Formal Verification of the Security for Dual Connectivity in LTE

Noomene Ben Henda, Karl Norrman and Katharina Pfeffer*
Ericsson Research, Färögatan 6 16480, Stockholm, SWEDEN
Email: {noamen.ben.henda, karl.norrman, katharina.pfeffer}@ericsson.com

***Abstract*—We describe our experiences from using formal verification tools during the standardization process of Dual Connectivity, a new feature in LTE developed by 3GPP. To the best of our knowledge, this is the first report of its kind in the telecom industry. We present a model for key establishment of this feature and provide a detailed account on its formal analysis using three popular academic tools in order to automatically prove the security properties of secrecy, agreement and key freshness. The main purpose of using the tools during standardization is to evaluate their suitability for modelling a rapidly changing system as it is developed and in the same time raising the assurance level before the system is deployed.**

I. Introduction

a) Background: Research on formal verification of security protocols has been ongoing for two decades. State of the art academic tools like ProVerif [1] can verify protocols for unbounded number of runs and communicating peers. However, in many places where new security protocols are continuously designed, e.g., the telecom and internet standardization industry, the use of formal methods has not yet caught on. One possible reason is the fact that existing tools require security protocols to be modeled at a too high abstraction level. However in practise, it is often the case that the specifications of security features are deeply intertwined with low level details of the system. Another possible reason could be that during a standardization process, the protocol under study is constantly evolving. In fact it is common that changes and additional features are proposed, evaluated and possibly adopted during the process. In this paper, we investigate this on an industrial case study.

b) Contribution: We consider Dual Connectivity (DC) [10] (TS 33.401) for a terminal, a feature recently introduced by the 3rd Generation Partnership Project (3GPP) in the Long Term Evolution (LTE) standards. This feature allows a terminal such as a mobile phone to be connected to two base stations simultaneously. We present a model for the key establishment of DC and provide a detailed account on our experience using three academic tools for the formal analysis of the model. This account includes insights on the modeling and the attempts to adapt to the input languages, the results of the verification and our evaluation of the tools with respect to different parameters such as expressiveness, usability and performance. The tools we consider are Scyther [2], Tamarin [3] and ProVerif. To the best of our knowledge there has been no reports, so far, on the use of formal verification tools to support ongoing standardization process in the telecom industry.

First, we give an overview on security in LTE networks.

c) LTE *Security:* LTE is the most recent standard developed by 3GPP for the 4th Generation (4G) mobile communication systems. Among the objectives of LTE is to provide at least as good security as previous generations. For this purpose, LTE introduces a key hierarchy and mandates the use of specific keys from the hierarchy for each protocol between the terminal and the nodes in the network. At the root of the hierarchy is the key provided by the operator and securely stored in a smart-card. Each key in the hierarchy is shared between the terminal and a particular node. For example, the terminal and the base station serving as an access point to the terminal share a key called K_{eNB}. eNB stands for evolved NodeB and denotes base stations in LTE terminology. Keys like the K_{eNB} are typically used to derive ciphering and integrity protection keys in order to secure the communication between the terminal and a node in the network.

The LTE standards describe procedures for the establishment of the keys in the hierarchy. It is common that these procedures share the following pattern: Starting from an already established key, for example a K_{eNB}, the procedures establish a new key between the terminal and a target node, for example a new eNB during a handover. Such procedures typically have to satisfy at least the following security properties: *agreement*, *secrecy* and *freshness*. Agreement guarantees that the involved peers obtain the same key at the end of the run. Secrecy guarantees that no one, other than the involved peers, obtains the key. Finally freshness prevents key re-use.

d) Related work: Some recent work such as [4] and [5] is promoting the use of formal methods during the security standardization of communication systems. Most published work on formal analysis of telecom protocols is performed after the standardization and when the equipment has already been built and deployed. This is the case for the Authentication and Key Agreement (AKA) protocol [6]–[8]. All of that work focus on key establishment, whereas we include aspects of the security of the system related to the use of the established keys as well. The only public use of formal methods during telecom standardization that we are aware of is the work done by the USECA [9] project on WCDMA [10] (TR 33.902) where the BAN logic [11] is used without tool support to prove some

*Katharina was a master thesis worker from KTH during this work. The security for DC was designed and specified in 3GPP working group SA3 during the spring of 2014. While we actively took part in the standardization work in SA3, this study was conducted as a back-office activity.

properties of AKA.

There are several case studies using the tools we consider. The Scyther tool has, for example, been used for the analysis of the Naxos protocol [12], the IPsec exchange protocols IKEv1 and IKEv2 [13], some security features in WiMAX [14] and [15]. Case studies using the ProVerif tool include the analysis of the Bluetooth device pairing protocol [16], the just-fast-keying protocol [17], the secure file sharing protocol in Plutus file systems [18], authentication in 3G where both GSM and WCDMA access is used [7], some of the LTE security procedures [19], and a privacy study on WCDMA [20]. The Tamarin tool has been used for the verification of group key agreement protocols [21], as well as TLS, TESLA and other protocols [3].

The work in [22] provides a comparison of several tools including also Scyther and ProVerif, in addition to the AVISPA tool suite [23]. This study focuses on performance and particularly on state space coverage. Furthermore, it relies on purely academic protocol examples such as the Needham-Schroeder protocol.

II. Dual Connectivity

There are two architectural options for DC, but in the following we will only consider one of them since the other does not require any new security features.

There are two eNBs involved in DC. One of them maintains control of the radio resource management, and is called a Master eNB (MeNB). The MeNB controls the other eNB, which is called the Secondary eNB (SeNB). According to the trust model, the MeNB and SeNB communicate over a *secure channel*; that is, communication between the two is integrity protected, replay protected and encrypted. This is also the case for the communication between the MeNB and the terminal. Furthermore, the trust model assumes that none of the entities can be compromised.

The security parameters in the MeNB include the $\mathrm{K_{eNB}}$ shared with the terminal and the list of encryption algorithms identifiers supported by the terminal. The $\mathrm{K_{eNB}}$ is the root of key hierarchy for DC. In case it is compromised, all keys derived from it, including any future keys for offload, would also be compromised. This motivates why the trust model assumes that the MeNB nodes cannot be compromised.

In general, all radio control traffic between a terminal and an eNB is transported over a *signalling radio bearer*, which is transported over a secure channel. Data traffic is transported over *Data Radio Bearers (*DRB*)*. Each such radio bearer has a unique identifier. In DC, the terminal maintains one radio control connection with the MeNB, but can have multiple data bearers established with both the MeNB and SeNB. Regardless of whether they are signalling or data radio bearers, the same encryption key is used for all of them. To ensure that the traffic on each bearer is encrypted using a unique key stream, the identifier of the radio bearer is included in the initialization vector of the encryption algorithm.

In order to keep track of the used bearers, the MeNB maintains a counter called Small Cell Counter (SCC). When the MeNB offloads a first DRB between the SeNB and the terminal, the MeNB derives a key called S-$\mathrm{K_{eNB}}$, from the $\mathrm{K_{eNB}}$ and the SCC. All keys are generated using a key derivation function denoted by KDF. The MeNB then sends to the SeNB a message containing the newly derived S-$\mathrm{K_{eNB}}$, the identifier of the DRB to be used, and the list of identifiers for the encryption algorithms that the terminal supports (step 1 in Fig. 1). Upon reception of this message, the SeNB first configures the DRB and selects an algorithm from the list. Second, using the selected algorithm and the received S-$\mathrm{K_{eNB}}$, the SeNB derives the actual encryption key $\mathrm{K_{UPenc}}$. Last, the SeNB sends the selected algorithm identifier to the MeNB (step 2). The MeNB forwards the received algorithm identifier together with the SCC and the DRB identifier to the terminal (step 3). The terminal, now in possession of all the required parameters, derives the S-$\mathrm{K_{eNB}}$ and the encryption key $\mathrm{K_{UPenc}}$ in a similar manner to how it is done in the other nodes. Observe that in Fig. 1, we include the DRB identifier in the derivation of $\mathrm{K_{UPenc}}$. This is not the case in reality, since the identifier is supposed to be used in the initialization vector of the encryption function. On the other hand all these steps, i.e. the key derivation followed by the encryption, are performed internally by the same trusted entity and thus can be considered as a single atomic step. In this regard, our way of modeling is sound.

When the MeNB attempts to offload an additional DRB and there are unused DRB identifers, it does not derive a fresh S-$\mathrm{K_{eNB}}$. Instead, the MeNB offloads a new DRB to the SeNB without sending neither an S-$\mathrm{K_{eNB}}$ nor an SCC (step 5). When the SeNB does not receive any S-$\mathrm{K_{eNB}}$, it configures the new DRB using the same $\mathrm{K_{UPenc}}$ as all existing DRBs for this terminal. To the terminal, the MeNB sends only the new DRB identifier as well (step 6). The additional $\mathrm{K_{UPenc}}$ derivations in after setp 6 in Fig. 1 are due to our way of modeling the usage of the DRB identifiers as explained above. In absence of an SCC in step 6, the terminal re-uses the existing S-$\mathrm{K_{eNB}}$.

The set of DRB identifiers is finite and hence it is possible that the MeNB attempts to offload a DRB while all identifiers have previously been used. In this situation, the MeNB increments the SCC and derives a new S-$\mathrm{K_{eNB}}$ as if this was an initial offload. The only difference is that no algorithm identifier selection is required (step 8 and 9). In case the SeNB have already active DRBs for the terminal, the SeNB reconfigures all these DRBs to use an encryption key based on the new S-$\mathrm{K_{eNB}}$. Since the model is mainly capturing the key establishment it does not include subsequent data transmissions apart from what is needed to verify agreement (step 4, 7 and 10). Modeling further data transmissions on DRBs and offloads complicates the model without adding any substance. Therefore we don't include the reconfiguration of existing DRBs in the model of Fig. 1.

III. Model implementations

We consider the Scyther, Tamarin and ProVerif tools for the formal verification of DC. All of these tools can

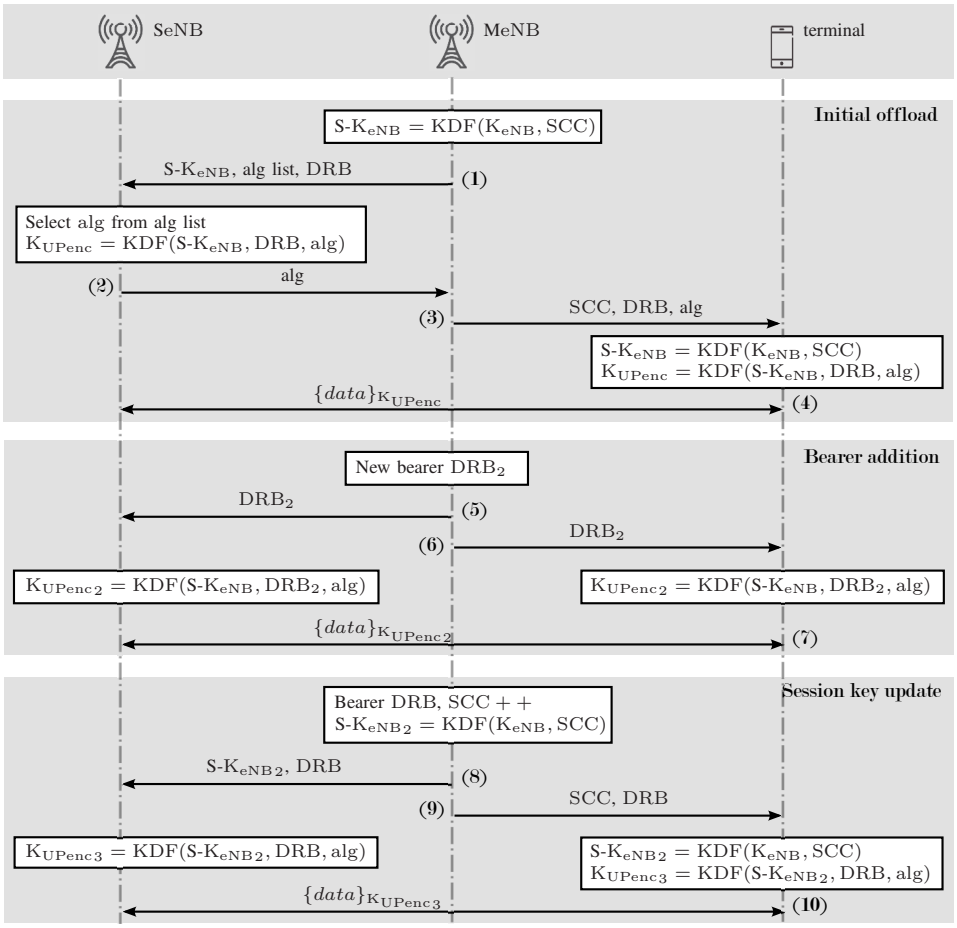

Fig. 1. Dual connectivity reference model.

achieve unbounded verification, i.e., they can prove the security properties for an unbounded number of agents and protocol runs. From an input language perspective, the tools support different level of abstraction where Scyther has the highest one and Tamarin the lowest. The tools implement the symbolic Dolev-Yao attacker model [24], with extensions such as agent compromise and secure channels. The problem is in general undecidable [25] so the tools may not terminate, return false attacks or terminate with inconclusive results.

In the following sections we describe how we implement the DC reference model of Fig. 1 using each tool. The goal is to automatically verify the secrecy and freshness of the K_{UPenc} key, and the agreement between the terminal and the SeNB on the selected encryption algorithm and K_{UPenc}. For shortage of space, the full versions of the models that can be used to reproduce our results are not included. They are available on demand.

A. Scyther

Scyther uses a backward search algorithm based on a symbolic representation of sets of protocol runs [26], an extension of the strand spaces [27]. A Scyther protocol model consists of a set of roles where each role is a list of events. There can be communication events (send and receive) and match events. Pattern matching is used in all events to enforce the structure and types of the arguments. The tool provides built-in types for modeling usual entities such as agents, nonces and keys. It has also support for user-defined ones. Cryptographic primitives are implemented by symbolic functions with predefined public and symmetric key infrastructures. Security properties are expressed by claim events within the roles.

Listing 1 includes all the declarations of the Scyther model.

```
1 hashfunction kdf;
2 usertype Alg;const alg-1, alg-2: Alg;
3 usertype Counter;const scc-1: Counter;const inc: Function;
4 usertype Bearer;const drb-1, drb-2: Bearer;
```

Listing 1. The declarations of the Scyther model

Most of the keywords are self-explanatory. Declarations preceded by **const** are public and thus known to the attacker. Since there is no support for complex data structure such as sets or lists, we assume a fixed set of predefined DRB identifiers listed in line 4. The terminal cryptographic capabilities are modeled in a similar manner (line 2). For the SCC, we use a constant to represent the initial value and a function to model the increment operation (line 6).

Listing 2 shows the MeNB role. The initial offload steps are implemented in lines 6-8. Lines 9-10 model one bearer

addition operation. In fact, because there is no support for control flow structures such as loops and conditionals, we unroll the bearer addition loop in the implementation. For that we assume a fixed number of possible identifiers, in our case two (`drb-1` and `drb-2`), and hence we only need to model one iteration of the loop since by then all available bearers would have been used. The remaining lines (12-13) are for the session key update steps. Observe that one can extend the role by alternating the bearer addition and key update steps resulting in an arbitrarily large model. For performance reasons, we consider this limited model as we could not get better results (termination) with larger ones.

```
1  protocol dc(MeNB, SeNB, UE)
2  {
3   role MeNB {
4    var a: Alg;
5    macro skenb-m-1 = kdf(k(MeNB, UE), scc-1);
6    send_1(MeNB, SeNB, {skenb-m-1, (alg-1, alg-2), drb-1}k(
         MeNB, SeNB));
7    recv_2(SeNB, MeNB, {a}k(MeNB, SeNB));
8    send_3(MeNB, UE, {scc-1, a, drb-1}k(MeNB, UE));
9    send_6(MeNB, SeNB, {drb-2}k(MeNB, SeNB));
10   send_7(MeNB, UE, {drb-2}k(MeNB, UE));
11   macro skenb-m-2 = kdf(k(MeNB, UE), inc(scc-1));
12   send_10(MeNB, SeNB, {skenb-m-2, drb-1}k(MeNB, SeNB));
13   send_11(MeNB, UE, {inc(scc-1), drb-1}k(MeNB, UE));
14   claim_MeNB1(MeNB, Reachable);
15  }
16  role SeNB { ...
```

Listing 2. The MeNB role in the Scyther model

Communication events take three arguments where the first two specify the involved roles and the third one is the actual message. For example the receive event of line 8 denotes that the MeNB receives from the SeNB a value of type `Alg` to be stored in the variable `a` encrypted with the shared key `k(MeNB,SeNB)`. Scyther does not include a primitive for secure channels. Encryption and integrity protection are supported in Scyther and thus can be easily added in the model. The replay protection is, however, more complex to model when counters cannot be used. Although it is possible to make a construction based on nonces, the required additional message exchange would most likely impede on the tool's performance. In our case only encryption is added to preserve at least confidentiality since otherwise, all the security properties would trivially fail. A direct consequence of the lack of replay protection is that Scyther fails to prove agreement and reports instead false attacks as one would expect.

For the algorithm selection step, the SeNB always replies with the same fixed value in the current model. This is because there is no support for choice. Nevertheless, it is possible to have several Scyther models for each possible algorithm value, but in this particular case they would be all equal up to the names of the algorithm constants. Our model is in fact independant of the names and therefore this is not needed.

Listing 3 shows the security claims in the end of the terminal role. The variable `kupenc-u-3` denotes the K_{UPenc} obtained in the session key update phase. Verification of agreement on the newly established key K_{UPenc} and the encryption algorithm, denoted by `a` in the model, requires an additional message exchange using this key (lines 4-5). The agreement claims (lines 3 and 6) are added accordingly as described in [28]. For secrecy of the key, we add a claim on the secrecy of the exchanged encrypted data (line 7). The reachability claim of line 8 is a sanity check. The tool will attempt to generate a trace that reaches the claim. This proves that the model can be fully executed and that there are no blocking events in the middle of the roles. The last two steps (lines 9-10) are a workaround for checking freshness. The variable `kupenc-u-2` denotes the K_{UPenc} obtained during the bearer addition phase. If Scyther cannot prove the last reachability claim (line 10), then this implies that it could not get passed the match event (line 9) and hence the keys are different. This is because a match event, in case its arguments are not equal, leads to the immediate termination of the role instance without executing the remaining events.

```
1   ...
2   fresh data: Nonce;
3   claim_UE7(UE, Running, SeNB, kupenc-u-3, a);
4   send_12(UE, SeNB, {data}kupenc-u-3);
5   recv_13(SeNB, UE, {data}kupenc-u-3);
6   claim_UE8(UE, Commit, SeNB, kupenc-u-3, a);
7   claim_UE9(UE, Secret, data);
8   claim_UE10(UE, Reachable);
9   match(kupenc-u-3, kupenc-u-2);
10  claim_UE11(UE, Reachable);
11 }
```

Listing 3. Last part of the terminal role in the Scyther model

B. Tamarin

Like Scyther, Tamarin also exploits strand spaces but relies on a constraint solving algorithm to verify the properties. Protocols are modeled as multiset rewrite systems [29]. This representation relies on facts and transitions. Facts are special symbols that can be applied to messages like functions are but not recursively. They can be freely introduced with the exception of few fact symbols used for example to model send and receive operations. The transitions are used to model the protocol behavior and the attacker actions.

A transition is defined by three sequences of facts: the left-hand side (LHS), the labels and the right-hand side (RHS) as shown in Listing 4. Executing a transition consumes all facts in the LHS, produces the ones in the RHS and emits the label facts as events. In the listing below, `Fr` and `Ltk` are facts; `ltk`, `A` and `B` are variables; and `h` is a (hash) function. The `Fr`, `Out` and `In` are built-in facts used respectively for creating fresh values, sending and receiving messages. Persistent facts are preceded by `!` and can always be used in transitions without being consumed. Prefixes are used to denote the types of variables. In particular `~` and `$` denote respectively a fresh and a public variable. The transition of Listing 4 models shared key provisioning. More precisely, upon its execution, a new fresh name `ltk` is generated. Then it is used for the creation of a key `h(ltk)`. Finally, it is associated non-deterministically with two public values `A` and `B` representing agent identifiers.

```
rule KeyProv:
 [Fr(~ltk)] --> [!Ltk($A, $B, h(~ltk))]
```

Listing 4. Key provisioning in the Tamarin model

Listing 5 contains the transitions implementing the initial offload of the MeNB role. Many of the limitations of the Tamarin language are dealt with in a similar manner to how it is done for Scyther. For instance, since there is no support for secure channels, all communication between the terminal and the MeNB and between the SeNB and MeNB is only encrypted. For that additional keys are used: `kenb` (line 4) and `kx2` (line 7). Furthermore, the algorithmic capabilities (line 9) and the bearers (line 5) are represented by public constant values. Finally, the SCC initially set to `'1'` (l3), is incremented by the built-in operation + though this is just syntactic sugar for usual functions.

Facts like `MeNBSession` (lines 8 and 10) can be used to store session data, in this particular case: the K_{eNB}, SCC, DRB identifier and the additional encryption key `kx2`. They can also be used as control states in the rules. This allows for modeling arbitrary control flows.

```
1 rule MeNB_Initial_Offload_1:
2 let
3  scc = '1'
4  skenb = kdf(<kenb, scc>)
5  drb = 'drb1'
6 in
7  [!Ltk(MeNB, SeNB, kx2),!Ltk(MeNB, UE, kenb)] -->
8  [MeNBSession(kenb, kx2, scc, drb),
9   Out(senc{skenb, <'alg1', 'alg2'>, drb}kx2)]
10 rule MeNB_Initial_Offload_2:
11  [MeNBSession(kenb, kx2, scc, drb), In(senc{alg}kx2)] -->
12  [Out(senc{scc, alg, drb}kenb)]
```

Listing 5. The MeNB rules in the Tamarin model

The algorithm selection operation is modeled by two rules for the SeNB with the same RHS such that the effect is that the SeNB chooses non deterministically from the tuple. In Listing 6, one of the rules is shown. The other one is obtained by replacing `alg1` by `alg2` in the label and the LHS.

```
1 rule SeNB_Initial_Offload_1:
2 let
3  kupenc = kdf(<skenb, alg1, drb>)
4 in
5  [!Ltk(MeNB, SeNB, kx2), In(senc{skenb, <alg1, alg2>, drb}
       kx2)]
6  --[SeNBRunning(kupenc, alg1)]->
7  [SeNBSession(kx2, skenb, alg1, drb), Out(senc{alg1}kx2)]
```

Listing 6. One of the SeNB selection rules in the Tamarin model

The security properties are first order formulas of label facts over the protocol traces with support for quantification over the positions (in the traces). This provides enough expressive power to represent all the properties that we want to verify. Listing 7 shows how the freshness property is defined. The notation `Ex` denotes the existential quantifier and `#i` a temporal variable (for trace positions). The lemma expresses that there should be no two occurrences of the `SessionKey` label fact with the same key `k`.

```
lemma key_freshness:
 not(Ex k #i #j. SessionKey(k) @ i & SessionKey(k) @ j & i
     < j)
```

Listing 7. The key freshness lemma in the Tamarin model

C. ProVerif

ProVerif translates protocol models to Horn clauses which are then subject to a resolution algorithm. The tool relies on an approximation [1] that enables unbounded verification. The input language of ProVerif is a typed variant of the applied pi calculus [30]. The complete specification can be found in the user manual [31]. A protocol model consists of a set of process macros representing each a particular role. Each macro is a sequence of events. As an example, Listing 9 shows the process macro of the MeNB role.

Security properties are expressed as queries on the attacker knowledge, or on arbitrary user defined events that are added in process macros in a similar manner to how claims are added in Scyther models. ProVerif can prove reachability properties and correspondence assertions [32] related to such events. Correspondence properties are of the form "if some event is executed, then another event has previously been executed", and can be used for checking various types of agreement properties [28].

Unlike the other tools, ProVerif provides support for secure channels and for better modeling of capabilities. In Listing 8, `mem` (line 3) denotes the set membership predicate where the arguments `a` and `uecaps` are of types representing respectively algorithms and algorithm sets. The event in line 3 binds the new name `a` to a value satisfying the predicate in the remaining events. This models the selection operation performed by the SeNB upon reception of the terminal capabilities.

```
1 let SeNB() =
2     in(secch, (=OFF, skenb: key, uecaps: algs, drb:
          bitstring));
3     let a: alg suchthat mem(a, uecaps) in
4     out(secch, (OFF, a));
5     ...
```

Listing 8. A sample from the SeNB process macro in the ProVerif model

Without the support for counters, the SCC and DRB are modeled by constants in a similar manner to how it is done in the Scyther model.

Each process macro is split in three separate phases (lines 6 and 8). This is a feature used to restrict process executions by preventing the execution of events of a phase until all the other processes have executed all the events of the previous phases. Using this feature, we group the steps for initial offload (lines 2-5), the steps for bearer addition (line 7) and the steps for session key update (lines 9-12) in 3 different phases. We then aim to prove secrecy and agreement on the established key and algorithm choice in each phase.

```
1 let MeNB(kenb: key, uecaps: algs) =
2     let skenb: key = kdf(kenb, ctobstr(SCC1)) in
3     out(secch, (OFF, skenb, uecaps, DRB1));
4     in(secch, (=OFF, a: alg));
5     out(secch, (OFF, SCC1, a, DRB1));
6     phase 1;
7     out(secch, (ADD, DRB2));
8     phase 2;
9     let skenb_2: key = kdf(kenb, ctobstr(inc(SCC1))) in
10    out(secch, (RFR, inc(SCC1), DRB1));
11    out(secch, (RFR, skenb_2, DRB1));
12    event menbReachable().
```

Listing 9. The MeNB process macro in the ProVerif model

Communication is modeled by the **in** and **out** events (such as in lines 3-4). Such events take two arguments representing the communication channel and the message. In message

TABLE I
VERIFICATION RESULTS

Tool	Scyther	Tamarin	ProVerif
Secrecy	+	+	++
Freshness	+	+	-
Agreement	-	-	++

TABLE II
TOOL EVALUATION

Tool	Scyther	Tamarin	ProVerif
Usability	++	+	+
Expressiveness	-	++	-
Performance	+	-	++

tuples, the first element is reserved for constants modeling message headers (`OFF`, `ADD` and `RFR`). Although this does not appear in the reference description of Fig. 1, it is a good practise when working with ProVerif. In fact ProVerif uses pattern matching when evaluating received messages so the pattern $=$ `OFF` in line 4 matches only messages with the `OFF` header. This has the effect of restricting the number of feasible traces and hence increasing the chances for termination. It is also in line with real implementations that do check the type of received messages.

IV. DISCUSSION

A. Verification Results

All the verification results are summarized in Table I. We used a laptop with an i5 Intel processor on which we executed the tools in a 1GB virtual machine running a 32 bit version of Linux. The convergence times where in the worst case in the order of two hours. The verification with the Scyther and Tamarin tools provided weaker results as all the properties could only be proven in the bounded models. In fact both tools have support for bounded verification. For the Scyther tool the user specified bound puts a limit on the number of runs. For Tamarin on the other hand, the limit is on the proof depth and it is not clear to us how to relate it to the protocol parameters. We admit this was a setback since our choice of the tools was in the first place partly motivated by their ability to deliver unbounded verification.

B. Tool Evaluation

In Table II, we compare the tools on three different dimensions: Usability (ease of use), expressiveness of the input language, and performance. We emphasize that this particular industrial case study does not exploit all the capabilities of the tools. Nevertheless, DC exhibits patterns and relies on features that are common in telecom security procedures.

On the usability dimension, we consider Scyther to be the most user-friendly tool for its simple input language and graphical interface. The Tamarin tool has also a graphical interface but requires that the user is familiar with some aspects of its theoretical foundation. The ProVerif tool is provided with an extensive tutorial which we believe compensates for the lack of gui-support to a great extent. For the performance dimension, there were no considerable differences in convergence times. Nevertheless ProVerif has a clear advantage since for Scyther and Tamarin convergences could only be achieved in the bounded models.

On the expressiveness dimension, Tamarin provides support for arbitrary control flow structures such as loops and conditionals. ProVerif offers a good compromise between language expressiveness and performance but in general because of the lack of support for global state, in both ProVerif and Scyther, we could not properly model the bearer addition loop of DC. Neither could we express the freshness property for these tools. The workaround described in Section III-A requires polynomial number of models to test equality between all possible pairs of the generated encryption keys. Finally, the lack of support for secure channels in Scyther and Tamarin poses a problem for agreement properties.

C. Applicability During Standardization

Most parts of DC as described here were agreed on relatively early in the standardization process. Nevertheless, other variations of the design were discussed afterwards that the model could not easily cater for. For example, it was discussed whether the same encryption key should be used for all DRBs in the SeNB, or the SeNB should reconfigure all existing DRBs when a new S-K_{eNB} is received from the MeNB. This is a quite typical exercise where tool support would simplify such decisions and increase the assurance level in a standardization process. In our case, this would require modeling lower level details of user plane encryption than the tools can handle.

None of the tools were able to verify the freshness property in the unbounded modification model. In fact, only Tamarin has support for the required modeling features, namely counters and loops. For the other tools, we had to resort to unrolling the bearer addition loop. In addition to counters, other constructions such as finite sets, e.g., for the bearer and encryption algorithm identifiers, are very common in access network security protocols. Most of the case studies for the tools under consideration are targeted at protocols where state changes are not preserved between runs. This is however the most common case in many security systems, such as telecom networks.

Another variation discussed was whether the SCC should be a counter or a random nonce. The choice fell on a counter due to the possibility that a nonce may repeat. In fact, despite being called nonces, in real systems they have finite domains. The counter on the other hand is predictable by an attacker. Since all the tools are based on a symbolic attacker model, it cannot be expected that probabilistic and computational attacker capabilities are taken into account.

V. CONCLUSION

In this work, we considered the security aspect of Dual Connectivity, a new LTE feature ongoing standardization. We described a model for the key establishment in this feature and formally analyzed it using the Scyther, Tamarin and

ProVerif tools. Our goal was to automatically prove secrecy, agreement and freshness of the established key. None of the tools provide alone full support for all the features in this case study, but used in combination as we did would be a good practice in standardization processes in order to increase the assurance in proposed designs. The process of formal modeling in itself helps in that one have to reflect on every design choice and clearly formulate the security goals.

ACKNOWLEDGMENT

Thanks to our colleagues and all the anonymous reviewers for their feedback on initial versions of this article.

REFERENCES

[1] B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," in *CSFW*. IEEE Computer Society, 2001, pp. 82–96.

[2] C. J. F. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols," in *CAV*, ser. Lecture Notes in Computer Science, A. Gupta and S. Malik, Eds., vol. 5123. Springer, 2008, pp. 414–418.

[3] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," in *CAV*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds., vol. 8044. Springer, 2013, pp. 696–701.

[4] D. Basin, S. R. K. Miyazaki and, and D. Watanabe, "Improving the Security of Cryptographic Protocol Standard," in *IEEE Security and Privacy*, 2015, p. to appear.

[5] N. Smart, V. Rijmen, M. Stam, B. Warinschi, and G. Watson, "Study on cryptographic protocols," 2014. [Online]. Available: https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/study-on-cryptographic-protocols/at_download/fullReport

[6] J.-K. Tsay and S. F. Mjølsnes, "A Vulnerability in the UMTS and LTE Authentication and Key Agreement Protocols," in *MMM-ACNS*, ser. Lecture Notes in Computer Science, I. V. Kotenko and V. A. Skormin, Eds., vol. 7531. Springer, 2012, pp. 65–76.

[7] C. Tang, D. A. Naumann, and S. Wetzel, "Symbolic Analysis for Security of Roaming Protocols in Mobile Networks - [Extended Abstract]," in *SecureComm*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, Eds., vol. 96. Springer, 2011, pp. 480–490.

[8] J. Fang and R. Jiang, "An analysis and improvement of 3GPP SAE AKA protocol based on strand space model," in *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on*, Sept 2010, pp. 789–793.

[9] "USECA - UMTS Security Architecture," http://www.acts-useca.org/.

[10] "3GPP The Mobile Broadband Standard," http://www.3gpp.org/specifications/.

[11] M. Burrows, M. Abadi, and R. M. Needham, "A logic of authentication," *ACM Trans. Comput. Syst.*, vol. 8, no. 1, pp. 18–36, 1990. [Online]. Available: http://doi.acm.org/10.1145/77648.77649

[12] C. J. F. Cremers, "Session-state Reveal Is Stronger Than Ephemeral Key Reveal: Attacking the NAXOS Authenticated Key Exchange Protocol," in *ACNS*, ser. Lecture Notes in Computer Science, M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, Eds., vol. 5536, 2009, pp. 20–33.

[13] ——, "Key Exchange in IPsec Revisited: Formal Analysis of IKEv1 and IKEv2," in *ESORICS*, ser. Lecture Notes in Computer Science, V. Atluri and C. Díaz, Eds., vol. 6879. Springer, 2011, pp. 315–334.

[14] A. Taha, A. Abdel-Hamid, and S. Tahar, "Formal Verification of IEEE 802.16 Security Sublayer Using Scyther Tool," in *Network and Service Security, 2009. N2S '09. International Conference on*, June 2009, pp. 1–5.

[15] ——, "Formal analysis of the handover schemes in mobile WiMAX networks," in *Wireless and Optical Communications Networks, 2009. WOCN '09. IFIP International Conference on*, April 2009, pp. 1–5.

[16] M. Jakobsson and S. Wetzel, "Security Weaknesses in Bluetooth," in *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, ser. CT-RSA 2001. London, UK, UK: Springer-Verlag, 2001, pp. 176–191. [Online]. Available: http://dl.acm.org/citation.cfm?id=646139.680779

[17] M. Abadi, B. Blanchet, and C. Fournet, "Just Fast Keying in the Pi Calculus," in *ESOP*, ser. Lecture Notes in Computer Science, D. A. Schmidt, Ed., vol. 2986. Springer, 2004, pp. 340–354.

[18] B. Blanchet and A. Chaudhuri, "Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2008, pp. 417–431.

[19] N. B. Henda and K. Norrman, "Formal analysis of security procedures in LTE - A feasibility study," in *Research in Attacks, Intrusions and Defenses - 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings*, ser. Lecture Notes in Computer Science, A. Stavrou, H. Bos, and G. Portokalidis, Eds., vol. 8688. Springer, 2014, pp. 341–361. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11379-1_17

[20] M. Arapinis, L. I. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar, "New privacy issues in mobile telephony: fix and verification," in *ACM Conference on Computer and Communications Security*, T. Yu, G. Danezis, and V. D. Gligor, Eds. ACM, 2012, pp. 205–216.

[21] B. Schmidt, R. Sasse, C. Cremers, and D. A. Basin, "Automated verification of group key agreement protocols," in *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 2014, pp. 179–194. [Online]. Available: http://dx.doi.org/10.1109/SP.2014.19

[22] C. J. F. Cremers, P. Lafourcade, and P. Nadeau, "Comparing state spaces in automatic security protocol analysis," in *Formal to Practical Security - Papers Issued from the 2005-2008 French-Japanese Collaboration*, ser. Lecture Notes in Computer Science, V. Cortier, C. Kirchner, M. Okada, and H. Sakurada, Eds., vol. 5458. Springer, 2009, pp. 70–94. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02002-5_5

[23] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron, "The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications," in *CAV*, ser. Lecture Notes in Computer Science, K. Etessami and S. K. Rajamani, Eds., vol. 3576. Springer, 2005, pp. 281–285.

[24] D. Dolev and A. C.-C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[25] N. A. Durgin, P. Lincoln, and J. C. Mitchell, "Multiset rewriting and the complexity of bounded security protocols," *Journal of Computer Security*, vol. 12, no. 2, pp. 247–311, 2004.

[26] C. J. F. Cremers, "Unbounded verification, falsification, and characterization of security protocols by pattern refinement," in *ACM Conference on Computer and Communications Security*, P. Ning, P. F. Syverson, and S. Jha, Eds. ACM, 2008, pp. 119–128.

[27] F. J. Thayer, J. C. Herzog, and J. D. Guttman, "Strand spaces: Why is a security protocol correct?" in *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 3-6, 1998, Proceedings*. IEEE Computer Society, 1998, pp. 160–171. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/SECPRI.1998.674832

[28] G. Lowe, "A Hierarchy of Authentication Specification," in *CSFW*. IEEE Computer Society, 1997, pp. 31–44.

[29] I. Cervesato, N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov, "A meta-notation for protocol analysis," in *Proceedings of the 12th IEEE Computer Security Foundations Workshop, CSFW 1999, Mordano, Italy, June 28-30, 1999*. IEEE Computer Society, 1999, pp. 55–69. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/CSFW.1999.779762

[30] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *POPL*, C. Hankin and D. Schmidt, Eds. ACM, 2001, pp. 104–115.

[31] B. Blanchet, B. Smyth, and V. Cheval, *ProVerif 1.88: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*. [Online]. Available: http://prosecco.gforge.inria.fr/personal/bblanche/proverif/

[32] B. Blanchet, "Automatic verification of correspondences for security protocols."