Modeling and Verification of a Tree-based GDH protocol in Event B

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

The correctness of group key protocols in communication systems remains a great challenge because of dynamic characteristics of group key construction as we deal with an open number of group members. In this paper, we propose a solution to model group key protocols and to verify their required properties, in particular secrecy property, using the event-B method. Event-B deals with tools allowing invariant checking, and can be used to verify group key secrecy property. We define a well-formed formal link between the group protocol model and the event-B counterpart model. Our approach is applied on a tree-based group Diffie-Hellman protocol that dynamically outputs group keys using the logical structure of a balanced binary tree.

1 Introduction and Motivation

Security protocols are used to establish secure channels between communicating systems. Great care needs to be taken in developing and implementing robust protocols. 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 protocols. Therefore they require a more precise definition before we look at how to verify them. An example of such properties is secrecy, which deals with the fact that secret data should remain secret and not compromised. However, for group key protocols, this 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. Therefore, systems designed for two-party protocols may not be able to model a group key protocol, or its intended security properties because such systems require an abstraction to a group of fixed size to be made before the automated analysis takes place. This can eliminate chances of finding attacks on these protocols.

In [14] 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 the 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 the time required to verify such property.

Event-B [17] 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. A strong point of event-B is the availability of tools that support automatic invariant checking such as Click'n'Prove [3], B4free [8], and RODIN [17]. Most theoretical aspects of the method, such as the formulation of proof obligations, are done automatically by the tool. Provers are also designed to run automatically and reference a large library of mathematical rules, provided with the system.

The B method has been used to verify semi-formal specifications. Several works were elaborated in this context such as the translation from UML to B [16], state-charts to B abstract machine notation [19] and UML activity diagrams to event-B [22]. In this context, we propose a solution mapping group protocol models presented in [14] in event-B to verify required secrecy properties.

In order to model a group key protocol in event-B first order logic, the semantics of the event-B language should be formally related to the protocol model. In this paper, we define well-formed conditions to guarantee that the event-B invariant is equivalent to the security property in the group key protocol model. These conditions are particular to the group key protocol model, and are essential to establish the equivalent event-B model. We show how an event-B model can be structured from group key protocols model and then used to give a formal semantics to protocols which support proofs of their correctness.

The Diffie-Hellman basic protocol [12] has been used extensively to design several group key protocols. Kim *et al.* [15] designed a static group key exchange protocol (tree based Group Diffie-Hellman (TGDH)) that outputs group keys using the logical structure of a balanced binary tree. The TGDH protocol design is based on an extension of the basic DH protocol by [20] *et al.*. The designers of the protocol provide an informal and non-intuitive simple proof for the secrecy property in their work. We apply our approach on the TGDH protocol and use invariant checking to formally verify secrecy property that ensures the correctness of key construction. We provide an automatic proof based on a sound semantical link between group key protocols and event-B models.

In the proposed approach, we model protocol events and traces as events and messages as sets in event-B. Secrecy property is defined as an event-B invariant. We then use the event-B first-order logic prover tool, Click'n'Prove, to perform invariant checking under the assumption that basic DH key is correct. The dynamic case was 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 contributions of this paper include the formal link between event-B semantic and security protocols, the use of event-B first-order prover to verify secrecy property as an event-B invariant, which is not a straightforward task, since we have to define a correct semantical link between the two models. Finally, applying the method on a group key protocol with join/ leave events, in order to verify secrecy property using first-order automatic theorem proving in event-B. To achieve this, certain distributive features in the protocol were abstracted away, focusing on key construction problems.

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 Section IV, we present our methodology and define a sound formal link between the protocol model and event-B semantics. 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 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 [13] and the inductive method of Paulson [18] 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 [10].

Events-based verification of security protocols was used by Crazzolara [9, 10] 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 [11] proposed an operational semantics for security protocols. The work provides a generic description of the interpretation of such security protocols and what it means for a pro-

tocol 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 [14].

Stouls and Potet [21] 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 [5], 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.* [6] presented a tool to build symbolic labeled transition systems from B specifications. The resulting symbolic transition system represents all 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 [7] 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.

Compared to the above, we address, in this paper, security property for group oriented protocols, which has special features that cannot be modeled in any of these approaches, such as the concept of group secrecy and dynamic group events. In addition, we consider events that are specific for group key protocols, something that was not treated by the B method based work of Butler [7]. This paper tackles the verification problem by using event-B as the target first-order logic model. In this context, we propose a solution translating group protocol models presented in [14] into event-B to verify required secrecy properties.

3 Preliminaries

In this section we follow the protocol model and its notations used throughout this paper and the formal semantics of event-B.

3.1 Group Key Protocols

We follow the formal definition for group key protocols presented in [14]. 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 $\mathbb{K} \subset \mathbb{M}$.

Finally, we define a safety property ϕ for a given 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 $\mathbb{G} \models \phi$.

3.2 Event-B

Event-B [2] is a variant of the B method introduced by Abrial [1] 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 (or systems) [1]. 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 =
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 3.1 and the invariant preservation is given in Lemma 3.2 [17].

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

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 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 3.1 and 3.2 above [17].

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 [17]:

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

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

Most theoretical aspects of the event-B formal method, such as the formulation of proof obligations, are done automatically by tools such as Click'n'Prove and B-Toolkit [3]. Provers are also designed to run automatically and reference a large library of mathematical rules, provided with the system. This makes B well adapted for large scale and wide range of systems [4].

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 logic 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. We tackle this problem by using event-B as the target first-order logic model benefiting from the automation and the expressiveness of the logic and the availability of supporting tools.

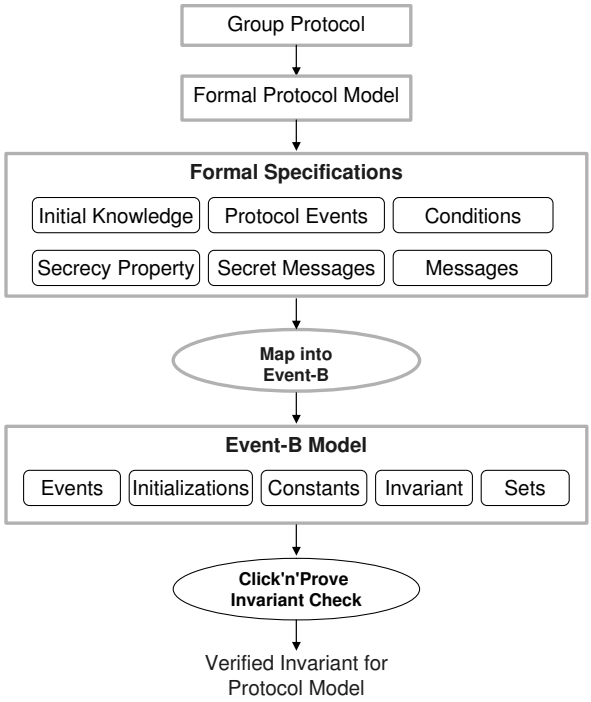

Figure 1: Verification Methodology

The proposed verification methodology consists of a number of steps as shown in Figure 1. In the first step, the group key protocol is specified formally using the model proposed in [14] 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 presented in Figure 2. From this mapping we 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. 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 2, 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 describing the required secrecy properties.

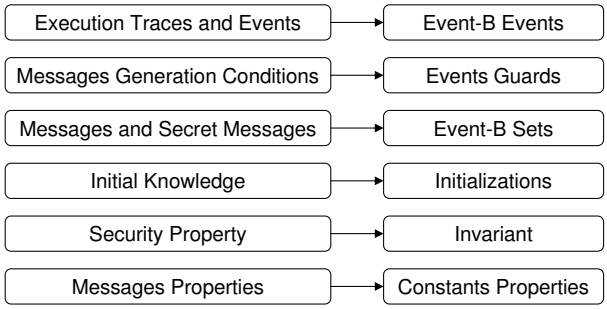

Figure 2: Mapping protocol primitives into event-B

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 regards to M, implies the safety property ϕ, with regard to $\mathbb{G}$.

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. From Figure 2, there is a map from public messages and secret messages to event-B sets and a map from 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.

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

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 3.5 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 $\mathbb{K}_0$.

In following, we 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 the relation 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 $(I_{init} \Rightarrow \phi_{init})$

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

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 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 $\mathbb{K}_0 \cap \mathbb{S} = \emptyset$ 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 corresponds to Lemma 3.3 considering that $R_i = e_i$.

Case 2 $(I_E \Rightarrow \phi_E)$

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

- $R = (e \in \mathbb{E}) \wedge (\exists m \in \mathbb{K}, m' \in \mathbb{M} \cdot m' = e(m))$

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 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 the 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 particular event-B tool.

4 Application: Secrecy in TGDH Protocol

In this Section, we apply the approach proposed in this paper 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.

We first introduce the basic Tree-based Group Diffie-Hellman protocol (TGDH) as it is designed in [15]. 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 [15].

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

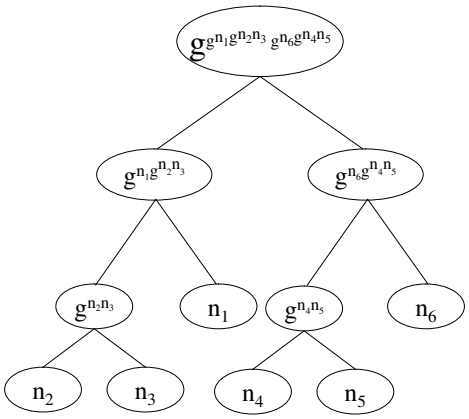

Figure 3: Tree-based GDH protocol binary tree structure

computed from any members secret share and all blind keys on the co-path to the root, these 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 3 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 [15].

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 the keys 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*, 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* 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. The authors of [15] provided an informal proof that their protocol satisfies these security property. In this work, we

provide a formal proof for group 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 has knowledge to all blind sub-keys cannot find a full path to calculate the root key.

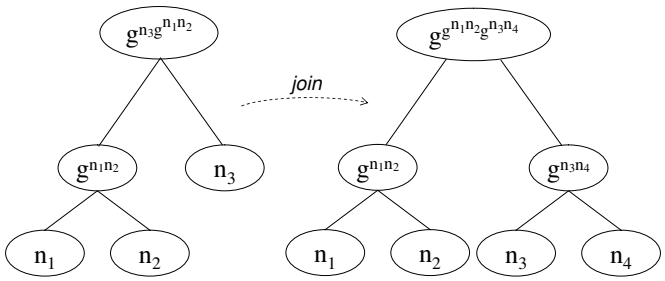

Figure 4: Join event in TGDH protocol

We illustrate our method on a group protocol composed of three members, then we apply a join event for a forth member. Figure 4 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 configuration, we show all group protocol components, including secrecy property, and the equivalent event-B model including the invariant, before the join event takes place:

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

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

$$\mathbb{K}_0 = \{n_i,\ g^{n_i}\}$$

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

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

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.

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

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

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

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

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

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

Figure 5 shows event-B model for the protocol components. 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).

In the event-B model, 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, $(K \cap S = \emptyset) \wedge GKey \notin K \wedge K \subset MS \wedge S \subset MS$. To be consistent with the group structure, we also defined the set of blind keys in event-B as follow:

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

Figure 6 represents an event-B definition that captures the behavioral semantics of a basic message update performed by the protocol. This event will result in updating the intruder's set of knowledge. 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 joining member, n_4.

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

To translate our initial protocol, we first consider the static case of key construction under

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 */
$K \cap S = \emptyset \wedge GKey \notin K \wedge K \subset MS \wedge S \subset MS$
...
INITIALISATION
$BLINDKEYS := \{g^{N_1}, g^{N_2}, g^{N_3}, g^{g^{N_1 N_2}}, \dots\}$;
$MS := N_1, g^{N_1}, N_2, g^{N_2}, \dots\}$;
$K := g^{N_1}, g^{N_2}, \dots$
$S := \{N_1, N_2, g^{n_1 n_2}, g^{n_1 n_2 g^{n_3}}, \dots\}$
EVENTS $eventB\_tgdh \triangleq \dots$ /*for a protocol event*/
END

Figure 5: Event-B Model of the Protocol Components.

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 Figure 5, 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

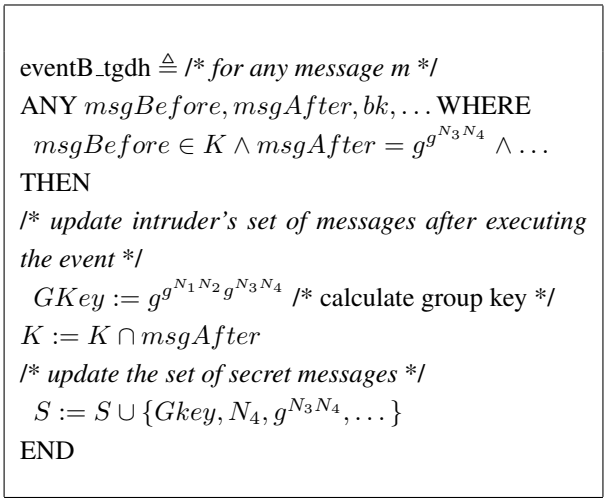

Figure 6: Event-B model capturing the semantics of the protocol join event

or authentication property.

In addition, to modeling the relationship between the secret keys and blind keys an exponent operator is needed, therefore, the set of possible blinded keys is directly related to the number of participants represented by the tree level, i.e, the model size in event-B is directly related to the number of participants. Hence, a huge set of keys should be modeled, where the automatic generation of these keys is infeasible because no exponent operator is supported by event-B. Therefore, applying invariant checking becomes limited by the issue of generating this set manually. 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 Conclusion

The correctness of group key protocols in communication systems remains a great challenge because of the sensitivity of the services provided. In this paper, we illustrated the need for a verification methodology for secrecy property in group key protocols. While many approaches in the literature target cryptographic properties for two parties protocols, the verification problem for group key protocols is more challenging because properties for these protocols are not trivial extensions of the two-parties models. For example, the fact that a group member computes a bad key can remain undiscovered by the group, specially for a large group.

In this paper, we provide 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. The contributions of this paper include defining a well-formed connection between event-B invariant and the group key protocol model including its secrecy property. These conditions guarantee that the invariant verified in event-B is equivalent to the secrecy property. In addition, we provide a mechanized approach using first-order logic proving system in the context of group key protocols verification. We applied this approach on a group key protocol, the tree based Group Diffie-Hellman protocol [15] and provided invariant checking for secrecy under the static and the dynamic case by applying a single event (join/leave). We also considered the limitation of tree size that can fit into first-order logic model. We found that, under certain assumptions, only group members can generate the correct key. The results we achieved are very promising and we believe that our method can be applied efficiently on protocols of similar complexity level. We provided a formal link from the group key protocol model to event-B in order to use specific features in event-B to describe protocol actions and verify the required secrecy property. The authors of the protocol we use (TGDH) provided an informal, non-intuitive and simple proof for secrecy property in their work. In this paper we provided a formal, tool supported, and automatic proof based on a sound method.

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 [3]. However, invariant checking is adequate to model properties that describe secrecy. In addition, we were able to conduct invariant checking for a limited number of tree levels due to the lack of an exponent operator in the prover. As future work, we intend to extend the event-B based model to be able to check for parameterized number of participants. In addition, it will be more interesting to consider more dynamic properties: forward and backward secrecy, and the most interesting case is key independence. However, in order to achieve this, major modifications of the approach are required using refinement in event-B.

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, France, 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 *Lecture Notes in Computer Science*, 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] N. Benaissa, D. Cansell, and D. Méry. Integration of Security Policy into System Modeling. In *Formal Specification and Development in B*, volume 4355 of *Lecture Notes in Computer Science*, pages 232–247. Springer-Verlag, 2007.

[6] 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 *Lecture Notes in Computer Science*, pages 299–318. Springer-Verlag, 2005.

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

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

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

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

[11] C. Cremers and S. Mauw. Operational Semantics of Security Protocols. In *Scenarios: Models, Transformations and Tools*, volume 3466 of *Lecture Notes in Computer Science*, pages 66–89. Springer-Verlag, 2005.

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

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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

[22] 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.