Cryptographically Sound Security Proof for On-Demand Source Routing Protocol EndairA

István Vajda

Abstract: We present the first cryptographically sound security proof of a routing protocol for mobile ad-hoc networks. More precisely, we show that the route discovery protocol does not output a non-existing path under arbitrary active attacks, where on a non-existing path there exists at least one pair of neighboring nodes without communication connection during the run of the route discovery protocol. The proof relies on the Dolev-Yao-style model of Backes, Pfitzmann and Waidner, which allows for mapping results obtained symbolically within this model to cryptographically sound proofs if certain assumptions are met.

1. Introduction

Routing is a fundamental networking function, which makes it an ideal starting point for attacks aiming at disabling the operation of an ad hoc network. Therefore, securing routing is of paramount importance. Several "secure" routing protocols have been published (see [17] for an overview). Unfortunately, the analysis of ad hoc routing protocol security features is typically informal. The flaws in routing protocols can be very subtle, therefore it is very difficult to discover them by informal reasoning, as was the case also for the Ariadne protocol proposed in [18]. In paper [15] we have shown an attack against this thought-to-be-secure protocol: it was a routing state pollution attack which caused an honest node to accept such routes that do not exist in the underlying network topology. Inspired by Ariadne, we have proposed a new source routing protocol endairA [3] (which is the reverse of Ariadne), because instead of signing the route request, intermediate nodes sign the route reply.

Considering cryptographic protocols in general, especially, owing to the distributed-system aspect of multiple interleaved protocol runs, to make proofs is awkward for humans. Therefore, automation of proofs has been an obvious challenge since the publication of first cryptographic protocols. One way to produce such proofs is the cryptographic approach, the alternative is the formal-methods approach. In cryptographic approach the security is proved by complexity theoretical reduction. In case of traditional formal-methods approach the cryptographic details (error probabilities, computational restrictions) are abstracted away, like in the so called Dolev-Yao model the proof is conducted in a symbolic model applying formal verification methods (model checkers, theorem provers).

A proof for a cryptographic protocol (e.g. secure routing protocol) in cryptographic model is an "all-in-one" procedure, it has to handle the complex behavior of the adversary by assuring that all possible attack attempts are "enlisted" and checked,

Technical University of Budapest E-mail: vajda@hit.bme.hu

which latter means typically reduction to hard problem involving probabilistic and complexity theoretic considerations. The proof is conducted by hand and potentially error-prone. In general, there is no insurance, that we can take into account all the potential attacks. Furthermore, the black box models for cryptographic elements are typically ad-hoc, in the sense, that there is no secure composition theorem in support.

A more sound way of proof, if we can separate the formal and the cryptographic part by producing a completely symbolic model and performing the proof in this model but having the assurance that if we finally replace the symbolic cryptographic operations with real ones the protocol which is proved to be secure in symbolic model remains secure also in the real cryptographic model. This is what is done in the BPW approach. It provides the needed composable cryptographic library [8]. To model the active attacks this library works in reactive setting, it is a stateful system with possible operations (commands) for honest participants and adversaries, depending on prior cryptographic operations and network actions. The security proof has to be given only for the symbolic system, and it remains "automatically" sound in the real system, assuming the use of provably secure cryptographic primitives with minor additions. The definition of the library is careful: e.g. when in a security proof digital signatures are substituted by a black box model it is typical to assume the security against essential forgery under chosen-message attacks, however such stereotypical approach may lead to severe failures in reactive systems (see [10] for reactively secure digital signatures).

Protocols are rewritten into algorithms using the commands of the symbolic system of the library, which provides a much more granular description than the usual definitions of protocols, in fact, it is a runnable system. At any time step the possible next actions of the participants can precisely be followed, which is especially important in the case of adversary, who is restricted to a set of adversarial commands. Even in the case when the proof is made by hand efficient techniques can support correct security analysis, e.g. the technique of invariants.

The BPW model was applied by its authors to provide the first cryptographically sound security proof for the Needham-Schroeder-Lowe public key authentication protocol [11] and the Kerberos [13]. In this paper we give the first cryptographically sound security proof for a routing protocol, the endairA, which also provides a new look at endairA.

The structure of the paper is the following. Section 2. gives a short summary of related works. In Section 3 we recapitulate protocol endairA and also present the symbolic setting of the BPW approach. Section 4. gives the definition of secure route discovery as well as the security claim about endairA. In Section 5. we present the proof. In Section 6 we show a related example for modular design and analysis based on the general composition theorem [11]. Conclusions are drawn in Section 7. The protocol in BPW symbolic model is given Appendix A.

2. Related work

Overwhelming majority of security analysis of routing protocols is informal and potentially error-prone, therefore it happened many times that after thorough inspection of the protocol researchers of the community came up with a vital attack,

e.g. Ariadne [17], SAODV [1], secure TinyOS beaconing [5]. The usual reason, as it was mentioned above, is that the proof is based on an informally obtained and incomplete "list" of potential attacks and it applies only to these attacks. Typical situation is when a new general type of attack is published and triggers a wave of papers, e.g. Sybil attack [16], sinkhole attack [18], route diversion attacks [18]. Many DoS attacks have been recognized in the literature [18], [22].

Pure formal approaches have been applied for automatic verification of small topologies of N nodes (N ~ 3-10) using e.g. SPIN model checker [17], typically to check simple liveness properties (see paper [21]).

In paper [15] we proposed a proof framework, which was a cryptographic approach for proving security of routing protocols in ad hoc networks. This framework was based on the simulatability approach, known in cryptography. In this framework we gave a proof for the security of endairA protocol [3]. This result was criticized in [6] by showing an attack against endairA: if the attacker is able to use wormholes, which is a type of proprietary communication channel connecting compromised nodes, the source of route discovery procedure may accept routes that do not exist in the underlying network topology. However, the adversary model [3] explicitly excluded both the Sybil and the wormhole attacks. These type attacks have also been included in the adversary model given by Ács in [5], where also the definition of secure route discovery is adjusted. The proof in [5] uses the same proof framework as the original proof.

However, as it was mentioned above, such an all-in-one proof for protocols is potentially error prone, it cannot give the guarantee that all possible actions of the modeled adversary are taken into account, the proof at one grasp works on quasi symbolic model of the protocol while remaining in the real word together with the necessary probabilistic and asymptotic considerations. The significance of the results in [3], [5], [14] first of all is that they took the attention of the community working in the field of ad-hoc network security to an important, powerful approach of simulatability and the importance of formal definition of the security goal.

Just within the field of routing protocols for ad hoc networks there are several protocols in stack waiting for formal security proof. Therefore, it seems important to call the attention of the community to a proof framework which provides sound modular design and provides the potential for obtaining rigorous security proofs.

Our belief is that the BPW approach is a big step to provide such clear-cut technology. Here we give a proof for the security of endairA in the BPW-model.

3. The EndairA protocol

The protocol endairA was proposed to provide a provably secure routing protocol in the context of the simulatability model [15],[3]. First we recapitulate the protocol as presented in [15],[3], [5]. In this kind of (usual) description a verbal explanation gives the details. Section 3.2. and the Appendix give the formal description of the protocol in the symbolic BPW setting.

Figure 1. shows an example route discovery, where the initiator of the route discovery is S, the target is T, and the intermediate nodes are A and B:

$S \rightarrow * :$ {rreq, S, T, id, ()}
$A \rightarrow * :$ {rreq, S, T, id, (A)}
$B \rightarrow * :$ {rreq, S, T, id, (A,B)}
$T \rightarrow B :$ {rrep, S, T, (A,B), (sig_T)}
$B \rightarrow A :$ {rrep, S, T, (A,B), (sig_T , sig_B)}
$A \rightarrow S :$ {rrep, S, T, (A,B), (sig_T , sig_B, sig_A)}

Fig.1. An example for the operation and messages of endairA.

The initiator of the route discovery process generates a route request, which contains the identifiers of the initiator and the target, and a randomly generated request identifier. Each intermediate node that receives the request for the first time appends its identifier to the route accumulated so far in the request, and re-broadcasts the request. When the request arrives to the target, it generates a route reply (a route contains different identifiers). The route reply contains the identifiers of the initiator and the target, the accumulated route obtained from the request, and a digital signature of the target on these elements. The reply is sent back to the initiator on the reverse of the route found in the request. Each intermediate node that receives the reply verifies that its identifier is in the node list carried by the reply, and that the preceding identifier (or that of the initiator if there is no preceding identifier in the node list) and the following identifier (or that of the target if there is no following identifier in the node list) belong to neighboring nodes. Each intermediate node also verifies that the digital signatures in the reply are valid and that they correspond to the following identifiers in the node list and to the target. If these verifications fail, then the reply is dropped. Otherwise, it is signed by the intermediate node, and passed to the next node on the route (towards the initiator). When the initiator receives the route reply, it verifies if the first identifier in the route carried by the reply belongs to a neighbor. If so, then it verifies all the signatures in the reply. If all these verifications are successful, then the initiator accepts the route.

3.1. The adversary model

The adversary can capture some of the honest nodes and she may be able to compromise their cryptographic secrets. It launches its attack from these compromised (adversarial) nodes. The adversary is able to overhear the communication of honest nodes which are neighbors of it. We well consider two adversary models (Model A and B), where in Model B we allow for the adversary to use also proprietary channels.

In both model we assume an adversary unifies the resources of all adversarial nodes. The adversary is able to transmit information between any two adversarial node. The adversary is able to share any information among all her nodes she obtained via any of its adversarial nodes (public identifiers, secret keys, elements from monitored runs). The adversary decides freely about the actual assignment of identifiers (and corresponding secret keys) to the adversarial nodes.

3.2. The endairA protocol in the BPW model

An overview of the endairA symbolic system is shown in Fig.2. using the usual notations in the BPW approach.

In the analysis of endairA ([3]) it was assumed that participants know the set of their neighbors, securely at least for honest elements. The protocol machine M_u of an honest node *u* is assumed to be initialized with the following information:

- set Id , the set all identifiers,
- list N_u , the list of neighbors,
- set $Nonce_u$, a set of nonces.

$Nonce_u$ is the set of request identifiers, which is an initially empty set. Each new request identifier is added to this set. Checking against this set, a participant can detect and drop a request message if received repeatedly.

In order to capture that keys have been generated and distributed, we assume that suitable entries already exist in the database *D* of the trusted host *TH*.

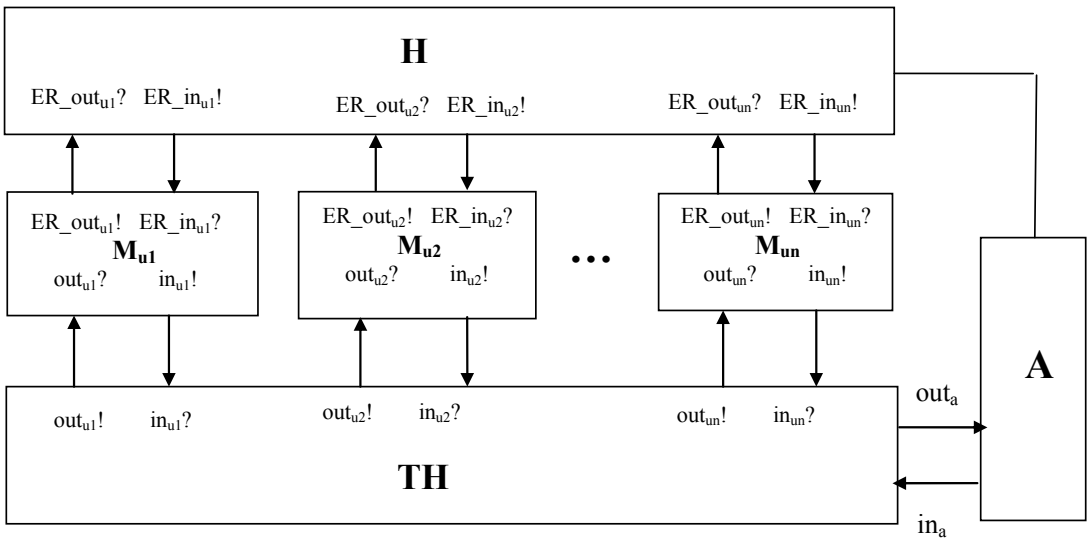

Fig.2: Overview of the endairA symbolic system

Communication channels, which are overheared also by the adversary are insecure, while all other channels are secure.

When user *u* sends a broadcast message to his neighbors, it is done by the following send command:

$send(N_u, l^{hnd})$ at port $in\_u?$:
Let $l^{ind} := D[hnd\_u = l^{hnd} \wedge type = list].ind$. If $l^{ind} \neq \downarrow$, then output
$(u, v, ind2hnd\_v(l^{ind}))$ at $out\_v?$, for all $v \in N_u$.

The protocol in symbolic setting is given in the Appendix.

The analysis of endairA ([3]) assumed that participants know the set of their neighbors. In Section 6. we show a way to include also the task of neighbor acquisition, which is a short example for modular design and analysis.

4. The security property

Assume adversary model Model A.

By usual definition a route is defined by the sequence of identifiers of nodes along the route. However, because the adversary is able to shuffle around freely with the identifiers of adversarial nodes, we demand only, that the adversarial nodes constitute an existing chain of communication together with honest nodes along the discovered route from the initializing node to the target node and we can not make any requirement about the identity of adversarial nodes participating in the route. The formal definition of Discovered Route Requirement, Req^{ER} is the following:

Definition 1: (Discovered Route Requirement) For any pairs *u*, *v* of honest users if $\exists t_j \in N:\ output(ok, v, x_4^{hnd})$ at $ER\_out_u!$ a time t_j, where $v_{j-1}, v_{j-2}, ..., v_0$ are the honest nodes on the discovered route x_4, then
$\exists t_0, t_1, ..., t_{j-1} \in N,\ t_0 < t_1 < ... < t_{j-1} < t_j$ such that
$t_i : (continue\_prot, m_i^{hnd})$ at $out_{v_i}?,\ i = 0,1,..., j-1,$
where $v_{j-1}, v_{j-2}, ..., v_0$ are all the different honest nodes, in the given order, on an existing route from the initializing node *u* to the target node *v* $(= v_0)$.

Theorem 1: (*Security of the endairA protocol based on the BPW model*)
Let $Sys^{ER,id}$ be the symbolic system under the BPW model and Req^{ER} the integrity property then $Sys^{ER,id} \models^{perf} Req^{ER}$, i.e. the integrity property holds for all traces arising in runs of $Sys^{ER,id}$.

Theorem 2: (*Security of the endairA protocol*) $Sys^{ER,id} \models^{poly} Req^{ER}$, i.e. the integrity property holds for polynomially bouded users and adversaries and with negligible error probability.

The proof of the cryptographic realization is based on the integrity preservation theorem, and the composition theorem of the BPW approach (the proof of Theorem 2. below is adaptation of a proof from [11]).

Proof of Theorem 2: In the BPW cryptographic library the real cryptographic operations are computationally at least as secure as the ideal ones, see [8]. Therefore, when we substitute the ideal operations by real ones in the symbolic BPW system, then by the composition theorem ([7]) the resulting real system is computationally at least as secure as the symbolic one. By the thorem of preservation of integrity properties ([10]) if we have two systems *Sys1*, *Sys2* such that *Sys1* is computationally at least as secure as *Sys2*, then *Sys2* $|=^{poly}$ *Req* implies *Sys1* $|=^{poly}$ *Req*. ■

5. Proof of security in the symbolic setting

The proof is based on a set of invariants. Invariants of the symbolic system are statements about the state of the symbolic system which hold at all times in all runs of the system.

Consider the following chain of signatures:

$$Sig_m\left(Sig_{m-1}\left(Sig_{m-2}\left(...Sig_1\left(x\right)...\right)\right)\right), \qquad (1)$$

where $Sig_i\left(y\right)$ is an entry in database *D* of trusted host *TH*, of type *sig* generated for list *y* by a participant having handle to *y* as well as to the secret signing key sk_i, $i=1,...,m$.

Invariants:

Inv. 1 *(Correct time order of signatures)* The order of signatures in the chain uniquely determines the order of time when they were generated.

Inv. 2 *(Correct time order of having control over TH)* The order of signatures in the chain uniquely determines the order of time the control held by a node having a handle to the corresponding secret key.

Inv. 3 *(Existing route)* At the time of route acquisition there existed a communication route between any two nodes which had handle to secret signing keys corresponding to two signatures in the chain.

Proof of invariants:
Correct time order of signatures.
When proving the invariant, we prove, that if an invariant holds at time t in a run of the system it will still hold at time *t+1* (one time unit is needed for the system to make one step).
Assume a new chain of signatures (1) appears in the database *D* at time *t+1*. The type of the corresponding new entry is **sig**. In the symbolic system, within a time step one command can be executed. The signature is the closing signature (Sig_m) of the chain

generated by command $sign(s^{hnd}, l^{hnd})$, where s^{hnd} is a handle to the secret signing key, and l^{hnd} is a handle to the list representing content $Sig_{m-1}\left(Sig_{m-2}\left(...Sig_1(x)...\right)\right)$.

Assume the invariant is violated at time *t+1*, i.e. the new signature that corresponds to a chain (1) is such that, there exists at least one signature in the chain (with index *j*), which has been generated earlier in time than at least one signature on the right to it in the chain (with index *k*), $m \geq j > k \geq 1$. Because Sig_m is generated at time *t+1*, therefore *j* must be smaller than *m*. The handle to the list representing content $Sig_{m-1}\left(Sig_{m-2}\left(...Sig_1(x)...\right)\right)$ must have existed already at time *t* (because otherwise the handle should have to be obtained by the signer also at *t+1* , but time step *t+1* is already reserved for a sign command), so $Sig_{m-1}\left(Sig_{m-2}\left(...Sig_1(x)...\right)\right)$ must have existed at time *t*. However, by assumption, the invariant holds up to time t. ■

Correct time order of having control.
In the symbolic system a signature can be generated only by a participant who has a handle to the secret signing key. A secret signing key is never transmitted to any other node by an honest node. According to invariant *Correct time order of signatures* the order of signatures in the chain uniquely determines the order of time when they were generated. It follows that those honest nodes whose secret keys have been used to calculate signatures in the chain, must have had control over *TH* in the time order corresponding to the order of signatures in the chain. Obvious statement follow for the adversary: the time gaps between periods when honest nodes controlled the system have been filled by the actions of the adversary. ■

Existing route.
Let nodes *w* and *z* be two honest nodes such that node *w* (and *z*) has a handle to secret signing key to signature Sig_j (and Sig_k, resp.) in the chain, where $m \geq j > k \geq 1$. According to *Correct time order of signatures* Sig_j and Sig_k were generated at time t_j and t_k, respectively, where $t_j > t_k$. The arguments of command $sign(s^{hnd}, l^{hnd})$, are a handle s^{hnd} to the secret signing key, and also a handle l^{hnd} to the list representing content. Signature Sig_j can only be generated if *w* has a handle l^{hnd} to a list *l* to be signed, which describes content $Sig_{j-1}\left(...Sig_k\left(...Sig_1(x)...\right)\right)$. By recursively parsing list *l*, *w* can get a handle also to Sig_k (in other words, knowing a chain of signatures we know also the components). *send* is the only command by which a participant may get a handle to an entry generated by an other participant. However a *send* command can successfully be executed only if there is an appropriate channel between two participants. It follows that node *w* can have a knowledge of Sig_k only if there exist a route from node *z* to node *w*. ■

Proof of Theorem 1:

Request steps of the protocol consist of only cleartext messages, therefore the adversary may be able to modify it by her wish. Therefore, we skip these steps in the security proof below, and we start with the reply message of the target. (Minimal formal checking of the arrived request message is assumed by the target, for instance, the identifiers are assumed to be different.)

Assume that honest protocol machine M_u outputs (ok, v, x_4^{hnd}) at $ER\_out_u!$ at time t_5. By definition of Algorithm 2, this can happen only if there was an input $(continue\_prot, m^{hnd})$ at $out_u?$ at time $t_4 < t_5$.

Because steps 73-104. have been carried out successfully (i.e. without an Abort event) by M_u, it follows that the signatures in the signature list are verified successfully by applying public keys corresponding to the identifiers in the identifier list, in an orderly manner.

Hence from invariants Inv. 1 and Inv. 2 it follows that if $v_{j-1}, v_{j-2}, ..., v_0$ are the honest nodes on the discovered route x_4, then $\exists t_0, t_1, ..., t_{j-1} \in N$, $t_0 < t_1 < ... < t_{j-1} < t_j$ such that the control has been at these nodes, formally, t_i: $(continue\_prot, m_i^{hnd})$ at $out_{v_i}?$, $i = 0,1,..., j-1$.

Furthermore, $v_{j-1}, v_{j-2}, ..., v_0$ are different nodes, which is ensured by the checking procedure according to the protocol.

Finally, invariant Inv. 3 ensures that $v_{j-1}, v_{j-2}, ..., v_0$ are nodes on an existing route, in that order, from the honest initializing node *u* to the honest target node v $(= v_0)$, which concludes the proof. ■

Note that in the protocol of endairA there are additional steps, which are not referred to in the above security proof. In Algorithm 2., essentially, only steps 83-104. (and step 105) are referred to, where the source node checks the accumulated identifier – signature list pair. Corresponding checks at intermediate node are skipped in the proof (neighbor checks and checks of accumulated identifier – signature list pairs at intermediate nodes). Consequently, these checks do not add to the ("final") security of the algorithm (under adversary model Model A).

Now we consider security under Model B, which models the case, when we assume the existence of proprietary channels exclusively available for the adversary for communication within the adversary. Using such channels the adversary may produce a route between two honest nodes which does not exist in the view of honest nodes.

Example 1: A route B-A1-A2-C, where B and C are honest nodes with no existing route between them, furthermore A1 and A2 are adversarial nodes, where nodes A1 and A2 are connected by a proprietary channel. The adversary might try to make honest node B to believe that honest node C is its neighbor. Checking against a safe neighbor list defeats such an attack. However, B and C will accept a route B-A1-A2-C where A1 (A2) is a neighbor of B (C) , respectively.

When we check neighbor list in Algorithm 2., it gives only a partial protection against detection of a non-existing route.

The Discovered Route Requirement for Model B is Definition 1. with the following modification:
"…where $v_{j-1}, v_{j-2}, ..., v_0$ are all the different honest nodes, in the given order, such that honest nodes which are neighbors on the discovered route are also neighbors according to the neighbor lists."

Informally, above, we have proved that endairA meets this security requirement. For a formal proof we have to modify the proof of invariant "Existing Routes" (denoted Inv.3*) and sightly also the proof of Theorem 1.

Inv. 3* *(Existing route*)* At the time of route acquisition there existed a communication route between any two honest nodes which had handle to secret signing keys corresponding to two signatures which are neighbors in the signature chain.

Proof of Inv 3*:
Let node *w* and *z* be any two honest nodes such that node *w* (and *z*) has a handle to secret signing key to signature Sig_{j+1} and Sig_j, respectively, in the chain of signatures, where $m > j \geq 1$. According to *Correct time order of signatures* Sig_{j+1} and Sig_j were generated at time t_{j+1} and t_j , respectively, where $t_{j+1} > t_j$. We want to show that there exists a route between nodes *w* and *z.* According to Algorithm 2. node *w* generates signature Sig_{j+1} for content $Sig_j\left(...Sig_k\left(...Sig_1\left(x\right)...\right)\right)$ only if in the view of node *w* there is an existing channel between node *w* and node *z*, i.e. node *z* is on the neighbor list of node *w.* ■

Now we give the needed modification at the end of the proof of Theorem 1:

"…Source node *u* also verifies, if its neighbor on the discovered route is also a neighbor according to its neighbor list, invariant Inv. 3* ensures that $u, v_{j-1}, v_{j-2}, ..., v_0$ ($v = v_0$) are honest nodes such that neighbors on the discovered route are also neighbors according to the neighbor lists, which concludes the proof."

6. Composition of subsystems

For short protocols like endairA or the Needham-Shroeder-Lowe [11] protocol the analysis of the symbolic system can be carried out by hand, however, even the Kerberos [13] is already at the borderline of the by-hand capabilities. Therefore, design of protocols should be carried out with a modular analysis in mind. From provably secure components we wish to build provably secure protocols relying on appropriate composition theorems. A general composition theorem was given in [12].

Fig 2. shows the overview of four systems ***SYS_0***, ***SYS_1,*** ***SYS_2*** and ***SYS_3***, each of which is made of two connected subsystems, where Sys_1^x is the endairA subsystem, Sys_0^x is the neighbor acquisition subsystem (x=r: real, x=s: symbolic, x=i: ideal). Furthermore, "≥" denotes "computationally at least as secure" and "=" stands for "perfectly as secure as". By Sys_0^i we want to model the assumption of the a priori knowledge of neighbors.

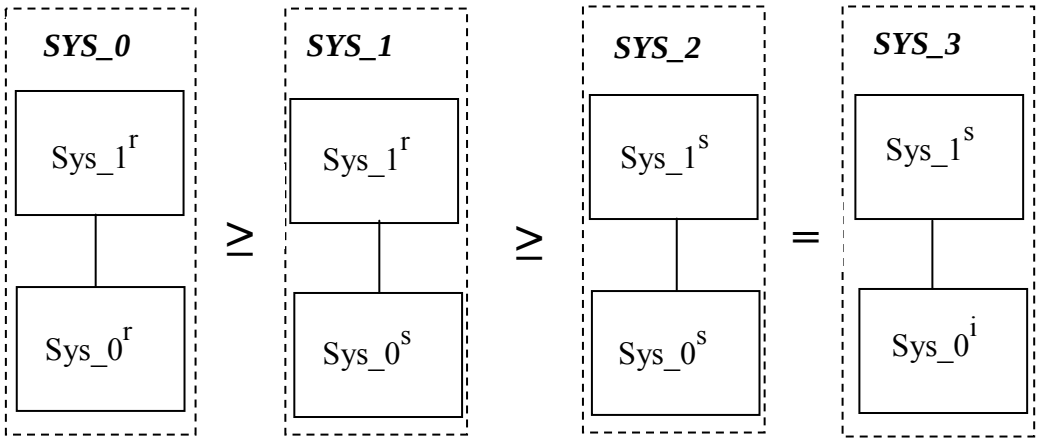

Fig.3: Overview of the modular protocol with security relationships : Sys_1^r is the real endairA component, Sys_0^x is the neighbor acquisition component (x=r: real, x=s: symbolic, x=i: ideal).

Henceforth, we consider system ***SYS_3***, which corresponds to the BPW symbolic endairA system extended with the ideal neighbor acquisition system. Fig.4. shows the overview of the extended proof system.

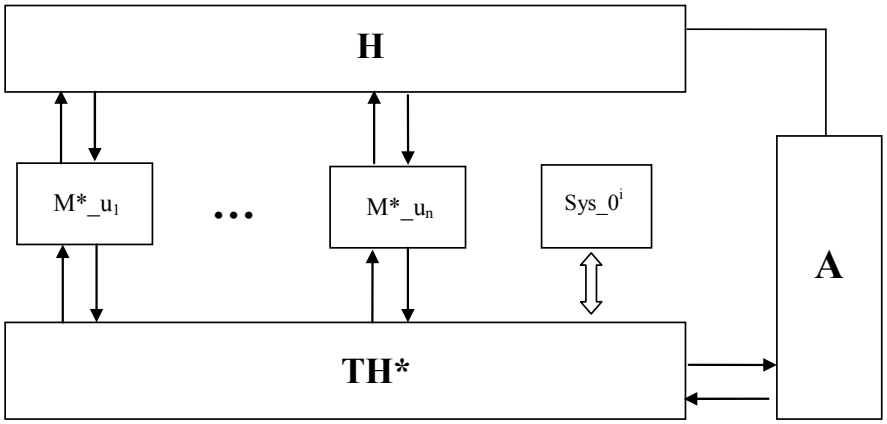

Fig.4.: Overview of the extended endairA symbolic system: *M*_u* is the endairA symbolic system, Sys_0^i is the neighbor acquisition ideal system.

Protocol machine *M*_u* contains the Sys_1^s system of user *u*. System Sys_1^s and an own copy of Sys_0^i are "physically" within the same node, which can be modeled by secure channels between *M*_u* and the common Sys_0^i in the extended proof system in Fig.4. Sys_0^i has communication channels also to the trusted host *TH**.

At its input port Sys_0^i accepts input *(u)*, where *u* is a node identifier. Sys_0^i sends *neighbor_req(u)* request to the ideal host *TH**, which latter sets a handle Ne_u^{hnd} for *M*_u* to the list of the corresponding neighbors. Sys_0^i replies with an *ok* message to Sys_1^s. Henceforth, Sys_1^s has access to the neighbor list via handle Ne_u^{hnd}.

*TH** is an extended version of *TH* of Fig.2. *TH** stores a representation of the communication graph *G* of the network. The adversary (via a *graph_adv* command) is allowed to set the identifiers of adversarial nodes and the state of the links between adversarial nodes.

According to Fig.3., if we prove security in ***Sys_3*** , we get a proof by composition and integrity preservation theorems also in ***Sys_0***.

7. Conclusions

In this paper we gave the first cryptographically sound security proof for a routing protocol for ad hoc networks. Sound separation of formal and cryptographic aspects is the important first step in the security analysis of cryptographic protocols. The BPW-approach is a strong candidate on this way.

In order to be able to analyze protocols, except short ones, modularization is necessary not only along cryptographic/formal issues of the analysis, but also the design should support the composable component approach, where the protocol is built from components which can be analyzed separately such that from the provably secure components we can build provably secure protocol by applying composition theorems. Obviously the natural design approach is when these components are formed along natural service interfaces within the protocol. Remaining at the application of this paper, the corresponding problem is to brake down a routing problem into such components.

References

[1] G. Ács, L. Buttyán, and I. Vajda. Provable security of on-demand distance vector routing in wireless ad hoc networks. In *Proceedings of the Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, 2005.

[2] G. Ács, L. Buttyán, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2006.

[3] G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11), 2006.

[4] G. Ács, L. Buttyán, and I. Vajda. The security proof of a link-state routing protocol for wireless sensor networks. In *Proceedings of the 3rd IEEE Workshop on Wireless and Sensor Networks Security (WSNS)*, 2007.

[5] G. Ács. Secure Routing in Multi-Hop Wireless Networks. *PhD Thesis*. Technical University of Budapest, 2009.

[6] T.R. Andel. Can Ad Hoc Routing Protocol be Shown Provably Secure? *Technical Report/TR-060615*, Computer Science Department, Florida State University, 2006.

[7] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001.

[8] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. *IACR Cryptology ePrint Archive*, Report 2003/015, http://eprint.iacr.org/, January 2003.

[9] M. Backes and B. Pfitzmann, and M. Waidner. Reactively Secure Signature Schemes. C. Boyd and W. Mao (Eds.): ISC 2003, *LNCS 2851*, pp. 84–95, 2003.

[10] M. Backes and C. Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 675–686. Springer, 2003.

[11] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. *Journal on Selected Areas in Communications*, 22(10):2075–2086, 2004.

[12] M. Backes and B. Pfitzmann. A General Composition Theorem for Secure Reactive Systems. *Theory of Cryptograpy Conference (TCC 2004)*, *LNCS 2951*, pp. 336-354, 2004.

[13] M. Backes, I. Cervesato, A.D. Jaggard, A. Scedrov and J-K. Tsay. . A cryptographically sound security proof for Basic and Public key Kerberos. Computer Security – *ESORICS 2006*, LNCS, Volume 4189/2006, 362-383.

[14] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1998.

[15] L. Buttyán, I. Vajda: Towards provable security for ad hoc routing protocols. *2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, Washington DC, USA (2004) 94-105

[16] J. R. Douceur. The sybil attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[17] G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2004.

[18] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy Magazine*, 2(3):28–39, 2004.

[19] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks Journal*, 11(1), 2005.

[20] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, volume 1, pages 293–315, September 2003.

[21] K. Saghar, W. Henderson and D. Kendall. Formal modelling and analysis of routing protocol security in wireless sensor networks. *Proceedings of the 10th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET 09)*, June, pp. 179-184, 2009

[22] A.D. Wood and J.A. Stankovic. Denial of service in sensor networks. In *IEEE Computer*, volume 35, pages 54–62, Sep 2002.

Appendix: endairA protocol in symbolic setting

Algorithm 1: initialization of a new protocol run.

Input: (new_prot, v) at ER_in_u ? for honest node u

1. $id^{hnd} \leftarrow gen\_nonce()$
2. $u^{hnd} \leftarrow store(u)$
3. $v^{hnd} \leftarrow store(v)$
4. $rreq^{hnd} \leftarrow store(rreq)$
5. $l_1^{hnd} \leftarrow list()$
6. $l_2^{hnd} \leftarrow list(rreq^{hnd}, u^{hnd}, v^{hnd}, id^{hnd}, l_1^{hnd})$
7. $send(N_u, l_2^{hnd})$

Algorithm 2:

Input: $(continue\_prot, m^{hnd})$ at out_w ? for honest node w

1. $x_i^{hnd} \leftarrow list\_proj(m^{hnd}, i)$ for i=1,…,5
2. $w^{hnd} \leftarrow store(w)$
3. $x_i \leftarrow retrieve(x_i^{hnd})$ for i=1,2,3
4. **if** $(x_1 \notin \{rreq, rrep\})$ **then**
5. Abort
6. **end if**
7. $typ_i = get\_type(x_i^{hnd})$ for i=4,5

{RREQ message}

8. **if** $(x_1 = rreq) \cap \big((x_2 \notin Id) \cup (x_3 \notin Id) \cup (typ_4 \neq nonce) \cup (typ_5 \neq list)\big)$ **then**
9. Abort
10. **end if**
11. **if** $(x_4^{hnd} \in Nonce_w)$ **then**
12. Drop_message
13. **end if**
14. $Nonce_w = Nonce_w \cup \{x_4^{hnd}\}$

{RREQ arrives to an intermediate node}

15. **if** $(x_1 = rreq) \cap (x_3 \neq w)$ **then**
16. $l_3^{hnd} \leftarrow list(x_5^{hnd}, w^{hnd})$
17. $l_4^{hnd} \leftarrow list(x_1^{hnd}, x_2^{hnd}, x_3^{hnd}, x_4^{hnd}, l_3^{hnd})$
18. $send(N_w, l_4^{hnd})$
19. **end if**

{RREQ arrives to the target node}

20. **if** $(x_1 = rreq) \cap (x_3 = w)$ **then**

21. $s^{hnd} \leftarrow sign(ske_w^{hnd}, m^{hnd})$
22. $l_5^{hnd} \leftarrow list(m^{hnd}, s^{hnd})$
23. $send(N_w, l_5^{hnd})$
24. **end if**

{RREP message}

25. **if** $(x_1 = rrep) \cap \big((x_2 \notin Id) \cup (x_3 \notin Id) \cup (typ_4 \neq list) \cup (typ_5 \neq list)\big)$ **then**
    Abort
26. **end if**

{RREP arrives to an intermediate node}

27. **if** $(x_1 = rrep) \cap (x_2 \neq w)$ **then**

28. j=0 {checking neighbors in the accumulated list of identifiers}
29. **do**
30. j=j+1
31. $y_j^{hnd} \leftarrow list\_proj(x_4^{hnd}, j)$
32. **if** $y_j^{hnd} = \downarrow$ **then**
33. Drop message
34. **end if**
35. $y_j \leftarrow retrieve(y_j^{hnd})$
36. **while** $y_j \neq w$

37. $y_{j+1}^{hnd} \leftarrow list\_proj(x_4^{hnd}, j+1)$
38. **if** $y_{j+1}^{hnd} = \downarrow$ **then** $y_{j+1} = x_3$
39. **else**
40. $y_{j+1} \leftarrow retrieve(y_{j+1}^{hnd})$
41. **end if**
42. **if** $j = 1$ **then** $y_{j-1} = x_2$
43. **end if**

44. **if** $(y_{j-1} \notin N_w) \cup (y_{j+1} \notin N_w)$ **then**
45. Abort
46. **end if**

{checking the list of signatures}

47. i=1
48. $k_1^{hnd} \leftarrow list(x_1^{hnd}, x_2^{hnd}, x_3^{hnd}, x_4^{hnd})$
49. $z_1^{hnd} \leftarrow list\_proj(x_5^{hnd}, 1)$
50. $q_1^{hnd} \leftarrow list\_proj(x_4^{hnd}, 1+j)$
51. $q_1 \leftarrow retrieve(q_1^{hnd})$
52. $l_{5+i}^{hnd} \leftarrow list()$

53. **while** $z_i^{hnd} \neq \downarrow$ **do**

54. $type_i = get\_type(z_i^{hnd})$
55. **if** $(type_i \neq sig)$ **then**
56. Abort
57. **end if**
58. $b \leftarrow verify(z_i^{hnd}, pke_{q_i}^{hnd}, k_i^{hnd})$
59. **if** $b = false$ then
60. Abort
61. **end if**
62. i=i+1
63. $l_{5+i}^{hnd} \leftarrow list(l_{5+i-1}^{hnd}, z_{i-1}^{hnd})$
64. $k_i^{hnd} \leftarrow list(k_{i-1}^{hnd}, l_{5+i}^{hnd})$
65. $z_i^{hnd} \leftarrow list\_proj(x_5^{hnd}, i)$
66. $q_i^{hnd} \leftarrow list\_proj(x_4^{hnd}, i+j)$
67. $q_i \leftarrow retrieve(q_i^{hnd})$
68. **end while**

69. $s_1^{hnd} \leftarrow sign(ske_w^{hnd}, m^{hnd})$
70. $l_6^{hnd} \leftarrow list(m^{hnd}, s_1^{hnd})$
71. $send(N_w, l_6^{hnd})$
72. **end if**

73. **if** $(x_1 = rrep) \cap (x_2 = w)$ **then** {RREP arrives to the source node}

74. $p_1^{hnd} \leftarrow list\_proj(x_4^{hnd}, 1)$
75. **if** $p_1^{hnd} = \downarrow$ **then**
76. $p_1 = x_3$
77. **else**
78. $p_1 \leftarrow retrieve(p_1^{hnd})$
79. **end if**

80. **if** $(p_1 \notin N_w)$ **then**
81. Abort
82. **end if**

rows 83-104: {rows 47-68 are repeated here, with j is set to 0}

105. $output(ok, v, x_4^{hnd})$ at ER_out_w!
106. end if