Formal Analysis of Security of LTE Handovers: Pre- & Post-Compromise

Rhys Miller, Ioana Boureanu and Zhili Sun

University of Surrey
{rhys.miller,i.boureanu, zhili.sun}@surrey.com

Abstract. Recent security analyses in 4G, Long Term Evolution (LTE), or even 5G, have mostly focused on the Authentication Key Agreement (AKA) protocol, rather than on other 3GPP security standards, such as handover procedures. To this end, we formally analyse the security of 4G/LTE handover protocols, called X2 and S1, beyond what existing works have achieved in this field. We conduct a thorough security analysis of X2 and S1 in ProVerif, a formal protocol verifier. We verify classical key-agreement properties, as well as post-compromise security (i.e., the secrecy of past and future session keys once the current session key is compromised). We find numerous attacks using ProVerif. We also exhibit a post-compromise security vulnerability which is a cross-protocol attack (between X2 and S1) that allows for the session keys of user equipments be compromised forever. We put forward secure versions of X2 and S1, and we use ProVerif to show that our new designs are indeed secure. In fact, we propose a series of such patches, including minimal ones that do not elude all attacks but better preserve usability and efficiency.

1 Introduction

As the Fifth Generation of mobile networks (5G) is being developed, an overwhelming proportion of key 5G functionalities relies on legacy versions of mobile communication networks, especially Long Term Evolution (LTE) or Fourth Generation (4G) networks.

Handover protocols allow mobile devices to swap connections between *cells* (or towers) of the mobile network. This process occurs seamlessly: as we physically travel out of the radius of one cell and into that of another, our phones search for a better signal with neighbouring cells.

In 4G/LTE, the cells that devices connect to are called "*E-UTRAN Node B (eNodeB)*" (or simply *node*). Figure 1 gives a high-level view of a handover: phone A switches from being connected to eNodeB A to eNodeB B. If the two nodes can negotiate this handover process amongst themselves, they do so via the *X2 interface*. Otherwise, where the nodes need to involve the *core* of the network for the handover, they do so via the *S1 interface*. The X2 and S1 interfaces are standardised by 3GPP (3rd Generation Partnership Project) in TS 36.423-900 and TS 36.413 [4,5].

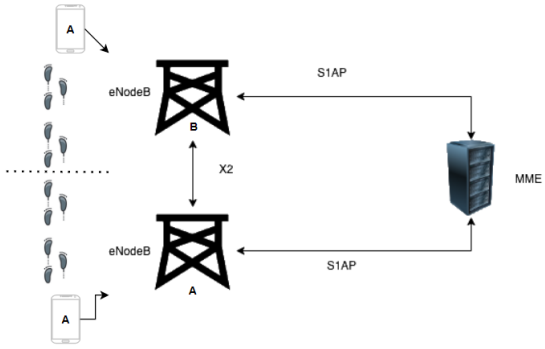

Fig. 1: LTE X2/S1 Interfaces – High Level Overview

Contributions.

1. We formally analyse the key-agreement security of the X2 and S1 protocols, along with backward- and forward-security (i.e., post-compromise security) of the session keys established. Our formal analysis uses ProVerif.

We do this in a comprehensive series of models and following the hierarchy of key-agreement properties. In ProVerif, we show several attacks against key-agreement, and for the first time in ProVerif, we show that X2 lacks post-compromise security (namely, backward security) of the main session key.

2. We also exhibit a cross-protocol attack that shows that the lack of post-compromise security in X2 can be exploited, resulting in the entire set of handover interfaces (i.e., S1 included) being made vulnerable to key compromise.

In practice, this means that a victim UE can have its secure mobile communications fully compromised for a long period of time, namely until it rejoins the network.

3. We propose patches for X2 and S1, model these new protocols in ProVerif, and prove our new variants secure against all previous threats found. For efficiency, we also propose a new variant of X2 that does not offer post-compromise security but does offer full key-agreement security.

4. In our work, we also compare with and analyse in ProVerif other patches suggested for X2, primarily the proposal in [22]. Whilst these patches were for different dis-agreement attacks (i.e., counter-based), we show them to be insecure for key agreement. We also note that they cannot counteract the serious cross-protocol attack.

Organisation. In Section 3, we present our security analysis in ProVerif of X2 and S1 using Lowe's agreement hierarchy [20]. In Section 4, we present our analysis of X2 against post-compromise security in ProVerif, as well as showing a cross-protocol attack that exploits this flaw in both X2 and S1. In Section 5, we propose our patches and give the results of analysing them in ProVerif (as well as those of analysing other orthogonal patches from [22]). In Section 6, we conclude. We provide most figures and attack traces in appendices, due to space constraints.

2 Background & Related Work

2.1 Background

In this section, we give a brief introduction to the 4G/LTE X2 and S1 handover protocols. From here on, we refer to these simply as *X2* and *S1*.

The X2 [1, 4, 6] and S1 [2, 5, 6] protocols are interactions between a *user equipment (UE)*, a *source node (SN)*, a *target node (TN)*, and the *core (or MME)* of the network. The message diagrams for X2 and S1 are given in Figure 5 and Figure 6, respectively, in Appendix 7.1.

Both protocols begin with the UE periodically producing *measurement reports (MRs)* as it geographically moves around; these MRs are capturing the strength of the radio signals to and from eNBs, to determine if the UE is on the most efficient network. The SN receives the measurement report and the node with the best signal is selected for the handover via the X2 or S1 interface, depending on what is supported by the SN.

X2. If the X2 interface is available, the SN derives a new cryptographic key K^*_{eNB}, based on K_{eNB} which the SN and the UE already share. Now, the SN sends the **Handover Request** (msg. 4 in Fig. 5) to the TN. This message contains the freshly-generated K^*_{eNB} along with a list of security algorithms supported by the UE. The TN uses K^*_{eNB} and one of the security algorithms to derive new *access-stratum (AS)* security keys, i.e., K_{RRCint}, K_{RRCenc}, K_{UPenc}, which will later be used to communicate securely with the UE. Upon receiving these, the TN sends the **Handover Request Ack** (msg. 6 in Fig. 5) to the SN. Then the SN sends **Handover Command** (msg. 8 in Fig. 5) to the UE, which informs it of the algorithm to use to calculate the K^*_{eNB} and the AS keys. The UE computes the latter and detaches from the SN.

Next, the SN gives the TN details (e.g., counts) to serve the UE using **Status Transfer** (msg. 11 in Fig. 5). The UE and the TN have a **Synchronization** phase (msg. 12 in Fig. 5), and timing-alignment using **Timing Advance and UL Allocation** (msg. 13 in Fig. 5). Finally, the UE sends the TN **Handover Confirm** (msg. 14 in Fig. 5). This contains an encryption of data using one of the established AS keys and a MAC over the security context used by the UE. From now on, the UE is supposed to receive and send packets from and to the target eNB.

The last message sent during the protocol is a **Packet Switch Request** (message 15) from the TN to the MME, which informs the MME that the user equipment has now switched to a new cell.

Note that there are some delays in the protocol, which we excluded from the description; these are to allow for asynchronous communication.

S1. If the SN and TN do not have an X2 interface, then the S1 protocol is performed. **Handover Required** (msg. 3 in Fig. 6) is sent to the MME, which includes the ID of the TN and relevant information about the SN.

Then, in **Derive the Security Context** (msg. 4 in Fig. 6), the MME computes the new security context for the UE; this contains the *Next Hop Chaining Count (NCC)* and the *Next Hop (NH)* key. In **Handover Request** (msg. 5 in Fig. 6) the TN receives the security context and the ID of the SN, which it uses

to derive $K_{eNB}{}^*$ and the AS security keys (K_{RRCint}, K_{RRCenc}, K_{UPenc}). The rest proceeds as in X2.

ProVerif. ProVerif [9] is an automated verifier of cryptographic protocols in the Dolev-Yao or symbolic analysis model [10]. It can handle an unbounded number of sessions of each protocol role. This tool has been successfully used to analyse secure air-to-ground communications [11], the TLS 1.3 Draft-18 [12], as well as e-voting protocols [15, 18].

2.2 Related Work

Due to space constraints, we cover primarily lines of work that view analyses of 4G/LTE handovers which are *formal*. Henda and Norrman [8] formally analyse the X2 and S1 protocols, using ProVerif [9]. The verification focused on showing reachability properties for correct executability, but included some agreement and secrecy analysis of the AS keys. This is not extensive and shows all agreements to hold. We disprove this using ProVerif—in fact we can disprove all their models if more specific queries are added.

Formal verification of handover procedures is also included in [13, 14]. Yet, unlike our work or in [8], [14] does not focus on 4G/LTE handovers, but instead on Universal Mobile Telecommunications Service (UMTS) ones, i.e., as user equipments move from a 3G to a 4G network. In [13], X2 and S1 are included in the verification, yet the focus is on payload security, guessing attacks, and analysis of compromising secure channels between the SN and the MME. We cannot find the ProVerif files used in [13], and the modest excerpts available in the paper do not focus on X2 and S1, but on 4G/LTE-UMTS handovers and payload security only. In the context of payload security, they check the backward and forward security of AS keys, but it is not clear which keys this is for (e.g., K_{eNB} or the AS keys).

Unlike [13], we look at 4G/LTE X2 and S1 as full protocols and not just at their payload security. Instead, we focus on session integrity and key agreement properties, where we model channel (in)security as per the 3GPP specifications. In these settings, we discover new attacks, as well as proposing patches for X2 and S1.

Finally, Han and Choi [17], look at the key-update procedures in 4G/LTE handovers to protect against rogue base stations. They show, via pen-and-paper analysis, that this key management can lead to the UE and the eNode to become desynchronised. The core of the attack exploits the signalling with the NCC counter (used by the core and the UE) to calculate the NH key and K_{eNB}, as performed in S1. Along similar lines, [22] also show that this type of desynchronisation attack exists using ProVerif. They also propose a patch which is similar to a patch we propose on X2 in order to attain backward security. Unfortunately, whilst their enhanced X2 does protect against desynchronisation attacks, using their own ProVerif code, we show that pure key-agreement properties on K_{eNB} do not hold in their patched protocol.

Other lines of work (such as [21]) look at the security of special handovers over mobile/4G heterogeneous networks. Since they are not closely related to this work, we do not discuss these works here.

3 Pre-compromise Security of LTE Handovers

In this section, we discuss our results on formal verification of "standard" security of the X2 and S1 protocols, in `ProVerif`. By "standard", we mean analysing security without considering that any party in the network is compromised, i.e., *pre-compromise*. *All the formal models and verification results discussed henceforth can be found at:* `github.com/rhysjohnmiller/4G_Handover_Verification`.

We model the four main parties in X2 and S1, i.e., the UE, the SN, the TN and the Core/MME, as "atomic" processes in our ProVerif code, respectively denoted as `UESeNodeB`, `TeNodeB`, `MME`. In ProVerif, we model all channels as public (i.e., available to the attacker), with the ones between SN, TN, and MME having added integrity[1] via the use signatures back and forth. In the lines 2,3,4 of ProVerif code below, the $sk*$ and $pk*$ represent the corresponding public/secret keys used for these signatures.For the queries/security-analysis discussed below, we consider two global processes, in two different ProVerif files:

```
PROCESS1
1 (UE(uecaps, kenb, cellidSource) ) |
2 (SeNodeB(uecaps, kenb, cellidSource, skSE, pkTE)) |
3 (TeNodeB(cellidTarget, kenb, skTE, pkSE, pkMME)) |
4 (MME(nh_2, kenb, skMME, pkTE))
```

```
PROCESS2
1 ! new .... let...
2 !(UE(uecaps, kenb, cellidSource) ) |
3 !(SeNodeB(uecaps, kenb, cellidSource, skSE, pkTE)) |
4 !(TeNodeB(cellidTarget, kenb, skTE, pkSE, pkMME)) |
5 (MME(nh_2, kenb, skMME, pkTE)),
```

`PROCESS1` contains just one instance of each of the four "atomic" processes. `PROCESS2` contains unbounded number of process and sessions of the SN, TN and UEs and 1 MME process with multiple sessions. As standard, in the above, `new ... let...` denotes the fresh declarations of all parameters describing the "atomic" processes and their instances under composition.

The X2 and S1 are (proxied) key-agreement protocols, as the access-stratum keys established are used to communicate securely between the UE and the new-serving eNode. Thus, we analyse these protocols against several security requirements to do with agreement and synchronisation [16,20] (i.e., session security and key-establishment security). In our ProVerif files, we also have several functional-correctness and helper/sanity-checking queries proven. Primarily, these show that the SN and TN operate in the prescribed manner, and are not detailed here.

3.1 X2 Verification in ProVerif

We now focus on discussing the security analysis of X2 in ProVerif, that is formal analysis w.r.t. message/key agreement and synchronisation goals [16].

Legitimate, Forged vs "Blind" HO Requests. In 4G/LTE networks, a handover which happens without a Measurement Report sent from the UE is known as a "blind" handover as opposed to a "full" handover. That is, in a "blind" HO request, the SN asks the TN to start serving a UE, without having received any prior information from that UE.

[1] From the 3GPP specifications in 36.331 [6], we know that the channels eNodes–MME are secure, and this is what we model. Signing, as opposed to encrypting and signing, is sufficient for the Dolev-Yao verification led herein.

We consider three queries below, on `PROCESS1`, to show that there is a real problem identifying the type/issuers of a HO requests.

```
(*****Query 1 UE-Aliveness UE-SN-TN *****)
(*attack trace found: TN is asked by SN to do a HO
for no reason, i.e., no UE has asked,
 TN asked to swap for no reason*)

query b:bitstring, b1:bitstring, b2:bitstring;
   event(TNReceivesHORequestFromSE(b)) ==>
   event(UESendsMeasurementReport(b1,b2)).


(*****Query 2 UE-Aliveness UE-SN-TN-SN *****)
(*attack trace: SN receives back even HO Req Ack
from TN with no UE wishing to transfer*)

query a:alg, b1:bitstring, b2:bitstring;
   event(SNReceivesHOCommandFromTE(a)) ==>
   event(UESendsMeasurementReport(b1,b2)).


(*****Query 3  UE-Aliveness UE-SN-TN-SN *****)
(*attack trace: SNode is actually finishing HO
commands in the due steps, with no input for UE*)

query a:alg, b1:bitstring, b2:bitstring, b3:bitstring;
   (event(SNReceivesHOCommandFromTE(a)) &&
   event(SNSendsHORequiredToTE(b3)) )  ==>
   event(UESendsMeasurementReport(b1,b2)) .
```

Based on query1, we show that in the X2 HO protocol, the next serving node (TN) cannot tell apart the case of the real HO request from that of a legitimate "blind" request. And, via query 2, we show that SN cannot tell apart a legitimate HO request from a faked one. Actually, the three queries show that the SN and/or TN can carry out entire HOs procedures without knowing if the original HO request was legitimate, faked, real or "blind".

Queries's Practical Significance. This may be equivalent, on the face of it, with just faking a message on public channel and resulting in a Denial of Service (DoS) attack. But, arguably, it is somewhat more worrisome in the light of the fact, that X2 will be used as the base for 5G's version of the protocol known as *XN*. These advancements will lead to many HO requests, and the cells/nodes should be able not only to tell apart a blind handover from a real one, and not allow the case of forged requests.

The attack traces show the falsification of the ProVerif queries above. The adversary impersonates a UE via a HO command with no UE being present. Not only does the TN start the handover process, but both the SN and the TN execute every stage of the HO Preparation phase, one by one, up to message 6 in Figure 5.

Formal Meaning. The attack traces refuting queries 1-3 above represent the falsification of recent aliveness, as defined in the Lowe hierarchy of agreement requirements [20] and reformulated in [16]. The first query is a failure of aliveness of UE with TN as partner, and the second and third queries with SN as partner for said UE.

Attacker Blocking HO Commands This blocking was tested against Process1. Note that message 8 in X2's Fig. 5 is on a public channel, i.e., it is easily intercepted and blocked by an attacker. This is the crux of the fact that attacks above do become more serious.

```
(*****Query 4 Non-inj. Synchronisation Query *****)
(*attack trace:   TNode starts transferring over +
original ask for UE, but UE not continuing*)

query b1:bitstring, ba: bitstring, bb: bitstring,
        a:alg;
  event(startTransfer(b1)) &&
  event(UESendsMeasurementReport(ba,bb))
  ==> event(UEReceivesHOComand(a)).
```

The trace falsifying query 4 shows that a legitimate UE can send a HO request (message 1) but the attacker blocks his receipt of message 10 (HO Command),

yet the SN and TN will continue the handover procedure and the TN will start transferring (i.e, receive message 11).

In Appendix 7, in Figure **??** we show this attack-trace refuting query 4.

In our ProVerif files we show the refutation of query 4 above, but also variations thereof.

To this end, the fact the HO Command can be blocked from getting to the UE and that the computations by the different processes in LTE handovers are asynchronous means that there is no guarantee that when a TN "transfers" to serve a legitimate UE (query 4) or (in a variation of query 4) that in fact any UE what-soever is ready to also be served by this TN. This translates, in its different fashions, in a DoS-style attack against the UE and the SN/TN: the UE in query 4 would not be served by the SN anymore and the TN would wish to serve this UE now, yet the UE is not ready to swap over to the TN. If nothing else, the QoS (quality of service) for that UE would drop significantly. So, whilst the asynchronicity of X2/S1 handovers likely has been put in place in order to have the mobile service not disturbed during the HOs, the very asynchronicity opens for adversarial service disruption.

Formal-Security Goal Violations Formally, the attack traces refuting query 4 above are falsification of non-injective synchronisation, as defined in the Lowe hierarchy of agreement requirements [20] and reformulated in [16]. The traces show a lack of synchronisation between the UE and the SN and/or TN.

Key-Agreement Attacks. We use queries 5–9 below to show a series of key-agreement attacks.

```
(*****Queries 5*****)
(*attack trace: UE has  a kenbstar
whereas the TN does not, or vice-versa*)
query k: key;
  (event(kenbstarUE(k)) ==>  event(kenbstarTE(k))).
query k1:key, k2:key;
  (event(kenbstarUE(k1)) ==> event(kenbstarTE(k2)) && (k1=k2)).


(*****Queries 6*****)
(*attack: UE has a kenbstar
whereas the SN does not, or vice-versa*)
query k:key;
  (event(kenbstarUE(k)) ==> event(kenbstarSE(k))).
query k1:key, k2:key;
  (event(kenbstarUE(k1)) ==> event(kenbstarSE(k2)) && (k1=k2)).


(*****Queries 8*****)
(*attack trace: TN and UE disagree on kenbstar*)
query k1:key, k2:key;
  (inj-event(kenbstarUE(k1)) &&
   inj-event(kenbstarTE(k2)) ==> (k1=k2)).

(*****Queries 9*****)
(*attack trace: UE, TN and SN disagree on kenbstar*)
 query k:key;
    ( inj-event(kenbstarUE(k)) ==>  (inj-event(kenbstarTE(k))
 ==> inj-event(kenbstarSE(k))) ).
```

Queries 5–6 are refuted in ProVerif, against `PROCESS1`, in trivial ways: the attacker can block message 8 to the UE, or fake the presence of a UE to the SN/TN via a fake message 1. More serious key-agreement attacks are the case, as shown by the failure of queries 8 and 9 against `PROCESS2`. Whilst they all fail, a trace on the first query 8 (found interactively) involves one user-equipment UE in two sessions thereof, UE-sess1 and UE-sess2, and 1 TN, as seen in Figure 2. In practice, this means that an attacker basically can chooses which session if the UE counts with respect it computing session-keys with a TN.

Note that, via message 14, the TN checks a MAC check on the security algorithm passed to the UE in message 8 in order for the latter to compute K^*_{eNB}. So, the HO Command s from two different UEs cannot be cross-wired by the attacker, but the sessions by one UE can, as per the attack in Figure 2.

Formal Meaning. Formally, the attack trace refuting query 5 above is a falsification of weak agreement w.r.t. to K^*_{eNB} between the TN and the UE, as per Lowe's hierarchy of agreement requirements [20]. From a verification perspective, it

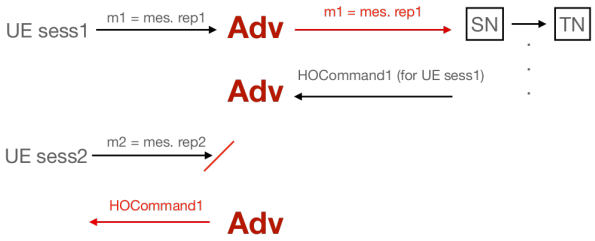

Fig. 2: Lack of Injective Agreement on K^*_{eNB} between a UE and TNs

means that the failure of the aliveness of the UE (queries 1-3) implies the failure of the non-injective synchronisation (query 4) and the weak agreement (queries 5,6):

failure of query 1, 2, 3⇒ failure of queries 5,6.

Formally, the session-interleaving attack via query 8 is a falsification of injective agreement to K^*_{eNB} between the TN and the UE, as per Lowe's hierarchy of agreement requirements [20].

Summary of Results on X2. Table 1 shows all the queries discussed above for X2. The failures in query 1-3 imply the failures in queries 4-7, and failure in queries 5-7 implies the failure in queries 8-9, yet since meaning of the attacks in queries 8-9 is different to that in queries 5-7, we recounted their ProVerif here checking too.

Table 1: ProVerif Verification of X2: Results Summary

Query	Meaning	PROCESS	Results
1–3	aliveness of UE	1	no
4	non-injective sync of UE and TN	1	no
5	weak key-agreement UE -TN	1	no
6	weak key-agreement UE-SN	1	no
7	weak key-agreement SN-TN	1	no
8	injective key-agreement UE-TN	2	no
9	injective key-agreement UE-TN-SN	2	no

3.2 S1 Verification in ProVerif

The verification of S1 follows the same ideas as that of X2, and as such we do not discuss it to the same length, due to space constraints. For instance, the significance of the attacks found (e.g., adversarial faking/blocking HO requests, key-agreement failure, etc.) is the same on S1 as it was on X2.

[2] See the long version [3] for queries 4 and 7, which leads to showing that the TN can be fooled into running a full HO without any UE present, and respectively to showing that the SN and TN can disagree on k^*_{eNB} due to a forged message 1.

Legitimate, Forged vs "Blind" HO Requests These were tested against Process1. As in the X2 case, an attacker can send bogus S1 Measurement-Reports and block HO Commands from going to the UE, which translate in the TN and SN running all the HO preparation phase with no UE, this this involving the MME, step by step too. These aliveness and synchronisation attacks are shown by the refutation of the 4 queries below.

```
(***** Query 1 *****)
(*TN is asked to do an HO  by the attacker,
no UE present*)
query b:bitstring, b1:bitstring, b2:bitstring;
    event(TNReceivesHORequestFromMME(b))
    ==> event(UESendsMeasurementReport(b1,b2)).

(*****Query 2*****)
(*SNode is finishing  a HO on attacker's calls,
i.e., no UE has asked*)

query a:alg, b1:bitstring, b2:bitstring;
    event(SNReceivesHOCommandFromMME(a)) ==>
    event(UESendsMeasurementReport(b1,b2)).

(* ****Query 3*****)
(*SNode is finishing HO Commands in the due steps,
with no input for a real UE*)

query a:alg, b1:bitstring, b2:bitstring,
   b3:bitstring;
  (event(SNReceivesHOCommandFromMME(a))
   && event(SNSendsHORequiredToMME(b3)) )  ==>
   event(UESendsMeasurementReport(b1,b2)).

(*****Query 4*****)
(*TNode is transferring over, with
no original ask for UE*)

query b1:bitstring, ba:bitstring, bb:bitstring;
  event(startTransfer(b1)) ==>
  event(UESendsMeasurementReport(ba,bb)).
```

The "cross-wiring" of UE/TN identities attacks (i.e., the injective agreement attacks on UE and TN identities) shown on X2 also apply to S1. When compared to X2, these attacks (which we have shown on X2 as well) could be seen as more dangerous as the core is more involved in the S1 procedures than in the X2 ones. I.e., even the resources of the core as spent uselessly on a full but fake HO. By and large, we see that (at least but not only) the TNs and/or UEs can be fooled as to which particular UE is be served/attached by/to which node TN in the network.

Key-Agreement Attacks. The same lack of weak-agreement on K^*_{eNB} for the UE and TN/MME as in X2 for SN/TN applies to S1, exhibited by query 5-6 below.

```
(*****Query 5*****)
(*TN and UE key disagreement*)
query k:key;
  (event(kenbstarUE(k)) ==> event(kenbstarTN(k))).

 (*****Query 6*****)
(*MME and UE key disagreement*)
query k:key;
  (event(kenbstarUE(k)) ==> event(kenbstarMME(k))).

  (*****Query 9*****)
query k:key;
inj-event(kenbstarTN(k))
&& event(UEReceivesHOComand(k))
==> inj-event(kenbstarMME(k)).

(*****Query 10*****)
query k:key;
event(kenbstarTN(k))
&& event(UEReceivesHOComand(k))
==> event(kenbstarMME(k)).
```

The attack trace refuting query 9 above shows that the S1 protocol is in fact susceptible to replay attacks as well, as the adversary replays a message for a TN in one session to another session of the same TN (see Section 7 for a link to the attack). In turn, this can cause the TN and the UE be desynchronised on the computed key. Yet, the non-injective variation of query 9 (i.e., query 10 above) does hold.

Summary of Results on S1. We recap in one table Table 2 the main queries we analysed S1. As discussed before for X2, the failures for S2 in query 1-3 imply the failures in queries 4-7, and failure in queries 5-7 implies the failure in queries 8, yet because the meaning of the attacks in queries 8 is different to that in queries

5-7, we deemed it suited to exhibit them in ProVerif as well. In the case of S1, multi-session execution leads to attacks not possible on single-session executions: query 9 is falsified when query 10 is not.

Table 2: ProVerif Verification of S1: Results Summary

Query	Meaning	PROCESS	Results
1–3	aliveness of UE	1	no
4	non-injective sync of UE and TN	1	no
5	weak key-agreement UE -TN	1	no
6	weak key-agreement UE-MME	1	no
7	weak key-agreement MME-TN	1	no
8	injective key-agreement UE - MME	2	no
9	injective key-agreement UE, MME, TN	2	no
10	non-injective key-agreement UE, MME, TN	2	yes

Recall that all the formal models and verification results discussed above can be found at `github.com/rhysjohnmiller/4G_Handover_Verification`.

4 Post-Compromise Security of the Access-Stratum

Now, we will discuss the post-compromise security of the access stratum (AS) keys in LTE Handovers. We discuss these keys in Section 2, but first we detail the key-derivation process.

4.1 AS Keys

As a UE joins the network, during the AKA protocol, the core will weakly-authenticate the UE via the IMSI and the two get to share a long-term key called K_{ASME}. This key K_{ASME} is then employed, during S1 handovers, to derive the short-term key K_{eNB}.

AS "base-key" for source node. After the AKA protocol, the core will allow the UE to connect to an eNode, the core will send this eNode a K_{eNB} key linked to this UE. This is done during a procedure called "AS Security Setup"; You can see this in S1 at message 6-9 in Figure 6 and in X2 at message 5 in Figure 5. This is described in Figure 3 below, in step 4.

New AS security base-key for target nodes, The eNode to which the UE is connected to begin with (and which has a K_{eNB} as per the above) is the so-called "source node" (SN) in S1/X2 protocols. Such an X2 source node can also compute the K^*_{eNB} key: that is, K^*_{eNB} is obtained from K_{eNB} and agains static data — the TN's ID and the frequency of the channel. I.e., K^*_{eNB}=$KDF(K_{eNB}$,TN_ID,$freq)$ (where KDF stands for "key derivation function"). Then, in X2, the SN sends this K^*_{eNB} to the TN, along with the security algorithms to use to compute the AS keys. In S1, the TN receives this K^*_{eNB} from the MME. A UE can do the same aforesaid

[3] See the long version [3] for queries 1–4, 7–8.

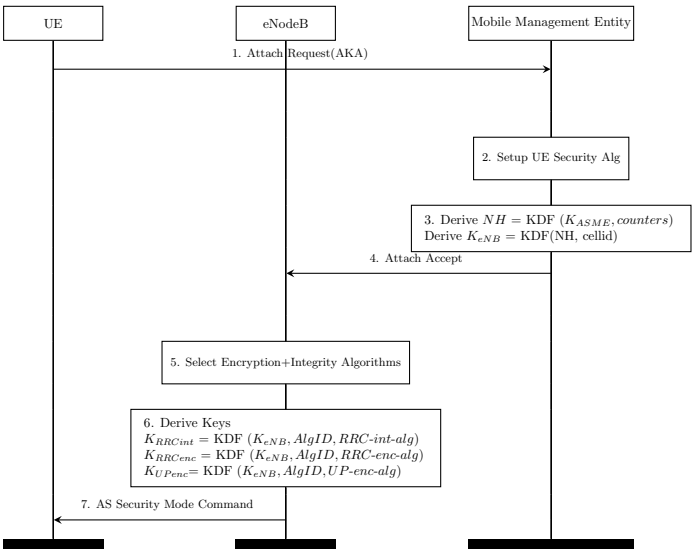

Fig. 3: AS Security Setup

operations in order to get a fresh key K^*_{eNB} to use with a new, target cell. In this sense, K^*_{eNB} is just a new K_{eNB} to use between a UE and a (target/new) node.

New AS security keys. As shown on Figure 3 in step 6, the AS keys $\mathrm{K_{RRCint}}$, $\mathrm{K_{RRCenc}}$, $\mathrm{K_{UPenc}}$ are obtained simply from K^*_{eNB} and given security algorithms for KDF passed by the SN to the TN and UE (to the latter in clear).

4.2 No Post-Compromise Security for the AS Keys

First, we recall two "post-compromise" security notions as per the 3GPP specs [7]:
– ***Backward security:*** requires that future session keys be not computable if current session keys are compromised.
– ***Forward security:*** requires that session keys is be not computable if current session keys are compromised.

It is known, from the 3GPP specification [7], that X2 has no backward or forward security of K_{eNB} or the AS keys. In simple terms, if the current K_{eNB} leaks for a UE, then follow-up K^*_{eNB} and the AS keys can be computed without any hardship due to the fact inside the KDFs the inputs are node-identifiers and security algorithms which can be collected/intercepted from the network, or exhaustively searched). Equally, if the currently used K^*_{eNB} leaks, then the previous K_{eNB} can be computed via the reverse of the KDFs on the same inputs.

ProVerif-found Post-Compromise Insecurities for X2. Our ProVerif analysis also exhibits the lack of backward-security in X2 w.r.t. K^*_{eNB}.

Note. We actually show X2's lack of backward security[2] for K^*_{eNB} in ProVerif, as well. Moreover, as far as we know, no pre-existing work that has fixed this lack of post-compromise security in X2; in the next section, we provide such a patch for X2, which we also prove to have post-compromise security in ProVerif.

We now present an obvious backward-security attack on the AS keys. It is known, from the 3GPP specs TS 33.401 that X2 has no backward security of K_{eNB}, but *in fact we show below that there is feasibly no backward security of the AS keys not just in X2, but in multiple settings.*

No Backward-Security for K_{eNB} and AS Keys in X2 or S1. Concretely, the below backward-security attacks apply to the S1 protocol, OR the X2 protocol, OR even just on the AS setup when the UE joins the network.

Assumption. Let one K_{eNB} corresponding to a UE leak to the attacker. Otherwise, this attacker proceeds as follows:

1. The attacker knowing the leaked K_{eNB} can compute all the output table of $KDF(K_{eNB}$,TN_ID,$frequency)$, one entry therein being the correct K^*_{eNB} for a new target node and the UE in cause.
   – This is because the TN_ID and $frequency$ are over an arguably small, easy-to-iterate space.
   – Moreover, the attack would normally be run with TN_ID being fixed/-targeted, as it is visible in a run of S1 and X2 (run which can be even induced/faked by the attacker), as part of the HO Command going from the SN to the UE: i.e, in messages 8, 10 on Fig. 5, Fig. 6 respectively.
2. To compute the AS keys for the said UE and the new TN, the attacker also needs to know the so-called security algorithms used for this.
   – The attacker could brute-force this space, since the number of security algorithms supported is low.
   – However, these algorithms are also sent in clear as part of each run of S1 and X2 (in messages 8, 10 on Fig. 5, Fig. 6 respectively), as part of the HO Command going from the SN to the UE.
   – Alternatively, if the attacker envisaged this UE for its attack from when they joined the network (e.g., mass surveillance or high-profile individuals being given a new phone in a new role), then –right after the UE joined the network– in the AS setup (Fig. 3), a set security algorithms used to compute the AS keys are sent to the UE in clear, and the UE is likely to receive the same again [19].

So, this attacker gets the AS keys and can now decrypt AS messages between a given UE and a new TN.

The attacker would normally collect encrypted AS messages between the UE and the/a TN (one that the UE was seen to connect to). Meanwhile, the attacker waits to finish the offline computations necessary in aforesaid steps of the attack. But thereafter, the attacker can decrypt the pre-collected, encrypted AS messages.

Discussions about this AS-security attack. This attack basically says that if a current K_{eNB} leaks for a UE, then follow-up K^*_{eNB} that the said UE would use to communicate securely with given node can be computed without much hardship (due to the fact that the identifiers of such nodes, the security algorithms and such

[2] Whilst [13] spoke of this attack, their ProVerif code is not available, and it not clear from the paper if the verification was on the AS keys or K^*_{eNB}.

needed ingredients can be collected from the network, or exhaustively searched). Indeed, note that we presented the attack in a fashion in which the id of the new serving node (and security algorithms) is (are) not known, but if it is (are), then the attacker is easier.

One pertinent question that remains is why would an adversary undertake such an attack vs. say just "bugging" the phone. In this vain, note that we presented the attack in a way independent of how the K_{eNB} does leak: i.e., the key may leak from an eNode or from the UE itself, depending also on the attacker's targets/possibilities. In that sense, all attacks depend on the attacker's capabilities. For ours, on the one hand, adversarial radio manufacturers may illicitly get such a K_{eNB} from an eNode. On the other hand, other type of attackers may have access to the UE, but this access may be limited; for instance, they may get to a current from a phone via side-channels when such a K_{eNB} is computed, but be unable to get the long-term, SIM-stored ASME details.

Finally, it is true that the feasibility of this attack (described generically above) does vary with its parameters. It is clearly much easier if the attack is run in the following fashion: attacker retrieves a K_{eNB} between a UE and SN now, knows that the UE will connect to a given TN in some moments, listens in the X2/S1 handover when this new connection is supposed to occur and collects the encrypted AS data between said UE and said TN and decrypts as soon as the K^*_{eNB} is computed (offline) by the attacker. For the latter computation, in this flavour of the attack, the attacker only needs to guess the right frequency of the network to compute K^*_{eNB} from a leaked K_{eNB}.

Cross-Protocol Attack: No Backward Security Forever. We now show that even if, in principle, S1 can be used to retrieve backward security (as core sends a "fresh" K^*_{eNB}), an adversary can bypass that via a cross-protocol attack. I.e., because X2 and S1 are compatible, a given adversary can always enforce X2 over S1 and so, once it has learnt one K_{eNB} for a UE, then all its AS security is compromised until it re-joins via AKA the network.

What is more, in this attack, we see that if just one SN in the network gets compromised, and *not even* the one to which the victim UE is currently connected, then all AS security for that UE is compromised too.

Attack Setting. The attacker is a radio manufacturer and manager of an eNode. It targets a given UE and it knows the whereabouts of this UE. This attacker also can control the network to some extends (i.e., sniff and block messages).

The steps of the attacks are as follows:

1. The attacker knows that the UE has been connected, for a while, to a node SN1.
   The attacker $\mathcal{A}$ knows that the UE was connected to a node SN2, before being served by SN1.
   The attacker knows that the UE will soon have to swap to a node TN.
   SN1 which only supports S1.
2. Now, the attacker $\mathcal{A}$ now gets control over SN2, which it can corrupt, unlike SN1.
   So, the attacker gets the $K1_{eNB}$ which this SN2 had for the UE when the UE was connected to SN2 before being connected to SN1.
   Then, the attacker uses the lack of backward-security in X2 and computes the current K_{eNB} for this UE, i.e., $K2_{eNB}= KDF(K1_{eNB},SN1,freq)$.

3. When the UE asks to start the handover, to happen via S1 between its current serving SN1 and the node TN, the attacker $\mathcal{A}$ intercepts the communication and blocks the request from reaching TN.
4. Instead, the attacker $\mathcal{A}$, acting as SN2, uses X2 and asks TN to swap and serve this UE.

   In this way, the K_{eNB} key does not get refreshed by the core/MME of the network and $\mathcal{A}$ still controls its re-computation.
5. The attacker $\mathcal{A}$ computes $K3_{eNB}$= $KDF(K2_{eNB}, TN, freq')$ and sends it to TN.
6. The attacker $\mathcal{A}$ sends the HO Command to the UE.
7. The attacker can derives the AS keys to decrypt all communication between the UE and the TN, and those between UE and SN1 (if it collected them).

A representation of this attack is given below.

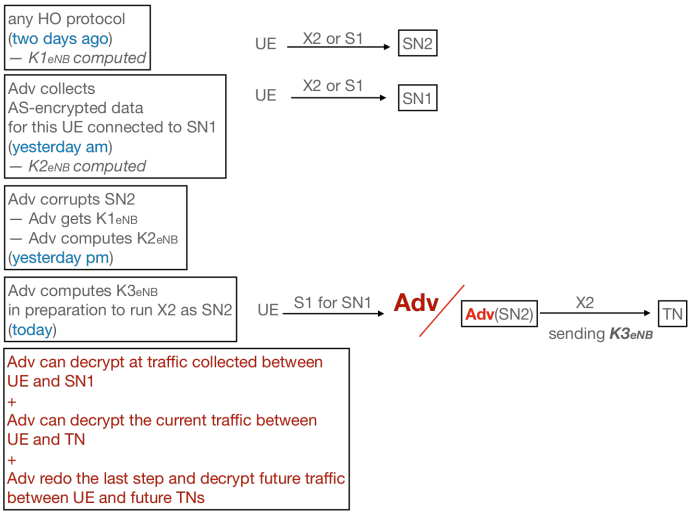

Fig. 4: AS Backward-Security Lost Forever via Cross-protocol Attacks

Via this attack, we see that X2's lack of backward-security and the S1-X2 compatibility allow that, if *only one eNode gets corrupted* (i.e., SN2) and the attacker can sniff/inject/block message into the network, then the security of the access stratum can be completely/permanently comprised, for S1 too!

5 Improved & Secure LTE Handovers

We formally proved that the X2 and S1 handovers are broken from the viewpoint of key-establishment security and backward-security of AS keys *even for S1*. Now, we show a series of patches for X2 and S1, balancing security and efficiency.

5.1 Our New X2 and S1 Designs

On Identification in Handovers. Via the failure of our queries 1-3, 6,7 on X2 and S1 we how that there is weak identification of the UE when it comes to the network eNodes. As we put it, one cannot tell between a forged HO Request, a blind HO request by an eNode, and legitimate HO request by an UE.

Outcome 1. Better identification of UEs to TNs (and vice-versa) is desirable. In turn, this mandates the Measurement Report be UE-identifying in a cryptographic sense, i.e., be sent encrypted (+ MAC-ed) so that attackers cannot get the information inside or modify it.

Key-Agreement Attacks. As our ProVerif traces show in Section 7, we see that most key-agreement attacks stem from the fact that HO Command is sent in clear and the attacker can manipulate it, confusing two session of the same UE as to K^*_{eNB} they computed from a TN's viewpoint.

Outcome 2. The HO Command needs to be sent encrypted and with more session-identifying information, to avoid session-interleaving attacks against (injective) key-agreement.

On Recentness/Synchronisation in X2/S1. In X2 and S1, parties act asynchronously, and that is for usability, i.e., smooth service. Yet, the attacks we show that a lack of synchronisation/step-confirmation between the UE and TN/SN is possible. I.e., the UE does not compute the K^*_{eNB} when the SN/TN do.

Outcome 3. The UE and (at least) the TN should contain a stronger mechanism of key/session confirmation[3], to guarantee injective key-agreement and avoid desynchronisation. To check recentness and avoid replay attacks, as well as aid with protecting agains post-compromise security attacks (see below), timestamps should be used in the key-confirmation process.

On Post-compromise Security.

Given the backward-security attacks we showed, stemming from compromised eNodes, we can infer that SNs should not singlehandedly compute keys for TNs. I.e., S1 is better than X2, from a post-compromise security perspective.

However, with usability being at stake, we would not recommend that X2 be scrapped. Instead, post-compromise insecurity can be improved via added encryption between the UE and SN (see Outcome 1, Outcome 2 above), as corrupt SNs would no longer get certain info (i.e., identifier, security algorithm). Also, in tandem with Outcome 3 above, we recommend that the AS key be also based on more session/fresh data (e.g., nonces, timestamps) which an attacker (i.e., compromised SN cannot retrieve). This would make exhaustive search for inputs to the AS keys' generation infeasible. Moreover, if this added freshness was contributed to by the UE as well as the TN, this would provide better balance of security and trust.

Finally, because of our cross-protocol post-compromise attack, we recommend that the target nodes can request to perform an S1 on demand via a specific command, to ensure a core-based refreshment of K^*_{eNB}.

Outcome 4. The AS key should be derived based on unpredictable, fresh data, ideally with the proxy SN incapable of retrieving (all of) it. Further, target nodes should be able request to perform an S1 on demand, via a specific command,

[3] The `Handover Confirmation` message contains a MAC keyed on K^*_{eNB} applied to the security context used by the UE, but this still allows for two sessions of the same UE to be confused, interleaved, replayed

to allow exceptional core-based refreshment of K^*_{eNB} as a prompt recovery of post-compromise security.

5.2 Our New X2 and S1

By implementing the aforementioned learning outcomes 1–4 issued from our ProVerif verification & our cross-protocol attacks, we will now suggest our patched versions of X2 and S1.

Our More Secure & Usable S1. In Figure 7, we present a patch of the S1 protocol w.r.t. issues found, yet keeps as close as possible to S1. We do not reiterate every single S1 message as per Figure 6, instead we keep the numbering in place and stress on Figure 7 only the messages that we change. The most important aspects are marked in red. Our new S1 is described below.

We keep the normal S1 setup is: there is a UE connected to a SN, sharing a current K_{eNB}. First, the UE generates a timestamped nonce T_1 (note that SIMs have this capability, so all UEs can do this); then, the UE no long sends the S1 Measurement Report (MSR) in clear, but the UE encrypts the MSR (in authenticated mode) under K_{eNB} after it appended its identity (SUPI – Subscription Permanent Identifier), and aforesaid T_1 to the MSR.

Second, as part of the Handover Decision, the SN sends to the MME more details than before: the ID of the TN, and all data received from the UE, all in authenticated encrypted form (e.g., encrypt-then-sign).

Third, the MME checks the freshness of the timestamp T_1. Fourth, the MME generates nh_2 not just based on the old nh and the security algorithms suited to the SUPI, but also the T_1 timestamp nonce, i.e., nh_2=$KDF(K_{ASME};NCC,T1)$; this makes the nh_2 key less predictable and resistantto leakage of K_{ASME} and NCC.

Fifth, in the HO Request, the MME sends the TN all the data needed, in authenticated encrypted form, included the T_1. So, the AS keys generated by the TN will also use T_1 as an extra input into the KDF.

After the normal S1 tunnels are created, in message 10 of the S1 protocol, the SN gets from the MME the security algorithms to pass on to the UE; at this stage, the T_1 and TN_{id} are ferried back and the SN would re-check them. Such verifications are denoted "Session checks" on Fig. 7, throughout.

Unlike in the normal S1, message 11 is now encrypted with the current K_{eNB} (shared by the SN and the UE) and this new HO Command contains the T_1, and SUPI echoed back, so the UE can check these are the data from his current request.

In step 13, unlike in standard S1, the UE uses T_1 to generate the new AS keys (i.e., T_1 is an extra input to the S1 KDF), i.e., K_{RRCint} = KDF (K_{eNB},$AlgID$,$RRC\text{-}int\text{-}alg$,$T_1$).

Finally, in the first data-packets (steps 18 and 19) there is a key-confirmation based mainly on T_1 (but also SUPI and TN_{id}), being rechecked.

Discussions on our new S1. The main additions are: (1) the securing of the Measurement-Report message and the HO Command message via encryption under the current K_{eNB}; (2) the end-to-end secure identification of the UE and TN (msgs 1–5); (3) addition of the timestamped nonce to yield verifiable session-freshness and be used in all AS-keys generation, to stop replay attacks and make post-compromise security attacks impractical; (4) secure key confirmation

steps (we enhanced msg. 14, 18, 19 with further session checks). Of course, we could have added several new nonces, and do a 2-by-2 confirmation of session/key integrity/agreement, etc. However, this would be too costly on the efficiency of HO and detrimental to service quality during HOs (and it is not backed by the analysis we undertook).

Note. In this new S1, we do not protect against the fact that the SN can be actively malicious during the S1 protocol. We aim to protect against post-compromise security by malicious SNs in X2, but this does not arise in S1 (since the SN does not compute K^*_{eNB}). However, in this new S1, we could have made the MME add his own nonce to the AS-keys' derivation and send the ingredients for the HO Command encrypted with K_{ASME} such that a malicious SN could not modify it in its transit towards the MME. But we chose to leave such more costly protections aside, and balance security, risk and trust instead. We take a different approach for X2. This is inline with our verification results for both protocols.

Our More Secure & Usable X2. In Figure 8 below, we propose a patch for X2, by implementing the aforementioned learning outcomes 1–4 issued from our attacks.

Note. Unlike in S1, in the case of X2, we explicitly protect against the SN doing post-compromise security attacks. Concretely, in our new X2, we do not mandate the SN to compute the K^*_{eNB} for the TN, but rather make TN do it. I.e., the SN becomes a passive proxy, not an active proxy.

What is more, for post-compromise security, in the new X2, the TN also add his randomness T_2 to the computation of the AS keys. Also, the K^*_{eNB} and AS-keys are computed and confirmed in two separate stages. First, K^*_{eNB} is confirmed between the UE and TN (msg 10), and MME (msg 11). Then, so that we do not allow the potentially malicious SN to see TN's randomness T_2, the core/MME is used to send the TN's randomness to the UE (and confirm other parts of the session); the K_{ASME}-key is used to encrypt this step between the MME and the UE (msg 12). Then, the UE uses it to compute the AS keys and uses the AS keys with the TN.

Note that our new X2 is lighter than a standard S1, in that the TN does do most of the computation, and the MME is used only in the last step. Yet, as an alternative, in this new X2 , the TN is enhanced with an optional command to ask that S1 be run instead of X2.

Lighter, New X2. We note that if we wish to incorporate just learning outcomes 1–3 into a new X2, i.e., add no protection for post-compromise security, then we can produce a much lighter X2. This one would not involve the MME and would not require the K^*_{eNB} and AS keys be computed and confirmed in two separate stages. The rest of it follows the new X2 that we presented above. This protocol attains all the key-agreement security that the current X2 does not apart for backward and forward security of the K^*_{eNB} and AS keys.

We present this protocol in Figure 9 and its ProVerif file can be found at: *github.com/rhysjohnmiller/4G_Handover_Verification.*

5.3 Verification Results of Our Patches & Other Variants

Our newly proposed X2 and S1 variants *provably* fix all the attacks we showed on X2 and S1, including the post-compromise attacks.

Note. The fact that the K^*_{eNB} key and the AS keys are computed/confirmed in two stages in our new-X2 protocol is similar to the way in which [22] patches X2 w.r.t. the desynchronisation attacks they found. In [22], the authors create and use some temporary AS-keys in the way we employ K^*_{eNB}: compute that first and, if confirmed, then compute final AS keys. Whilst [22]'s X2 also involves the MME to add security, there are differences between our new X2 and the [22]'s X2, with the most notable one being our encryption of the HO Request and HO Command. On this We looked at the ProVerif modelling in [22] and re-checked it against more queries. We found that whilst their solution does indeed protect against desynchronisation attacks, it does not offer injective agreement on K^*_{eNB}; the ProVerif file with our verification of the X2 proposed in [22] is to be found at `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/X2/X2_ndevices.pv` where query 5 from X2 was tested. As we explained in the learning outcomes, our new X2 attains the injective agreement on K^*_{eNB} primarily due to the aforesaid encryption and addition of the T_1 nonce. We reflect this in the table below as well.

Table 3: (New) S1 and X2 – Verification Results Summary

	Non-Injective Sync	Injective Agreement on K*$_{eNB}$	Weak Agreement on K*$_{eNB}$	Post-Compromise Security on K^*_{eNB}
X2	Fail	Fail	Fail	Fail
Our new X2	Pass	Pass	Pass	Pass
X2 in [22]	Fail	Fail	Pass	Fail[4]
S1	Fail	Fail	Fail	Not checked
Our new S1	Pass	Pass	Pass	Not checked

6 Conclusion and Future Work

This paper analysed the LTE handover procedures called X2 and S1 in ProVerif, both from a key-agreement perspective and from a post-compromise security perspective. We found flaws in both protocols and aspects. We also show, outside of ProVerif, that the lack of post-compromise security in X2 affects S1 as well, which means that the secure communications in mobile networks, for a given UE, can be lost totally for a very long time (i.e., until they re-join the network). We propose patches for X2 and S1, which we prove secure in ProVerif. We also prove that previous patches (for other attacks) are insecure for our setting.

The next steps for this research will be to explore the verification results presented here in formal models for XN and NG, which are the upcoming 5G handovers. Along with formal verification for XN and NG, we plan to explore the feasibility of these attack in a real-world environment, on a test-bed scenario.

References

1. EMM Procedure 6. Handover without TAU - Part 2. X2 Handover. Technical report. Library Catalog: www.netmanias.com.

[6] Whilst the backward security on K^*_{eNB} fails on [22]'s X2, the one on AS keys passes.

2. EMM Procedure 6. Handover without TAU - Part 3. S1 Handover. Technical report. Library Catalog: www.netmanias.com.
3. Formal Analysis of Security of LTE Handovers: Pre- & Post-Compromise. long version of this paper.
4. 3GPP. LTE; Access Network (E-UTRAN); X2 Application Protocol (X2AP). Technical Specification (TS) 36.423, 3rd Generation Partnership Project (3GPP), 10 2012. Version 11.2.0.
5. 3GPP. LTE-Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application Protocol (S1AP). Technical Specification (TS) 36.331, 3rd Generation Partnership Project (3GPP), 09 2014. Version 12.3.0.
6. 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification. Technical Specification (TS) 36.331, 3rd Generation Partnership Project (3GPP), 01 2016. Version 13.0.0.
7. 3GPP. Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE. Technical Specification (TS) 36.403, 3rd Generation Partnership Project (3GPP), 07 2018. Version 15.4.0.
8. N. Ben Henda and K. Norrman. Formal analysis of security procedures in LTE - a feasibility study. In A. Stavrou, H. Bos, and G. Portokalidis, editors, *Research in Attacks, Intrusions and Defenses*, Lecture Notes in Computer Science, pages 341–361. Springer International Publishing.
9. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.
10. B. Blanchet. Security protocol verification: Symbolic and computational models. In *Principles of Security and Trust*, pages 3–29, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
11. B. Blanchet. Symbolic and computational mechanized verification of the ARINC823 avionic protocols. In *30th IEEE Computer Security Foundations Symposium (CSF'17)*, pages 68–82, Santa Barbara, CA, USA, Aug. 2017. IEEE.
12. B. Blanchet and B. Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. Research report RR-8906, Inria, Apr. 2016. Available at https://hal.inria.fr/hal-01306440.
13. P. B. Copet, G. Marchetto, R. Sisto, and L. Costa. Formal verification of LTE-UMTS and LTE-LTE handover procedures. 50:92–106.
14. P. B. Copet, G. Marchetto, R. Sisto, and L. Costa. Formal verification of LTE-UMTS handover procedures. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 738–744. IEEE.
15. V. Cortier, A. Filipiak, and J. Lallemand. Beleniosvs: Secrecy and verifiability against a corrupted voting device. In *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*, pages 367–381, 2019.
16. C. Cremers and S. Mauw. Operational semantics of security protocols. In *Scenarios: Models, Transformations and Tools*, pages 66–89. Springer, 2005.
17. C. Han and H. Choi. Security Analysis of Handover Key Management in 4G LTE/SAE Networks. *IEEE Transactions on Mobile Computing*, 13(2):457–468, Feb. 2014. Conference Name: IEEE Transactions on Mobile Computing.
18. L. Hirschi and C. Cremers. Improving automated symbolic analysis of ballot secrecy for e-voting protocols: A method based on sufficient conditions. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*, pages 635–650, 2019.
19. R. P. Jover. LTE security, protocol exploits and location tracking experimentation with low-cost software radio.
20. G. Lowe. *A Hierarchy of Authentication Specifications.* CSFW '97. IEEE Computer Society, USA, 1997.

21. N. Qachri, O. Markowitch, and J.-M. Dricot. A formally verified protocol for secure vertical handovers in 4g heterogeneous networks. 7(6):309–326.
22. B. Sun, J. Chu, L. Hu, H. Li, and G. Shi. A secure and effective scheme providing comprehensive forward security to lte/sae x2 handover key management. *KSII Transactions on Internet & Information Systems*, 11(9), 2017.

7 Attack Traces in Proverif

Due to space constraints, below, we exemplify only some ProVerif attack traces/

1. The attack trace corresponding to the **failure of Non-injective Synchronisation between UE and TN in X2** can be found at `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/Figures/failNonInjSync.png`.
   As per the *X2* section of Section 3, this is the failure of our Query 4 on X2, to be found in file `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/X2/X2.pv`.

2. The **failure of weak-agreement K^*eNB between UE and TN in X2** can be found at `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/Figures/failWeakAgree.png`.
   As per the *X2* section of Section 3, this is the failure of our Query 5 on X2, to be found in file `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/X2/X2.pv`.

3. The attack trace for the **failure of injective agreement K^*eNB between UE and TN in X2** can be found at `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/Figures/failInjAgreeX2.png`.
   As per the *X2* section of Section 3, this is the failure of our Query 8 on X2, to be found in file `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/X2/X2_ndevices.pv`.

4. In the **failure of injective agreement K^*eNB between UE, TN and MME in S1** you can see the attack trace at `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/Figures/failInjAgreeS1.png`.
   As per the *S1* section of Section 3, this is the failure of our Query 9 on S1, to be found in file `https://github.com/rhys-john-miller/4G_Handover_Verification/blob/main/S1/S1_ndevices.pv`.

7.1 S1 and X2 Application Protocols – Diagrams

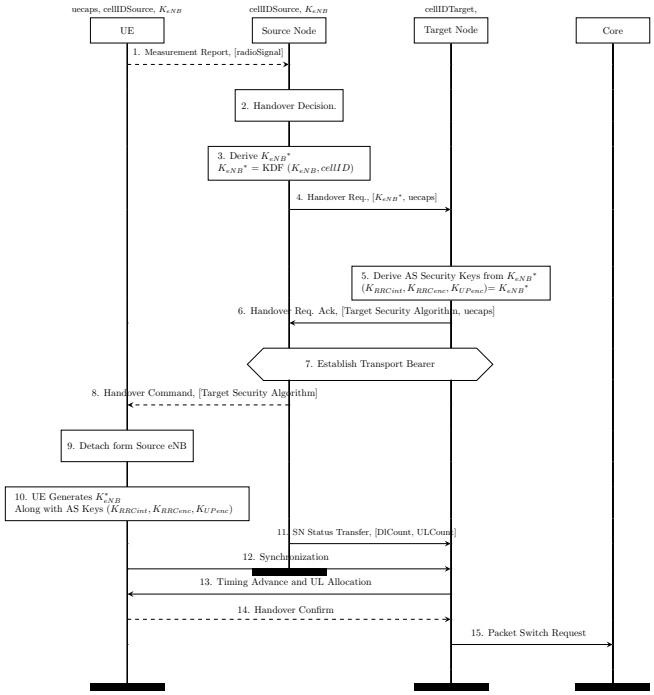

Fig. 5: The X2 Application Protocol

7.2 Our New S1 and X2 Application Protocols: Diagrams

Fig. 6: The S1 Application Protocol

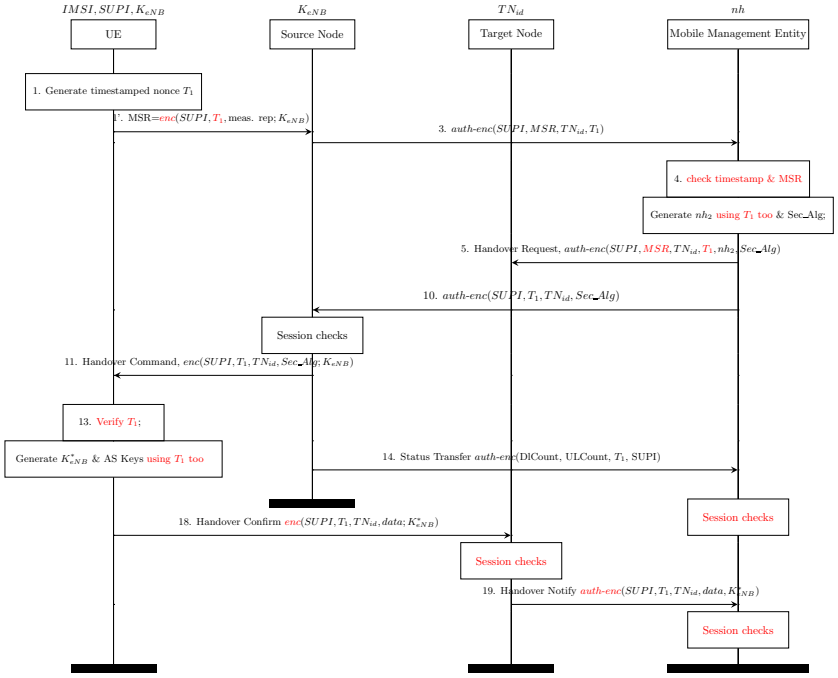

Fig. 7: New S1 Application Protocol

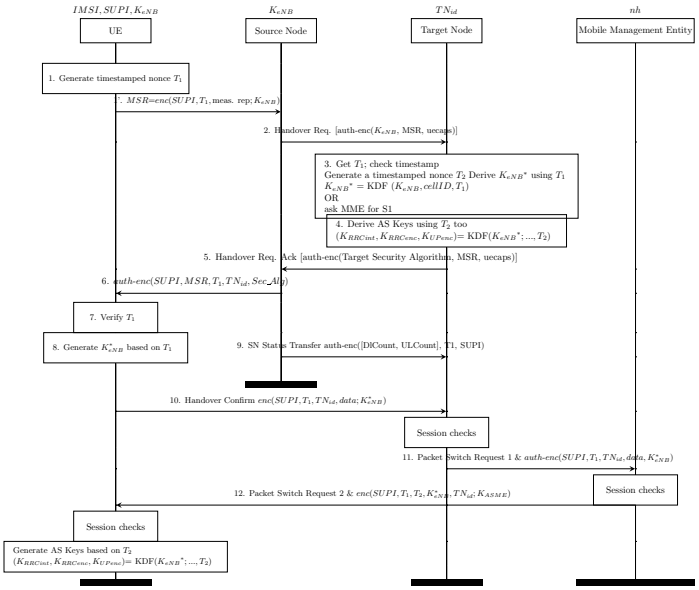

Fig. 8: New X2 Application Protocol – All Secure, incl. Post-Compromise Security

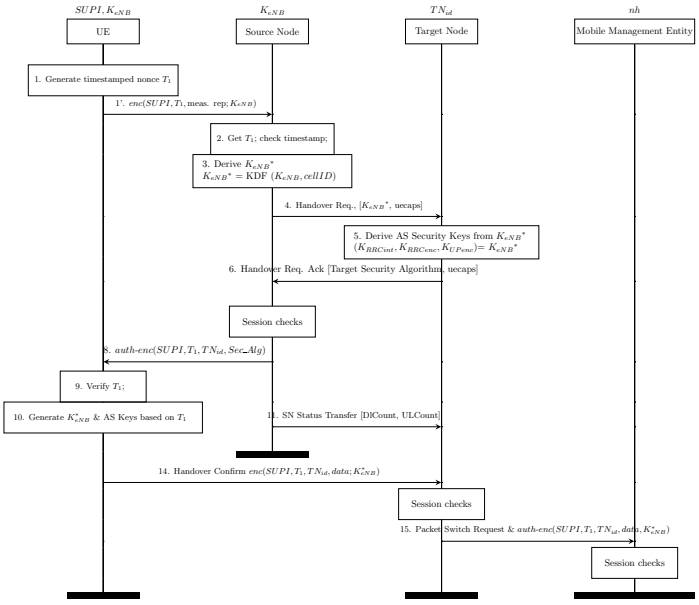

Fig. 9: New X2 Application Protocol – Key-Agreement Secure, without Post-Compromise Security