Formal Verification of Key Establishment for Dual Connectivity in Small Cell LTE Networks

Vassilios G. Vassilakis*, Ioannis D. Moscholios†, Michael D. Logothetis‡, Michael N. Koukias‡
* Dept. of Computer Science, University of York, York, United Kingdom
† Dept. of Informatics & Telecommunications, University of Peloponnese, Tripolis, Greece
‡ Dept. of Electrical & Computer Engineering, University of Patras, Patras, Greece
e-mail: vv573@york.ac.uk, idm@uop.gr, {mlogo,mkoukias}@upatras.gr

***Abstract*—Dual connectivity (DC) has been included in the Release 12 of the long-term evolution (LTE) standard. In this paper, we perform a formal security verification of the key establishment protocol for DC in small cell LTE networks. In particular, the security verification is performed using a popular tool called Scyther. The considered security properties include secrecy and reachability. We also simulate a key leakage and show that some security claims in this case can be falsified.**

Keywords–Dual connectivity; long-term evolution (LTE); key establishment; Scyther tool.

I. INTRODUCTION

The exponential growth in mobile data traffic in the last years necessitates the development of novel applications and equipment to satisfy customer needs [1]. Standards bodies, such as the 3rd generation partnership project (3GPP), are working towards defining and enhancing the specifications for mobile communications. The current mobile broadband technology developed by 3GPP is known as long-term evolution (LTE) and was initially specified in 3GPP's Release 8 document series.

Since Release 12 [2], LTE provides enhanced support for small cells, such as dual connectivity (DC) and carrier aggregation. The main motivation for these enhancements is to support dense small cell deployments by reducing the mobility signaling and improving user data rates by using macro and small cells together. In this context, heterogeneous networks (HetNets) [3] are considered as a promising approach to enhance network capacity. In a HetNet, small cells typically provide improved capacity in hot spots, whereas macro cells are responsible for reliable wide-area coverage and fast moving user equipments (UEs). A macro base station in the LTE terminology is known as the evolved Node B (eNB).

According to the DC framework, the control plane, which transmits system information and handles user connectivity, is spit from the user plane, which transmits user data [4]. This split provides greater flexibility in terms of network management and administration, since the small cells can be used to operate in the user plane only, while the macro cells operate in both the control plane and the user plane by providing additional quality-of-service (QoS) support [5]. The DC feature of LTE, essentially, allows a UE to be connected to two cells simultaneously.

In this paper, we perform a formal security verification of the key establishment protocol for DC in small cell LTE networks [6]. In particular, the security verification is performed using a popular tool called Scyther [7]. The considered security properties include secrecy, agreement, and key freshness. Our considered model for the key establishment is in line with [8].

The rest of the paper is structured as follows. In Section II, we describe our considered model for DC in small cell LTE networks and introduce the necessary notation. In Section III, we describe the considered key establishment protocol for DC. In Section IV, we briefly introduce the Scyther's input language. In Section V, we describe our Scyther implementation of the key establishment protocol. In Section VI, we present our security verification results for the protocol. Finally, in Section VII, we conclude and discuss possible future directions. Also, in Table I, we present the list of abbreviations used in this paper.

II. DUAL CONNECTIVITY IN LTE NETWORKS

In this section we describe our considered model for DC in small cell LTE networks.

A. Preliminaries

Consider a UE and two eNBs involved in DC. The first eNB is called a Master eNB (MeNB) and is responsible for maintaining the control of the radio resource management. The second eNB, called Secondary eNB (SeNB), is controlled by the MeNB and provides additional radio resources to the UE. In a realistic scenario the MeNB is typically a macro eNB, while the SeNB could be a small cell eNB.

All the radio control traffic between the UE and the MeNB is transported over the established signaling radio bearer (SRB). All the data traffic between the UE and the eNBs (both MeNB and SeNB) is transported over the established data radio bearers (DRBs) with each eNB. Each DRB has a unique identifier (ID). The ID of the i-th DRB between the UE and the MeNB is denoted by DRB_i^{mu}. Similarly, the ID of the j-th DRB between the UE and the SeNB is denoted by DRB_j^{su}.

From UE's perspective, one SRB is established with the MeNB and multiple DRBs can be established with the MeNB and the SeNB.

B. Trust Model

The considered trust model is as follows [8]:

- A secure channel between the MeNB and the SeNB.
- A secure channel between the MeNB and the UE.
- The channel between the UE and the SeNB is insecure. This introduces a particular challenge for the selection of an appropriate key establishment mechanism.
- The MeNB, the SeNB, and the UE cannot be compromised.

 ISBN: 978-1-61208-596-8

TABLE I. LIST OF ABBREVIATIONS

3GPP	3rd Generation Partnership Project
AES	Advanced Encryption Standard
DC	Dual Connectivity
DL	Downlink
DRB	Data Radio Bearer
eNB	evolved Node B
HetNet	Heterogeneous Network
KDF	Key Derivation Function
ID	Identifier
LTE	Long Term Evolution
MeNB	Master eNB
QoS	Quality of Service
SCC	Small Cell Counter
SeNB	Secondary eNB
SRB	Signaling Radio Bearer
UE	User Equipment
UL	Uplink
UP	User Plane

In particular, the aforementioned secure channels provide *encryption*, *integrity*, and protection against *replay attacks*.

C. Security Parameters

The security parameters of the considered model are as follows [8]:

- $Algs$: A list IDs of encryption algorithms supported by the UE and shared with the MeNB.
- SCC: The small cell counter (SCC) that is used to keep track of the established DRBs.
- $KDF()$: A key derivation function (KDF).
- K_{MeNB}: The secret key shared between the UE and the MeNB. This is used to derive other keys.
- K_{SeNB}: The secret key shared between the UE and the SeNB. The UE can derive this key using formula (1), discussed below. The SeNB receives this key from the MeNB, which in turn derives this key using formula (1).
- $K_{UPenc,j}$: The secret key shared between the UE and the SeNB for DRB_j^{su}. This key is used to encrypt the user plane (UP) traffic. That is, the data traffic between the UE and the SeNB for the j-th DRB. The UE and the SeNB can derive this key using formula (2).

D. Adversary Model

For the adversary, we adopt the widely used Dolev-Yao model [9]. According to this model, the adversary is able to intercept all messages. They are also able to forward, drop, or replay old messages. However, the adversary is not able to decrypt messages, unless they are in possession of the required keys.

III. KEY ESTABLISHMENT PROTOCOL

A. First Data Radio Bearer (DRB)

Below we describe the key establishment process for the first DRB between the UE and SeNB. At the end of the process, the UE and the SeNB are in the possession of the shared key $K_{UPenc,1}$, which will be used to encrypt the traffic in the first DRB.

- Step 1: The MeNB generates the key K_{SeNB} based on (1).
- Step 2: The MeNB sends to the SeNB a message containing: K_{SeNB}, $Algs$, and DRB_1^{su}.
- Step 3: The SeNB establishes the first DRB with the UE.
- Step 4: The SeNB selects some encryption algorithm ID $a \in Algs$ and derives the key $K_{UPenc,1}$ via (2), where $j = 1$.
- Step 5: The SeNB sends the selected algorithm ID, a, to the MeNB.
- Step 6: The MeNB sends to the UE a message containing: SCC, DRB_1^{su}, and a.
- Step 7: The UE derives the key $K_{UPenc,1}$ via (2), where $j = 1$.

$$K_{SeNB} = KDF(K_{MeNB,SCC}) \tag{1}$$

where SCC is the small cell counter (SCC) that is used to keep track of the used DRBs.

$$K_{UPenc,j} = KDF(K_{SeNB}, DRB_j^{su}, a) \tag{2}$$

B. Subsequent Data Radio Bearers (DRBs)

Having described the key establishment process for the first DRB, below we describe the key establishment for subsequent DRBs between the UE and SeNB. At the end of the process, the UE and the SeNB are in the possession of the shared key $K_{UPenc,j}$ ($j > 1$). At this stage the SeNB is already in possession of the key K_{SeNB} and has selected some encryption algorithm a.

- Step 1: The MeNB sends to the SeNB the ID of the j-th DRB, DRB_j^{su}, where $j > 1$.
- Step 2: The SeNB establishes the j-th DRB with the UE.
- Step 3: The SeNB derives the key $K_{UPenc,j}$ via (2).
- Step 4: The MeNB sends to the UE the ID of the j-th DRB, DRB_j^{su}.
- Step 5: The UE derives the key $K_{UPenc,j}$ via (2).

IV. SCYTHER'S LANGUAGE

Scyther is a widely used formal verification tool and has been designed for the automatic verification of security protocols. In this section, we briefly describe the Scyther' input language. Additional language features are also introduced and explained later, in Section V, which describes the protocol implementation. Scyther's language is loosely based on C/Java-like syntax [7]. At the most basic level, Scyther manipulates the *terms*. Terms could be *atomic* or *complex*. An atomic term could be any identifier, for example a string of alphanumeric characters that represents user data. Atomic terms can be combined into more complex terms by operators, such as pairing or encryption. This could refer, for example, to encrypted user data. Encryption of a term m with a key k is denoted in Scyther as $\{m\}k$.

The main purpose of the language is to describe protocols which are defined by a set of roles. A role could represent, for example, a network node, such as eNB or UE. Roles, in turn are defined by a sequence of events, such as sending/receiving terms (e.g., signaling messages or user data) and security

 ISBN: 978-1-61208-596-8

claims. In particular, message sending and receiving can be specified by the pair $send\_i(S, R, m)$ and $recv\_i(S, R, m)$, where i denotes the i-th message, S is the sender, R is the receiver, and m is the message. Claims are used to specify security requirements, such as message secrecy and node aliveness. For example, $claim(X, Alive)$ means that role X claims to be alive, and $claim(Y, Secret, m)$ means that role Y claims that the message m must be unknown to an adversary.

Scyther has a predefined symmetric key infrastructure: $k(X, Y)$ denotes the long-term symmetric key shared between the roles X and Y. Also, Scyther has a predefined adversary model, which is based on the Delev-Yao model, mentioned in Section II. This greatly simplifies the implementation task, since there is no need to implement the adversary model from scratch.

V. Protocol Implementation using Scyther

In this section, we describe the Scyther implementation of the key establishment protocol of Section III. In the interest of space, we only present the implementation for the first DRB. The key establishment for subsequent DRBs is implemented in a similar way.

A. Initial Definitions

In Figure 1, we present the initial type definitions and term declarations. As we observe, the KDF has been defined as a hash function (line 2). In Scyther, this is done by using the keyword *hashfunction*. Next, in lines 3-6 we define four types: ALG, SCC, DRB, and DATA. A user-defined type can be specified in Scyther using the keyword *usertype*. The names of the these types are self-explanatory. For example, the ALG type will be used to represent various encryption algorithms.

Having specified the user-defined types, in lines 8-10 we declare various terms that will be used by the protocol. The keyword *const* specifies terms whose value remains constant throughout the whole protocol verification process. For example, the constants $a1$ and $a2$ have been declared of ALG type and will refer to different encryption algorithms supported by the UE, such as the SNOW-3G and the advanced encryption standard (AES).

```
1   # Type Definitions
2 hashfunction KDF; # key derivation function
3 usertype ALG; # encryption algorithm
4 usertype SCC; # small cell counter
5 usertype DRB; # data radio bearer
6 usertype DATA; # user data
7   # Term Definitions
8 const a1, a2: ALG;
9 const scc: SCC;
10 const drb1, drb2: DRB;
```

Figure 1. Initial Definitions.

```
11 # Protocol Specification: Key Establishment for Dual Connectivity
12 protocol DC−LTE(MeNB,SeNB,UE){
13   # MeNB role specification
14   role MeNB {
15   macro kmenb=k(MeNB,UE); #shared key between MeNB and UE
16   fresh n1: Nonce;
17   macro ksenb=KDF(kmenb,scc,n1);
18   send_1(MeNB,SeNB,{ksenb,(a1,a2),drb1}k(MeNB,SeNB));
19   claim(MeNB,Secret,ksenb);
20   var a: ALG;
21   recv_2(SeNB,MeNB,{a}k(MeNB,UE));
22   send_3(MeNB,UE,{scc,a,drb1}mkenb);
23   claim(MeNB,Reachable);
24   }
```

Figure 2. The MeNB Role Implementation in Scyther.

B. The MeNB Role

In Figure 2, we begin specifying the protocol and start with the MeNB role. A protocol definition (line 12) specifies the protocol name and takes as a parameter a sequence of roles, which are then defined in the protocol's body. The MeNB role definition is shown in lines 14-24, starting with the role's name in line 14. In line 15, we assign the value of the shared key K_{MeNB} to the term $kmenb$. Recall that the key K_{MeNB} is shared between the MeNB and the UE. Hence, it is predefined in Scyther as $k(MeNB, UE)$. The keyword *macro* is used to define abbreviations for particular terms. In line 16, we define a new term $n1$ of type *Nonce*. Nonce is a predefined type and is used to define terms that must be used only once. The keyword *fresh* is used to declare a *freshly generated* term of arbitrary/random value. This means that different instances of the same role will generate different values. In line 17, the term $ksenb$, which refers to the key K_{SeNB}, is assigned its value according to (1). Note that the value assignment in line 17 is also based on $n1$. This is to ensure that different MeNB instances will generate different keys K_{SeNB}. Line 18 declares the first *send* event. The MeNB sends to the SeNB an encrypted message that contains the key K_{SeNB}, a list of encryption algorithms supported by the UE, and the ID of the DRB. The corresponding *recv* event appears in line 28 of Figure 3, which defines the SeNB role and will be explained in Subsection V-C. Returning back to Figure 2, line 19 declares the first *claim* event. The MeNB claims that the key K_{SeNB} must be unknown to an adversary. In line 20, using the keyword *var*, we declare the term a as a variable of type *Nonce*. Variables are used to store received terms. In this case, the variable a will be used to store the encryption algorithm ID received from $recv_2$ in line 21. The corresponding $send_2$ event is defined in line 31 of Figure 3. Having received from the SeNB the selected algorithm ID, the MeNB forwards it to the UE together with other required parameters, such as the SCC and the ID of the DRB (line 22). These parameters will be used by the UE to determine the key $K_{UPenc,j}$, based on (2), with $j = 1$, which corresponds to the first DRB. Finally, line 23 declares the *claim* event using the keyword *Reachable*. This is used to check whether this claim can be reached at all. It returns true if and only if there exists a trace in which this claim occurs.

C. The SeNB Role

The specification of the SeNB role is provided in Figure 3. Having provided a detailed explanation for the specification of the MeNB role in Subsection V-B, most of the code in Figure 3 should now be self-explanatory. Hence, below we explain only some selected parts of the code. In line 32 we define a term $dataDL$ of DATA type. This term represents the downlink (DL) data that will be sent to the UE. As shown in line 34, $dataDL$ is encrypted using the key $K_{UPenc,j}$, with $j = 1$. Similarly, the term $dataUL$ will be used to store the received

```
25     # SeNB role specification
26   role SeNB {
27     var n1: Nonce;
28     recv_1(MeNB,SeNB,{ksenb,(a1,a2),drb1}k(MeNB,SeNB));
29     claim(SeNB,Secret,ksenb);
30     macro kupenc1=KDF(ksenb,drb1,a1);
31     send_2(SeNB,MeNB,{a1}k(MeNB,SeNB));
32     fresh dataDL: DATA;
33     fresh n2: Nonce;
34     send_4(SeNB,UE,{dataDL,n1}kupenc1);
35     claim(SeNB,Secret,dataDL);
36     var dataUL: DATA;
37     var n3: Nonce;
38     recv_5(UE,SeNB,{dataUL,n3)kupenc1);
39     claim(SeNB,Secret,dataUL);
40     claim(SeNB,Reachable);
41   }
```

Figure 3. The SeNB Role Implementation in Scyther.

```
42     # UE role specification
43   role UE {
44     var a: ALG;
45     var n1, n2: Nonce;
46     recv_3(MeNB,UE,{scc,a,drb1}k(MeNB,UE));
47     var dataDL: DATA;
48     recv_4(SeNB,UE,{dataDL,n1}kupenc1);
49     claim(UE,Secret,kupenc1);
50     claim(UE,Secret,dataDL);
51     fresh n3: Nonce;
52     fresh dataUL: DATA;
53     send_5(UE,SeNB,{dataUL,n3}kupenc1);
54     claim(UE,Secret,dataUL);
55     claim(UE,Reachable);
56   }# end of UE tole specification
57 } #end of protocol specification
```

Figure 4. The UE Role Implementation in Scyther.

uplink (UL) data from the UE (line 38). In lines 35 and 39, the SeNB claims that these DL and UL data will remain secret (i.e., unknown to the adversary).

D. The UE Role

The specification of the UE role is provided in Figure 4 and should be self-explanatory.

VI. Security Verification

In this section, we present the security verification results for the key establishment protocol. The verification is based on the Scyther implementation of Section V. In particular, the purpose of the verification is to verify whether the claims defined in various role specifications are true. In the Result Window, shown in Figure 6, Scyther outputs a single line for each claim. The first column shows the protocol in which the claim occurs, the second column shows the role, etc.

We observe that for each claim the verification process returns its status (i.e., whether the claim has been verified), some claim-specific comments, and the identified trace patterns (if applicable). For all the secrecy claims (lines 19, 29, 35, 39, 49, 50, and 54) we observe that the status is *OK* and *Verified*, i.e., no attacks have been found where an adversary gains knowledge of the confidential information. If a claim is false, the corresponding status message will be *Fail*.

```
30     ...
31   macro kupenc1=KDF(ksenb,drb1,a1);
32   send_!(SeNB,SeNB,kupenc1);
33     ...
```

Figure 5. Leaking the key $K_{UPenc,1}$ to the adversary.

For all the reachability claims (lines 23, 40, and 55) we observe that the status is also *Verified* and the Comments column says *At least 1 trace pattern*. Furthermore, by clicking the *1 trace pattern* button in the Patterns column, it is possible to see a trace identified Scyther.

In the following sections we simulate a key leakage and perform the Scyther verification process. Assume that the key $K_{UPenc,1}$ is leaked to the adversary by the SeNB. This can be simulated in Scyther using a *send* event with an exclamation mark, as follows: $send\_!(X, X, m)$, where X is the role from which the leak occurs and m is the leaked message. Hence, the required code modification of Figure 2 includes adding a new *send* event, as shown in line 32 of Figure 5.

The new verification results are shown in Figure 7. We observe that the status of five claim events changed from *OK* to *Fail*. In particular, the failed claims are: one for the actually leaked key and the other four for the compromised DL/UL data, as a result of key leakage. For each failed claim the Comments column says *At least 1 attack*. By pressing the corresponding button in the Patterns column, we can view one of the possible attack graphs for $dataDL$, which is shown in Figure 8.

In Figure 8, we see one instantiation, Bob, of the SeNB role (denoted as Run #1) and one instantiation, Charlie, of the MeNB role (denoted as Run #2). There is also Alice in the UE role, but Alice is not involved in the particular attack. The boxes represent creation of a run (i.e., an instance of a protocol role), communication events of a run, and claim events. The arrows represent ordering constraints. According to lines 32 and 33 of Figure 3, Bob generates two fresh terms: $dataDL$ and $n2$. Bob's run number is appended to his terms, in order to distinguish them from any homonymous terms of other runs. Hence, these terms appear in the graph as $dataDL\#1$ and $n2\#1$. Bob also generates a variable $n1$ (line 27), which will receive its value from $n1\#2$, generated by Charlie. After receiving ($recv\_1$) the first message from Charlie, Bob generates the key $K_{UPenc,1}$ (not shown in the graph) and leaks ($send\_!$) the key to the adversary, which is represented by an orange oval. When later Bob sends the message $dataDL$ to Alice ($send\_4$), the adversary intercepts this message and decrypts it. Hence, the secrecy claim has been falsified (black rectangular box in Figure 8).

VII. Conclusion and Future Work

In this paper, we perform a formal security verification of the key establishment protocol for dual connectivity in small cell LTE networks. In particular, the key establishment protocol has been implemented and verified using the Scyther tool. The protocol verification includes security properties, such as secrecy and reachability. Although all the security claims have been verified, further protocol analysis is required to identify potential attacks. In our future work, we intend to perform additional security analysis using other popular formal verification tools, such as Tamarin [10] and ProVerif [11].

 ISBN: 978-1-61208-596-8

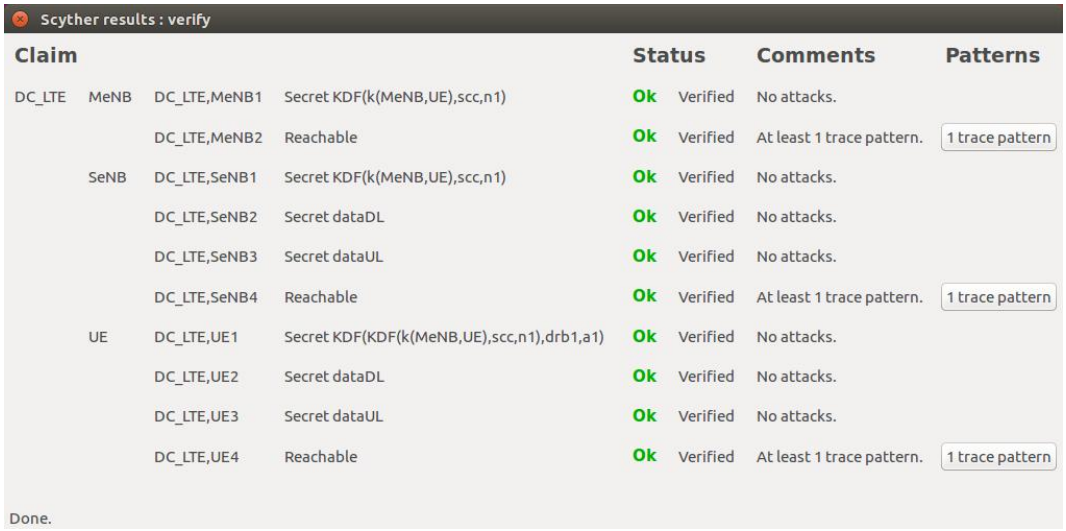

Figure 6. Security verification results using Scyther: Key establishment protocol.

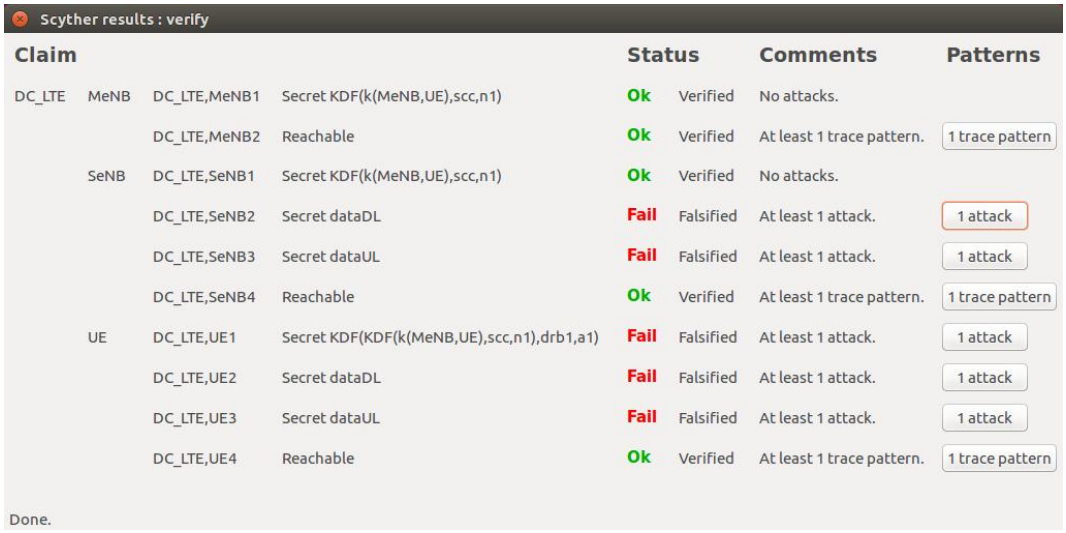

Figure 7. Security verification results using Scyther: Simulating key leakage.

REFERENCES

[1] Cisco, Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper, Feb. 2017.

[2] D. Astely, E. Dahlman, G. Fodor, S. Parkvall, and J. Sachs, "LTE release 12 and beyond," IEEE Commun. Mag., vol. 51, no. 7, July 2013, pp. 154-160.

[3] J. G. Andrews, "Seven ways that HetNets are a cellular paradigm shift," IEEE Commun. Mag., vol. 51, no. 3, March 2013, pp. 136-144.

[4] A. Zakrzewska, D. López-Pérez, S. Kucera, and H. Claussen, "Dual connectivity in LTE HetNets with split control-and user-plane," Proc. IEEE Globecom Workshops, Atlanta, USA, Dec. 2013, pp. 391-396.

[5] V. G. Vassilakis, I. D. Moscholios, A. Bontozoglou, and M. D. Logothetis, "Mobility-aware QoS assurance in software-defined radio access networks: An analytical study," Proc. 1st IEEE Conference on Network Softwarization (NetSoft), London, UK, April 2015, pp. 1-6.

[6] S. C. Jha, K. Sivanesan, R. Vannithamby, and A. T. Koc, "Dual connectivity in LTE small cell networks," Proc. IEEE Globecom Workshops (GC Wkshps), Austin, USA, Dec. 2014, pp. 1205-1210.

[7] C. J. F. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," Proc. International Conference on Computer Aided Verification, Princeton (CAV), USA, July 2008, pp. 414-418.

[8] N. B. Henda, K. Norrman, and K. Pfeffer, "Formal verification of the security for dual connectivity in LTE," Proc. 3rd FME Workshop on Formal Methods in Software Engineering, Florence, Italy, May 2015, pp. 13-19.

[9] D. Dolev and A. Yao, "On the security of public key protocols," IEEE Transactions on Information Theory, vol. 29, no. 2, March 1983, pp. 198-208.

[10] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," Proc. 25th International Conference on Computer Aided Verification (CAV), Saint Petersburg, Russia, July 2013, pp. 696-701.

[11] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," Proc. 14th Computer Security Foundations Workshop (CSFW), Cape Breton, Canada, June 2001, pp. 82-96.

 ISBN: 978-1-61208-596-8

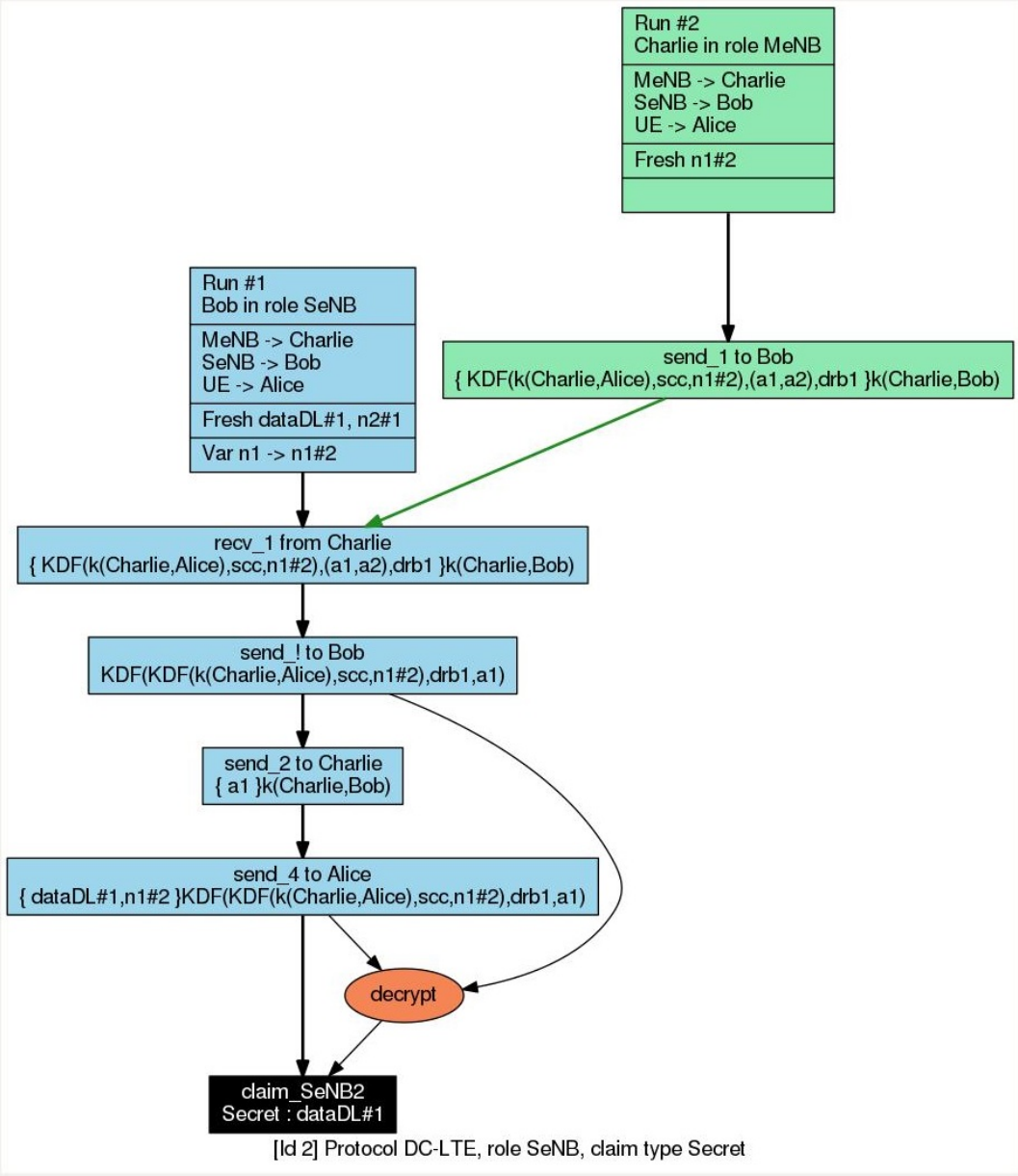

Figure 8. Security verification results using Scyther: Attacking the secrecy of downlink data.