasoning. These methods have successfully tackled a number of protocols though in an ad hoc fashion. They make an informal spring from a protocol to its representation and do not address how to build up protocol representations in a compositional fashion [11].

Events-based verification of security protocols was used by Crazzolara [11] using mappings between process algebra, Petri nets, strand spaces and inductive models. The authors established precise relationships between the Petri nets semantics and transition semantics, strand spaces, inductive rules, trace languages, and event structures. They show how event-based models can be structured in a compositional way and so used to give a formal semantics to security protocols which support proofs of the correctness of these protocols. They demonstrated the usefulness of their Petri nets semantics in deriving proof principles for security protocols and apply them to prove an authentication property.

Cremers [12] proposed an operational semantic for security protocols. The work provides a generic description of the interpretation of such security protocols and what it means for a protocol to ensure some security property. This work imposes explicit static requirements for valid protocols, and verifies that the model is parametric with respect to the matching function and intruder network capabilities. Other related work that treats group key protocols verification, specifically DH based protocols, are discussed in more details in [13].

Stouls and Potet [14] proposed a method to automatically enforce an abstract security policy on a network. They used the B refinement process to build a formal link between concrete and abstract terms, which is dynamically computed from the environment data. The method is applied on a case study modeling a network monitor. A different approach to achieve a similar objective was proposed in [15], where the authors addressed the proof-based development of system models satisfying a security policy. They used OrBAC [16] models to express the security policies in order to state permissions and prohibitions on actions. An abstract B model is derived from the OrBAC specification of the security policy and then the model is refined to introduce properties that can be expressed in OrBAC. The refinement guarantees that the resulting B model satisfies the security policy.

Bert *et al*. [17] presented a tool to build symbolic labeled transition systems from B specifications. The resulting symbolic transition system represents all the behaviors of the initial B event system. The tool, called GeneSyst, was illustrated on a security property for a model of a smart card purchase transaction protocol. Butler [18] combined CSP and B method refinement in order to verify authentication property. The work does not present a new theoretical framework, instead it describes the use of the above methods to treat refinement of secure communication systems.

Chridi *et al.* [19] presented a decision procedure for

the sub-class of Well-Tagged protocols with Autonomous Keys to analyzing Web Services manipulating sequences of items. Dalal *et al.* [20] provided a comparative and evaluation study for tools used in verification of security protocols. In another related work, Li [21] and Pand extended the strand space method to include time and timstamps to model the notion of recency in Kerberos protocol.

Compared to the above, we address security properties for group oriented protocols, which have special features that were not addressed in any of these approaches, such as the concept of group secrecy and forward secrecy and dynamic group events. In addition, we consider events that are specific for group key protocols that were never treated by the Event-B method. In [13], we used the rank function based inference system to model and verify two parties Diffie-Hellman protocol, while in [5], we presented an approach for modeling and verification of group key protocols by using event-B first-order logic invariant checking. The method is based on a formal link between the semantics of group key protocols model and event-B based on a well-formed connection between event-B invariant and the group key protocol model including its secrecy property. This paper extends the work in [5] to verify forward secrecy property using event-B refinement. We define two models for the group protocol: an abstract model and a refined model. The first one captures secrecy property as an invariant for the abstract model, and the second one captures forward secrecy as an invariant for the refined model.

Many methods were developed to verify certain aspects of security protocols such as secrecy and authenticcation. These methods have successfully tackled a number of protocols though in an ad hoc fashion. On the other hand, using formal methods like model checking can be efficient in the verification of authentication property, while modeling and reasoning about properties like forward secrecy requires first-order-logic based methods such as Event-B.

3. Event-B Method

Event-B [2] is a variant of the B method introduced by Abrial [3] to deal with reactive systems. An event consists of a guard and an action. The guard is a predicate built on state variables and the action is a generalized substitution which defines a state transition. An event may be activated once its guard evaluates to true and a single event may be evaluated at once. The system is assumed to be closed and it means that every possible change over state variables is defined by transitions; transitions correspond to events defined in the model. The B method is based on the concept of machines [3]. A machine is composed of descriptive and operational specifications:

SYSTEM <*name*>
SETS <*sets*>
VARIABLES <*variables*>
INVARIANT <*invariants*>
INITIALISATION <*initialization of variables*>
EVENTS <*events*>
END

A descriptive specification describes what the system does by using a set of variables, constants, properties over constants and invariants which specify properties that the machine's state verify. This constitutes the static definition of the model. Operational specification describes the way the system operates. It is composed of a set of atomic events described by generalized substitutions. An event has a guard and an action, and it may occur only when its guard evaluates to true. An event has one of the general forms where the *SELECT* form is just a particular case of the *ANY* form. *SELECT* takes the form

Name event =
  ANY *P* WHERE
    G
  THEN
    R

and similarly a *SELECT* statement takes the form

Name event =
  WHEN
    G
  THEN
    R

3.1. Event-B Invariant Checking

The consistency of an event-B model is established by proof obligations which guarantee that the initialization verifies the invariant and that each event should preserve the invariant. The guard and the action of an event define a before-after predicate for this event. It describes a relation between variables before the event holds and after this. Proof obligations are produced from events in order to state that the invariant condition is preserved. Let *M* be an event-B model with *v* being variables, carrier sets or constants. The properties of constants are denoted by $P(v)$, which are predicates over constants, and the invariant by $I(v)$. Let *E* be an event of *M* with guard $G(v)$ and before-after predicate $R(v,v')$ that indeed yields at least one after value v'. The initialization event is a generalized substitution of the form $v : init(v)$. Initial proof obligation guarantees that the initialization of the machine must satisfy its invariant: $Init(v) \Rightarrow I(v)$.

Each event *E*, if it holds, has to preserve the invariant. The feasibility statement is illustrated in Lemma 3.1 and

the invariant preservation is given in Lemma 3.2 [4].

Lemma 3.1. $I(v) \wedge G(v) \wedge P(v) \Rightarrow \exists v' \cdot R(v,v')$

Lemma 3.2. $I(v) \wedge G(v) \wedge P(v) \wedge R(v,v') \Rightarrow I(v')$

An event-B model *M* with invariant *I* is well-formed, dented by $M \models I$, only if *M* satisfies all proof obligetions. The B syntax for generalized substitutions defines three predicates: a relation *R*, the subsets of the pre-states where *G* is true of the states in $domain(R)$, and the subset of the pre-state where *P* is true. Let *S* be restricted to evaluations that satisfy the invariant, $S \triangleq \{v | I(v)\}$. Each event can be represented by a binary relation *rel* formally defined as $rel \triangleq \{v \mapsto v' | I(v) \wedge G(v) \wedge R(v,v')\}$. The fact that the invariant $I(v)$ is preserved by event *rel* is simply formalized by saying that *rel* is a binary relation built on *S*: $rel \subseteq S \times S$. It is shown that this binary relation yields to both Lemmas 3.1 and 3.2 above [4].

Lemma 3.1 guarantees that the active part of the relation is a total relation, *i.e.*, when all predicates *I*, *P*, and *G* hold, formally, $G(v) \wedge P(v) \subseteq domain(R(v,v'))$, while Lemma 3.2 guarantees that the postcondition of any operation must satisfy the machine invariant. The initial proof obligation guarantees that the initialization of a machine must satisfy its invariant.

We distinguish special rules for the initialization events. We use $R_I(v,v')$ to denote the predicate of the generalized substitution associated with this event. Then we obtain the following initialization statements [4]:

Lemma 3.3. $P(v) \Rightarrow \exists v' \cdot R_I(v,v')$

Lemma 3.4. $P(v) \wedge R_I(v,v') \Rightarrow I(v')$

3.2. Event-B Refinement

Refinement is a technique to deal with the development of complex systems. It consists in building, starting from an abstract model, a sequence of models of increasing complexity containing more and more details. These details could be introduced when using new variables, adding details to abstract events or adding new events. A model in the sequence is followed by a model it refines. The invariant of the refined model is not weaker than the model it refines and it may contain new variables. The events are the same but may be redefined. It is also used to transform an abstract model into a more concrete version by modifying the state description [22]. The abstract state variables, *v*, and the concrete ones, v_c, are linked together by means of a gluing invariant $J(v,v_c)$. A number of proof obligations ensures that 1) each abstract event is correctly refined by its corresponding concrete version, 2) each new event refines skip, 3) no new event takes control forever, and 4) relative deadlock fairness is preserved. Suppose that an abstract model A_M with variables *v* and invariant $I(v)$ is refined by a concrete model C_M with variables *vc* and gluing invariant $J(v,v_c)$. If $R_A(v,v')$ and $R_C(v_c,v_c')$ are, respectively, the abstract and concrete before-after predicates of the same event, we have to prove the following statement:

$$\left(I(v) \wedge J(v,v_c) \wedge R_C(v_c,v_c')\right) \Rightarrow \exists v' \cdot \left(R_A(v,v') \wedge J(v',v_c')\right)$$

This statement means that under the abstract invariant $I(v)$ and the gluing invariant $J(v,v_c)$, a concrete step $R_C(v_c,v_c')$ can be simulated (v') by an abstract one $R_A(v,v')$ in such a way that the gluing invariant $J(v',v_c')$ is preserved. A new event with before-after predicate $R(v_c,v_c')$ must refine skip $(x'=x)$. This leads to the following statement to prove:

$$I(v) \wedge J(v,v_c) \wedge R_C(v,v_c') \Rightarrow J(v,v_c').$$

Moreover, we must prove that a variant $V(v_c)$ (valuation of variable *v*) is decreased by each new event (this is to guarantee that an abstract step may occur). We have thus to prove the following for each new event with before-after predicate $R_c(v_c,v_c')$:

$$I(v) \wedge J(v,vc) \wedge BA(v,v') \Rightarrow Val(v_{c'}) < Val(v_c).$$

At last, we must prove that a concrete model does not introduce more deadlocks than the abstract one. This is formalized by means of the following proof obligation:

$$I(v) \wedge J(v,v_c) \wedge G(A_M) \Rightarrow G(C_M)$$

where $G(A_M)$ stands for the disjunction of the guards of the events of the abstract model, and $G(C_M)$ stands for the disjunction of the guards of the events of the concrete one. The essence of the refinement relationship is that it preserves already proved system properties including safety properties. The invariant of an abstract model plays a central role for deriving safety properties; the goal is to obtain a formal statement of properties through the final invariant of the last refined abstract model.

4. Event-B Semantics Based Verification Methodology

In order to reason about group protocols in the first-order logics, a map between the group protocol model and event-B model semantics is defined. The event-B tool guarantees the correctness of the invariant w.r.t the event-B model. The map from group protocols to event-B model guarantees certain equivalence between the two models, under certain conditions. Secrecy property is semantically implied in event-B invariant in a defined and proved lemma. Then, a theorem is defined to guarantees that once an event-B invariant is proved against event-B mode, we can conclude that the secrecy property is correct for the group protocol mode.

In the event-B method, compared to higher-order logic, the number of protocol participants that can be considered is limited, finally, modeling forward and backward simultaneously requires more than two levels of abstraction, hence, generates more proof obligations, which will reduce automation as more interaction with the tool will be required to discharge these obligations, this is an open issue to be addressed in future work. **Figure 1** depicts the formal links in the proposed event-B approach.

The security property is defined in the event-B model based on mapping sets, events, and invariants. For forward secrecy, we use refinement of secrecy. The soundness of the event-B model is then established based on a well-formed link to the group protocol model. The validity of the event-B invariant against its model is checked using the Rodin invariant checking tool. This way, we establish the formal link from the Rodin langauge to the group protocol model.

Let $\mathbb{G}$ be a group key protocol model, and let $\mathbb{M}$ be a set of all possible messages (messages space). We choose $\mathbb{S}$ to represent the secret messages space, the set of all secret messages, $\mathbb{S} \subset \mathbb{M}$. Thereafter, we define $\mathbb{E}$ to be the set of all events, or dynamic operations, *i.e.*, join, leave, merge, and split. An event is a term from the message space to the message space, $\mathbb{S} : \mathbb{M} \to \mathbb{M}$. It represents an action the user can perform on the system to update his/her own set of knowledge.

Let $\mathbb{K}_0$ be the set of initial knowledge of the intruder, where $\mathbb{K}_0 \subset \mathbb{M}$. The initial knowledge of the information is collected before executing the protocol events. This information is usually publicly known, $\forall m \in \mathbb{M} \cdot m \in \mathbb{S} \Rightarrow m \notin \mathbb{K}_0$. We then define $\mathbb{K}$ as the set of knowledge of the intruder that is updated by executing events. The system starts with the initial set of knowledge and the set of events, then, by executing a sequence of events, it updates this set. $\mathbb{K}_0 \subseteq \mathbb{K}$ and $\mathbb{K} \subset \mathbb{M}$.

We define a safety property ϕ for a group key protocol model $\mathbb{M}$. This property states that the system cannot execute an event in $\mathbb{E}$ in order to generate a message in $\mathbb{S}$, and is formally modeled as follows:

$$\phi = \forall e \in \mathbb{E} \cdot m' = e(m) \Rightarrow m \notin \mathbb{S}.$$

If this property is correct for the protocol $\mathbb{G}$, then we can write $\mathbb{G} \models \phi$.

Forward secrecy guarantees that a passive adversary who knows a contiguous subset of old group keys (say $\{K_0, K_1, \cdots, K_i\}$) cannot discover any subsequent group key K_j for all i and j, where $j > i$. We will follow this definition in our model for the rest of the paper.

The proposed verification methodology consists of a number of steps as shown in **Figure 2**. In the first step, the group key protocol is specified formally using the model presented before in order to obtain precise protocol specifications. In addition, the secrecy property expected to be checked by the system is described informally. In the second step, the obtained specification is translated into event-B specification using mapping relations to obtain an event-B model that captures the features of the group protocol model. Next, a secrecy property ϕ is specified as an invariant of the resulting

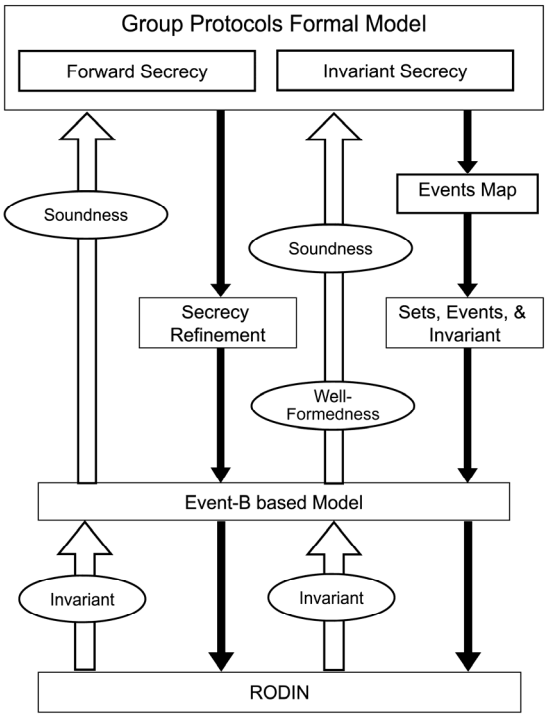

Figure 1. Event-B based approach.

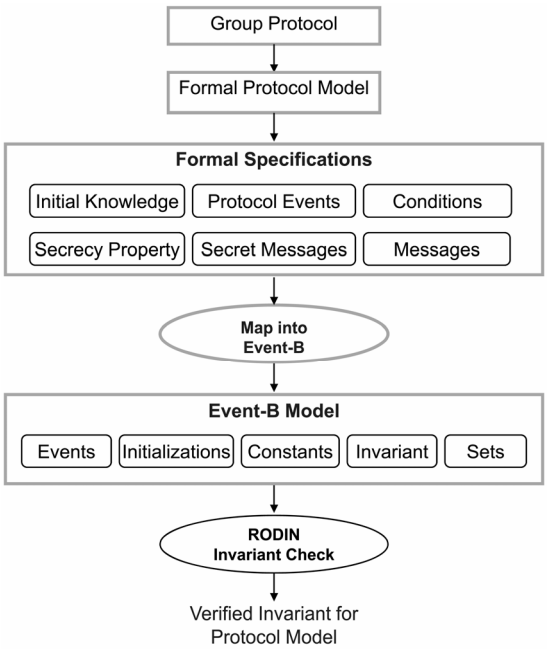

Figure 2. Verification methodology.

event-B model *I*, and a forward secrecy property ϕ_f is specified as an invariant of the refined event-B model. Messages can be defined as a set with an enumeration of all possible secret and known messages. The intruder initial knowledge, $\mathbb{K}_0$, is directly defined as variable or set in the event-B initialization list. Secret messages are defined similarly. Protocol initial constraints, such as $\mathbb{K}_0 \subset \mathbb{M}$ and $\mathbb{S} \subset \mathbb{M}$, are defined as properties that will be included in the invariant. Protocol join or leave events are defined as event-B operations that update the intruder's knowledge and the set of secret messages, including the new generated key. Finally, the property is checked from the obtained global system specification using the event-B invariant checking in Rodin platform.

In **Figure 3**, protocol events and execution traces are mapped into event-B events, messages generation conditions are mapped into events guards, and messages sets are used to generate event-B model constants properties. The initial knowledge is defined as event-B initializetions, messages are mapped directly into sets, and finally the secrecy property is defined as an invariant for the event-B model. The generation of the target event-B model requires treating three parts: the static part which includes initializations and the constant properties of the protocol, the dynamic part that represents events of the protocol, and finally, enriching the resulting model with invariants describing the required secrecy properties.

The event-B semantics is close to the protocol model semantics. This relationship is demonstrated by establishing a well-formed link between the semantics of both models. To achieve this link, we are interested in showing that if the invariant *I* holds for event-B system *M*, then the safety property ϕ must hold for the group protocol model $\mathbb{G}$. Formally, $(M \models I) \Rightarrow (\mathbb{G} \models \phi)$. In terms of equivalence between the two models, we can say that the event-B model *M* is an abstract model of the protocol model $\mathbb{G}$, with regards to the security property, if the property ϕ holds in the model $\mathbb{G}$, and the invariant *I* holds in the abstract model *M*. To illustrate this equivalence, we need to show that $I \Rightarrow \phi$. Therefore, it is enough to show that the invariant *I*, with regards to *M*, implies the safety property ϕ, with regard to $\mathbb{G}$.

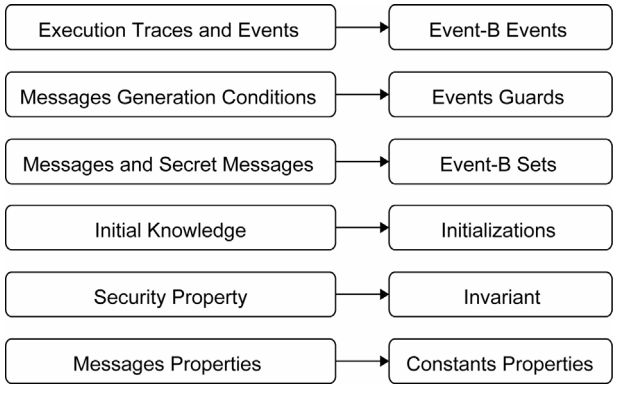

Figure 3. Mapping protocol primitives into Event-B.

4.1. Verification of Secrecy as Event-B Invariant

To show that $I \Rightarrow \phi$, we need to establish a well-formed link between event-B invariant and the safety property. We split this formal link into two parts: the first deals with the initialization, and the second deals with executing the events. For this, we need to relate messages in $\mathbb{G}$ to variables in *M* by mapping public messages and secret messages to event-B sets and messages sets to event-B constants properties. This map relates the variable *m* over the set of messages $\mathbb{M}$ directly to the variable *v* over event-B carrier sets and constants. The semantical correspondence between the variable *m* and the variable *v* is defined by this map.

Theorem 4.1. *Secrecy Soundness.*
A group protocol, *G*, satisfies its secrecy property, ϕ, if there is an equivalent abstract event-B model, *M*, that satisfies an event-B invariant, *I*, and implies the property ϕ. More formally, for a model *G*, we need to find an abstract model *M* where $G \succcurlyeq M$ means that the event-B model *M* can be abstracted from the group protocol *G*. This abstraction will be defined later. Let $(M \models I)$, and $(I \Rightarrow \phi)$ be correct lemmas, then,

$$(G \succcurlyeq M) \wedge (M \models I) \wedge (I \Rightarrow \phi) \Rightarrow (\mathbb{G} \models \phi).$$

The proof is divided into two parts, in the first we assume that $G \succcurlyeq M$ holds, then we prove the theorem based on that. In the next stage we prove each lemma separately and we identify the relation between *G* and *M* such that $G \succcurlyeq M$ holds.

Proof. Given $(G \succcurlyeq M)$, $(M \models I)$, and $(I \Rightarrow \phi)$, we can deduce

$$(M \models I) \wedge (I \Rightarrow \phi) \Rightarrow (M \models \phi)$$

$$(G \succcurlyeq M) \wedge (M \models \phi) \Rightarrow (G \models \phi)$$

We establish the abstract relation between *G* and *M* such that $G \succcurlyeq M$ holds, then we prove the lemma $(I \Rightarrow \phi)$. The lemma $(M \models I)$ is assumed to be correct in the event-B tool.

Definition 4.1. A group protocol model *G*, is abstracted to an event-B model, *M* under certain conditions and semantically correct map from *G* to *M*. $G \succcurlyeq M$ is defined as follows:

For every component and condition in *G* there is an equivalent one in *M*. A protocol model is composed of $\mathbb{M}, \mathbb{S}, \mathbb{K}, \mathbb{K}_0, \mathbb{E}, \phi$, we map each component in *G* into an equivalent one in *M*.

Messages sets are mapped into an event-B variable by defining *v* over the set $\mathbb{M}$, and messages sets relations are mapped to event-B constants properties. $P(v)$ is a

function of $\mathbb{M}$. These relations include the predicates about sets that should always hold.

Messages generation conditions are mapped into events guards. These conditions include predicates that should hold prior to executing an event, like having the appropriate key to encrypt or decrypt a message.

$$G(v) = condition(\mathbb{M}, \mathbb{E})$$

The secrecy property, ϕ, is mapped into an event-B invariant, *I*. This map is defined in Lemma 4.1.

An event in $\mathbb{E}$ is mapped into event *R*, an event, $e_c \in \mathbb{E}$ with a precondition *condition*, where $m' = e_c(m_1, m_2, \cdots)$, the message generated from executing the even, can be defined concretely using an event-B statement such as:

Name event =
ANY mWHERE
condition
THEN
$R(m, m')$

This map defines the relation $G \succcurlyeq M$.

To show that $I \Rightarrow \phi$, we need to establish a well-formed link between event-B invariant and the safety property. We split this formal link into two parts: the first deals with the initialization, and the second deals with executing the events. First we relate messages in $\mathbb{G}$ to variables in *M*. In **Figure 3**, we describe a map from public messages and secret messages to event-B sets and a map from messages sets relations to event-B constants properties. This map relates the variable *m* over the set of messages $\mathbb{M}$ directly to the variable *v* over event-B carrier sets and constants. The semantical correspondence between the variable *m* and the variable *v* is defined by this map.

We define the invariant *I* as $I = I_{init} \wedge I_E$, where I_{init} is the invariant predicate under the initial conditions, and I_E is the invariant predicate under executed events. Similarly, we define the safety property $\phi = \phi_{init} \wedge \phi_E$.

Lemma 4.1. $(I \Rightarrow \phi) = ((I_{init} \Rightarrow \phi_{init}) \wedge (I_E \phi \Rightarrow \phi_E))$

Proof. We define the well-formed conditions that guarantee the correctness of this Lemma in two steps, we first show that $(I_{init} \Rightarrow \phi_{init})$. We identify the initial events and initial set of messages in $\mathbb{G}$ under which the formula $(I_{init} \Rightarrow \phi_{init})$ holds. Then we define the predicates *P*, *I*, *G*, and *R* presented in Lemmas 3.1 and 3.2 for the protocol model $\mathbb{G}$ such that Lemma 4.1 holds.

The definition of the group key protocol must satisfy the initial soundness conditions: $\mathbb{K}_0 \cap \mathbb{S} = \varnothing$ and $\forall e_i \in \mathbb{E}_i \cdot m' := e_i(m) \Rightarrow m' \notin \mathbb{S}$, where e_i is an initial event that can be applied on the intruder's initial set of messages. We choose $R_I = \mathbb{E}_0$ to be the set of events that can be executed on $\mathbb{K}_0$.

We will define the constants property *P* and the initialization predicate R_I for the model $\mathbb{G}$ that will satisfy Lemmas 3.3 and 3.4. Then we define *P*, *R*, the predicate guards *G*, and the invariant *I* for the model $\mathbb{G}$ that will satisfy Lemmas 3.1 and 3.2.

Case 1. $(I_{init} \Rightarrow \phi_{init})$

- $P(m) = (\mathbb{K}_0 \neq \varnothing) \wedge (\mathbb{K}_0 \subset \mathbb{M}) \wedge (\mathbb{K} = \mathbb{K}_0)$
- $R_I = (e_i \in \mathbb{E}) \wedge (\exists (m' \in \mathbb{M}, m \in \mathbb{K}_0) \cdot m' := e_i(m))$
- $I(m) = m \in \mathbb{K}_0 \Rightarrow m \notin \mathbb{S}$

The message generation event $m' := e_i(m)$ is equivalent to the transition relation $R_I(v, v')$. This yields the formula $P(m) \Rightarrow \exists e_i \in \mathbb{E}_i \cdot m' := e_i(m)$ which is exactly Lemma 3.3 considering that $R_I = e_i$.

The invariant definition for the model $\mathbb{G}$ is $I(m) = m \in \mathbb{K} \Rightarrow m \notin \mathbb{S}$. We need to show that the invariant *I* holds for both $I(m)$ and $I(m')$. Since the protocol is initially sound, then both $I(m)$ and $I(m')$ hold by the fact that $\mathbb{K}_0 \cap \mathbb{S} = \varnothing$ and that the initial events cannot generate secret messages in *S*. If $m' := e_i(m)$ then $m' \notin \mathbb{S}$. Therefore we can write $(P(m) \wedge (m' := e_i(m))) \Rightarrow I(m')$, which corresponds to Lemma 3.3 considering that $R_I = e_i$.

Case 2. $(I_E \Rightarrow \phi_E)$

- $P(m) = (\mathbb{K} \subset \mathbb{M})$
- $I(m) = (m \in \mathbb{K} \Rightarrow m \notin \mathbb{S})$
- $G(m) = \left(\left(\{m\}_k := encr(m, k) \Rightarrow k \in \mathbb{K} \right) \wedge \left(m := decr(\{m\}_k, k) \Rightarrow m \in \mathbb{K} \right) \right)$
- $R = (e \in \mathbb{E}) \wedge (\exists m \in \mathbb{K}, m' \in \mathbb{M} \cdot m' = e(m))$

This message generation event is equivalent to the transition relation $R(v, v')$. Therefore, applying the predicates *P*, *I*, and *G* will lead to the relation *R*. We can write the formula $P(m) \wedge I(m) \wedge G(m) \Rightarrow \exists e \in \mathbb{E} \cdot m' = e_i(m)$ which is equivalent to Lemma 3.1 considering that the relation *R* is equivalent to an existing event $e \in \mathbb{E}$.

The validity of the invariant $I(m')$ for the model $\mathbb{G}$ is expressed by the validity of the predicates *P*, *I*, *R*, and *G*, where $m' := e(m)$. This can be written as $I(m) \wedge P(m) \wedge G(m) \wedge R \Rightarrow I(m')$, which corresponds to Lemma 3.2.

Under these conditions, we guarantee that when the invariant holds in event-B model, the secrecy property definition holds for the group key protocol model. These predicates should be considered carefully when providing the event-B implementation. Properties that can be expressed as invariants are verified using the translation process and the event-B tool.

This completes the proof of Theorem 4.1. The major restriction on this method is that it can reason about the execution of a single protocol event, *i.e.*, join or leave. However, this is enough to model and verify group secrecy when a member joins or leaves the group.

4.2. Verification of Forward Secrecy Using Event-B Refinement

Figure 4 illustrates the modifications required on the verification methodology in order to support forward secrecy. When the invariant *I* holds for event-B system *M*, the safety property ϕ must also hold for the group protocol model $\mathbb{G}$. The verification methodology for forward secrecy is built on top of the methodology we use for secrecy. In order to apply invariant checking on forward secrecy, we will consider the model *M* as an abstract one, and define a refined model, M_c, and a gluing invariant *J* linking variables of the abstract model to those of the concrete or refined one (M_c). In addition to the previous map defined from *G* to *M*, we will use the variable v_c to represent the set of intruders messages in the refined model. Therefore, $J(v, v_c)$ will represent the gluing invariant which represents forward secrecy property be- fore the relation R_c. In addition, we use $I(v_c)$ to rep- resent the invariant in the refined model, which corre- sponds to the secrecy property of the refined protocol model, G_c. The relation R_c will be defined the same way as the relation *R*.

In event-B, refinement can be done with events or variables. In our case, the group protocol join or leave events have the same semantics in both secrecy and forward secrecy, therefore, it will have the same definition in both the abstract and refined event-B models, *i.e.*, $R_c = R$. We use $R(v, v')$ to represent join or leave events, which update the intruder's set of knowledge, the variable *v* here. In the refined model, we will use the same relation, we call it $R_c(v_c, v_c')$, that will update the intruder's set of knowledge, v_c, in the refined model M_C.

The correctness of forward secrecy, ϕ_f, with regards to the event-B concrete model M_c is achieved through the correctness of the gluing invariant $J(v', v_{c'})$. **Figure 5** below illustrates the link between the abstract and refined model to achieve a model for forward secrecy in event-B.

A group protocol, G_c, satisfies its secrecy property, ϕ_f, if there is an equivalent event-B model, *M*, that satisfies an event-B invariant, *I*, and a refined event-B model M_c that satisfies an event-B invariant *I*, and a gluing variable $J(v, v_c)$ that implies ϕ_f in the existence of a relation $R_c(V_c, V_{c'})$.

Formally, given $(G_c \equiv M_c)$, $(I_c \Rightarrow \phi_f)$, $(M \models I)$, and $(M \models I_c)$, then

$$(G_c \equiv M_c) \wedge (M_c \models I_c) \wedge (J \Rightarrow \phi_f) \Rightarrow (G_c \models \phi_f).$$

Assuming $(G_c \equiv M_c)$, $(J \Rightarrow \phi_f)$, and $(M_c \models J)$, we can deduce:

$$(M_c \models J) \wedge (J \Rightarrow \phi_f) \Rightarrow M_c \models \phi_f .$$

$$(G_c \equiv M_c) \wedge M_c \models \phi_f \Rightarrow (G_c \models \phi_f).$$

This way, we guarantee that once a refined event-B model is verified against its invariant, then the group protocol model is verified against its forward secrecy property.

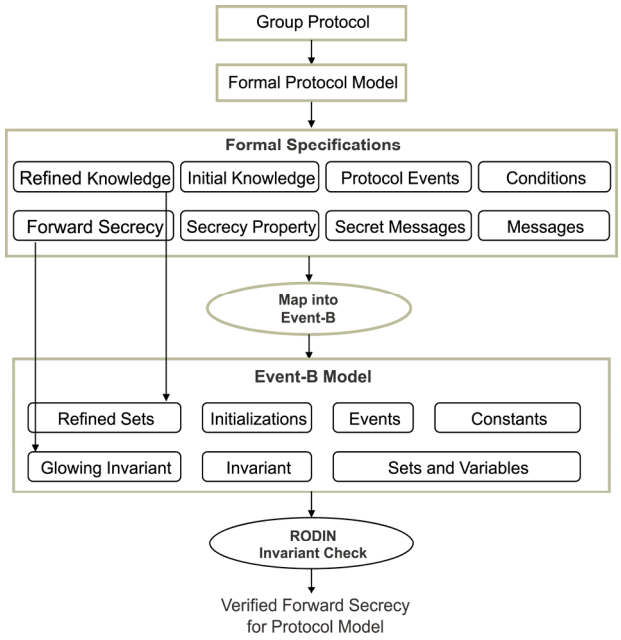

Figure 4. Refined Event-B method for forward secrecy.

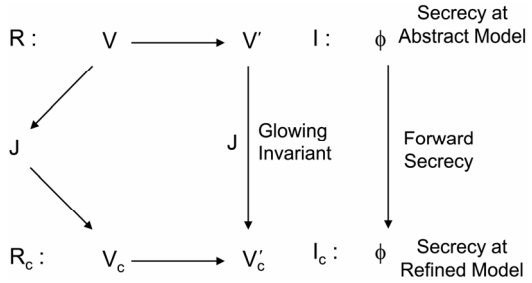

Figure 5. Relationship between abstract and refined models.

Event-B invariant checking cannot reason about backward secrecy because invariants cannot be used in a reverse manner, *i.e.*, refining the intruder's knowledge back in time. In backward secrecy the intruder is assumed to be an active user in the group while trying to discover older secret shares prior to his/her membership. Therefore, refinement of secrecy can only be used for forward secrecy. Based on this, key independence (collusion) cannot also be modeled in this method as is.

5. Case Study: The TGDH Protocol

In this section, we apply the approach on a group key protocol that generates a key in a distrusted group. We show how the conditions defined for the correctness of the above model can be concretely applied on a real protocol. The intended secrecy property, along with its conditions, are efficiently defined and checked as event-B

invariant.

The Tree-based Group Diffie-Hellman protocol (TGDH) is intended for secure key generation. **Figure 6** shows a binary tree structure that represents the group members, their own secret shares, and the secret sub-keys on every node up to the root. As part of the protocol, a group member can take on a special sponsor role, which involves computing intermediate keys and broadcasting to the group. Each broadcasted message contains the senders view of the key tree, which contains each blind key known to the sender [1].

A group key can be computed from any members secret share and all blind keys on the co-path to the root. Blind keys are the siblings of the nodes on the key path. The members own secret share and all sibling blind keys on the path to the root enable a member to compute all intermediate keys on its key-path, including the root group key.

The group key is calculated by each member based on his/her key-path and blind keys. For instance, for a member M_3 at node n_3, the key-path is the set of messages $\left\{n_3, g^{n_2 n_3}, g^{n_1 g^{n_2 n_3}}\right\}$. The set of blind keys ordered as they appear up to the root is $\left\{g^{n_2}, g^{n_1}, g^{g^{n_6 g^{n_4 n_5}}}\right\}$. The group key at the root is calculated directly using the two sets:

$$GroupKey = g^{g^{n_1 g^{n_2 n_3}}\, g^{n_6 g^{n_4 n_5}}}$$

The protocol designers presented four types of security properties: group key secrecy, forward secrecy, backward secrecy, and key independence. The authors of [1] provided an informal proof that their protocol satisfies these security property. In this work, we provide a formal proof for group key secrecy property under certain conditions. This property can be described as a *correct key construction property*, which guarantees that only group members, who are of knowledge to their own private shares, can calculate the group key at root. On the other hand, an adversary, who knows all blind sub-keys cannot find a full path to calculate the root key.

We illustrate our method on a group protocol composed of three members, then we apply a join event for a fourth member. **Figure 7** shows the modification on the tree structure when a new member joins the group, we define the group protocol components before and after this event takes place. Assuming that a passive adversary is monitoring the group activity, the knowledge set is built based on the blind keys interchanged between members. Based on this, we show all group protocol components, including secrecy property, and the equivalent event-B model including the invariant, before the

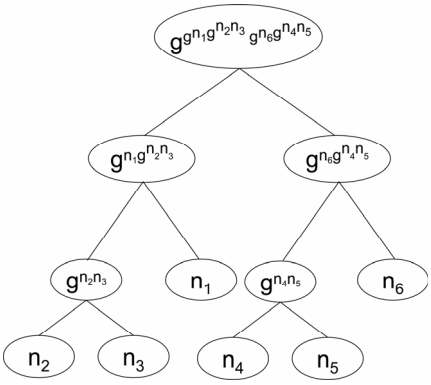

Figure 6. Tree-based GDH protocol binary tree structure.

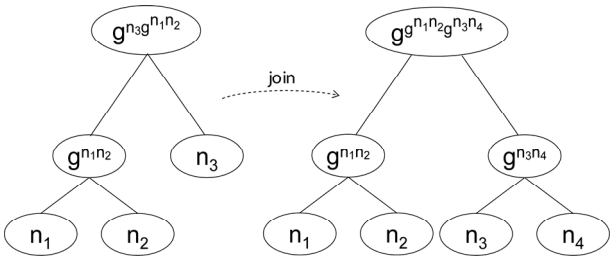

Figure 7. Join event in the TGDH protocol.

join event takes place:

$$M = \left\{n_1, n_2, n_3, g^{n_1}, g^{n_2}, g^{n_3}, g^{n_1 n_2}, g^{g^{n_1 n_2}}, g^{n_3 g^{n_1 n_2}}\right\}$$

$$S = \left\{n_1, n_2, n_3, g^{n_1 n_2}, g^{n_3 g^{n_1 n_2}}\right\}$$

$$K_0 = \left\{n_i, g^{n_i}\right\}$$

$$K = \left\{n_i, g^{n_i}, g^{n_1}, g^{n_2}, g^{n_3}, g^{g^{n_1 n_2}}\right\}$$

$$GroupKey = g^{n_3 g^{n_1 n_2}}$$

where the sets of messages $M \subset \mathbb{M}$, $K_0 \subset \mathbb{K}_0$, $K \subset \mathbb{K}$, and $S \subset \mathbb{S}$.

Then, we show the same components after the join event of a new member with a new secret contribution n_4. Note that group key secrecy has the same definition and should be valid always, before and after a join (or leave) event takes place.

$$M = \left\{n_1, n_2, n_3, n_4, g^{n_1}, g^{n_2}, g^{n_3}, g^{n_4}, g^{n_1 n_2}, g^{n_3 n_4}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}}, {}^{g^{n_1 n_2} g^{n_3 n_4}}\right\}$$

$$S = \left\{n_1, n_2, n_3, n_4, g^{n_1 n_2}, g^{n_3 n_4}, {}^{g^{n_1 n_2} g^{n_3 n_4}}\right\}$$

$$K = \left\{n_i, g^{n_i}, g^{n_1}, g^{n_2}, g^{n_3}, g^{n_4}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}}\right\}$$

$$GroupKey = g^{g^{n_1 n_2} g^{n_3 n_4}}$$

$$\phi = GroupKey \notin \mathbb{K} \wedge \mathbb{K} \cup \mathbb{S} = \varnothing$$

5.1. Secrecy Model in Event-B Invariant

The event-B model for the protocol components is described below. We describe the current status of the group by initializations appropriate sets for messages and keys. The secrecy property is defined as an invariant that combines a set of conditions to be satisfied at the initialization and after executing the event: $K \cap S = \varnothing$. Some of the protocol characteristics can also be encoded within this invariant, such as $K \subset M \wedge S \subset M$. We also define an event to represent the protocol action (join/leave).

The TGDH protocol components are defined within the event-B model. The group key has basically the same definition, and secrecy property is defined as an event-B invariant that contains, in addition to group key secrecy, certain conditions on messages sets to ensure the consistency of the map,

$$\left(K \cap S = \varnothing\right) \wedge GKey \notin K \wedge K \subset M \wedge S \subset M \ .$$

An event-B definition that captures the behavioral semantics of a join event which will result in updating the intruder's set of knowledge is described as follows:

MACHINE M0
SEES P0
VARIABLES
K GKey S
pwn
INVARIANTS
inv1: $K \subset Key \wedge GKey \in Key \wedge S \subset M$
inv2: $pwn \in N \times N \mapsto K$
inv3: $K \cap S = \phi \wedge GKey \notin K$
EVENTS
Initialisation
begin
act1: $M := \{n_1, n_2, n_3, pwn(g, n_1), pwn(g, n_2), pwn(g, n_3), pwn(g, n_1 * n_2), pwn(g, n_2 * n_3), pwn(g, pwn(n_1 * n_2)), pwn(g, n_3 * g(n_1, n_2))\}$
act2: $K := \{n_i, pwn(g, n_i), pwn(g, n_1), pwn(g, n_2), pwn(g, n_3), pwn(g, pwn(g, n_1 * n_2))\}$
act2: $Key := \{pwn(g, n_3 * pwn(g, n_1 * n_2))\}$
act3: $S := \{n_1, n_2, n_3, pwn(g, n_1 * n_2), pwn(g, n_3 * pwn(g, (n_1 * n_2)))\}$
act4: $GKey := \{pwn(g, n_3 * pwn(g, n_1 * n_2))\}$
end
Event *join*
any
n_4
where
grd1: $n_4 \notin M$
then
act5: $M := M \cup \{n_4, pwn(g, n_4), pwn(g, n_3 * n_4), pwn(g, pwn(g, n_3 * n_4) * pwn(g, (n_1 * n_2)))\}$
act6: $S := S \cup \{n_4, pwn(g, n_3 * n_4), pwn(g, pwn(g, n_3 * n_4) * pwn(g, (n_1 * n_2)))\}$
act7: $GKey := pwn(g, pwn(g, n_3 * n_4) * pwn(g, (n_1 * n_2)))$
act8: $Key := Key \{GKey\}$
act8: $K := K \{n_4, pwn(g, pwn(g, n_3 * n_4))\}$
Event *IntEvnt*
any
$m1, m2$
where
grd1: $m1 \in K \wedge m2 \in K$
then
act1: $K := K \cup \{pwn(m1, m2)\}$
END

The intruder set of knowledge is modeled with K in the above model, and is initialized with the public set of knowledge. We assume the intruder has access to the knowledge of users leaving the group. Therefore, with the execution of each event, this knowledge is updated dynamically. The function *pwn* models the exponent operator used to calculate keys. After the join event is executed, the new key will be generated and added to that set of secret keys based on the contribution of the joined member, n_4, as follows:

$$GKey = g^{g^{n_1 n_2} g^{n_3 n_4}}$$

For the verification of the invariants, we first consider the static case of key construction under the assumption that basic DH key construction (on tree leaf nodes) is correct. We then consider the dynamic case by applying events such as join and leave and verify the correctness of key construction for a bounded tree size and bounded number of events. The event-B invariant has been proven totally. The number of generated proof obligations are three, all proof obligations are discharged, and then the initial model of the group key protocol is validated. The event-B invariant, I, defined in the Rodin platform above, implies the group protocol secrecy semantically, $I \Rightarrow \phi$. The event-B tool guarantees that $M \models I$. We have shown in the previous section that the group protocol G is mapped into an event-B model M. Therefore, we can conclude the correctness of the secrecy property ϕ for the protocol model G, $\mathbb{G} \models \phi$.

The proposed solution allows us to verify the required property, however, one limitation of our approach is its applicability on bounded number of participants and protocol events, this issue will be addressed in the future work. Another limitation is due to the fact that we verify

the property under the execution of a single event. However, this approach is sufficient for the target property, where key distribution is abstracted away because we are concerned only with modeling key construction but not key distribution or authentication property. Even though there are some limitations for the approach, event-B can be used in modeling specific protocols behaviors, like key construction, and tree-based protocol primitives can be modeled directly in event-B for safety properties verification.

5.2. Forward Secrecy Model with Event-B Refinement

We illustrate the method on a group protocol composed of four members, then we apply a leave event for a specific member. **Figure 8** shows the modification on the tree structure when a new member leaves the group, we define the group protocol components before and after this event takes place. This represents the abstract model. When a new member joins the group, the knowledge set is built based on the blind keys interchanged between group members and the observing member's old set of knowledge who left the group earlier (**Figure 8**). This represents the refined model.

In **Figure 8(a)**, the current set of messages, secret set of messages, and the current group key are defined, respectively, as follows:

$$M = \left\{ n_1, n_2, n_3, g^{n_1}, g^{n_2}, g^{n_3}, {}^{n_1 n_2}, g^{g^{n_1 n_2}}, g^{n_3 g^{n_1 n_2}} \right\}$$

$$S = \left\{ n_1, n_2, n_3, g^{n_1 n_2}, g^{n_3 g^{n_1 n_2}} \right\}$$

$$GroupKey_a = g^{n_3 g^{n_1 n_2}}$$

Then, after member M_4 joins the group, as shown in **Figure 8(b)**, the above variables become:

$$M = \left\{ n_1, n_2, n_3, n_4, g^{n_1}, g^{n_2}, g^{n_3}, g^{n_4}, g^{n_1 n_2}, g^{n_3 n_4}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}}, g^{n_3 g^{n_1 n_2}}, g^{g^{n_1 n_2} g^{n_3 n_4}} \right\}$$

$$S = \left\{ n_1, n_2, n_3, n_4, g^{n_1 n_2}, g^{n_3 n_4}, {}^{g^{n_1 n_2} g^{n_3 n_4}} \right\}$$

$$GroupKey_b = g^{g^{n_1 n_2} g^{n_3 n_4}}$$

The next step, is that we let member M_1, who will be assumed to be dishonest later, leave the group, where a new secret share n_2' is generated as in **Figure 8(c)**. The above variables become:

$$M = \left\{ n_1, n_2, n_2', n_3, n_4, g^{n_1}, g^{n_2}, g^{n_2'}, g^{n_3}, g^{n_4}, g^{n_1 n_2}, g^{n_3 n_4}, g^{g^{n_1 n_2}}, {}^{g^{n_3 n_4}}, g^{n_3 g^{n_1 n_2}}, g^{g^{n_1 n_2} g^{n_3 n_4}}, g^{n_2' g^{n_3 n_4}} \right\}$$

$$S = \left\{ n_2', n_3, n_4, g^{n_3 n_4}, g^{n_2' g^{n_3 n_4}} \right\}$$

$$GroupKey_c = g^{n_2' g^{n_3 n_4}}$$

Finally, **Figure 8(d)** is the join (or similarly leave) event on which we will check invariant for the refined model. The event represents the join event for the new member M_5. M_1 is a dishonest user who will take advantage of this event. We first illustrate secrecy, then forward secrecy:

$$M = \left\{ n_1, n_2, n_2', n_3, n_4, n_5, g^{n_1}, g^{n_2}, g^{n_2'}, g^{n_3}, g^{n_4}, g^{n_5}, g^{n_1 n_2}, g^{n_3 n_4}, g^{n_5 n_2'}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}}, g^{g^{n_5 n_2'}}, g^{n_3 g^{n_1 n_2}}, g^{g^{n_1 n_2} g^{n_3 n_4}}, g^{g^{n_5 n_2'} g^{n_3 n_4}}, g^{n_2' g^{n_3 n_4}} \right\}$$

$$S = \left\{ n_2', n_3, n_4, n_5, g^{n_3 n_4}, g^{n_5 n_2'}, g^{n_2' g^{n_3 n_4}}, g^{g^{n_5 n_2'} g^{n_3 n_4}} \right\}$$

$$GroupKey_d = g^{g^{n_5 n_2'} g^{n_3 n_4}}$$

The intruder knowledge, *K*, as viewed by a member monitoring the group from outside, before M_5 joins the group is defined as:

$$K = \left\{ n_i, g^{n_i}, g^{n_1}, g^{n_2}, g^{n_2'}, g^{n_3}, g^{n_4}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}} \right\}$$

Then we show the set of messages when member M_5 joins the group:

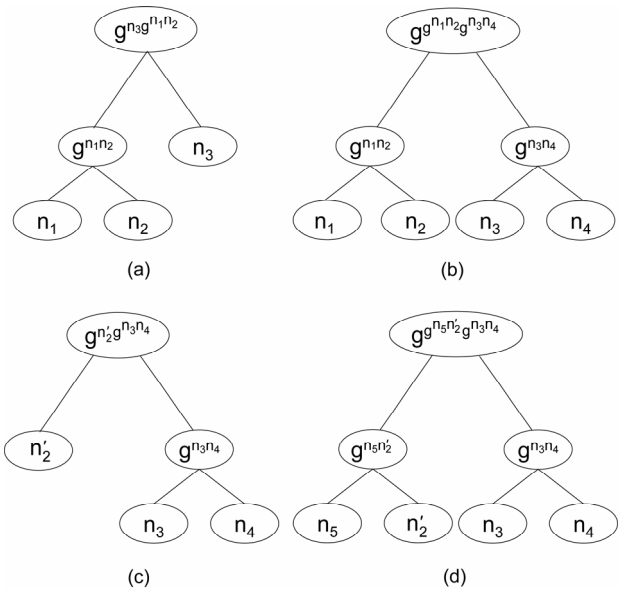

Figure 8. Forward secrecy in the TGDH protocol. (a) M_1 is member of the group; (b) M_4 joins the group; (c) M_1 leaves the group; (d) M_5 joins the group.

$$K = \left\{ n_i, g^{n_i}, g^{n_1}, g^{n_2}, g^{n_2'}, g^{n_3}, g^{n_4}, g^{n_5}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}}, g^{g^{n_5 n_2'}} \right\}$$

Secrecy implies that the intruder monitoring the group should not be able to calculate the group key $GroupKey_d$ (or any secret share or sub-key).

In forward secrecy, the set of messages K is refined with the knowledge gained by user M_1 while member in the group, and is denoted K':

$$K' = \left\{ n_i, n_1, g^{n_i}, g^{n_1}, g^{n_2}, g^{n_2'}, g^{n_3}, g^{n_4}, g^{g^{n_5 n_2'}}, g^{n_3 g^{n_1 n_2}}, g^{n_5}, g^{n_1 n_2}, g^{g^{n_1 n_2}}, g^{g^{n_3 n_4}} \right\}$$

$$GroupKey_d = g^{g^{n_5 n_2'} g^{n_3 n_4}}$$

$$\phi_f = GroupKey_d \notin \mathbb{K}' \wedge \mathbb{K}' \cup \mathbb{S} = \varnothing$$

Group key protocol secrecy properties were defined as invariants in Rodin platform, the tool generates proof obligations that were successfully discharged using Event-B proof control. The results achieved here are important because our method allows semi-automated verification of these properties under dynamic operation of the protocol. In addition, the method allows modeling protocols at different levels of abstraction, where the model can be further refined in order to include more details about the protocol operation.

6. Conclusions

In this paper, we provided invariant checking approach for group key secrecy and forward secrecy. We used event-B invariants to model and verify group key secrecy, then, on top of this, we used event-B refinement to model and verify forward secrecy. For this purpose, a formal link between the semantics of the group protocol model and event-B was established. The result was combining the event-B and group protocol model to be able to use specific features in event-B to model protocol actions and verify the required property. However, we restrict the group protocol model to be verified to certain conditions in order to guarantee the correctness of the method and the applicability of first-order logic theorem proving. This includes the number of participants, abstracting the exponentiation operator for Diffie-Hellman style protocols, and finally, applying a single protocol event.

We applied this approach on a group key protocol, the tree based Group Diffie-Hellman protocol and provided invariant checking for secrecy under the static and the dynamic case by applying a single event (join/leave). In contrast to our work, the authors of this protocol [1] provided an informal, non-intuitive and simple proof for secrecy property.

Even though our approach can model and verify only invariant properties, while liveness related properties cannot be modeled in our approach, we believe that invariant checking is adequate to model secrecy related properties. This is due to the target model and the verification tool, namely, event-B and Rodin platform. Event-B supports only safety related properties which are modeled and verified as invariants. In addition, a limited class of liveness properties can be modeled using invariants, such as termination, while more general liveness properties are not supported yet in Event-B. Hoang and Abrial [23] have an ongoing research for reasoning about liveness properties in Event-B.

As future work, an interesting issue to be considered is modeling and verifying liveness security properties using event-B. In addition, an extension of the event-B based model to handle a parameterized number of participants shall be explored. It will also be interesting to investigate modeling backward secrecy and key independence using event-B.

REFERENCES

[1] Y. Kim, A. Perrig and G. Tsudik, "Tree-Based Group Key Agreement," *ACM Transactions on Information and Systems Security*, Vol. 7, No. 1, 2004, pp. 60-96. doi:10.1145/984334.984337

[2] J. Abrial, "Modelling in Event-B: System and Software Engineering," Cambridge University Press, Cambridge, 2009.

[3] J. Abrial, "The B-Book: Assigning Programs to Meanings," Cambbridge University Press, Cambridge, 1996. doi:10.1017/CBO9780511624162

[4] C. Metayer, J. Abrial and L. Voisin, "RODIN Deliverable 3.2: Event-B Language," *Technical Report Project IST*-511599, School of Computing Science, University of Newcastle, Newcastle, 2005.

[5] A. Gawanmeh, L. J. B. Ayed and S. Tahar, "Event-B Based Invariant Checking of Secrecy in Group Key Protocols," *Local Computer Networks*, IEEE Computer Society Press, New York, 2008, pp. 950-957.

[6] M. Steiner, G. Tsudik and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," *Conference on Computer and Communications Security*, ACM Press, London, 1996, pp. 31-37.

[7] J. Abrial, M. Butler, S. Hallerstede and L. Voisin, "An Open Extensible Tool Environment for Event-B," *International Conference on Formal Methods and Software Engineering*, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Vol. 4789, 2006, pp. 588-605.

[8] "Rodin Platform," 2011. http://www.event-b.org

[9] F. F´abrega, "Strand Spaces: Proving Security Protocols Correct," *IOS Journal of Computer Security*, Vol. 7, No. 2-3, 1999, pp. 191-230.

[10] L. Paulson, "The Inductive Approach to Verifying Cryptographic Protocols," *IOS Journal of Computer Security*, Vol. 6, No. 1-2, 1998, pp. 85-128.

[11] F. Crazzolara and G. Winskel, "Events in Security Protocols," *ACM Conference on Computer and Communications Security*, ACM Press, London, 2001 pp. 96-105.

[12] C. Cremers and S. Mauw, "Operational Semantics of Security Protocols," *Scenarios*: *Models*, *Transformations and Tools*, *LNCS*, Springer-Verlag, Berlin, Vol. 3466, 2005, pp. 66-89. doi:10.1007/11495628_4

[13] A. Gawanmeh, A. Bouhoula and S. Tahar, "Rank Functions Based Inference System for Group Key Management Protocols Verification," *International Journal of Network Security*, Vol. 8, No. 2, 2009, pp. 207-218.

[14] N. Stouls and M. Potet, "Security Policy Enforcement through Refinement Process," *Formal Specification and Development in B*, *LNCS*, Springer-Verlag, Berlin, Vol. 4355, 2007, pp. 216-231. doi:10.1007/11955757_18

[15] N. Benaissa, D. Cansell and D. Mery, "Integration of Security Policy into System Modeling," *Formal Specification and Development in B*, *LNCS*, Springer-Verlag, Berlin, Vol. 4355, 2007, pp. 232-247. doi:10.1007/11955757_19

[16] A. Abou El Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel and G. Trouessin, "Organization Based Access Control," *International Workshop on Policies for Distributed Systems and Networks*, IEEE Computer Society Press, New York, 2003, pp. 120-131.

[17] D. Bert, M. Potet and N. Stouls, "GeneSyst: A Tool to Reason about Behavioral Aspects of B Event Specifications. Application to Security Properties," *Formal Specification and Development in Z and B*, *LNCS*, Springer-Verlag, Berlin, Vol. 3455, 2005, pp. 299-318. doi:10.1007/11415787_18

[18] M. Butler, "On the Use of Data Refinement in the Development of Secure Communications Systems," *Formal Aspects of Computing*, Vol. 14, No. 1, 2002, pp. 2-34. doi:10.1007/s001650200025

[19] N. Chridi, M. Turuani and M. Rusinowitch, "Decidable analysis for a Class of Cryptographic Group Protocols with Unbounded Lists," *Computer Security Foundations Symposium*, IEEE Computer Society Press, New York, 2009, pp. 277-289.

[20] N. Dalal, J. Shah, K. Hisaria and D. Jinwala, "A Comparative Analysis of Tools for Verification of Security Protocols," *International Journal of Communications*, *Network and System Sciences*, Vol. 3, No. 10, 2010, pp. 779-787. doi:10.4236/ijcns.2010.310104

[21] Y. Li and J. Pang, "Extending the Strand Space Method with Timestamps: Part I the Theory," *International Journal of Communications*, *Network and System Sciences*, Vol. 1, No. 2, 2010, pp. 45-55.

[22] J. Abrial, "Extending B without Changing It (for Developing Distributed Systems)," 1*st Conference on the B method*, *Putting into Practice Methods and Tools for Information System Design*, Institut de Recherche en Informatique de Nantes, 1996, pp. 169-190.

[23] T. Hoang and J. Abrial, "Reasoning about Liveness Properties in Event-B," *International Conference on Formal Engineering Methods*, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Vol. 6991, 2011, pp. 456-471.

