Framework for Security Proofs for On-demand Routing Protocols in Multi-Hop Wireless Networks

István Vajda

Abstract: We present a framework for security proofs for on-demand routing protocols. The framework relies on the composable cryptographic library by Backes, Pfitzmann and Waidner (BPW). The idea is to break down the security requirement against the system (the protocol) into security requirement against the elements of the system, the honest protocol machines in the BPW symbolic model. The practical income of this approach is simplified, structured and cryptographically sound security proof as well as it provides guidelines for designing provably secure protocols.

1. Introduction

Routing is a fundamental networking function, therefore a successful attack against routing disables the operation of a network. Several "secure" routing protocols have been published. Unfortunately, the analysis of routing protocol security features has typically been informal so far.

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 and the proof is conducted in a symbolic model applying formal verification methods (model checkers, theorem provers). The abstraction of cryptography has almost always been based on the classical Dolev-Yao model. The benefit of such a formal approach is that, at least in principle, it can exhaustively test the protocol against all types of possible attacks within the formal model, and if we can find an attack in the formal model, the attack can be used successfully against any cryptographic implementation of the protocol. However, if we find a proof, we cannot be sure that it can be carried over to cryptographic implementation, where the adversary is much more powerful.

A sound way of proof, if we can separate the formal and the cryptographic part by producing a 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], which makes possible, that in a reactive scenario cryptographically sound proof can be given for the real protocol via the analysis of the abstracted protocol.

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

When we prove the security in the symbolic system, then we have to show that the security requirement $\mathrm{Re}q_{Sys}$ (i.e. the specification of the secure service expected from the system) is met in the presence of the adversary. If the attacker can successfully attack the symbolic system, then she is able to attack the "logic" of the protocol, because all crypto elements are ideally attack-proof in this system. In this paper we propose a structured analysis of the symbolic system in the case of considered classes of routing protocols (on-demand source routing (SR), on-demand dynamic vector distance routing (VDR)). On this way we introduce a second layer of abstraction by defining ideal honest protocol machines and the corresponding security requirement $\mathrm{Re}q_m$. We show that requirement $\mathrm{Re}q_{Sys}$ is met if and only if $\mathrm{Re}q_m$ is fulfilled by honest protocol machines. Therefore, for a concrete protocol, we have to show that the implementation of this ideal machine, which implementation is in the symbolic model is a secure substitution of the ideal machine. In other words, we use the composition technique twice, first to abstract the real crypto elements based on the general composable cryptographic library and second to abstract the implementation details specific to a given protocol in the symbolic model. The ideal machine grasps the security expectation of a protocol class (class of RSR and class of VDR protocols).

The advantage of our approach is twofold. At first, the security analysis of the routing protocol is simplified to the analysis of a component, an honest protocol machine in the symbolic model. Such a structured analysis supports clear-cut and considerably less error-prone derivations. At second, the abstraction of the symbolic model gives a deeper insight into the necessary, essential security measures to comply with the security requirement, consequently, it provides also guidelines for designing secure protocols within the considered class of protocols.

The structure of the paper is the following. Section 2 gives a short summary of the related works. In Section 3 we introduce the approach of idealized symbolic protocol machines for SR routing protocols. In Section 4 we apply the technique for VDR protocols with an example security analysis of protocol ARAN. Conclusions are drawn in Section 6.

2. Related work

In the last decade several efforts have been made to provide cryptographically sound security proofs. For the time being, the BPW approach is the only one, which supports such proofs with a general composable cryptographic library [8] and corresponding composition theorems [12]. 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]. Applying the BPW approach, we gave the first cryptographically sound security proof for a routing protocol, the endairA [23].

An overwhelming majority of security analysis of routing protocols is informal and potentially error-prone, therefore it happened many times that after thorough inspection of a protocol, researchers of the community came up with a vital attack, e.g. Ariadne [3], 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 it triggers a wave of papers, e.g. Sybil attack [16], sinkhole attack [18], route diversion attacks [18].

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 [14]. In this framework we gave a proof for the security of endairA protocol [3]. Sybil and wormhole attacks were included in the adversary model by Ács in [5], where the definition of secure route discovery was adjusted, accordingly. The proof in [5] uses the same proof framework as the original proof in [15]. As it was mentioned above, an all-in-one proof for protocols is potentially error prone, in general, it cannot give the guarantee that all possible actions of the modeled adversary are taken into account, the proof at one grasp works on a quasi symbolic model of the protocol, while remaining in the real word with the necessary probabilistic and asymptotic considerations. Therefore, the significance of the results in [3], [5], [14], first of all, is that these results took the attention of the community working in the field of ad-hoc network security to the important, powerful technique of simulatability and the importance of the formal definition of the security goal of a protocol.

This paper provides a framework for the structured analysis and design of cryptographically sound provably secure routing protocols. The approach we describe in this paper, as we hope, will contribute to calling the attention to the efficiency of composition techniques, the structured design and analysis of secure protocols in general, especially, in the field of wireless ad-hoc and sensor networks.

3. Idealization of protocol machines in the symbolic system

3.1. On-demand route acquisition and the adversary model

In case of on-demand routing protocols a route is established between a source and a destination when it is needed, i.e. when the source wants to send something to the destination.

The source generates a route request message (RREQ), which contains the identifiers of the source (S) and the destination (D), furthermore request identifier(s) as well as further protocol specific information. The request message is broadcast by the source (originator node) (see Fig.1.). When a request packet is received by an intermediate node, first it checks the request identifier, and if it has already been received in a request packet, then the request packet is dropped. Otherwise, the packet is processed according to the protocol and the output is broadcast. This procedure is repeated until the request reaches the destination.

Destination node generates a route reply message (RREP) and unicasts back to the node from which the message has been received.

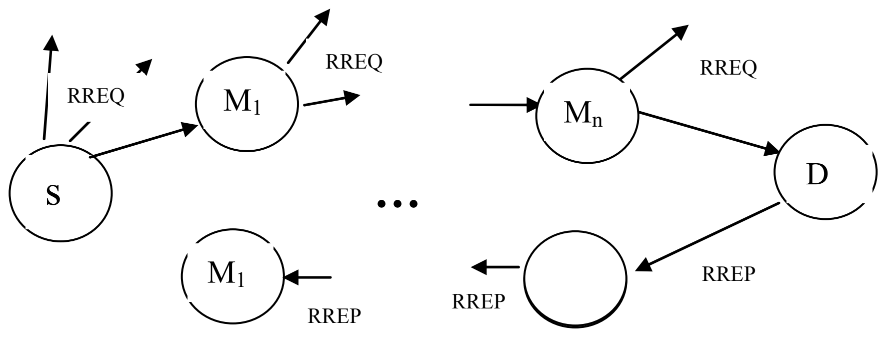

Fig.1.: Message flow in on-demand route acquisition protocol

The adversary freely uses all the resources of the adversarial nodes (identifiers, keys, communication channels). Therefore, we can not identify the adversarial nodes in a secure way, i.e. we cannot be sure in the true identity of an adversarial node. We assume that all communication channels are publicly available, i.e. there are no proprietary channels (e.g. used by the adversary to produce communication channels between honest nodes during route acquisition which do not exist for standard communication). In Section 5, we will return to the special case of existing proprietary channels.

The adversary is not omnipotent, it has no additional resources and capabilities, just those available via the set of compromised nodes, except the assumption that the adversary is able to initialize protocol runs with any of honest originator and honest destination nodes.
A communication channel between two honest nodes is secure if and only if neither of these honest nodes has a communication channel to any of the adversarial nodes. Consequently the adversary is able to collect run information (protocol messages) only via legal communication channels of compromised nodes. The adversary is not able to schedule honest machines at her wish during the run of the protocol, she is able to activate honest machines just via the legal way by sending syntactically correct protocol messages from an adversarial nodes.

The adversary complies with the syntactic rules of the protocol and she does not launch DoS-type attacks.

The aim of the attacker is to attack the security requirement: the attacker tries to make the originator node accept a “non-existing” route, a false route, which does not comply with the security requirement. (Note, we are working in the symbolic world, therefore we have to meet security requirement perfectly, not within the accuracy of some negligible error probability.)

3.2. Decomposition of the security requirement

Fig.2. shows the overview of the symbolic system with usual notations of the BPW approach. Protocol machines $M_{u_i}, i = 1, 2, ..., n$ are honest. Compromised protocol machines (adversarial machines) $M_{u_i}, i = n+1, ..., N$ are incorporated into the adversary machine *A*. User machine *H* communicates with protocol machines: initializes a new run and receives the output of the run. A secure protocol has to meet – formally defined - security requirement at the service layer *I*, which lays between machine *H* and the honest protocol machines.

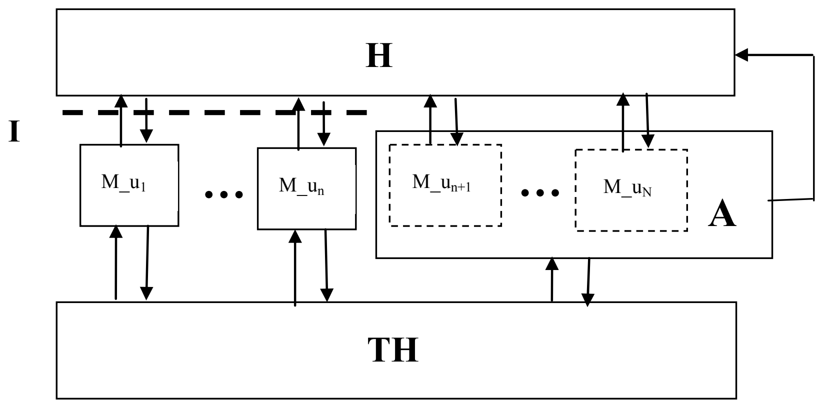

Fig. 2.: Overview of the symbolic system (service layer S)

The adversary is also allowed to initialize new protocol runs via user machine *H*. Trusted host *TH* is an important element of the symbolic system. Protocol machines (honest and adversarial) communicate with each other via *TH*. All cryptographic primitives are moved from protocol machines into *TH* and are available for protocol machines by sending commands to *TH*. Machine *TH* contains also a database, which stores the history of all operations made by protocol machines with cooperation of *TH* (see [8]).

User machine H initializes new protocol runs by sending appropriate input to a protocol machine M_u_i and at the end of the run the same machine will send the result (a discovered route in case of an SR protocol) to machine *H*, via service interface *I* (Fig.3).

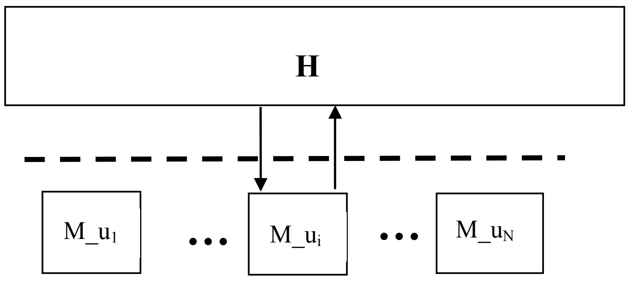

Fig.3. User machine initializes session and receives result via protocol machine M_{u_i}

Definition 1: Security requirement *Req* for SR protocol
For any pairs *S*, *D* of honest users if $v_1, v_2, ..., v_j$ are the honest nodes on the discovered route, then $v_1, v_2, ..., v_j$ are all the different honest nodes, in the given order, on an existing route from the originator node *S* to the destination node *D* $(= v_j)$.

□
Now we introduce the concept of decomposition of the security requirement *Req*.

Protocol machines process the protocol message serially in time, i.e. the output of a machine becomes the input of the subsequent machine. When honest node u_i receives a syntactically correct input message z, it outputs message $T(z, w_i)\,(= T_{u_i}(z))$, where transformation *T* is defined by the protocol, w_i is a binary string, the components of which are quantities known by node u_i (e.g. secret keys, time etc.). This way the serial processing corresponds to nesting a series of transformations: message $T_{u_i}(T_{u_j}(...(s)...)$ is output by node u_i, when it has received an input $T_{u_j}(...(s)...)$ from node u_j, where *s* is an initializing message.

For verification purposes protocol entities can roll backward in the series of nested transformation they have received as deep as they have to (and permitted to) according to the protocol. Specially, the session originator node is able (must be able) to parse a syntactically correct message and to reproduce the outputs of all the intermediate nodes by removing transformations one by one. So, if not removed or modified by the adversary the outputs ("transformations") of honest intermediate nodes arrive to the originator node, too. The session originator node processes these outputs and forwards the resulted routing information to service interface *I*.

Alternatively, we can imagine that there is a direct virtual channel from honest nodes to the service interface via which they can send their contribution to the set of routing information collected during the session (illustrated in Fig.4.).

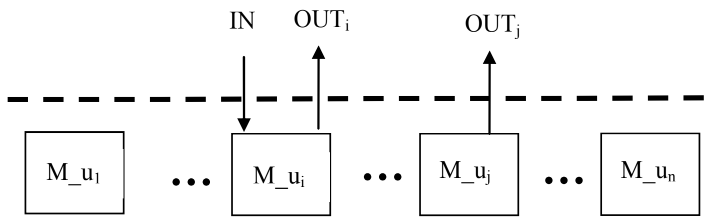

Fig.4. Decomposition of the output: machine M_{u_i} initializes the session and all participating nodes report directly to the service interface

Let $\mathrm{Re}q_{u_i}$ denote the security requirement for the virtual channel of an honest protocol machine M_{u_i}.

Definition 2: Security requirement $\mathrm{Re}q_{u_i}$ for an ideal honest SR protocol machine

The components of the output of the virtual channel are the following:

1. identifier of machine M_{u_i}
2. session identification information (minimally S, D, session id)
3. identifier of those protocol machines in time order, which have processed the protocol message before it arrived to machine M_{u_i},

where an identifier corresponds to the true identity of the machine if it is an honest machine, otherwise, it is one from the set of the identifiers of adversarial machines.
□

A few comments follow to Definition 2:

Phrasing "have processed" means that the corresponding protocol machine (honest or adversarial) have performed a syntactically correct transformation on its input message.

Implicitly, this also means that transformations cannot be removed later on by the adversary. Informally, if during a session at time t_1 a protocol message arrives to an ideal honest protocol machine, it fixes the past processing events happened to the protocol message before time t_1 such a way that an adversary can not rewrite the history of processing the protocol message (assuming that the adversary avoids causing an abort of the session).

An adversarial node might just forward an input message between two nodes, which action is not considered here as "processing" the message, when we call these two nodes pseudo neighbors.

The ideal machine has no oracle capabilities. E.g. even an ideal machine cannot distinguish honest and adversarial nodes from their outputs.

In the RREQ and RREP phase an intermediate machine may run different programs. E.g. there are protocols, where intermediate machines do just plaintext operations in the RREQ phase of the session. Security requirement refers to outputs of intermediate nodes they intend to send safely to the originator node.

A virtual channel from an honest machine M_{u_i} to interface *I* complying with requirement $\mathrm{Re}q_{u_i}$ will shortly be called an *ideal_channel*. Furthermore, requirements *Req* and $\mathrm{Re}q_{u_i}$ will be called global and local, respectively.

Theorem 1 (Decomposition of global security requirement)
Security requirement *Req* by Definition 1. is fulfilled if and only if honest nodes comply with local security requirements $\mathrm{Re}q_{u_i}$ by Definition 2.

Proof of Theorem 1:
$\mathrm{Re}q_{u_i} \to \mathrm{Re}q$: Assume ideal honest nodes $C_1, C_2, ..., C_n$ have processed the protocol message during the session, before it arrived as an input message to ideal honest node C_{n+1}. By induction, we assume, in this input message the order of nodes $C_1, C_2, ..., C_n$ corresponds to the true time order and none of the honest nodes have been deleted from the protocol message by the adversary. By $\mathrm{Re}q_{C_{n+1}}$ the output message of ideal honest node C_{n+1} cannot be processed by nodes following node C_{n+1} during the session such that it could cause any change in series $C_1, C_2, ..., C_n, C_{n+1}$.

$\mathrm{Re}q \to \mathrm{Re}q_{u_i}$: If any of the honest machines would not use an ideal_channel, then $\mathrm{Re}q$ could not be met. Indeed, because honest machines run the same protocol (in symbolic model), none of them would comply with security requirement $\mathrm{Re}q_{u_i}$ and the adversary is able to successfully manipulate the sub-series of honest nodes participating in the session.
■

The merit of this approach is that the security evaluation of the protocol against requirement $\mathrm{Re}q$ is simplified to the security evaluation of a protocol machine u_i against requirement $\mathrm{Re}q_{u_i}$ (recall, all honest protocol machines run the same protocol).

A design approach could also be built by using the above method in a reversed way, where first we would design ideal intermediate nodes and then we would extend this strength to the whole protocol.

3.3. Design considerations for the implementation of the ideal_channel

According to Definition 2. the ideal protocol machine sends output to interface *I* via an *ideal_channel*, which protects the protocol message from hostile deletion or modification of the series of intermediate honest machines.

It arises the question of how we can implement such channel. First, we cite the case of two known protocols, the Ariadne [15] and the endairA [3] protocols, which use the technique of nested digital signatures, then an example construction follows.

1.) Ariadne:
During the RREQ phase an intermediate node *C* appends its public identifier, *id_c* to the received message *m* and signs the result:

$$(m, id_C)sign_C \tag{1}$$

It is an insecure implementation of the ideal_channel. E.g. adversary *A* is able to remove signature ending of the chain, and substitute it to get:

$$(m, id_A)sign_A$$

Paper [3] shows an attack which exploits this weakness.

2.) endairA:
This protocol implements the ideal_channel by an implicit way (Fig.5.).

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

Fig.5. An example for the operation and messages of endairA. Originator node is S, destination node D, intermediate nodes are A and B, sig_x is calculated on all preceding fields.

Destination node *D* signs the sequence of identifiers accumulated as plaintext information during the RREQ phase of the session (call it *maybe_route*), which if untouched by the adversary, is a real route from node *S* to node *D*. Node *D* produces message

$$(..., maybe\_route)sign_D \tag{2}$$

and uses it as part of the initializing message of the RREP phase. The transformation performed by an intermediate node *C* on the input message *m* is a signature:

$$(m)sign_C \tag{3}$$

When an intermediate honest node signs the input message, it confirms the validity of routing information derivable from the input message: looks for his own identifier on maybe_route and checks if the preceding identifier on this route is consistent with the ending signature of the input message. Nodes can verify also all preceding signatures, however, this is only a syntactic checking and – by assumption - the adversary is careful to output syntactically correct messages to avoid aborting the session (see [23] for detailed analysis).

3.) An example implementation of the ideal_channel:

Construction (1) of nested signatures can be made secure by the following transformation:

$$\left(\left(m, id_C\right) enc_S\right) sign_C \qquad (4)$$

Informally, the adversary is not able to do any modifications to an honest transformation: the encryption with the public key of *S* prevents the adversary to eliminate an honest transformation. The last signature could be replaced easily by the adversary, however, he could not adjust the corresponding identifier within the encryption operation.

Note, an honest node does not have to be able to authenticate the identity of the node from which he received the protocol message. Indeed, the task for an honest node is to bond the received message and its own identifier securely such away that the session originator node is able to verify all the preceding transformations. Therefore, construction (4) can be simplified into:

$$\left(m, sign_C\right) enc_S \qquad (4')$$

Example 1:
Applying construction (4’), Fig. 6. shows an example route discovery protocol, where the originator node is S, the destination node is D, and the intermediate nodes are A and B. Let the double operation {{…} sig$_X$} enc$_S$ be denoted as {…} sig-enc$_{XS}$:

S → * : rreq, S, D, id, ()
A → * : rreq, S, D, id, (A)
B → * : rreq, S, D, id, (A,B)
D → B : (A,B,D), (m) sig$_D$;m= rrep, S, D, id
B → A : (A,B,D), ((m) sig$_D$) sig-enc$_{BS}$
A → S : (A,B,D), (((m) sig$_D$) sig-enc$_{BS}$) sig-enc$_{AS}$

Fig.6. An example protocol

Note, that destination node applies lighter transformation than sig-enc. For the session phase identification information, only a signature is calculated by node *D*. Indeed, the role of the encryption transformation is to bond the series of previous nodes, processing the protocol message. Destination node *D* is the first node processing the session phase information.
Construction (4’) is lavish upon heavy crypto operations compared to endairA. Note, however, if RSA technology is applied, then an encryption transformation with small public exponent might add only a minor increment to the complexity of the number of modular multiplications.
□

In construction (4-4') public key technology is applied. Symmetric key technology could be used in a scenario, where *S* is a base station sharing secret keys with each node of the network (typical scenario in a wireless sensor network). In this case, a construction analogue to (4) is the following:

$$\left((m)enc_C\right)MAC_C \tag{5}$$

If we consider a scenario, where from security reasons more then one base station is used, it may happen that the complexity of symmetric key management increases to a level, where a public key construction, like (4-4') becomes a feasible candidate.

4. On-demand distance vector routing

RDVR protocols do not use source routing messages, but routing decisions are based on traditional routing tables. Each node maintains a routing table, where each entry of the table contains the following information: the identifier of the destination, the identifier of the next hop on the route towards the destination as well as the believed route cost (e.g. hop count in case of SAODV [25], time delay in case of ARAN [22]). When an intermediate node receives a packet to be forwarded to a given destination, it looks its routing table to see who is the next hop towards the destination and then forwards the packet to that next hop.

Protocol machines store and use the routing information (e.g. the next hop towards a node) they derive, and they do not have to forward it toward the originator node, as in SR protocols. Virtually speaking, the ideal_channel is trivially given.

Informally, the security requirement for a on-demand distance vector routing (RDVR) protocol, is that every routing entry of all honest nodes must be "correct". A routing entry at a node *C*, is a quadruple: (*C*,*E*,*F*,*c*), where *E* is the identifier of the *next hop node* towards *ending node F*, and *c* is the route cost value. A node cost *C(B)* is assigned to each node *B*, which is a non-negative additive quantity, giving the cost of using the node to process and forward the protocol message. Route cost c is the sum of all node cost values along the route [1].

Now we formalize the corresponding security requirements $\mathrm{Re}\,q$ and $\mathrm{Re}\,q_{u_i}$.

Definition 3: Security requirement ($\mathrm{Re}\,q$) for an RDVR protocol
Routing entries of honest nodes must be correct, where a routing entry (*C*,*E*,*F*,*c*) is correct, if

1.) there exists a route starting at node *C* and ending at node *F* via next hop *E* such that on this route:

1.1.) each honest node (*U*) has a routing entry with ending node *F* such that the next hop points to

1.1.1) an honest node (*V*), if *U* and *V* are (direct) neighbors,

1.2.2) an honest node (*V*), if *U* and *V* are pseudo neighbors,

1.1.3) any of the adversarial nodes, otherwise.

1.2.) the sum of costs over honest nodes on the route is less than or equal *c*

Any setting of a new entry in a routing table by an honest node *C*, is preceded by a corresponding run of the protocol, with the participation of node *C*.

Routes with a lower cost are preferred. It is, therefore, natural to assume that the adversary wants to make routes appearing less costly, than they are. This means that if node believes that there exists a route between itself and target with a cost c, while in reality, there exist only routes between them with a cost higher than c, then the system should certainly be considered to be in an incorrect state.

In case of SR protocols we wanted to find an existing route. Here, in case of RDVR protocols we want also find existing routes, and additionally we want to label them with correct cost values.

Definition 4: Security requirement $\mathrm{Re}q_{u_i}$ for an ideal honest RDVR intermediate protocol machine participating in a session by processing the protocol message

The ideal honest machine sets the corresponding routing entry:

1.) ending node id is set to the true identity of the node launching the session phase (RREQ phase, RREP phase),

2.) next hop id is set to sender id from whom the input has been received,

where the sender id is the true identity of the sender if it is an honest machine, otherwise it is one from the set of the identifiers of the adversarial machines.

3.) cost of usage of a honest machine in a session cannot be influenced by the adversary

□

Note, in implemented protocols the typical components of cost comes from transmission delay and computational complexity.

Theorem 2: Global Security requirement $\mathrm{Re}q$ of Definition 3. is met if and only if local security requirement $\mathrm{Re}q_{u_i}$ of Definition 4. is fulfilled by any honest node u_i, positioned on the discovered route.

Proof of Theorem 2:

$\mathrm{Re}q_{u_i} \rightarrow \mathrm{Re}q$:

Assume an honest node *C* takes part in a session and sets the routing entry according to requirement $\mathrm{Re}q_{u_i}$. If this entry is (*C*,*E*,*F*,*c*), then *E* is the identifier of the next hop node towards ending node with identifier *F* , which entry is set according to $\mathrm{Re}q_{u_i}$.

Requirements $\mathrm{Re}q$ of Definition 3 are met:

$\mathrm{Re}q$ - 1.) There exists a route starting at node *C* and ending at node *F*:

by $\mathrm{Re}q_{u_i}$ -1.) the ending node id is set to the true identity of node launching the session phase, therefore there must exist a route via which such an identity information has arrived to node *C*.

$\mathrm{Re}q$- 1.1.) According to $\mathrm{Re}q_{u_i}$ -2.) honest nodes set next hop identifier such that it meets requirement $\mathrm{Re}q$- 1.1.)

$\mathrm{Re}q$- 1.2.) According to $\mathrm{Re}q_{u_i}$ -3.) costs over usage of honest nodes in a session cannot be influenced by the adversary, therefore the total cost of a route over all honest nodes cannot be adjusted by the adversary to a value below the sum of component costs

$\mathrm{Re}q \rightarrow \mathrm{Re}q_{u_i}$:
Consider an honest node *C*. According to $\mathrm{Re}q$ it sets correct routing entries. An honest node *C* sets its routing table only when it takes part in a corresponding session. Assume node *C* sets a routing entry, say entry (*C,E,F,c*), which is correct by $\mathrm{Re}q$. This means that in the corresponding session a route has been built through node *C* (or starts with node *C*) to the ending node *F* with next hop node *E*, where *E* is the identifier of a (direct or pseudo) neighbor honest node or it is the identifier of one of the adversarial nodes. Therefore, E is the true identity if the message is received from an honest nodes, otherwise it is one of the adversarial nodes.
If the cost of an honest node could be manipulated by the adversary, then the cost over a route could also be manipulated.
■

Example 2.
SAODV protocol [25] is an example of flawed RDVR protocol, where the security requirement $\mathrm{Re}q$ is not met (Fig.7.). Though the protocol uses a tricky one way hash chain mechanism to calculate the route cost measured in hop distance. It fails to meet the goal, because an adversarial node as an intermediate node in a session may decide not to increase the hop count (this attack was published by the authors of the protocol). Furthermore, there is no security mechanism in the protocol, which could help honest nodes to verify the true identity of a honest neighbor node, i.e. $\mathrm{Re}q_{u_i}$ -ii. is not met. Corresponding attack has been given in [5].

S → * : {rreq, S, D, id, MHC, TH, **sig**$_S$, HC, Hash=R}
A → * : {rreq, S, D, id, MHC, TH, **sig**$_S$, HC, h(Hash) }
B → * : {rreq, S, D, id, MHC, TH, **sig**$_S$, HC, h^2(Hash)}
D → B : {rrep, S, D, id, MHC, TH, **sig**$_D$, HC, h^3(Hash)}
B → A : {rreq, S, D, id, MHC, TH, **sig**$_D$, HC, h^4(Hash)}
A → S : {rreq, S, D, id, MHC, TH, **sig**$_D$, HC, h^5(Hash)}

Fig.7. An example for the operation and messages of SAODV
(id=session id, MHC=MaxHopCount, TH=TopHash, HC=HopCount, h(.)=hash function, R=random seed, sig$_x$ is calculated on all preceding fields except identifier x)

Recall, in the BPW approach hash function can only be modeled as random oracles (publicly available random functions), i.e. symbolic models of protocols using hash functions are not purely deterministic. Informally, the above attack exploits missing authentication of the protocol and has nothing to do with the hash chain, therefore this attack exists also in the BPW symbolic model with random oracle.
□

An implementation of $\mathrm{Re}q_{u_i}$ is a signature given on the session identification information (*m*) generated by *S* and *D* in the RREQ phase and the RREP phase, respectively, i.e. the core scheme is

$$(m)sign_C . \qquad (4)$$

Indeed, this is done in protocol ARAN [22] (Fig.8).

S → * : {rreq, D, $cert_S$, n, t, sig_S}
A → * : {rreq, D, $cert_S$, n, t, sig_S, sig_A, $cert_A$}
B → * : {rreq, D, $cert_S$, n, t, sig_S, sig_B, $cert_B$ }
D → B : {rrep, S, $cert_D$, n, t, sig_D }
B → A : {rrep, S, $cert_D$, n, t, sig_D, sig_B, $cert_B$}
A → S : {rrep, S, $cert_D$, n, t, sig_D, sig_A, $cert_A$}

Fig.8. An example for the operation and messages of ARAN. Nodes *A* and *B* are on the route from *S* to *D*. *n* and *t* stand for session nonce and time, respectively. $cert_X$ is the public key certificate of node *X*.

The general step of an intermediate node (say B) is the following: When B, receives the route request, then it verifies both signatures, and the freshness of the nonce. If the verification is successful, then B sets an entry in its routing table with S as target, and A as next hop. Then, B removes the certificate and the signature of A, signs the request, appends its own certificate to it, and rebroadcasts the resulted message. The operation is similar in the RREP phase.

Theorem 3: The protocol machine of ARAN in the symbolic model meets security requirement $\mathrm{Re}q_{u_i}$.

Proof of Theorem 3:
Honest node u_i sends a message to its output, if and only if, beforehand it has received an input message, which has not caused an abort event: the input message was syntactically correct and it arrived with an appropriate nonce value (in RREQ phase a fresh nonce, in RREP phase the nonce value received in RREQ phase).

The input message has arrived from an honest node or from the adversary. If it arrived from the adversary, then the outer signature of the message is an authentic signature produced with the use of the secret signing key of an adversarial node, because – in the symbolic model – the adversary is not able to produce a valid signature in the name of an honest node. Therefore, the next hop identifier is set to the true identifier of the honest node which is direct or pseudo neighbor, otherwise it is set to one of the adversarial nodes.

Similarly, the identifier of the target node is set to the identifier of the (honest) node, the secret signing key of which has been used to produce the inner signature in the input message. Once again this identifier cannot be false, because the adversary is not able to produce a fake signature in the symbolic model.

The adversary is not able to change transmission or processing speed of an honest machine, therefore the security againts route cost attacks is implied by the fulfillment of security requirements 2.-3.) by Definition 4.
■

5. Proprietary channels

A proprietary channel is a communication link for exclusive use of the adversary. A proprietary channel carries protocol messages during route acquistion only, later on these channels are not offered by the adversary for transmission of payload messages. It can be used to attack the route acquistion protocol: the adversary might be able to connect two honest intermediate nodes via a proprietary channel which are, otherwise not neighboring nodes (call them *artificial neighbors*), and this way produces a non-existing route accepted by the originator node.

Note, if nodes *B* and *C* are honest nodes, which are artificial neighbors, then there must exist two adversarial nodes *D* and *E* which are *real neighbors* to nodes *B* and *C*, respectively, i.e. they are along a route in the following order: *B* – *D* – (adversarial nodes)…proprietary channel…(adversarial nodes) -*E*-*C*.

Accordingly, we have to weaken the security requirement against a discovered route.

Definition 1': Security requirement $\mathrm{Re}q$ for SR protocol in case of existing proprietary channels:
For any pairs *S*, *D* of honest users if $v_1, v_2, ..., v_j$ are the honest nodes on the discovered route, then $v_1, v_2, ..., v_j$ are all the different honest nodes taking part in the session, in the given order, such that honest nodes, which are real neighbors become also neighbors on the *discovered route*.
□

I.e. honest nodes, which are not real neighbors, especially, which are even not nodes along any (real) route, may appear neighbors on the discovered route.

Definition 3': Security requirement $\mathrm{Re}q$ for RDVR protocol in case of existing proprietary channels:
The following points are changed compared to Definition 3:

1'.) there exists a maybe artificial route starting at node *C* and ending at node *F* via next hop *E* such that on this maybe artificial route (on an artificial route there exist at least one pair of artificial neighbors)

1.1.2’) an honest node (*V*), if *U* and *V* are pseudo neighbors, where intermediate adversarial nodes may be connected via proprietary channel.

□

Requirements against ideal honest protocol machine do not change (Definition 2. and Definition 4.).

6. Conclusion

In this paper, we presented a new framework for design and analysis of cryptographically sound secure on-demand source routing and on-demand dynamic vector distance routing protocols. This framework provides a structured and simplified technique, which applies two step idealization procedure. The first step follows the BPW approach, where the real crypto operations are abstracted by the composable cryptolibrary and the result is the protocol in the BPW symbolic model. In the second step symbolic honest protocol machines are abstracted into ideal honest protocol machines. We have shown examples for the application of the proposed technique. Composability seems a strong technique on the way to make the security analysis of protocols more transparent and structured.

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] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, 2002.

[23] I.Vajda: Cryptographically Sound Security Proof for On-Demand Source Routing Protocol EndairA. *Cryptology ePrint Archive Report 2011/103.* http://eprint.iacr.org/2011/103.pdf

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

[25] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2002.