Modeling and Verification of Group Protocol Forward Secrecy using Event-B Refinement

Amjad Gawanmeh[1], Leila Jemni Ben Ayed[2], and Sofiène Tahar[1]

[1]Department of Electrical and Computer Engineering,

Concordia University, Montreal, Canada

Email: {amjad, tahar}@ece.concordia.ca

[2]Research Laboratory of Technologies of Information and Communication

Tunis, Tunisia

Email: leila.jemni@fsegt.rnu.tn

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 secrecy and 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.

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: *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 key, 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 [23] was introduced by extending B [1] 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. Notice that this advantage is important if we compare this approach to classical model checking where the transition system describing the model is refined and enriched.

In our previous paper [15] we have presented the proposed approach and some mapping relations giving a semantic of Group Key protocol model based on event-B allowing, hence, the verification of secrecy properties. Proposed mapping concerns only an abstract level of the initial model In this paper we prove the correctness of the proposed solution and we extend the proposed solution to verify distributed protocols secrecy by using event-B refinement. 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 (which is usually more complex). Moreover, in the refinement, it is not needed to re-prove these properties again while the model complexity increases. This advantage is important if we compare this approach to classical model checking where the transition system describing the model is refined and enriched.

We propose a solution to model group key protocols and to 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 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 paper extends the work in [**?**] to verify forward secrecy using event-B refinement.

In [17], we introduced a rank functions based inference system for verification of secrecy in group key protocols that is implemented in higher-order logic theorem proving. Implementing the inference system in higher order logic theorem proving required a lot of effort and time, in addition, verifying properties is achieved interactively with theorem proving tool because of the decidability problem on higher order logic. This paper complements our previous work, by providing an event-B based automatic invariant checking for a similar class of properties. This allows us to avoid user interaction with the theorem proving tool, and reduce time required to verify such property. In order to complement the previous methods, an event-B based automatic invariant checking is provided for a similar class of properties. This allows us to avoid user interaction with the theorem proving tool, and reduce the time required to verify such property. In this method, once a protocol is proven to be secure in the static case, it can be easily proven to be secure in the dynamic case by considering a protocol event, where, in this case, the number of members and messages in the protocol are abstracted under the execution of a single event.

We apply our approach on a group key protocol, the tree based Group Diffie-Hellman (TGDH) protocol [27] 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 tool Click'n'Prove [3] 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 overview preliminary definitions and notations we use in this paper. In Section IV, we present our methodology to verify secrecy and forward secrecy in event-B. In Section V, we apply our approach on a TGDH protocol. Finally, Section VI concludes the paper with future work hints.

2 Related work

The last years have seen the emergence of successful application 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 [14] and the inductive method of Paulson [24] 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 [10, 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], in the same issue, proposed an operational semantic for security protocols. The work provides a generic description of the interpretation of such security protocols and what it means of 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 [17].

Stouls and Potet [28] 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. They applied their method on a case study modeling a network monitor. A different approach to achieve a similar objective was proposed in [6], where the authors addressed the proof-based development of system models satisfying a security policy. They used OrBAC 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.* [7] 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 [8] combined CSP and B method refinement approach 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.

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 B method. In [**?**], 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 [**?**] 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.

3 Preliminaries

4 Group Key Protocols and Attacks

Cryptographic protocols are used for establishing secure communication in order to allow group members to exchange or establish keys to encrypt and authenticate messages within the group. In a group communications context, new members can join or leave the group, therefore the group requires that membership changes cause the group key to be refreshed. It is intuitive to generate a new key so that a new member is prevented from decoding messages exchanged before he/she joined the group. If a new key is distributed to the group when a new member joins, the new member cannot decrypt previous messages encrypted with the old key even if he/she has recorded any combination of earlier messages. In addition, changing the group key prevents a leaving group member from accessing the group communication if it can receive the messages. If the key is changed as soon as a member leaves, that member will not be able to decipher group messages encrypted with the new key. However, distributing the group key to valid members is a complex problem.

An example on group key protocols is the Diffie-Hellman key exchange [13]. It is based on public-key cryptography and provides a practical solution to the key distribution problem. It

enables two parties, which have never communicated before, to establish a shared secret key by exchanging messages over a public channel.

There are many extended protocols in the literature that are based on the DH agreement. For instance, Kim *et al.* [20] designed a static group key exchange protocol based on the extension of the basic DH protocol by Steiner *et al.* [27]. The protocol is called Tree-based Group Diffie-Hellman (TGDH) and it outputs group keys using the logical structure of a balanced binary tree. The Internet Key Exchange (IKE) [18], and its latest version IKEv1 [19] are also based on the basic Diffie-Hellman protocol.

Protocols are vulnerable to attacks, an example of such an attack exists in the Diffie-Hellman key establishment scheme. Despite the fact that the protocol is designed to work in the existence of passive adversary, it is vulnerable to "man in the middle" attack in the existence of active adversary. This Diffie-Hellman scheme has no way to ensure authentication. A man-in-the-middle could pretend to be user B and establish a shared key with user A, therefore, reading all messages that A believes to be sending to B. Similarly, the intruder could be doing the same with B, with certain control over the network. The intruder monitors a run of the protocol or part of it and at some time replays one or more of the messages. The intruder tricks an agent into revealing some information, possibly by inducing him to perform specific steps of the protocol.

4.1 Group Protocol Model

In this section we present our formal model and the notations we will use throughout this paper.

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, $\mathbb{E} : \mathbb{M} \rightarrow \mathbb{M}$. 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, $\forall m \in \mathbb{M} : m \in \mathbb{S} \Rightarrow m \notin \mathbb{K}_0$. 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 $\mathbb{K} \subset \mathbb{M}$.

$\mathbb{K}_f$: set of knowledge of a user who is not currently a member of the group, i.e. has no access to the current group's secret shares, and who was previously a member of the group. $\mathbb{K}_f \cap \mathbb{K} = \emptyset$. Also $\mathbb{K}_f \subset \mathbb{S}$, where $\mathbb{S}$ is the set of secret messages when this user was member of the group. This is used to express the knowledge a member accumulated during activity as a member of the group, and it will be used to model forward secrecy.

4.2 Secrecy

Only members of the group should have access to keys. The important issue here, is wether we want to allow users who just joined the group to have access to previously used keys, also wether we want to allow users who just left the group to have access to keys that will be generated henceafter. To ensure the secrecy of old and new keys, every protocol uses a mechanism for keys generation to guarantee that they cannot be calculated using the current group session information including the key itself.

In the following, we give the formal definition of group secrecy, forward secrecy and backward secrecy.

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 $\{K_0, K_1, \ldots, K_i\}$) 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. We formally model the forward secrecy requirement as follows: This can be expressed as $\phi_f \equiv (\mathbb{K} \cup \mathbb{K}_f) \cap \mathbb{S} = \emptyset$.

5 Overview of the Event B method

5.1 Event B model

Event-B is a variant of the B method introduced by Abrial to model reactive systems [1]. 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, which means that every possible change over state variables is defined by transitions; transitions correspond to events defined in the model [2]. A machine is composed of descriptive and operational specification:

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 =
SELECT
G
THEN
R

The consistency of an event-B model is established by proof obligations which guarantee that the initialization verify 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 its invariant. The feasibility statement is illustrated in Lemma 5.1 and the invariant preservation is given in Lemma 5.2 [23].

Lemma 5.1. $I(v) \ \wedge \ G(v) \ \wedge \ P(v) \Rightarrow \exists v'.R(v, \ v')$

Lemma 5.2. $I(v) \ \wedge \ G(v) \ \wedge \ P(v) \ \wedge \ R(v, v') \Rightarrow \ I(v')$

An event-B model M with invariant I is well-formed, dented by $M \models I$, only if M satisfies all proof obligations. 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. rel is formally defined as $rel \triangleq \{v \mapsto v' \mid I(v) \ \wedge \ G(v) \ \wedge \ R(v, \ v')\}$. 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 5.1 and 5.2 above [23].

Lemma 5.1 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 5.2 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.

The B syntax for generalized substitutions defines three predicates: a relation R and, 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. rel is formally defined as $rel \triangleq \{v \mapsto v' \mid I(v) \ \wedge \ G(v) \ \wedge \ R(v,\ v')\}$. 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 5.1 and 5.2 above [23].

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 [23]:

Lemma 5.3. $P(v) \Rightarrow \exists v'.R_I(v, v')$

Lemma 5.4. $P(v) \ \wedge \ R_I(v,\ v') \Rightarrow \ I(v')$

5.2 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 [2]. 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 $J(v, vc)$. 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:

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

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 $R_c(v_c, v'_c)$:

$I(v) \wedge J(v, vc) \wedge BA(v, v') \Rightarrow V(v'_c) < V(v_c)$.

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.

The event-B formal method supports an incremental development process, using refinement and B methodology is based on proofs of invariance and refinement. A strong point of B is support tools like B-Toolkit [3]. Most theoretical aspects of the method, such as the formulation of proof obligations, are done automatically by tools. Provers are also designed to run automatically and reference a large library of mathematical rules, provided with the system. The B method has been also used to verify semi formal specifications. Hence, a semi formal specification could be verified. Several works were elaborated in this context such as the translation from UML to B [21], statecharts to B abstract machine notation [26, 25] and UML Activity diagrams to event-B [5]. In this context, we propose in this paper a solution translating Group protocol models presented in [16] into event-B to verify required secrecy properties.

6 Event-B Semantics based Verification

It is a practical solution to verify a security property using model checking tools, when applicable. However, it is inconvenient because of two reasons: the state space explosion problem of model checking, and the limited expressiveness of proposition logic based-tools. Treating the problem at the first-order level requires applying a valid abstraction on the protocol in order to fit to the proving system. This abstraction should be based on a correct semantical link between the protocol model and the target model. In this method, we tackle this problem by using event-B as the target first-order model benefiting from the automation and the expressiveness of first order logic.

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 logics, the number of protocol participants that can considered is limited, it is also applicability on a single protocol event, finally, it cannot model backward secrecy and key independence. Figure 1 depicts the formal links in the proposed event-B approach.

The proposed verification methodology consists of a number of steps as shown in Figure 5. In

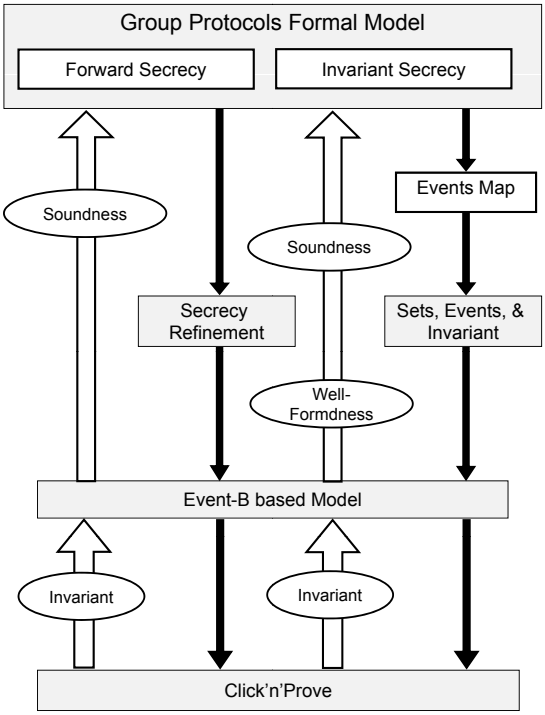

Figure 1: Event-B based Approach

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 mode. Next, the secrecy property ϕ is specified as an invariant of the resulting event-B model I, and 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 tool Click'n'Prove.

In Figure 4, 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 initializations, 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

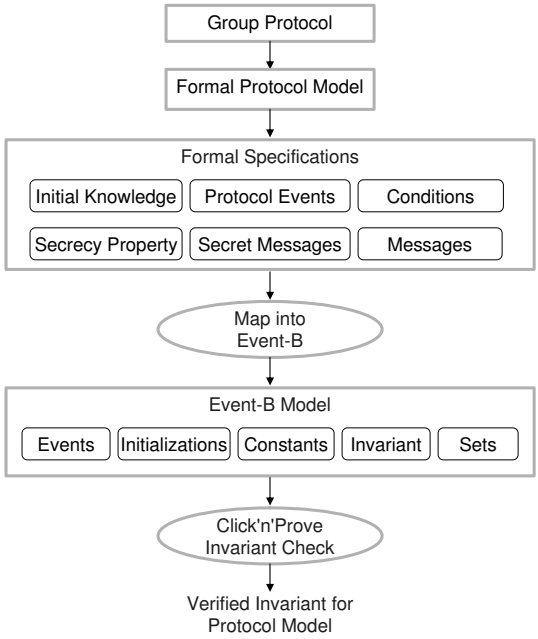

Figure 2: Overview of the Event-B Invariant based Method

describing the required secrecy properties.

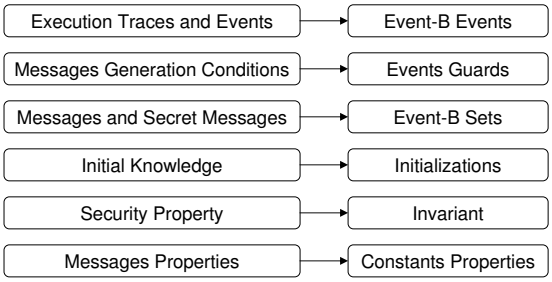

Figure 3: Mapping protocol primitives into event-B

7 Verification of Secrecy as Event-B Invariant

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 machine 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 a protocol Model $\mathbb{G}$ is equivalent to an event-B model M, with regards to the security property, if the property ϕ holds in the model $\mathbb{G}$, and the invariant I holds in the 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 regard to M, implies the safety property ϕ, with regard to $\mathbb{G}$.

Theorem 7.1. *Secrecy Soundness.*

A group protocol, G, satisfies its secrecy property, ϕ, if there is an equivalent event-B model, M, that satisfies an event-B invariant, I, that implies the property ϕ. More formally, for the definition $(G \triangleq M)$, let $(M \models I)$, and $(I \Rightarrow \phi)$ be correct lemmas, then,

$((G \triangleq M) \wedge (M \models I) \wedge (I \Rightarrow \phi) \Rightarrow (G \models \phi)$.

Proof. The proof is divided into two parts, we assume that lemmas are correct, then we proof the theorem based on that. In the next stage we proof each lemma separately.

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

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

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

□

□

We first define the equivalence relation between G and M, then we proof the lemma $(I \Rightarrow \phi)$. The lemma $(M \models I)$ is assumed to be correct in the event-B tool.

Definition 7.1. *A group protocol model G, is equivalent to an event-B model, M under certain conditions and semantically correct map from G to M. $G \triangleq M$ 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 variables 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 7.1.

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

$Nameevent =$

ANY m$WHERE$

$condition$

$THEN$

R((m, m')

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

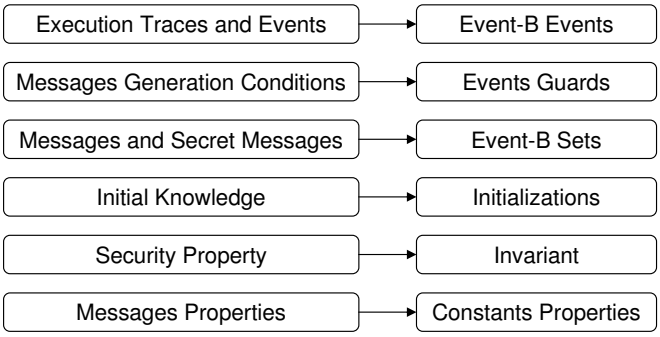

Figure 4: Mapping protocol primitives into event-B

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 4, 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 7.1. $(I \Rightarrow \phi) = ((I_{init} \Rightarrow \phi_{init}) \ \wedge \ (I_E \phi \Rightarrow \phi_E))$

Proof. ***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 5.1 and 5.2 for the protocol model $\mathbb{G}$ such that Lemma 7.1 holds.

The definition of the group key protocol must satisfy the initial soundness conditions: $\mathbb{K}_0 \cap \mathbb{S} = \emptyset$ and $\forall e_i \in \mathbb{E}_i . m' := e_i(m) \Rightarrow m' \notin \mathbb{S}$, 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 5.3 and 5.4. Then we define P, R, the predicate guards G, and the invariant I for the model $\mathbb{G}$ that will satisfy Lemmas 5.1 and 5.2.

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

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

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 5.3 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 $\mathbb{K}_0 \cap \mathbb{S} = \emptyset$ 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 is corresponds to Lemma 5.3 considering that $R_I = e_i$.

Case 2 $(I_E \Rightarrow \phi_E)$

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

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 5.1 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 5.2.

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 event-B tool.

This completes the proof of Theorem 7.1. 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.

8 Verification of Forward Secrecy in Event-B Refinement

In previous section, a well-formed link between the semantics of the group protocol and event-B models is established. Then we showed that when the invariant I holds for event-B machine 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.

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 before the relation R_c. In addition, we use $I(v_c)$ to represent the invariant in the refined model, which corresponds 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.

The proposed verification methodology consists of a number of steps as shown in Figure 5. 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 mode. Next, the secrecy property ϕ is specified as an invariant of the resulting event-B model I, and 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 tool Click'n'Prove.

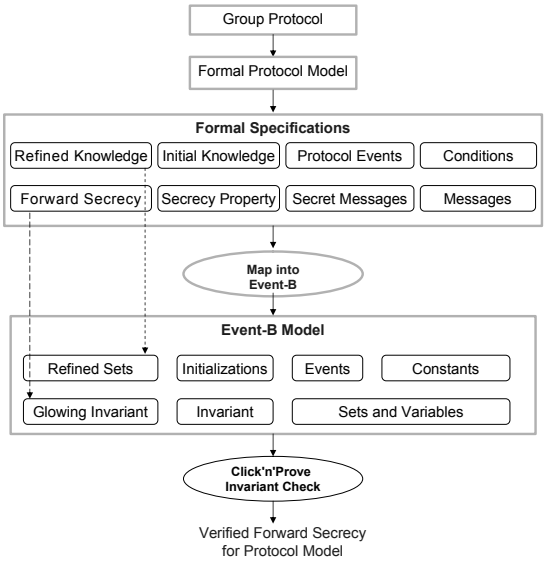

Figure 5: Refined Event-B Method for Forward Secrecy

In event-B, refinement can be done with events or variable. 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 6 below illustrates the link between the abstract and refined model to achieve a model for forward secrecy in event-B.

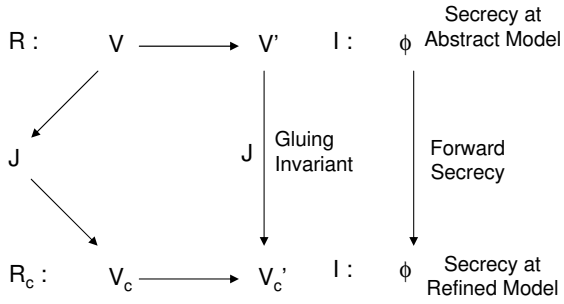

Figure 6: Relationship between Abstract and Refined Models

Theorem 8.1. *Forward Secrecy Soundness.*

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, let $(G_c \triangleq M_c)$, $(I_c \Rightarrow \phi_f)$, $(M \models I)$, and $(M_c \models I_c)$ be correct lemmas, then

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

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

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

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

□

□

Event-B invariant checking cannot reason about backward secrecy because invariant 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.

9 Application: Secrecy in 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 con-

cretely applied on a real protocol. The intended secrecy property, along with its conditions, are efficiently defined and checked as event-B invariant.

Tree-based Group Diffie-Hellman protocol (TGDH) is intended for secure key generation. Figure 7 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 [20]. All TGDH protocols have the following features:

- Each group member contributes an equal share to the group key, and the key is a function of all current group members shares.
- The share of each member is secret and is never revealed.
- When a new member joins the group, one of the old members changes its share, and new members' shares are factored into the group key.
- When an existing member leaves the group, its share is removed from the new group key, and at least one remaining member changes its key share.
- All protocol messages are signed, time-stamped, sequence-numbered, and type-identified by the sender.

After every membership change, all remaining members independently update the key tree structure and recompute identical key trees after any membership event. 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. Figure 7 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 [20].

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 $\{n_3,\ g^{n_2 n_3},\ g^{n_1 g^{n_2 n_3}}\}$. The set of blind keys ordered as they appear up to the root is $\{\ g^{n_2},\ g^{n_1},\ g^{g^{n_6 g^{n_4 n_5}}}\ \}$. 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 [20] 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

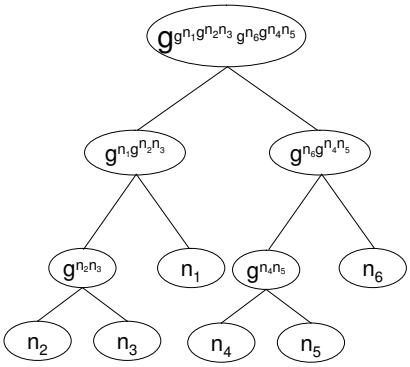

Figure 7: Tree-based GDH Protocol Binary Tree Structure

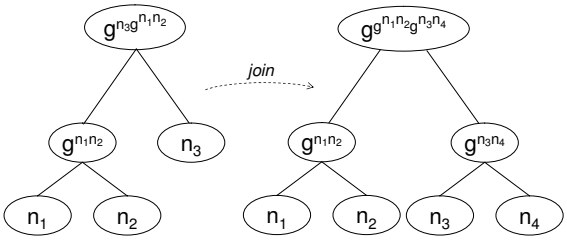

Figure 8: Join Event in the TGDH Protocol

their own private shares, can calculate the group key at root. On the other hand, an adversary, who has knowledge to 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 8 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 join event takes place:

$$\mathbb{M} = \{n_1,\ n_2,\ n_3,\ g^{n_1},\ g^{n_2},\ g^{n_3},\ g^{n_1n_2},\ g^{g^{n_1n_2}},\ g^{n_3g^{n_1n_2}}\}$$
$$\mathbb{S} = \{n_1,\ n_2,\ n_3,\ g^{n_1n_2},\ g^{n_3g^{n_1n_2}}\}$$
$$\mathbb{K}_0 = \{n_i,\ g^{n_i}\}$$
$$\mathbb{K} = \{n_i,\ g^{n_i},\ g^{n_1},\ g^{n_2},\ g^{n_3}, g^{g^{n_1n_2}}\}$$
$$GroupKey = g^{n_3g^{n_1n_2}}$$

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.

$$\mathbb{M} = \{n_1,\ n_2,\ n_3,\ n_4, g^{n_1},\ g^{n_2},\ g^{n_3},\ g^{n_4},\ g^{n_1n_2},\ g^{n_3n_4},$$
$$g^{g^{n_1n_2}},\ g^{g^{n_3n_4}},\ g^{g^{n_1n_2}g^{n_3n_4}}\}$$

SYSTEM *TGDHProtocol*
SETS
$BLINDKEYS$ /* set of Blind keys */
MS; /* set of messages */
K /* Intruder's set of knowledge*/
S /* Set of secret messages */
VARIABLES
$intruderKey, msgBefore, msgAfter, bk, Gkey$
INVARIANT
/* malicious participant cannot evaluate to GK */
$K \cap S = \emptyset \wedge GKey \notin K \wedge K \subset MS \wedge S \subset MS \ldots$
INITIALISATION
$BLINDKEYS := \{g^{N_1}, g^{N_2}, g^{N_3}, g^{g^{N_1 N_2}}, \ldots\}$;
$MS := N_1, g^{N_1}, N_2, g^{N_2}, \ldots\}$;
$K := g^{N_1}, g^{N_2}, \ldots$
$S := \{N_1, N_2, g^{n_1 n_2}, g^{n_1 n_2 g^{n_3}}, \ldots\}$
EVENTS $eventB\_tgdh \triangleq \ \ldots$ /*for a protocol event*/
END

$$\mathbb{S} = \{n_1,\ n_2,\ n_3,\ n_4,\ g^{n_1 n_2},\ g^{n_3 n_4},\ g^{g^{n_1 n_2} g^{n_3 n_4}}\}$$
$$\mathbb{K} = \{n_i, g^{n_i},\ g^{n_1},\ g^{n_2},\ g^{n_3},\ g^{n_4},\ g^{g^{n_1 n_2}},\ g^{g^{n_3 n_4}}\}$$
$$GroupKey = g^{g^{n_1 n_2} g^{n_3 n_4}}$$
$$\phi = GroupKey \notin \mathbb{K} \wedge \mathbb{K} \cup \mathbb{S} = \emptyset$$

9.1 Secrecy Model in Event-B Invariant

The event-B model for the protocol components is described below. We first define the event-B sets for blind keys ($BLINDKEYS$), the general set of messages (MS), the intruder's set of messages (K), and the set of secret keys (S). Then we define a number of variables over the above sets of messages. We describe the current status of the group by initializations where each of the above sets is concretely defined. 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: $K \cap S = \emptyset$. Some of the protocol characterizes can also be encoded within this invariant, such as $K \subset MS \wedge S \subset M$. We also define an event to represent the protocol action (join/leave).

The TGDH protocol components are defined within the event-B model, where the sets of messages MS, K, and S, are directly defined from the above sets $\mathbb{M}$, $\mathbb{K}$, and $\mathbb{S}$, respectively. 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 insure the consistency of the map, $(K \cap S = \emptyset) \wedge GKey \notin K \wedge K \subset MS \wedge S \subset MS$. These components are defined within the Click'n'Prove tool as follows:

To be consistent with the group structure, we also defined the set of blind keys in event-B as follows:

eventB_tgdh $\triangleq$ /* *for any message m* */
ANY $msgBefore, msgAfter, bk, \ldots$ WHERE
$msgBefore \in K \wedge msgAfter = g^{g^{N_3 N_4}} \wedge \ldots$
THEN
/* *update intruder's set of messages after executing the event* */
$GKey := g^{g^{N_1 N_2} g^{N_3 N_4}}$ /* calculate group key */
$K := K \cap msgAfter$
/* *update the set of secret messages* */
$S := S \cup \{Gkey, N_4, g^{N_3 N_4}, \ldots\}$
END

$$BLINDKEYS = \{g^{N_1}, g^{N_2}, g^{N_3}, g^{g^{N_1 N_2}}, \ldots\};$$

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 within the Click'n'Prove tool as follows:

After this event is executed in the tool, new blind keys will be generated and added to that set. The new secret group key is calculated based on the new contribution of the joined member, n_4.

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

In the verification with the Click'n'Prove tool, 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 proven automatically, and then the initial model of the group key protocol is validated. The event-B invariant, I, defined in the Click'n'Prove model 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, $G \models \phi$.

The proposed solution allows us to verify the required property, however, one limitation of our approach is related to the fact that event-B operations are defined only over finite sets. Therefore, a bounded number of participants and protocol events should be applied. 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.

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.

9.2 Forward Secrecy Model in 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 9 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 9). This represents the refined model.

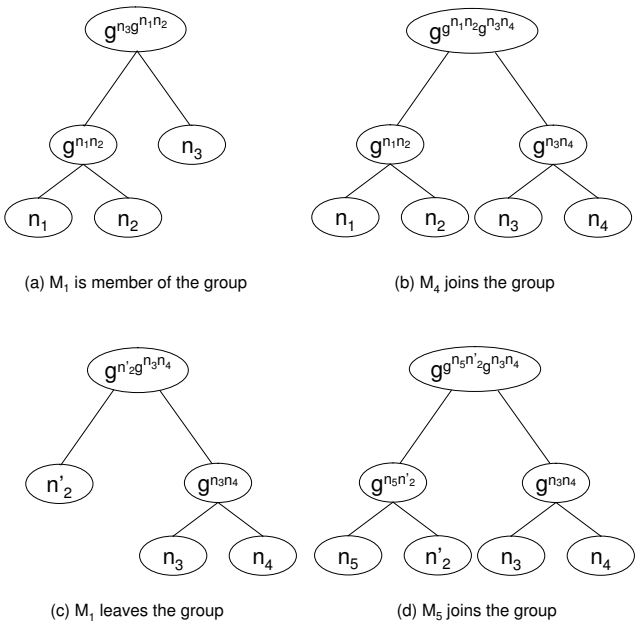

Figure 9: Forward Secrecy in the TGDH Protocol

In Figure 9 (a), the current set of messages, secret set of messages, and the current group key are defined, respectively, as follows:

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

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

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

Then, after member M_4 joins the group, as shown in Figure 9 (b), the above variables become:

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

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

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

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 9 (c). The above variables become:

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

$g^{g^{n_3 n_4}}, g^{n_3 g^{n_1 n_2}},\ g^{g^{n_1 n_2} g^{n_3 n_4}},\ g^{n_2' g^{n_3 n_4}}\}$

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

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

Finally, Figure 9 (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:

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

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

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

We define the set $\mathbb{K}$, as viewed by a member monitoring the group from outside, before M_5 joins the group:

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

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

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

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 $\mathbb{K}$ is refined with the knowledge gained by user M_1 while member in the group, and is defined as:

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

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

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

10 Conclusion

In this paper we provide automated invariant checking for group key secrecy and forward secrecy properties. 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 group protocols model and event-B was established. The result was combining 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 the TGDH protocol [20] provided an informal, non-intuitive and simple proof for secrecy property.

As a limitation of the approach, only invariant properties can be modeled and verified. This is due to the target model and verification tool, namely, event-B and Click'n'Prove tool. However, invariant checking is adequate to model many security properties. In addition, we were able to conduct invariant checking for a limited number of tree levels due to the lack of exponent operator in the prover.

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. However, in order to achieve this, major modifications of the B approach are required, mainly, tool support.

References

[1] J. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambbridge University Press, 1996.

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

[3] J. Abrial and D. Cansell. Click'n'Prove: Interactive Proofs within Set Theory. In *Theorem Proving in Higher Order Logics*, volume 2758 of *LNCS*, pages 1–24. Springer-Verlag, 2003.

[4] P. Behm, P. Desforges, and J. Meynadier. MÉTÉOR: An Industrial Success in Formal Development. In *Recent Advances in the Development and Use of the B Method*, volume 1393 of *Lecture Notes in Computer Science*, pages 26–26. Springer-Verlag, 1998.

[5] A. Ben-Younes and L. Jemni Ben Ayed. Using UML Activity Diagrams and Event-B for Distributed and Parallel Applications. In *Computer Software and Applications Conference*, pages 163–170. IEEE Computer Society Press, 2007.

[6] N. Benaissa, D. Cansell, and D. Méry. Integration of Security Policy into System Modeling. In *Formal Specification and Development in B*, volume 4355 of *LNCS*, pages 232–247. Springer-Verlag, January 2007.

[7] D. Bert, M. Potet, and N. Stouls. GeneSyst: A Tool to Reason about Behavioral Aspects of B Event Specifications. Application to Security Properties. In *Formal Specification and Development in Z and B*, volume 3455 of *LNCS*, pages 299–318. Springer-Verlag, April 2005.

[8] M. Butler. On the Use of Data Refinement in the Development of Secure Communications Systems. *Formal Aspects of Computing*, 14(1):2–34, 2002.

[9] ClearSy. B4free, http://www.b4free.com/, 2004.

[10] F. Crazzolara. *Language, Semantics, and Methods for Security Protocols*. PhD thesis, BRICS, Denmark, May 2003.

[11] F. Crazzolara and G. Winskel. Events in Security Protocols. In *ACM Conference on Computer and Communications Security*, pages 96–105. ACM Press, 2001.

[12] C. Cremers and S. Mauw. Operational Semantics of Security Protocols. In *Scenarios: Models, Transformations and Tools*, volume 3466 of *LNCS*, pages 66–89. Springer-Verlag, 2005.

[13] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[14] F. Fábrega. Strand Spaces: Proving Security Protocols Correct. *IOS Journal of Computer Security*, 7(2-3):191–230, 1999.

[15] A. Gawanmeh, L. Jemni Ben Ayed, and S. Tahar. Event-B based Invariant Checking of Secrecy in Group Key Protocols. Technical report, Concordia University, April 2008.

[16] A. Gawanmeh, A. Bouhoula, and S. Tahar. Rank Functions based Inference System for Group Key Management Protocols Verification. Technical report, Concordia University, October 2007.

[17] A. Gawanmeh, A. Bouhoula, and S. Tahar. Rank Functions based Inference System for Group Key Management Protocols Verification. *International Journal of Network Security*, 8(2):207–218, 2009.

[18] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, Internet Engineering Task Force. Available at http://www.ietf.org/rfc/rfc2409, November 1998.

[19] P. Hoffman. The Internet Key Exchange version 1 (IKEv1). RFC 4109, Internet Engineering Task Force. Available at http://ietf.org/rfc/rfc4109, November 2005.

[20] Y. Kim, A. Perrig, and G. Tsudik. Tree-based Group Key Agreement. *ACM Transactions on Information and Systems Security*, 7(1):60–96, 2004.

[21] H. Ledang. and J. Souquiere. Modeling Class Operations in B : a Case Study on the Pump Component. Technical Report A01-R-011, Laboratoire Lorrain de Recherche en Informatique et ses Applications, France, November 2001.

[22] C. Meadows and P. Syverson. Formalizing GDOI Group Key Management Requirements in NPATRL. In *ACM Conference on Computer and Communications Security*, pages 235–244. ACM Press, November 2001.

[23] 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, UK, 2005.

[24] L. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *IOS Journal of Computer Security*, 6(1-2):85–128, 1998.

[25] E. Sekerinski. Graphical Design of Reactive Systems. In *Recent Advances in the Development and Use of the B Method*, volume 1393 of *Lecture Notes in Computer Science*, pages 182–197. Springer-Verlag, 1998.

[26] E. Sekerinski and R. Zurob. Translating Statechart to B. In *Integrated Formal Methods*, volume 2335 of *Lecture Notes in Computer Science*, pages 128–144. Springer-Verlag, 2002.

[27] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.

[28] N. Stouls and M. Potet. Security Policy Enforcement Through Refinement Process. In *Formal Specification and Development in B*, volume 4355 of *LNCS*, pages 216–231. Springer-Verlag, 2007.