Systematic Improvement of Access-Stratum Security in Mobile Networks

Rhys Miller, Ioana Boureanu, Stephan Wesemeyer, Zhili Sun
University of Surrey, UK
rhys.miller@surrey.ac.uk, i.boureanu@surrey.ac.uk, s.wesemeyer@surrey.ac.uk & z.sun@surrey.ac.uk

Hemant Zope
Fraunhofer FOKUS, Germany, hemant.zope@fokus.fraunhofer.de

Abstract—**In mobile networks, the User Equipment (UE) secures some of the communication with its serving Radio Access Network (RAN) node ("base station") via a set of keys known as Access Stratum (AS) keys. Unfortunately, the level of secrecy of these keys varies with the mobile procedures re-establishing them. To improve the secrecy of the AS keys, we propose minimal changes to 5G & 4G handovers, i.e., the main AS-key establishment procedures. We show the minimality of our changes also via an implementation of one of our protocols in the 3GPP-compliant `Open5GCore` 5G testbed. We also systematically cross-compare standard handovers with our amended handovers using *`MobTrustCom`*: a framework to quantify especially trust but also communication complexity in mobile networks. Moreover, we use `Tamarin`, a formal security-protocol verification tool, to prove no loss of "classical" security yet an increase in AS-keys' secrecy brought by our improvements to handovers.**

Index Terms—**Formal Verification, Security, Mobile Communications**

1. Introduction

Modern 5th Generation Mobile Networks (5G) are proliferating fast, offering new services as well as promising better security compared to earlier generation networks [1]. One such security aspect deals with the application-layer traffic (or *Access Stratum* traffic in 5G terminology) between the User Equipments (UEs) and their serving RAN nodes. This AS channel is secured with a set of keys called *AS keys*. It is well-known [9] that, in some instances of the mobile procedures/protocols, the AS keys do not have *backwards security*, i.e., under some conditions, a party knowing the current value of a specific key might be able to compute another party's future AS keys. Not for AS keys, but for the keys that they are derived from, recent work [27] formally showed that rogue RAN nodes can make the lack of secrecy persist for longer, unhindered; no solutions were explored therein.

Access-Stratum Backwards Security. In this vein, we look at the backwards (in)security specifically of the AS keys, and, particularly, in the context of a set of mobile procedures called *handovers*. Handovers are initiated whenever the radio node, which has so far provided service to a user, needs to change, e.g., because the user is now out of its range. The most common handovers are: the *XN handover procedure (XN)* and *N2 handover procedure (N2)* in 5G, and their corresponding 4th Generation Mobile Networks (4G) versions: *X2 handover procedure (X2)* and *S1 handover procedure (S1)*.

Trust & Communication-Costs Playing into Security Gains. We are also interested in two aspects associated to security: trust and communication costs.

Firstly, with regards to trust, the UEs, i.e., devices with a Subscriber Identity Module (SIM), are becoming more complex while at the same time, the RAN-nodes are more diverse, with, for example, small cells appearing from various third-party providers [24], most of which run proprietary software. This arguably increases the number of potentially untrustworthy "players" present in a mobile network. Thus, to have better chances to catch and/or stop possible compromises, key-derivation should be done by a trusted party, or jointly by several parties: e.g., not just by a corruptible RAN-node, but rather by the trusted core or the core together with a RAN-node. It is thus important to quantify the different levels of trust placed on the many interacting components in mobile networks. In this work, we define notions of trust in mobile networks; we then apply these systematically to the handover protocols and other procedures, as well as our security-driven improvements of handovers, in order to cross-compare them amongst themselves while, at the same time, showing how security and trust attainment vary alongside.

Secondly, intuitively, a procedure with better trust levels may incur a higher communication costs than one with lower trust levels. So, especially when leveraging different trust values, quantifying also the communication costs in (any amendment of) any procedure is sensible. We therefore also define certain measures of communication costs in mobile networks, to evaluate our propositions for security improvements.

Our definitions of trust and communication measures in mobile procedures make up a framework which we call *`MobTrustCom`*. In this way, we are able to systematically look at how varying security gains in access-stratum mobile security require different levels of endowed trust in various mobile-network entities, as well as imply different communication costs of mobile procedures.

Contributions.

1. We propose a series of protocols which modify the XN/X2 handover procedure (X2) handovers in a systematic, as well as minimal way w.r.t. the 3GPP specifications [6, 8], in order to recover the backwards security for the AS keys.

2. To ascertain the minimality of our changes, we implement our of extensions to the XN/X2 protocol in the well-known and 3GPP-compliant `Open5GCore` 5G network testbed [22]; we show no loss of functionality or depreciation of efficiency.

3. We carry out formal verification of all our proposed improvements to XN/X2. We show that XN/X2 does not have backwards security of the AS keys and that our protocols gain that, whilst losing no other security guarantees.

4. We propose the *`MobTrustCom`* framework, motivated by measuring how much trust and what communication costs are associated to access-stratum security and potentially improvements to it in mobile networks. It provides fine-grained measures of trust, "split-trust" and explicit descriptions of communication costs in mobile networks. We apply `MobTrustCom` to some of the main protocols in 5G and 4G (handovers and other key-establishment protocols such as the Registration procedure (REG)) and compare our security-improving solutions amongst themselves, as well as against existing mobile procedures w.r.t. the `MobTrustCom`'s trust and communications cost.

2. Access-Stratum Keys & Handovers in 4G and 5G Networks

We now provide the background on certain aspects of mobile networks. First, we present a general overview of the network and the procedures relevant for this work (Section 2.1), and then we discuss the establishment and security of AS-keys (Section 2.2).

The details we provide next are based on the following 3rd Generation Partnership Project (3GPP) specifications: TS 23.401 [2], TS 33.401 [5], TS 36.423 [3] and TS 36.413 [6] (release 8 for 4G), and TS 23.316 [7], TS 23.501 [8], TS 23.502 [12], and TS 38.423 [10] (release 16 [11], for 5G). *Given this enumeration, we do not always re-cite all specifications throughout the paper.*

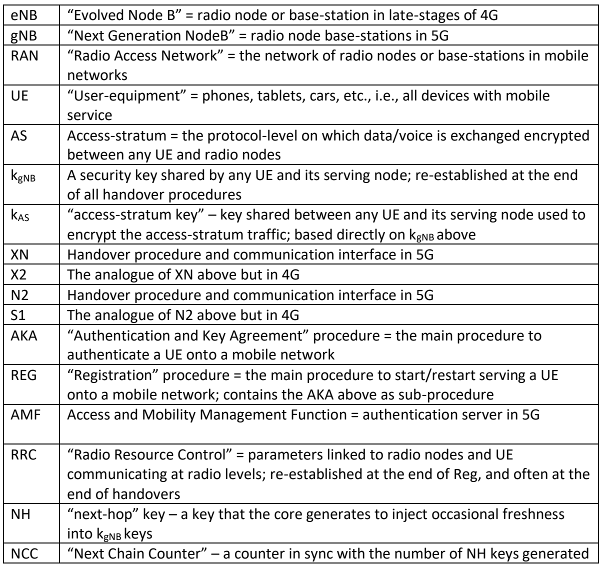
Glossary. We include below a summary of the most relevant acronyms and concepts herein:

2.1. The Network

In Figure 1, we give an overview of both the 4G and 5G network architectures.

Users. In 4G/5G networks, a UE receives service from its *operator*, whose main backend infrastructure we refer to as the *core network*. Each UE's SIM contains several long-term cryptographic secrets which it shares with the core. The ones relevant to this work are the K_{AMF} key (in 5G) and the K_{ASME} key (in 4G).

Radio Access Nodes. At any point, a UE is provided with mobile service through a radio "base-station" denoted *Evolved Node B (eNB)/ Next Generation NodeB (gNB)* in 4G/5G, respectively[1]. These nodes also communicate with the core network and other nodes. The combination of them forms the *RAN*.



eNB	"Evolved Node B" = radio node or base-station in late-stages of 4G
gNB	"Next Generation NodeB" = radio node base-stations in 5G
RAN	"Radio Access Network" = the network of radio nodes or base-stations in mobile networks
UE	"User-equipment" = phones, tablets, cars, etc., i.e., all devices with mobile service
AS	Access-stratum = the protocol-level on which data/voice is exchanged encrypted between any UE and radio nodes
k_{gNB}	A security key shared by any UE and its serving node; re-established at the end of all handover procedures
k_{AS}	"access-stratum key" – key shared between any UE and its serving node used to encrypt the access-stratum traffic; based directly on k_{gNB} above
XN	Handover procedure and communication interface in 5G
X2	The analogue of XN above but in 4G
N2	Handover procedure and communication interface in 5G
S1	The analogue of N2 above but in 4G
AKA	"Authentication and Key Agreement" procedure = the main procedure to authenticate a UE onto a mobile network
REG	"Registration" procedure = the main procedure to start/restart serving a UE onto a mobile network; contains the AKA above as sub-procedure
AMF	Access and Mobility Management Function = authentication server in 5G
RRC	"Radio Resource Control" = parameters linked to radio nodes and UE communicating at radio levels; re-established at the end of Reg, and often at the end of handovers
NH	"next-hop" key – a key that the core generates to inject occasional freshness into k_{gNB} keys
NCC	"Next Chain Counter" – a counter in sync with the number of NH keys generated

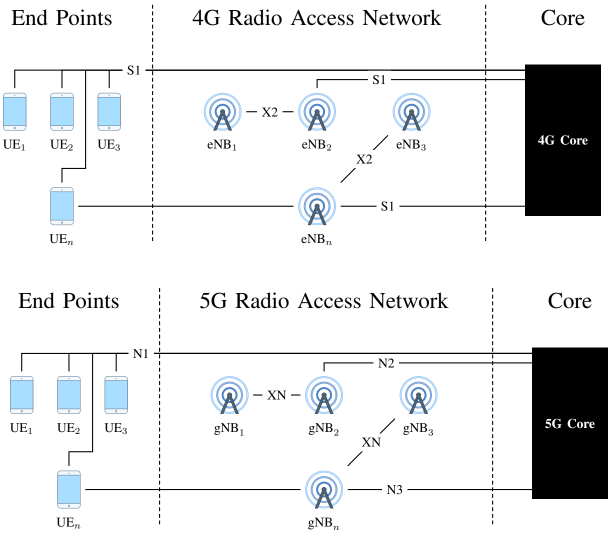


Figure 1. Telecommunication Network Diagrams

Communication Interfaces. All channels between the nodes themselves, as well as the nodes and the *core* are secure (i.e., confidential, integer and authenticated). This work is not concerned with this security aspect.

In Figure 1, some connections between entities are annotated by letters. These stand for communication interfaces: S1 interface (S1), X2 interface (X2), N2 interface (N2), XN interface (XN), etc. Each such interface largely corresponds to a *mobile procedure* being run over it.

Trends in Handover Interfaces/Procedures. We pause on handover interfaces/procedures, as they are the subject of this paper.

Firstly, we note that not all RAN-nodes are connected via an XN/X2 interface in 5G/4G, but all are connected to the *core* via an N2/S1 interface in 5G/4G. So, all nodes can execute N2/S1 *"handover" protocols* in 5G/4G, but not all can execute XN/X2 *"handover" protocols* in 5G/4G.

In fact, 4G's X2 was introduced to optimise the S1 handover procedure (S1), which involved costly communication with the core for deriving new cryptographic

[1] We refer to both eNBs and gNBs as *RAN-nodes* or simply *nodes*.

keys; instead, in X2, these just requires communication between the RAN nodes. Hence, in infrastructures updated after 2014, phones will always use X2 over S1 unless the involved RAN-nodes have not yet been updated to support X2. This carries forward similarly from 4G to 5G: that is, XN which is the new version of 4G's X2 is supported only by the newer infrastructures, whereas the N2 – the new version of 4G's S1 – is supported by all RAN-nodes.

So, especially since 5G pledged better quality-of-services, the trend is to move from S1/N2 to X2/XN, with any infrastructure update, as S1/N2 are disfavoured due to efficiency loss compared to X2/XN. See more at [13].

Access-Stratum (AS) Level. After any handover, a UE is served by a single RAN-node, communicating securely at the *AS level*: i.e., the interface over which the mobile-service messages are sent/received. *The backward (in)security of (re-)establishing the keys to secure the AS level is the subject of this work.*

2.2. Securing UE-to-RAN

We will now describe the role of the AS keys, their derivation and their backwards (in)security.

AS Keys. Actual mobile-service messages or *access-stratum messages* are exchanged securely between any one UE and a RAN-node. The two entities, using the *core*, first establish a secure channel by using the Authentication and Key Agreement (AKA) protocol, which is part of the *Registration (REG)* procedure. Messages sent over the channel between UE and its serving node are, in fact, secured with the so-called *Access Stratum keys*, which we broadly denote as K_{AS}. These AS keys[2] are first derived during REG [9, 5] and refreshed during several procedures such as handovers [10, 3].

AS Keys & The "Security Key" $K_{\mathbf{x}NB}$. The AS keys are derived using a so-called "security key" denoted K_{gNB} in 5G, and K_{eNB} in 4G; we call it $K_{\mathrm{x}NB}$. This $K_{\mathrm{x}NB}$ is (re-)established during a series of procedures and sub-procedures such as: the AKA procedure, the Handover procedures [10, 3] and the Radio Resource Control reconfigurations that happen when the UE "wakes" up and moves from Idle state to Connected state. The AS keys are computed from the $K_{\mathrm{x}NB}$ key as per Equation 1:

$$\mathsf{K}_{AS} = \mathsf{KDF}(K_{\mathrm{x}NB},\ \texttt{identifiers},\ \texttt{constants}), \quad (1)$$

where the "$\texttt{identifiers}$" generally identify a RAN-node, KDF stands for "key derivation function" and, in the current specifications [4], is a HMAC-SHA-256.

Figure 2 gives an overview of K_{AS} keys in the context of handovers and their secrecy over several parties therein. Figure 2 also aids with the next descriptions.

AS Keys' Computation. By Equation 1, a party who has the current $K_{\mathrm{x}NB}$ also knows or can easily retrieve the associated K_{AS}.

These keys are "ratcheted", meaning that a new $K_{\mathrm{x}NB}$(a.k.a $K^*_{\mathrm{x}NB}$) will be re-generated from keys pertaining to $K_{\mathrm{x}NB}$'s session, and $K^*_{\mathrm{x}NB}$ will generate the new K_{AS} (a.k.a. K^*_{AS}). See the "10:16 traffic" in Figure 2.

To determine which mobile-network parties know the current K_{AS} and if this entirely desirable or not, we need to understand better the aspects linked to Figure 2, described as **(a)** and **(b)** below.

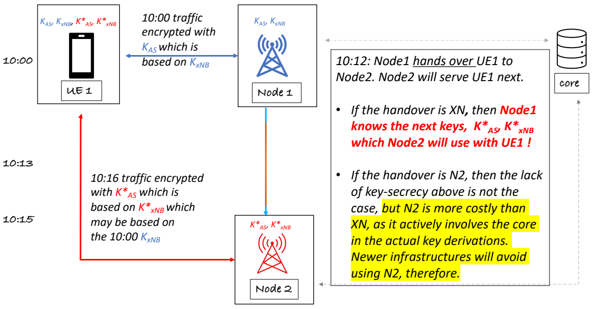
(a) Who has the latest $K_{\mathbf{x}NB}$. The security key $K_{\mathrm{x}NB}$ used to generate the latest AS key for a given UE and its serving RAN-node is known between the UE and its currently-serving RAN-node. The UE will always know the materials needed to generate the $K_{\mathrm{x}NB}$ and future versions, as the UE generates this key $K_{\mathrm{x}NB}$ itself. For the currently-serving RAN-node, the acquisition of the latest and future $K_{\mathrm{x}NB}$s is more involved.

(b) How is the latest $K_{\mathbf{x}NB}$ computed. The $K_{\mathrm{x}NB}$ keys are refreshed in the AKA/ REG procedure (see case **(b1)** below), in the Handover procedures (see case **(b2)** below) and during Radio Resource Control reconfigurations (see case **(b3)** below).

(b1) AS Keys' Computations in the Registration Procedure. The Registration procedure (REG) procedure is run between a UE and the *core*, and it is passively proxied by a RAN-node which will serve the UE thereafter. During REG, the *core* authenticates the UE and the procedure refreshes a series of keys at the UE's and the *core*'s end. The "lowest-level key" regenerated at the end of REG is called K_{AMF} in 5G and K_{ASME} in 4G, which is then used by both the UE and the *core* to derive $K_{\mathrm{x}NB}$. The *core* sends $K_{\mathrm{x}NB}$ to the node that proxied the REG procedure, and this node will then use this shipped $K_{\mathrm{x}NB}$to generate K_{AS}.

(b2) AS Keys' Computations in the Handover Procedures. A handover procedure is executed when a UE is being served by a node already, and the two share a current $K_{\mathrm{x}NB}$. Moreover, the node may or may not (depending on the procedures it ran in the past) have a fresh/unused *Next-Hop key (NH)*, sent previously to it by the *core* in another handover.

We will now look at the different handovers that can be run amongst a UE, its current serving *source node (sNode)*, its *target node (tNode)*, i.e., the node that will next serve the UE, and the *core*.


K_AS, K_xNB, K*_AS, K*_xNB
10:00
UE 1
10:00 traffic encrypted with K_AS which is based on K_xNB
K_AS, K_xNB
Node 1
core
10:13
10:15
10:16 traffic encrypted with K*_AS which is based on K*_xNB which may be based on the 10:00 K_xNB
K*_AS, K*_xNB
Node 2

10:12: Node1 hands over UE1 to Node2. Node2 will serve UE1 next.

- If the handover is XN, then **Node1 knows the next keys, K*AS, K*xNB which Node2 will use with UE1 !**
- If the handover is N2, then the lack of key-secrecy above is not the case, but N2 is more costly than XN, as it actively involves the core in the actual key derivations. Newer infrastructures will avoid using N2, therefore.

Figure 2. Access-Stratum Keys in Handovers: Birds-Eye View

(b2.1). In both the XN/X2 handovers, the sNode will use its latest $K_{\mathrm{x}NB}$ or, if available, NH to compute the so-called $K^*_{\mathrm{x}NB}$ and the sNode will (securely) send the $K^*_{\mathrm{x}NB}$ to the tNode. This $K^*_{\mathrm{x}NB}$ now becomes the current $K_{\mathrm{x}NB}$ used to compute the K_{AS} keys between the UE and the tNode.

At the end of XN/X2, the *core* sends (securely) a new NH to the tNode, which will use this NH value to compute

[2] There are several AS keys, which include K_{RRCINT}, K_{RRCENC}, K_{UPINT} and K_{UPENC}, used for integrity protection and encryption of the Radio Resource Control (RRC) and the User Plane (UP) traffic at the AS-level respectively. The only differences in their derivation is the key derivation function used (but not the actual seed-key) and some constants. This work is not concerned with their specific usage and we refer to them all as the ***access-stratum key(s)*** K_{AS}.

the next $K_{\text{x}NB}$ in future runs of the procedures in which $K_{\text{x}NB}$ may be refreshed (cases b1, b2, b3).

In XN/X2 handovers, we speak of *vertical key derivation (vkd)* or *horizontal key derivation (hkd)* respectively, when $K^*_{\text{x}NB}$ is computed from the latest *core*-issued NH value or when $K^*_{\text{x}NB}$ is simply ratcheted from the "node-resident" security key $K_{\text{x}NB}$. We summarise this distinction in the equations below:

$$\mathsf{K}^*_{\mathsf{xNB}} = \mathsf{KDF}(K_{\text{x}NB},\ tNode,\ \texttt{const}) \text{ hkd of } \mathsf{K}^*_{\mathsf{xNB}} \qquad (2)$$

$$\mathsf{K}^*_{\mathsf{xNB}} = \mathsf{KDF}(NH,\ tNode,\ \texttt{const}) \text{ vkd of } \mathsf{K}^*_{\mathsf{xNB}} \qquad (3)$$

We use XN^{vkd}, $X2^{vkd}$ and XN^{hkd}, $X2^{hkd}$ to differentiate between handovers using vertical/horizontal key derivation respectively.

(b2.2). In the N2 and S1 handovers, the derivation of a new $K^*_{\text{x}NB}$ always uses a stronger, *core*-coordinated instance of vertical key derivation. During these handovers, the *core* computes a new NH and sends it (securely) to the tNode. The tNode uses this NH to compute the $K_{\text{x}NB}$ and then derives the K_{AS} from this $K_{\text{x}NB}$. Similarly, the UE receives enough information from the *core* via the sNode to compute the $K^*_{\text{x}NB}$ and the K_{AS}, just as the tNode did.

Note 1. So, at *the end of* XN/X2 and *during* N2/S1 handovers, the tNode has a new NH. This NH will be used to derive $K_{\text{x}NB}$ within the very N2/S1 run, or $K^*_{\text{x}NB}$ — in a future execution by this node of XN/X2 or of other procedures like RRC reconfigurations.

(b3) AS Keys' Computations in the Idle-to-Connected UE State-Change. If a UE goes Idle whilst being served by one RAN-node and "wakes" up[3] from this state onto the same node (e.g., if the UE has not moved), a new $K^*_{\text{x}NB}$and a new K_{AS} will be re-computed by the UE and the node. In this case, if a fresh NH is present on the node, it will be used (in a vkd. for $K^*_{\text{x}NB}$) to generate the security key (see Note 1 above).

Note 2. In the Registration procedure and in variants[4] of N2/S1 handovers, K_{AMF}/K_{ASME} are used by the *core* and the UE to generate the new security key $K^*_{\text{x}NB}$. At that stage, these K_{AMF}/K_{ASME} keys, which are shared between the UE and the *core* and authenticate the former to the latter, are also re-generated.

AS Keys' Backwards Insecurity. Given Equations (1), (2), (3) and points **(b1)** - **(b3)**, it is clear that:

1) In XN/X2 handovers, a "detaching" source node can compute the future AS keys between the UE and the "attaching" target node (by point **b2.1** above).
   This is equivalent to saying that *the XN/X2 handovers do not have backwards security/secrecy of the access-stratum (AS) keys, relating to potentially corrupt (source) nodes*. This is known [9, 5], for both X2 in 4G and XN in 5G.

Note 3. There is generally only a case of *"one-hop" lack of AS keys' backwards security in XN/X2 handovers*, i.e., this lack can be remedied in two or more hops due to the target node applying vkd with a fresh NH value when renewing $K_{\text{x}NB}$.

2) The N2 and S1 handovers do have backwards security/secrecy of the AS keys, relating to potentially corrupt (source) nodes, by point **b2.2** above.
   These two points **b2.1** and **b2.2** on backwards security/secrecy of the AS keys in handovers are captured at one glance in Figure 2.
3) The REG/AKA procedures do have backwards security/secrecy of the access-stratum (AS) keys, relating to potentially corrupt nodes that served the UE prior to the current-serving node running the REG/AKA procedures, by point **b1** above.

Note 4: So, up to here, we see that N2/S1 handovers are more secure compared to XN/X2 w.r.t. the security of newly derived AS keys. But, due to the involvement of the core in these key derivations, N2/S1 require in fact additional as well as more "costly" messages than XN/X2 do. We explicitly quantify such extra costs in Section 6, see e.g. Table 3. In other words, the XN/X2 handovers are more efficient than N2/S1 handovers and hence preferred whenever the RAN-nodes support them, especially in newer infrastructures; the current trend is to add XN/X2 support and disfavour N2/S1 executions [13].

2.3. XN and X2 Handovers: Protocol Flows

Figure 3 describes the relevant details of XN/X2 handovers. It covers both horizontal and vertical key derivations, as per Equation 2 and Equation 3.

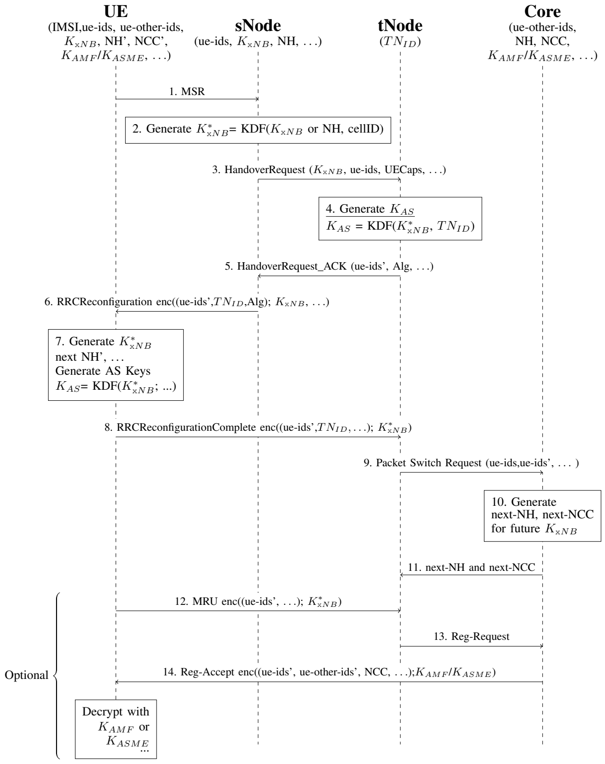

UE
(IMSI,ue-ids, ue-other-ids, $K_{\text{x}NB}$, NH', NCC', K_{AMF}/K_{ASME}, ...)
sNode
(ue-ids, $K_{\text{x}NB}$, NH, ...)
tNode
(TN_{ID})
Core
(ue-other-ids, NH, NCC, K_{AMF}/K_{ASME}, ...)
1. MSR
2. Generate $K^*_{\text{x}NB}$= KDF($K_{\text{x}NB}$ or NH, cellID)
3. HandoverRequest ($K_{\text{x}NB}$, ue-ids, UECaps, ...)
4. Generate K_{AS}
K_{AS} = KDF($K^*_{\text{x}NB}$, TN_{ID})
5. HandoverRequest_ACK (ue-ids', Alg, ...)
6. RRCReconfiguration enc((ue-ids',TN_{ID},Alg); $K_{\text{x}NB}$, ...)
7. Generate $K^*_{\text{x}NB}$
next NH', ...
Generate AS Keys
K_{AS}= KDF($K^*_{\text{x}NB}$; ...)
8. RRCReconfigurationComplete enc((ue-ids',TN_{ID}, ...); $K^*_{\text{x}NB}$)
9. Packet Switch Request (ue-ids,ue-ids', ...)
10. Generate
next-NH, next-NCC
for future $K_{\text{x}NB}$
11. next-NH and next-NCC
12. MRU enc((ue-ids', ...); $K^*_{\text{x}NB}$)
13. Reg-Request
14. Reg-Accept enc((ue-ids', ue-other-ids', NCC, ...);K_{AMF}/K_{ASME})
Optional
Decrypt with
K_{AMF} or
K_{ASME}
...

Figure 3. The XN/X2 Handover Protocols

We now explain Figure 3. UEs send reports on radio-signal strength called "measurement reports (MSR)" to their serving nodes. The XN/X2 procedure starts when a UE-serving sNode decides, usually based on these MSRs, to hand the UE over to a tNode. The sNode then generates

[3] If there is no user-plane activity to/from the UE for a certain time, the network "moves" the UE to an Idle state – to reduce the UE's power consumption. 5G introduced an additional *Inactive* state, which also triggers the re-computation of the AS keys.

[4] These variants are "N2/S1" with "K_{AMF}/K_{ASME}" re-keying.

the security key $K^*_{\mathrm{x}NB}$ using either a horizontal or vertical key derivation (Equation 2 or Equation 3) and, in message 3 of Figure 3, this new security key is sent to the tNode. At this stage, the tNode generates the new K_{AS} keys; this is as per equation Equation 1 with "identifiers" therein being the $tNode_{ID}$. In message 5, the tNode sends to the sNode the acknowledgment of the handover request which contains *ue-ids'*: i.e., tNode generates new ephemeral IDs for the UE to use after the handover[5]. In message 6, the sNode sends the "Radio Resource Control (RRC) reconfiguration message" to the UE containing details related to the tNode, UE's updated IDs, and the algorithms to use in order to do the right (type of) key-derivations. This message is encrypted with the current $K_{\mathrm{x}NB}$ that the UE and the sNode (still) share. The UE now generates the new security key $K^*_{\mathrm{x}NB}$; it will also regenerates other keys and counters, e.g., NH and Next Chain Counter (NCC). The UE then sends a "RRC reconfiguration complete" message (message 8 in Figure 3), encrypted with the new $K^*_{\mathrm{x}NB}$ key, to the tNode. This also serves as a key-confirmation step between the UE and the tNode, and it ensures they have computed/received the same $K^*_{\mathrm{x}NB}$ and will use the right IDs and (RRC) parameters. At this stage, the UE and the tNode effective share a new K_{AS} key, and could start communicating securely.

Yet, asynchronously, another part of the protocol continues, mainly between the tNode and the *core*. The tNode sends a "packet switch request" to the *core* signalling a handover is taking place and providing the necessary details (e.g., the IDs of the UE). Upon receipt, the *core* re-generates a series of parameters including counters, such as NCC and the NH key[6] labelled "next-NH". The *core* then sends the new NH and NCC to the tNode to use in the future computation of $K_{\mathrm{x}NB}$ by vkd. (see message 11 on Figure 3). At this stage, the tNode is confirmed by the core as the node to serve the UE now and the handover to the tNode by the sNode can be considered finished.

The Phase of Mobile Registration Updates (MRU): "*XN/X2_noMRU*". vs. "*XN/X2_MRU*". XN/X2 handovers often continue with messages 12 to 14 as per Figure 3, i.e., with a "Mobility Registration Update (MRU)". According to TS 23.502 [12] (Sections 4.9 and 4.2.2.), a "Mobility Registration Update" is akin to a shorter re-registration of the UE. It occurs at the end of a XN/X2 handover, if a) the UE has to be served/authenticated by a different server of the core (e.g., a new-area 5G Core Access and Mobility Management Function (AMF)), or b)"the UE needs to update its capabilities or protocol parameters", which have been negotiated in the Registration procedure, as well as in two other cases of no interest here. In 5G, these capabilities and protocol parameters include details such as settings and updates linked to "PDU (Protocol Data Unit)" sessions. These PDU sessions facilitate the management of complex subscribers' plans, their billing and quality-of-services related to UEs getting onto the the data network via all-to-often specific, local gateways/User Plane Function (UPF)s, to allow for even preferential treatment for certain services included in the subscribers' plans, e.g., WhatsApp traffic, and not for others. So, in 5G (especially in what is called "SA (standalone) 5G", i.e., full-capability 5G), this MRU part (messages 12 to 14 on Figure 3) occurs after handovers more frequently than in 4G.

In message 14, the *core* sends to the UE freshly-generated parameters such as new ephemeral IDs (e.g., new 5G-TMSI, 5G-GUTI) by which the core identifies the UE, denoted *ue-other-ids'* in Figure 3. This message is encrypted with K_{AMF} in 5G and with K_{ASME} in 4G.

To consider the protocol finished with respect to cryptographic key-agreement as well as key/parameter confirmation not between 2-by-2 parties but between all 3 parties involved (i.e., UE, tNode and *core*), the MRU phase would be necessary. In a sense, the extra complexity added by 5G vs 4G, whereby this MRU phase is more frequent, improves the multi-party security of XN/X2.

Note that when we explicitly require that an XN/X2 execution includes an MRU phase, we use the shorthand *XN/X2_MRU*. Conversely, when an XN/X2 execution is not followed by an MRU phase, we use the shorthand *XN/X2_noMRU*. If we simply write *XN/X2*, then –as per the 3GPP specifications– the corresponding XN/X2 execution may or may not contain an MRU phase.

Note 5: **Crucial Handover Exchanges for UE-tNode-*core* relating to AS Reconfiguration.** The AS-keys' establishment between the UE, the tNode and *core* is the focus of this work. To this end, messages 8, 9, 11, as well as 14, are the crucial messages that bring these 3 parties (i.e., UE, tNode, *core*) in agreement over the AS-channel reconfiguration.

3. Threat & Trust Model

Our new designs are adding provable AS backwards-security w.r.t. the following realistic threat model:

- the core and UEs are honest;
- the RAN nodes are honest-but-curious, i.e., they follow all the protocols but might use their onboard implementation/specification to execute legal options such as to downgrade the security [7];
- we assume a network attacker that listen onto and inject messages into channels, but all communication channels that are secure stay secure and attackers do not subvert cryptography.

This is a realistic level of threat, i.e., we do not assume fully malicious RAN-nodes, as that is less realistic to 3GPP (to whom we disclosed our findings). Moreover, for backwards secrecy in handovers, which is the property we will improve here, honest-but-curious is sufficient, as Section 2 explained.

In Section 5, this threat model is cast into the well-known Dolev-Yao (DY) model [21], which is slightly stronger (e.g., corrupted parties other than the one running the current protocol-sessions can be executed by the DY attacker, in parallel with the ones under security scrutiny). Therefore, all that is proven secure in the DY, formal model is secure in our threat model as given above.

[5] One of these is the C-RNTI, but we abstract this away here.

[6] Note that the *core* and the UE are in sync with respect to the NCC and Next-Hop key (NH) values, but the UE is one step behind on their derivation/increment, which is why in our figure we use "NH' " and "NH" for the UE and the *core*, respectively.

[7] Specifically, the RAN nodes may exploit the lack of backward security of the AS keys in XN/X2, to keep executing XN/X2 such as to ensure that future traffic remains readable to them when it should not be if, e.g., N2/S1 was run instead.

The threat model given above is kept throughout the paper. In Section 6 alone, we also add elements of trust measures to it, as follows:

- any UE is less trusted than any RAN node, which in turn is less trusted than the core;
- these trust measures compose under addition as would be expected: e.g., the core and a RAN node running together a procedure are more trustworthy than the core and UE running a version of that procedure together, etc.

Our trust model is also realistic: i.e., it is easier to corrupt a phone than a RAN node, which, in turn, is easier to corrupt than the core. In fact, in Section 6 we quantify the endowments/costs needed to gain trust: a lot is required to trust a UE, less to trust a base-station, while trust in the core is almost taken for granted.

4. Improving Access-Stratum Keys' Security in 4G/5G Handovers

From Note 3 inSection 2.2, XN/X2 protocols (be it $XN/X2^{vkd}$ or $XN/X2^{hkd}$, i.e., no matter the key-derivation options) do not have backwards security of AS keys: the detaching source node sNode can compute the AS keys to be shared between the attaching target node tNode and the UE.

We now propose a series of enhancements to XN/X2 which not only recover this key-secrecy loss, but do so via minimal changes to XN/X2 (e.g., adding a 32-bit nonce to one or more existing XN/X2 messages) to be as backwards-compatible as possible.

Main Idea in All Our Protocols. To recover AS-keys' backwards security, in principle, the tNode and/or the UE could generate at most one new secret each, to input into the AS keys' computation, and then find a way to share the newly-generated secret(s), without the sNode being able to obtain said secret(s). This would clearly break the chain of backwards insecurity of the AS keys w.r.t. sNode. Hence, we categorise protocols that achieve this aim of re-provision of AS-keys' backwards security into three classes, depending on which entities (i.e., UE or RAN node, or both) are actively involved in the augmented AS-key generation:

1) *Enhanced XN/X2 with Added AS Backwards Security via RAN Operations*: The tNode alone determines the final value of the AS keys via a locally generated secret and securely sends this secret to the *UE* across the network (without the sNode getting it).
2) *Enhanced XN/X2 with Added AS Backwards Security via UE Operations*: The *UE* alone determines the final value of the AS keys via a locally generated secret and securely sends this secret to the tNode across the network (without the sNode getting it).
3) *Enhanced XN/X2 with Added AS Backwards Security via UE-RAN Shared Operations*: The tNode and the UE both contribute with new secrets to the AS keys and securely exchange their new secrets (without the sNode getting them).

We give one concrete protocol in each of the classes above, in such a way that all our concrete protocols require minimal and backwards-compatible changes to XN/X2, i.e., we introduce no new messages in XN/X2's 3GPP specifications, and we try to leverage existing fields in these messages.

In Section 5, we formally verify that all our proposed protocols do indeed provide backwards security of AS keys. In Section 6, we systematically quantify the different efficiency and complexity trade-offs of our XN/X2 protocols vs. existing comparable protocols, using our `MobTrustCom` framework. In Section 7, we show a 5G implementation of our protocol realising " Enhanced XN/X2 with Added AS Backwards Security via RAN Operations ", called $XN/X2^{RANc}$. In this way, we also concretely re-ascertain how close it runs to the current XN/X2 5G-standard: i.e., in the implementation, we only use existing 5G messages and fields.

4.1. A Realistic, Enhanced XN/X2 with Added AS Backwards Security via RAN Operations

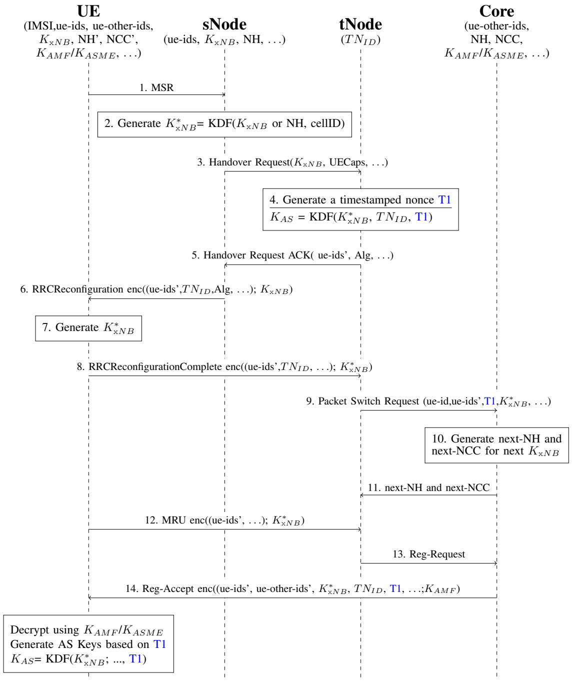

UE
(IMSI,ue-ids, ue-other-ids, K_{xNB}, NH', NCC', K_{AMF}/K_{ASME}, ...)
sNode
(ue-ids, K_{xNB}, NH, ...)
tNode
(TN_{ID})
Core
(ue-other-ids, NH, NCC, K_{AMF}/K_{ASME}, ...)
1. MSR
2. Generate K^*_{xNB}= KDF(K_{xNB} or NH, cellID)
3. Handover Request(K_{xNB}, UECaps, ...)
4. Generate a timestamped nonce T1
K_{AS} = KDF(K^*_{xNB}, TN_{ID}, T1)
5. Handover Request ACK(ue-ids', Alg, ...)
6. RRCReconfiguration enc((ue-ids',TN_{ID},Alg, ...); K_{xNB})
7. Generate K^*_{xNB}
8. RRCReconfigurationComplete enc((ue-ids',TN_{ID}, ...); K^*_{xNB})
9. Packet Switch Request (ue-id,ue-ids',T1,K^*_{xNB}, ...)
10. Generate next-NH and next-NCC for next K_{xNB}
11. next-NH and next-NCC
12. MRU enc((ue-ids', ...); K^*_{xNB})
13. Reg-Request
14. Reg-Accept enc((ue-ids', ue-other-ids', K^*_{xNB}, TN_{ID}, T1, ...;K_{AMF})
Decrypt using K_{AMF}/K_{ASME}
Generate AS Keys based on T1
K_{AS}= KDF(K^*_{xNB}; ..., T1)

Figure 4. `XN/X2`^{RANc} – a New Enhanced XN/X2 with Added AS Backwards Security via RAN Operations

Our `XN/X2`^{RANc} Protocol. We now present our first enhancement of the XN/X2 protocol, $XN/X2^{RANc}$. Again, the main idea is to provide backwards security of AS Keys by adding at least one extra ingredient to the AS-keys' derivation, which is and remains secret to the sNode during the XN/X2execution, and –if possible– to do so in a manner which is as aligned with the current XN/X2 specifications as possible.

Figure 4 shows our `XN/X2`^{RANc} protocol, in which this secret ingredient, a 4-byte nonce T_1, is generated by the tNode. This nonce T_1 is then input into the AS-keys' computation, as per step 4 on Figure 4.

Compared to XN/X2 (see Figure 3), just two messages (message 9 and message 14) are changed to send this nonce across from the tNode to the UE. Clearly, for the

UE to be able to compute the AS keys –now based on the T_1 nonce– as well, T_1 needs to be sent from the tNode to the UE, but not via the sNode to prevent the sNode from computing the AS keys. Given the communication links in handovers, T_1 can therefore only go from the tNode to the UE via the *core*, which can encrypt it with a key shared just between the *core* and the UE. Figure 4 reflects this in messages 9 and message 14, which now contain T_1 as described here.

Note that XN/X2^{RANc} enhances XN/X2in the same uniform way irrespective of whether horizontal or vertical key derivations are used inside XN/X2.

XN/X2^{RANc} : Proposition & Adoption. If our protocol were to be adopted, then a new, bespoke field for T_1 will need to be added to the 3GPP specifications, ideally inside the "security capabilities" and/or "security context" of the UE (see Section 4.2.2 of [12]), as the nonce carried within is an element contributing to the AS-keys' security.

4.1.1. XN/X2^{RANc} : Security.

XN/X2^{RANc} 's Threat Model. In all our premises and designs, we distrust sNodes: they may/will act on their ability to compute the AS keys between *UE* and tNode to decrypt the AS traffic between *UE* and the new-serving tNode. In XN/X2^{RANc} , we trust the *core* to act honestly; if the *core* were malicious, then it could obviously send T_1 to the sNode thus breaking the security of our protocol.

XN/X2^{RANc} – Provable AS Backwards-Secrecy. In Section 5, we formally show (in the Tamarin prover [26]) that XN/X2^{RANc} does add backwards security of AS keys to $XN/X2$ by just adding a 4-byte nonce to only two existing XN/X2 messages.

4.1.2. XN/X2^{RANc} : Efficiency w.r.t. 3GPP Messages.

XN/X2^{RANc} - Low Communication Impact. Note 5 in Section 2.2 indicates messages 8, 9 and 14 in Figure 3 are the best candidates for implementing XN/X2^{RANc} efficiently and sending T_1 from the tNode to the UE. As seen in Figure 4, XN/X2^{RANc} overloads only messages 9 and 14 of the XN/X2 handover, adding just one 4-byte field to each of these two messages.

However, note that since we use message 14 to deliver T_1 to the UE in order for it to compute the AS-keys, our XN/X2^{RANc} makes the MRU phase of XN/X2 obligatory after each handover execution. That is, a run of XN/X2^{RANc} is always a run of XN/X2_MRU. Thus, in practice, our XN/X2 would lead to MRUs happening more frequently in the mobile network.

The average increase in communication complexity w.r.t. mobile-network relevant procedures will be discussed systematically and in depth in Section 6.

XN/X2^{RANc} – Cost w.r.t. 3GPP Messages. When comparing XN/X2_MRU in Figure 3 with XN/X2^{RANc} in Figure 4, we see that –apart from the generation of T_1 inside the AS-key generation and from messages 9 and 14 containing this extra nonce– the XN/X2^{RANc} protocol is identical to the XN/X2_MRU protocols.

Only to evaluate communication complexity, we now describe where inside message 9 ("Path Switch Request") and message 14 ("MRU.Registration Accept") our added nonce T_1 could be inserted in a way in-keeping with the 3GPP XN/X2 standard [12]. For these matters, we understandably focus on XN (not X2), i.e., on 5G and not on 4G specifications.

Firstly, we chose this T_1 nonce to be a 32-bit string, since this size is in line with many other 3GPP fields [12].

Secondly, we note that neither the "Path Switch Request" nor the "Registration Accept" in the 3GPP specifications [12] have any "reserved for future use" fields. Thus, to send T_1 in a way in-line with the standard, one would be extending and using fields in the "Path Switch Request" and in the "MRU.Registration Accept" of the XN protocol.

- In the "Path Switch Request", two sets of UE-related items, which are always sent, are the UE's "security capabilities" and the UE's "user-location information" (see Section 4.2.2 of [12]). Sending T_1 in our of these fields is akin complexity-wise to "overloading" a "not-yet-filled-in" field/sub-field of these items, of 32 bit length or more, inside the "Path Switch Request" (message 9). For a given UE, there is always more than one such field not yet filled-in: e.g., unused algorithms in "security capabilities", timestamps inside "user-location information", etc.
- In the "MRU.Registration Accept", two possible sets of UE-related items are the UE's "PDU sessions list" and the UE's "user-location information" (see Section 4.2.2 of [12]). Again, to measure complexity in the absence of a bespoke field for our T_1, one could simulate this sending by using any "not-yet-filled-in" field/sub-field of these items, of 32 bit length or more, inside the "MRU.Registration Accept" (message 14). For a given UE, there is always more than one such field not yet filled-in: e.g., data in an unused PDU sessions, etc.

XN/X2^{RANc} – Negligible Impact on Mobile Traffic. The above discussions show that our lift of XN/X2 to XN/X2^{RANc} via the addition of a 4-byte nonce would impact very little on the overall communication costs of a general XN/X2 (or XN/X2_MRU) run executing with a number of normal fields (e.g., timestamps, algorithms, PDU sessions) which varies from UE to UE. In other words, an XN/X2-capable UE with several security features, or with several PDU sessions (which likely already runs MRUs after each handover) costs the network in terms of communication complexity hardly any less than a XN/X2^{RANc} -capable UE, which just sends an additional nonce in one such sub-field.

The above arguably makes XN/X2^{RANc} a realistic enhancement of XN/X2.

In Section 7, we show a concrete implementation of our XN/X2^{RANc} protocol on top of the XN 5G-procedure.

4.2. Enhanced XN/X2 with Added AS Backwards Security via UE Operations

Our XN/X2^{UEc} Protocol. Figure 7, in Appendix A, shows our protocol, XN/X2^{UEc} , which provides backwards security of AS keys in XN/X2 by using the UE to provide the required secret ingredient for the AS key generation.

The Idea of XN/X2^{UEc} . This time the UE generates the new 4-byte nonce, T_2, shown in red in Step 1 of Figure 7), to be "injected" into the AS keys' computation.

The UE needs to send T_2 to the tNode, without the sNode learning its value. This is achieved by encrypting T_2 with the UE's K_{AMF}/K_{ASME} key and sending it to the tNode which then forwards it to the *core* for decryption as the *core* shares the K_{AMF}/K_{ASME} key with the UE.

Concretely, in $\texttt{XN/X2}^{UEc}$, message 8 (RRC-Reconfiguration-Complete) is used to send the encrypted secret, T_2, to the tNode. So, while the sNode can, in theory, intercept and read this message, it cannot decrypt the secret as it is encrypted with the K_{AMF}/K_{ASME} key, which is only shared between the UE and the *core*. The tNode cannot decrypt it, either, but it can pass the encrypted T_2 on to the *core* for decryption, adding it to message 9 (Packet-Switch-Request). The *core* decrypts T_2 and sends it together with the new NH and NCC back to the tNode (in message 11). The tNode can now compute the new AS keys.

$\texttt{XN/X2}^{UEc}$: Proposition & Adoption. If our protocol were to be adopted, then a new, bespoke field for T_2 will need to be added to the 3GPP specifications, ideally inside the "security capabilities" and/or "security context" of the UE (see Section 4.2.2 of [12]), as the nonce carried within is an element contributing to the AS-keys' security.

4.2.1. $\texttt{XN/X2}^{UEc}$: Security.

The main discussions on the AS keys' security of $\texttt{XN/X2}^{UEc}$ is as for $\texttt{XN/X2}^{RANc}$, in Section 4.1.1.

Advantages of $\texttt{XN/X2}^{UEc}$ over $\texttt{XN/X2}^{RANc}$. As we will see formally in Section 6, $\texttt{XN/X2}^{UEc}$ has a better level of "joint/global" trust than $\texttt{XN/X2}^{RANc}$. Intuitively, this is because in $\texttt{XN/X2}^{RANc}$ the responsibility of AS-keys' generation remains with the RAN (like in XN/X2, we start from the premise of dis-trusting certain RAN nodes); meanwhile, in $\texttt{XN/X2}^{UEc}$, the *UE* also has its "say" in the AS-keys' generation, so RAN node cannot control the AS-keys' secrecy, even if they were to collude *à la* [27].

$\texttt{XN/X2}^{UEc}$ – Cost w.r.t. 3GPP-Specified Messages. To measure communication complexity, the way could override existing fields in XN/X2's 3GPP specifications is largely identical to the discussions above, in Section 4.1.2. Computation-wise, an extra encryption by the UE (i.e., $enc(T_2; K_{AMF}/K_{ASME})$) and decryption by the core (of the same $enc(T_2; K_{AMF}/K_{ASME})$)) are implied.

Advantages of $\texttt{XN/X2}^{UEc}$ over $\texttt{XN/X2}^{RANc}$. Two things are worth noting w.r.t. the advantages of $\texttt{XN/X2}^{UEc}$ over $\texttt{XN/X2}^{RANc}$ in the context of both being extensions of XN/X2. Firstly, whilst $\texttt{XN/X2}^{RANc}$ imposes that each execution be followed by an MRU phase (i.e., $\texttt{XN/X2}^{RANc}$ is a type of XN/X2_MRU), the $\texttt{XN/X2}^{UEc}$ protocol does not do so: $\texttt{XN/X2}^{UEc}$ is a extension of XN/X2 preserving the optionality of the MRU phase, i.e., when XN/X2 executes as XN/X2_noMRU, $\texttt{XN/X2}^{UEc}$ executes as $\texttt{XN/X2\_noMRU}^{UEc}$, and when XN/X2 executes as XN/X2_MRU, $\texttt{XN/X2}^{UEc}$ executes as $\texttt{XN/X2\_MRU}^{UEc}$. This means that, on average, w.r.t. mobile procedure executing, $\texttt{XN/X2}^{UEc}$ adds no overheads coming from extra MRUs. This will be discussed systematically in depth in Section 6. Secondly, note that the UE and the tNode get to share a new AS-key earlier on in the handover execution, compared to $\texttt{XN/X2}^{RANc}$. This is also in line with the normal XN/X2.

4.3. Additional New Handovers with Backwards Security of AS Keys

Enhanced XN/X2 with Added AS Backwards Security via UE-RAN Shared Operations – Our $\texttt{XN/X2}^{RANc-UEc}$ Protocol. Our final protocol is shown in Figure 8 in Appendix A. It improves XN/X2 to have AS keys' backwards security by having both the UE and the tNode generate new secrets to be "injected" into the AS keys' computation in such a way that the sNode does not learn these secrets.

The Idea of $\texttt{XN/X2}^{RANc-UEc}$. Simply put, $\texttt{XN/X2}^{RANc-UEc}$ combines both $\texttt{XN/X2}^{RANc}$ and $\texttt{XN/X2}^{UEc}$ into one protocol. Two 4-byte nonces, T_2 and T_1, are generated by the UE and the tNode in Step 1 and Step 5 (shown in red and blue in Figure 8), respectively. As before, both nonces need to be moved to the other party in such a way that the sNode does not learn their values. As in $\texttt{XN/X2}^{UEc}$, the RRCReconfigurationComplete request (message 9 in Figure 8) carries the encrypted T_2 while the PacketSwitch request (message 10), which is sent by tNode to the *core*, contains both T_1 and the encrypted T_2. The *core* decrypts T_2 and sends it back to the tNode as part of message 11, at which point the tNode can compute its AS keys. The *core* also sends T_1 as part of the Registration-Accept (message 14) to the UE, so that the UE can compute the same AS keys as the tNode.

4.3.1. $\texttt{XN/X2}^{RANc-UEc}$: Security & Efficiency.

The arguments here follow as per the cases of $\texttt{XN/X2}^{RANc}$ and $\texttt{XN/X2}^{UEc}$ above.

The communication costs of $\texttt{XN/X2}^{RANc-UEc}$ are clearly the highest of our proposed protocols, as it includes all the changes from both $\texttt{XN/X2}^{RANc}$ and $\texttt{XN/X2}^{UEc}$. However, the changes to XN/X2 are still fairly minimal, and can be made by overriding current field in 3GPP-specified messages of XN/X2.

$\texttt{XN/X2}^{RANc-UEc}$ arguably provides an even better trust-level than either $\texttt{XN/X2}^{RANc}$ or $\texttt{XN/X2}^{UEc}$ (as even more parties get to know how to compute the AS keys), albeit at a slightly increased communication costs. Again, this will be discussed in detail in Section 6.

Note 6. It is important to mention our protocols $\texttt{XN/X2}^{RANc}$, $\texttt{XN/X2}^{UEc}$ and $\texttt{XN/X2}^{RANc-UEc}$ only aim to improve the *backwards security* of the AS keys and are not meant to achieve better multi-party key-agreement for the AS keys. Whilst this latter strengthening is not our aim, it is, nevertheless, trivial to implement: e.g., in $\texttt{XN/X2}^{RANc-UEc}$, the core would need to send not only T_1, but also T_2 back to the UE thus allowing the UE to verify the various secrets used in the computation of the AS keys. Consequently, this incurs therefore even further (yet still negligible) communication complexity. Please see Note 9 in Section 6 for more details on this aspect.

5. Formal Verification

In Section 4, we demonstrated that our protocols, $\texttt{XN/X2}^{RANc}$, $\texttt{XN/X2}^{UEc}$ and $\texttt{XN/X2}^{RANc-UEc}$, implement minimal "patches" to XN/X2 to provide backwards security of the AS keys and without compromising existing key-establishment security. We now formally

	Lemma	Meaning	Proving Status	Time
1	correctness-no-T1	Full executability of our handover model with a single handover.	Proved	17.34s
2	correctness-with-T1	Full executability of $\texttt{XN/X2}^{RANc}$ with a single handover.	Proved	16.49s
3	secrecy-of-ASKey-no-T1	in XN, backwards secrecy of AS keys w.r.t. tNode.	Falsified	24.27s
4	secrecy-of-ASKey-with-T1	in $\texttt{XN/X2}^{RANc}$, backwards secrecy of AS keys w.r.t. tNode.	Proved	57.88s
5	secrecy-of-K^*_{xNB}-no-T1	in XN, backwards secrecy of K^*_{xNB} w.r.t. tNode.	Falsified	28.43s
6	secrecy-of-K^*_{xNB}-with-T1	in $\texttt{XN/X2}^{RANc}$, backwards secrecy of K^*_{xNB} w.r.t. tNode.	Falsified	33.84s
7	injective-agreement-ASkey-UE-tNode-no-T1	in XN, the UE and the tNode compute the same AS key in all sessions run together.	Proved	43.87s
8	injective-agreement-ASkey-UE-tNode-with-T1	in $\texttt{XN/X2}^{RANc}$, the UE and the tNode compute the same AS key in all sessions run together.	Proved	48.96s
9	injective-agreement-K^*_{xNB}-tNode-sNode	in both protocols, the sNode and the tNode compute the same K^*_{xNB} in all sessions run together.	Proved	13.64s
10	injective-agreement-K^*_{xNB}-sNode-UE	in both protocols, the sNode and the UE compute the same K^*_{xNB} in all sessions run together.	Proved	17.73s

TABLE 1. Main Verification Results

demonstrate these security statements, via formal verification in the Tamarin prover [26], a popular verification tool for security protocols in the DY model [21].

Threat Model. We follow the threat model presented in Section 3. That is, we assume honest UEs and honest core, but honest-but-curious RAN-nodes, i.e., all nodes follow the protocols but might use their "knowledge" to exploit the lack of backward security of the AS keys.

Recall that a honest-but-curious source node running XN/X2 will get to unwarrantedly know a next AS key because this source node just computed the K_{gNB} for the target node. We model these unwarranted AS-keys' computations by honest-but-curious RAN-nodes, as an "out of band" leakage to the DY attacker of a computed K_{gNB} key, and we show that the corresponding K_{AS} is indeed computable by the attacker: i.e., all the remaining parts of this computation are also derivable.

We do not model any other long-term key leakage, e.g., leaking K_{AMF}/K_{ASME} from the UE or the *core*, as this is not part of the honest-but-curious threat model.

Models for the XN/X2 Handover and our Extensions. We modelled the standard XN/X2 handover in Tamarin as well as all our protocols, $\texttt{XN/X2}^{RANc}$, $\texttt{XN/X2}^{UEc}$ and $\texttt{XN/X2}^{RANc-UEc}$, all including the MRU phase.

In this section, we will focus on the modelling of $\texttt{XN/X2}^{RANc}$ and refer the reader to our models for the other variants.

All our Tamarin files are available at https://fmsec.github.io/5gtechsec.github.io/.

Entities & Channels. Our models do not restrict the number of UEs, sNodes or tNodes, but we only allow one *core* which is a reasonable abstraction for our purposes. Communication between the sNode, tNode and *core* is via a secure channel, i.e., the DY attacker cannot interfere with message on the core network.

The channel between the UE and the other parties is not modelled as secure *a priori*, but, where required, messages to and from the UE will be sent encrypted using the appropriate keys, e.g., K_{xNB}, K^*_{xNB} and $\mathsf{K}_{AMF}/\mathsf{K}_{ASME}$.

Modelling XN/X2 & $\texttt{XN/X2}^{RANc}$. Our $\texttt{XN/X2}^{RANc}$ model encodes both the original, "standard" XN/X2 protocol as well as our "enhancements" in one single model. The variant executed during a proof is determined by setting a "flag" in the statement of the lemma to be proven: *(T1='no_T1'* indicates the standard protocol while *not(T1='no_T1')* is used for $\texttt{XN/X2}^{RANc}$. Specifying this flag forces Tamarin –using pattern matching or explicit equality checks– to include those rules that provide the state transitions for the relevant protocol while, at the same time, explicitly excluding rules that provide support for the other respective variant.

Modelling "one-hop" backwards (in)security of the AS key. Unlike other prior handover models, we model and focus on the key derivation of AS keys and their security. We thus made the following modelling choices:

1) Recall that the AS keys only differ by some constants passed to the KDF. We thus model just one AS key derivation as this is representative of all the other derived keys;
2) As we wish to investigate "one-hop" backwards (in)security of the AS key, we only need to model one single handover;
3) The nature of the handover (vkd or hkd), is unimportant for "one-hop" backwards security as the K^*_{xNB}will be known to both sNode and tNode using either method. So, given the AS derivation mechanism in XN/X2, the sNode can obtain the next AS key that the tNode will compute[8].

Modelling sNode's Knowledge of AS Keys. As stated in Section 3, we consider honest-but-curious sNodes. We model that by allowing a sNode to leak its current K_{xNB}. Tamarin's built-in DY attacker will try and compute all possible meaningful messages derivable from the leaked K_{xNB}, including the AS keys, thus effectively emulating the behaviour of an honest-but-curious sNode.

In the XN/X2 protocol, the attacker can calculate K^*_{xNB} from the leaked K_{xNB}, andthen compute the AS keys, too. However, the DY attacker cannot do the the latter in $\texttt{XN/X2}^{RANc}$, $\texttt{XN/X2}^{UEc}$ and $\texttt{XN/X2}^{RANc-UEc}$, since the computation of the AS keys therein includes at least one nonce unknown to the attacker.

We prove both these statements and several others formally in Tamarin, as the next subsection explains.

Properties Verified. For both XN/X2 and $\texttt{XN/X2}^{RANc}$, the main lemmas of interest are given in Table 1. They show that XN/X2 has no backwards security of the AS keys, while $\texttt{XN/X2}^{RANc}$ does without compromising any of XN/X2's other security

[8] Only a subsequent "second-hop" *vertical key derivation* handover by the tNode will prevent the sNode from learning future traffic. For this paper, we are not interested in the backwards (in)security of the AS keys of "second-hop" handovers.

(e.g., key agreement) properties. We now explain this in more detail.

- **Lemmas 1 & 2** formally show correctness of the execution, i.e., without an attacker both protocols execute the protocol and derive the required keys correctly;
- **Lemmas 3 & 7** formally show that XN/X2 has no backward security of the AS keys, but it does achieve injective key-agreement between the tNode and the UE;
- **Lemmas 4 & 8** formally show our protocol, $\texttt{XN/X2}^{RANc}$, achieves backward security of the AS keys while all expected key-agreement properties still hold;
- **Lemmas 5 & 6** formally show neither XN/X2 nor $\texttt{XN/X2}^{RANc}$ have backward-security w.r.t. the tNodes when it comes to K^{*}_{xNB}; indeed, these tNodes are given the next K^{*}_{xNB}s by the sNodes.
- **Lemmas 9 & 10** formally show that both XN/X2 and $\texttt{XN/X2}^{RANc}$ achieve injective agreement with respect to K^{*}_{xNB}, between the relevant parties, in two by two fashion.

$\texttt{XN/X2}^{UEc}$ & $\texttt{XN/X2}^{RANc-UEc}$ Verification. The Tamarin models for $\texttt{XN/X2}^{UEc}$ & $\texttt{XN/X2}^{RANc-UEc}$ contain the same set of aforesaid lemmas for the $\texttt{XN/X2}^{RANc}$ model. So, we also formally show that $\texttt{XN/X2}^{UEc}$ & $\texttt{XN/X2}^{RANc-UEc}$ provide backwards security of the AS keys as well, also without compromising on any of the existing XN/X2 properties.

Verification Statistics. The $\texttt{XN/X2}^{RANc}$ model is $\approx$ 850 lines of code (LoC): 500 of which implement the XN/X2 and $\texttt{XN/X2}^{RANc}$ protocols, while 350 LoC encode the various lemmas to prove or falsify. The Tamarin files for $\texttt{XN/X2}^{UEc}$ and $\texttt{XN/X2}^{RANc-UEc}$ are slightly shorter at $\approx$ 750 LoC, as they only model their own variant. All protocol-relevant lemmas "autoprove" using the standard Tamarin heuristic [29]. The proofs of the two source lemmas – needed to remove partial deconstructions in Tamarin [29] – need the provided oracle (see .py file within our shared files). The code was executed on a laptop with 16GB of RAM and a 4-core (8 threads) Intel Core®i7-1065G7 CPU running at 1.30GHz (3.9GHz maximum Turbo frequency) and timings of the relevant lemmas are included in Table 1. We used Tamarin 1.6.1 for our analysis.

6. A Trust & Communication Metric for Key-Establishment in Mobile Networks

We now present a new framework, *MobTrustCom*, which quantifies the level of trust and communication complexity in mobile networks from the viewpoint of key-establishment procedures in mobile communications (full Registrations and Mobile Registration Updates, Handovers, RRC Reconfigurations). It captures the *core*, the RAN and the UE involved in computing a new session-key, and quantifies (1) the trust level needed in the key-establishment procedure, and (2) the communication cost involved for the newly established key, e.g., how much traffic is incurred by the need to communicate with the *core* or just with the RAN.

6.1. Motivating the MobTrustCom Framework

In mobile procedures, the channel-securing keys can be computing by a RAN node alone, as is the case in XN/X2 handovers. Or, it can be computed by the core, as it is the case in N2/S1 handovers. Clearly, the former case requires more caution, as the RAN nodes run proprietary software. But, the latter case is computationally more expensive, as the core needs to get involved to compute a key that is only RAN-facing.

One would ideally like to catch possible compromises/misbehaviour by RAN nodes or compromised UEs in key-derivations or channel-securing settings. But that therefore means that such untrustworthy parties cannot compute said keys alone, but should do so jointly, under the proviso that not all entities are corrupted at once. In extremis, one could involve a trusted party such as the core in all key derivations, thus reducing the concerns caused by, e.g., compromisible RAN nodes. However, if for the sake of added trust, certain key-derivation procedures were to be jointly executed by more parties than usually or even with added involvement of the core, then the communication complexity of such adapted key-derivation procedures would clearly increase. In order words, there is an easy-to-see inversely proportional relation between the level of trust required and the communication complexity of these procedures. What is not clear is how to measure the variation of this relation over different mobile-network parties and different procedures in mobile-networks. This is where MobTrustCom comes in.

To be able to systematically as well as exhaustively quantify how measures of trust and communication complexity vary with the security of the access-stratum (i.e., channel) in mobile networks, we introduce MobTrustCom. Concretely, MobTrustCom quantifies trust-communication tradeoffs, primarily by associating trust "values" to mobile-network parties, i.e., core, node, UE, and communication-costs to mobile-networks links, i.e., UE-RAN, RAN-RAN, RAN-CORE.

6.2. MobTrustCom: a Framework for Quantifiable Trust & Communication Complexity in Mobile Key-Establishment

In what follows, we exemplify MobTrustCom mainly on handovers, but what we also explain how it applies to the REG, and RRC reconfigurations as well. This wide-application is also visible in the summative Figure 5.

Access Stratum Trust Scale. To assess the level of trust needed in the access-stratum-securing procedures, we introduce the notion of *Access Stratum trust scale*.

Definition 1 (Access Stratum Trust Scale.). The *AS trust scale* is a set $\{n_1, \ldots, n_6\}$ of increasing positives integers, i.e., $n_1 < n_2 < \ldots < n_6$, called *AS trust levels*:

- AS trust level n_6: if the UE were to compute the AS keys by itself and securely send its ingredients into the network.
  A procedure with AS trust level n_6 is said to have *UE controlled* AS keys' computation.
- AS trust level n_5: when the RAN alone effectively determines the value of the AS keys, and securely sends its ingredients into the network.

A procedure with AS trust level n_5 is said to have *RAN controlled* AS keys' computation.

- AS trust level n_4: if the RAN and the UE were to compute together the AS keys, and securely send its ingredients into the network.
  A procedure with AS trust level n_4 is said to have *RAN-UE- controlled* AS keys' computation.
- AS trust level n_3: if the core and the UE were to compute together the AS keys, and securely send its ingredients into the network.
  A procedure with AS trust level n_3 is said to have *core-UE- controlled* AS keys' computation.
- AS trust level n_2: when the core effectively determines the AS keys, and securely sends its ingredients into the network.
  A procedure with AS trust level n_2 is said to have *core controlled* AS keys' computation.
- AS trust level n_1: if the core, the RAN, and the UE together were to compute the AS keys.
  A procedure with AS trust level n_1 is said to have *core-RAN-UE controlled* AS keys' computation.

The monotonicity of the access-stratum trust scale is as follows: AS trust level n_1 represents the least trust needed, and AS trust level n_6 represents the highest trust needed:

Clearly, riskier aspects require more trust be invested.

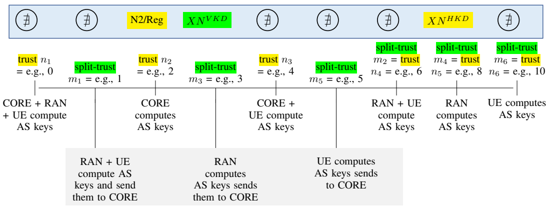

N2/Reg
XN^{VKD}
XN^{HKD}
trust n_1 = e.g., 0
split-trust m_1 = e.g., 1
trust n_2 = e.g., 2
split-trust m_3 = e.g., 3
trust n_3 = e.g., 4
split-trust m_5 = e.g., 5
split-trust m_2 = trust n_4 = e.g., 6
split-trust m_4 = trust n_5 = e.g., 8
split-trust m_6 = trust n_6 = e.g., 10
CORE + RAN + UE compute AS keys
CORE computes AS keys
CORE + UE compute AS keys
RAN + UE compute AS keys
RAN computes AS keys
UE computes AS keys
RAN + UE compute AS keys and send them to CORE
RAN computes AS keys sends them to CORE
UE computes AS keys sends to CORE

Figure 5. MobTrustCom's AS (Split)-Trust Levels

Our AS Trust Levels "In The Wild". We note that only *RAN controlled* (trust level n_5) and *core controlled* (trust level n_2) exist in real-life AS keys' calculations in handover procedures: i.e., in XN/X2 and N2/S1, respectively. The other trust levels correspond to AS keys' computations that could exist, but do not at the moment. Some cases are plausible but extreme:, e.g., *UE controlled* at trust level n_6, where the UE would calculate the AS keys of its own accord and send them to a RAN-node. It could be used, for instance, in emergencies: e.g., if the mobile generation of some RAN nodes is too low.

• Definition 1 looks *just* at which parties decide the value of an AS-key, i.e., it does not consider to which *exact* other parties this key –or the materials for it– are sent. This intuitively says that the AS trust levels n_6, n_4, n_5 can be refined and this is dealt with in Definition 2.

Definition 2 (**Access Stratum Split-Trust Scale.**). The *AS split-trust scale* is a set $\{m_1, \ldots, m_6\}$ of increasing positive integers, i.e., $m_1 < m_2 < \ldots < m_6$, called *AS trust levels* associated to AS trust levels n_6, n_4, n_5 in such a way that m_6= n_6, m_4= n_5, m_2= n_4 and are as follows:

1) AS split-trust level m_6 (= n_6): if the UE were to compute alone the AS keys (i.e., trust-level n_6) and securely send its ingredients just to the RAN.
2) AS split-trust level m_5 : if the UE were to compute alone the AS keys (i.e., trust-level n_6) and securely send its ingredients to the RAN as well as the core.
   Thus, we require $m_5 < n_6$, to encode that the trust n_6 is split/reduced.
3) AS split-trust level m_4 (= n_5): when the RAN alone effectively determines the value of the AS keys (i.e., trust-level 8), and it securely sends its ingredients to the UE, but it does not send its ingredients to the core as well.
4) AS split-trust level m_3: when the RAN alone effectively determines the value of the AS keys (i.e., trust-level n_5), but securely sends its ingredients to the UE as well as the core.
   Thus, we require $m_3 < n_5$, to encode that the trust n_5 is split/reduced.
5) AS split-trust level m_2 (= n_4): if the RAN and the UE were to compute together the AS keys (i.e., trust-level 6), and they do not send its ingredients to the core as well.
6) AS split-trust level m_1: if the RAN and the UE were to compute together the AS keys (i.e., trust-level n_4), and they securely send its ingredients to the core as well.
   Thus, we require $m_1 < n_4$, to encode that trust n_5 is split/reduced.

Apart from the inequalities in points 2, 4, 6 above, we also require[9] that: n_1 < m_1, n_2 < m_3, and n_2 < m_3.

The monotonicity of the AS split-trust scale is as follows: access-stratum split-trust level m_1 represents the least split-trust needed, and access-stratum split-trust level m_6 represents the highest split-trust needed.

Clearly, the smaller the amount of split-trust level needed, the lower the risk.

The inequalities in Definition 2 between the ms and ns show how some trust level can be lowered to split-trust level. Yet, we see that some trust-levels are associated to a split-trust level by Definition 2. Simply put, there is no way to lower the trust/risk needed in those cases. An example is the case of trust-level n_2. In this case, the core effectively determines the value of the AS keys, by calculating the $K_{\text{x}NB}$ key, and it already has to send this (or its elements) to the RAN and to the UE, as these keys are for the AS communication between the latter two entities; so, all entities already share the key material, so we cannot do anything to split the trust further.

Which Integers Can be (Split-)trust Levels. For Definition 2 and Definition 1, we have the following system of simultaneous equations:

$$\begin{cases} n_1 < n_2 < \ldots < n_6; m_1 < m_2 < \ldots < m_6 \\ m_6 = n_6; m_4 = n_5; m_2 = n_4 \\ m_5 < n_6; m_3 < n_5; m_1 < n_4 \\ n_1 < m_1; n_2 < m_3; n_2 < m_3 \end{cases}$$

For instance, the numbers 0 to 10 exhibit the full AS trust and split-trust scale as per Table 2. Table 2 shows in bold the points where the AS trust and split-trust coincide.

[9] This is based on the meaning associated to each level and the trust place on the entity directly involved. E.g., in n_2 the core generates the key material, in m_3 the RAN generates the key material, but sends to the core to split the trust.

TABLE 2. EXAMPLES OF AS TRUST & SPLIT-TRUST LEVELS

example value	0	1	2	3	4	5	**6**	-	**8**	-	**10**
trust level	n_1	-	n_2	-	n_3	-	$\mathbf{n_4}$	-	$\mathbf{n_5}$	-	$\mathbf{n_6}$
split-trust level	-	m_1	-	m_3	-	m_5	$\mathbf{m_2}$	-	$\mathbf{m_4}$	-	$\mathbf{m_6}$

Table 2 aids us to see that trusts of high levels (e.g., 10, 8, 6) can be lowered by splitting. This is how the split-trust level appear, i.e., m_1, m_3, m_5.

Note 7: The MobTrustCom framework exhausts all the combinations of endowing trust over the three main mobile parties. It is not our remit to justify if all these trust levels should exist in reality.

MobTrustCom 's trust and split-trust scales relating to AS-keys' establishment procedures are given in Figure 5.

Mobile Networks' Communication Cost. To capture aspects of communication costs, we give Definition 3.

Definition 3. Mobile Networks Communications Cost. *Mobile networks communications cost* is a tuple t=(c_1, c_2, c_3, c_4) of four positive digits ($c_1, c_2, c_3, c_4 \in \{1, 2, \ldots, 9\}$), called *cost factors*. Each cost factor is the expense of one unit of information (e.g., 1 bit / 1 byte) being sent in one direction, on a segment of the network. The cost factors are associated to the network as follows

$$UE \overset{c_2}{- - -} RAN \overset{c_1}{- - -} RAN \overset{c_3}{- - -} CORE$$

$$UE \overset{c_4}{- - - - - - - - - - - - - - - -} CORE,$$

and respect the following system of inequalities:

$$\begin{cases} c_1 < c_2 < c_4 & (1) \\ c_1 < c_3 < c_4 & (2) \\ c_1 + c_3 < c_4 & (3) \\ c_2 + c_3 < c_4 & (4) \\ c_1 + c_2 + c_3 > c_4 & (5) \end{cases}$$

The system of inequalities for the cost factors to satisfy are justified as follows. First, it is, in our view, least expensive to communicate between the nodes in the RAN (i.e., at a cost factor of c_1). The RAN-to-UE exchanges are radio messages as well as NAS messages, so they are generally more costly than the former (i.e., $c_1 < c_2$). The core-to-RAN exchanges are on authenticated and integer pre-established channels and can often be over different networks/providers, so that is estimated at an even higher cost (i.e., $c_3 > c_1$). Second, the core-to-UE communication is in fact proxied over the RAN and it is relatively rare, using authenticated encryption with renewable keys such as K_{ASME} (in 4G) and K_{AMF} (in 5G), which are re-calculated in different procedures (e.g., S1/N2, re-registration); so, we deemed this most expensive (i.e., $c_4 > c_2$ and $c_4 > c_3$). Thirdly, we chose the cost factors such that the cumulative cost of a message travelling over two segments (e.g., core-to-RAN at a cost of c_3 and the intra-RAN at a cost of c_1) will be close to but less than the cost of an exchange over one single but "longer" segment (e.g., UE-to-core at a cost of c_3); this is captured by inequalities (3) and (4). Finally, inequality (5), together with the fact that all cost factors are all digits, express that the cost factors are numbers of the same magnitude (i.e., similar number of bits), close together.

Example 6.1. Example of A Mobile Communication Cost. A tuple that satisfies Definition 3 is $(2, 3, 4, 8)$, given visually as:

$$UE \overset{c_2=3}{- - -} RAN \overset{c_1=2}{- - -} RAN \overset{c_3=4}{- - -} CORE$$

$$UE \overset{c_4=8}{- - - - - - - - - - - - - - - -} CORE$$

Note 8. **A Goal: AS Keys' Establishment with Lower Trust at Reasonable Communication Costs?** In Figure 5, we see that XN^{hkd} has the same trust level n_5 and split-trust level m_5, which –as per Table 2– could be 8. At the same time, we see that XN^{vkd} does lower the trust level of n_5 to the split-trust level of m_3, which –as per Table 2– could be 3. And, we see that N2 has a low trust level of n_2 (e.g., 2). By looking at the specifications, we see that XN^{hkd} and XN^{vkd} come at comparable total communication cost of, say, α. And, N2 comes at a total communication cost of, say, β, where $\beta > \alpha + 4\ bytes \times (c_3 + c_4)$.

So, can one improve the trust level of XN^{hkd}, possibly to as much as that of XN^{vkd}, but with a communication cost lower than N2? In the process, can we also better the backwards' security of the AS keys? This is what we pursued in our protocols presented in the previous section, particularly XN/X2^{RANc} .

MobTrustCom Framework, (Backwards) Key-Secrecy vs. Key -agreement Properties. MobTrustCom focuses on (splitting) trust and not on key-agreement. Simply put, to lower trust levels, we ask that the "secret ingredients" produced by one party A be sent to party B, but we do not ask these be returned by party B to party A for confirmed agreement amongst the two. In that sense, it is clear that the trust and especially split-trust scales in Definition 2 can be made finer if we wished to consider the extra case where the information sent by A to B is –additionally– either returned to A or not. If we did that, then MobTrustCom would cater for split trust in a way aligned to multi-party key-agreement properties.

6.3. MobTrustCom Comparison of the "Old" and New Handovers

In Table 3, we cross-compare XN/X2 handovers vs. their improvements XN/X2^{RANc} , XN/X2^{UEc} , XN/X2$^{RANc-UEc}$, relating to AS keys' backwards security and the MobTrustCom's cost and trust notions. The value of 4 corresponds to the byte-length of T_1 and T_2 in XN/X2^{RANc} , XN/X2^{UEc} , and/or XN/X2$^{RANc-UEc}$, and 16 is the byte-length of the encryption of T_2 with K_{ASME} (for X2) or K_{AMF} (for XN) in their XN/X2^{UEc} and XN/X2$^{RANc-UEc}$ improvements.

We say XN/X2 has a communication cost of "α" and N2 of "β" (with $\beta > \alpha$ as stated above).

The communication complexity of XN/X2^{RANc} is that of XN/X2 plus, first, the cost of ferrying T_1 over 2 specific messages; that gives an addition of $4\ bytes \times (c_3 + c_4)$. On top, XN/X2^{RANc} makes MRU phases obligatory with every execution, whereas XN/X2 does not impose that with every execution. The cost of an MRU phase is $x\ bytes \times (c_2 + c_1 + c_4)$. for some fixed $x > 16$. All is divided by some "*positive integer*" because these small-number-of-bytes additions would not be in placse for every UE, as some have already more options set than these

procedure	AS-key generation controlled by	AS-backwards security w.r.t RAN	example trust level	example split-trust level	communication complexity
XN/X2^{hkd}	RAN	no	8	8	α
XN/X2^{vkd}		no	8	3	α
XN/X2^{RANc} *(new/ours)*		yes	8	3	$\gamma := \alpha + \frac{4\ bytes \times (c_3+c_4)}{positive\ integer} + \frac{MRU\ comm.\ cost}{positive\ integer}$
XN/X2^{UEc} *(new/ours)*	UE	yes	10	1	$\gamma' := \alpha + \frac{16\ bytes \times (c_2+c_3)}{positive\ integer} + \frac{4\ bytes \times c_3}{positive\ integer}$
XN/X2$^{RANc\text{-}UEc}$ *(new/ours)*	RAN+UE	yes	6	1	$\gamma'' := \gamma + \gamma' - \alpha$
N2	core	yes	2	2	β, where $\alpha < \gamma < \gamma' < \gamma'' < \beta$

TABLE 3. COST VS. TRUST IN HANDOVERS WITH ENHANCED AS KEYS' BACKWARDS SECURITY (WHERE MRU_COMM. COST=$x \times (c_2 + c_1 + c_4)$, WITH x>16)

few extra bytes. This leads to the final communication complexity of XN/X2^{RANc}, denoted by "γ" in Table 3.

In XN/X2^{UEc}, the 4-byte value T_2 needs to be sent in an encryption of 16 bytes from the UE to the core, and decrypted from the core to the tNode. This gives γ'. Note again that XN/X2^{UEc} does not force obligatory MRUs so, there is no added overhead from that, unlike in XN/X2^{RANc}. So, in reality, γ' may be smaller on average than γ. i.e., XN/X2^{RANc} less efficient than XN/X2^{UEc}.

As we said, XN/X2$^{RANc-UEc}$ is XN/X2 augmented with the enhancements of both XN/X2^{RANc} and XN/X2^{UEc} resulting in the communication cost of γ''.

We note $\beta > \gamma''$: i.e., even our most communication-expensive protocol, XN/X2$^{RANc-UEc}$, is cheaper communication-wise than N2 handovers. And, as we can see, the split-trust of our protocols XN/X2$^{RANc-UEc}$ and XN/X2^{UEc} is also better than that of N2.

The MobTrustCom (split-)trust levels for the "old" handovers in Table 3 were explained via Figure 5. For our new XN/X2^{RANc}, XN/X2^{UEc} and XN/X2$^{RANc-UEc}$ handovers, these levels follow immediately from their constructions plus Definition 1 and Definition 2. For instance, in XN/X2^{RANc}, the RAN generates the T_1 secret ingredient for K$_{AS}$ and does not just share it with the UE, but also with the *core* (hence, trust-level of n_5, e.g., 8 and split-trust level of m_3, e.g., 3). In XN/X2^{UEc}, the UE alone generates T_2 (hence, trust-level n_6, e.g., 10), but all 3 parties get to know this value (hence, split-trust level as low as m_1, e.g., 1). Finally, in XN/X2$^{RANc-UEc}$, the UE and the RAN both generate K$_{AS}$-relevant secrets (hence, trust-level of n_4, e.g., 6), but the core gets to know these too (hence, the lowest split-trust level m_1, e.g., 1).

7. Implementing our Modified XN

We implemented an emulation of XN/X2^{RANc} in a 5G network. We used the well-known and 3GPP-compliant Open5GCore toolkit [22], developed by the Fraunhofer Society. Since we focused on 5G (not 4G), we refer to the implemented version of XN/X2^{RANc} as XN^{RANc}.

We chose to implement XN/X2^{RANc} as opposed to, e.g., XN/X2^{UEc}, due to limitations of the Open5GCore toolkit, which currently does not have a full UE-capability *for 5G messages* inside the Open5GCore toolkit.

7.1. Fraunhofer's Toolkit

Open5GCore is a research-grade testbed for 5G networks w.r.t. the latest 3GPP specifications. The Open5GCore toolkit offers several features of the *core*, most 5G interfaces (N1, N2, N3, XN), and some 5G NAS (non access-stratum) protocols partly or fully developed on this interfaces, including XN. Meanwhile, some of the UE's and the *core*'s features, as well as certain (parts of) procedures are still under development.

The source code of the Fraunhofer toolkit is written in C. As per our XN^{RANc} design (see Section 4), we modified this C implementation of the gNB and the AMF parts of the Fraunhofer setup w.r.t. the XN interface.

7.2. A Backwards-compatible Implementation of XN^{RANc} onto the Open5G Toolkit

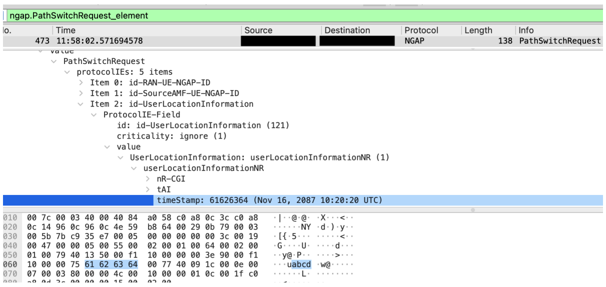
First, recall the discussions in Section 4.1.2: compared to XN, our XN^{RANc}'s edits are akin to overriding fields inside the "Path Switch Request" and the "MRU Registration Accept" messages of the 3GPP specifications [12] and that candidates for this were the "security capabilities", the "user-location information", etc. (see Section 4.2.2 of [12]). A sensible choice is the "user-location information" field, as it appears in both the "Path Switch Request" and the "MRU Registration Accept" messages. Next, a good candidate for actual overloading would be a sub-field able to encode an unformatted 32-bit long bitstring. Thus, we implemented the sending of our T_1 nonce inside the "timestamp" field within "id-UserLocationInformation", inside the "userLocationInformation" item.

An example of this modification, via a Wireshark trace, is shown on Figure 6.

This addition of the T_1 nonce as a timestamp (for now) was, of course, shown to not hinder functional correctness of the XN/X2: i.e., the core and the UE continue to function as expected w.r.t. all XN/X2 functionalities, since the timestamp field used is part of the 3GPP specification.

We did not implement the addition of the T1 nonce into the calculation of the AS keys (i.e., K_{AS}), as per our XN^{RANc} specification, since the Open5GCore toolkit does not yet support[10] the access-stratum functionalities.

[10] In the future, we will add the T_1 nonce to the AS keys, as the Fraunhofer toolkits grows to support the AS.



```
ngap.PathSwitchRequest_element
lo.     Time                    Source       Destination  Protocol  Length  Info
    473 11:58:02.571694578      [redacted]   [redacted]   NGAP         138  PathSwitchRequest
    value
      PathSwitchRequest
        protocolIEs: 5 items
          Item 0: id-RAN-UE-NGAP-ID
          Item 1: id-SourceAMF-UE-NGAP-ID
          Item 2: id-UserLocationInformation
            ProtocolIE-Field
              id: id-UserLocationInformation (121)
              criticality: ignore (1)
              value
                UserLocationInformation: userLocationInformationNR (1)
                  userLocationInformationNR
                    nR-CGI
                    tAI
                    timeStamp: 61626364 (Nov 16, 2087 10:20:20 UTC)
010  00 7c 00 03 40 00 40 84  a0 58 c0 a8 0c 3c c0 a8   ·|··@·@· ·X···<··
020  0c 14 96 0c 96 0c 4e 59  b8 64 00 29 0b 79 00 03   ······NY ·d·)·y··
030  00 5b 7b c9 35 e7 00 05  00 00 00 00 00 3c 00 19   ·[{·5··· ·····<··
040  00 47 00 00 05 00 55 00  02 00 01 00 64 00 02 00   ·G····U· ····d···
050  01 00 79 40 13 50 00 f1  10 00 00 00 3e 90 00 f1   ··y@·P·· ····>···
060  10 00 00 75 61 62 63 64  00 77 40 09 1c 00 0e 00   ···uabcd ·w@·····
070  07 00 03 80 00 00 4c 00  10 00 00 01 0c 00 1f c0   ······L· ········
```

Figure 6. Augmenting XN to $\texttt{XN}^{RANc}$: Backwards-Compatible Addition of T1 to Path Switch Request

7.2.1. Efficiency of the implemented $\texttt{XN}^{RANc}$. In Section 6, we looked at the communication-complexity increase of our $\texttt{XN}^{RANc}$ over the XN specification: i.e., in essence, the addition of 32 bits in one message ("Packet Switch Request") and 32 bits added to the payload of another message ("Mobile Registration Update"). That comparison carries forwarded in the implemented $\texttt{XN}^{RANc}$ vs. XN in the `Open5GCore`.

On top, we used the `Open5GCore` toolkit to perform handovers, that is normal XN alongside $\texttt{XN}^{RANc}$ executions, and looked at the Wireshark captures of these. To estimate the communication-complexity depreciation, one can look at the times of departure and arrival of, e.g., a "Packet Switch Request", in the `Open5GCore`'s XN vs those of the corresponding "Packet Switch Request" in the `Open5GCore`'s $\texttt{XN}^{RANc}$. We see no depreciation in these times over several trials, with different fields filled-in and not filled-in in the "Packet Switch Request" message; the times fluctuate both up and down with a deviation either up or down of approximately 30-50 nanoseconds.

Disclosure. We are working with 3GPP on the lack of AS-keys' backwards security in XN.

8. Related Work

Formal Verification of Keys' Security in 4G Handovers. [15] and [18, 19] verified 4G handover procedures in the Dolev-Yao model [21], using Proverif [16]: [15] showed that "classical" (not backwards) secrecy of K^*_{eNB} holds in X2 and S1, and [18, 19] showed in Proverif, for 4G alone, that the backward secrecy of K_{eNB} was lacking. Unlike us, they did not look at the backwards secrecy/security of AS keys.

Formal Analyses of Keys' Security in 5G Handovers. [28] proposed a Dolev-Yao models of 5G XN and N2 handovers (3GPP Release 16 specifications [11]) in Tamarin [26]. The paper found no confidentiality issues, as long as honest participants cannot be compromised. [28] did not consider the forward/backward secrecy of the K^*_{gNB} or the AS keys.

[27] took the Tamarin models in [28] and lifted them to more than handovers, e.g., including registration and handovers executed in series, as well as considering different tiers of dishonest RAN nodes. Unlike us, [27] does not focus on fixing AS backwards insecurity or on the AS keys, but rather on the k_{gNB} keys; they show that if several dishonest RAN nodes are colluding, k_{gNB} keys' leakage can be made to persist over a series of XN executions. This composition of handovers, due to its inherent complexity, yields sometimes intractable Tamarin models. Instead, our work (and more-trackable models) abstracts some of this complexity and focuses specifically on: key-derivation in handovers and backwards-secrecy as a security property; so, our Tamarin models of the XN/X2 protocols are simpler than those in [28, 27]. Moreover, we propose improvements and formally verify them. Lastly, we provide a backwards-compatible implementation over Open5GCore. Meanwhile, [27] has no practical study.

Finally, somewhat similarly, but in a computational model rather than a Dolev-Yao model, [17] looks at the "duration" of such leakages in handovers before a XN executions' chain is interrupted by, e.g., an N2 execution.

Our work can be considered as taking the message of AS-key backwards insecurity in XN and looking at ways of providing provably-secure enhancements to the XN/X2 protocols in a measurably efficient way while staying as close as possible to the 3GPP specifications.

General Analyses of Non-XN/N2 Handovers or other 5G Procedures. [30] carry out a Dolev-Yao verification, in the Scyther tool [20], of their "home-made" protocol called the Lightweight and Secure Handover Authentication Scheme (LSHA), based on intra-RAN authentication in handovers. Other such ad-hoc extensions of handovers may exist, but since they are far from "standard" handovers, we do not cover them further here. Equally, [23] is on formal verification of 5G procedures, but it focuses not on security, but on privacy. [14] looked at X2 not w.r.t. formal security but w.r.t. UE performance; it showed that certain delays in X2 handovers stem from synchronisations of the UE and the tNode. We do not cover further such security-unrelated works either.

Considerations on Trust and Communications in Mobile Network. Other considerations of communication complexity in mobile networks (e.g., [30, 25]) are generally not systematic, not even coarsely so like ours.

9. Conclusions

We proposed a series of protocols that modify the X2/XN handovers in minimal ways such as to get procedures which gain AS backwards security. We formally verified them alongside X2/XN, to show formally this security gain. To demonstrate the closeness to the 3GPP-specified X2/XN handovers, we also implemented one of proposed protocol in Fraunhofer's 3GPP-compliant `Open5GCore` 5G testbed and ascertained no functionality loss or efficiency depreciation. We presented a framework, *`MobTrustCom`*, for quantifying trust and communication cost in mobile networks. We applied the `MobTrustCom` to handovers and other key-establishment protocols (such as REG) and used it to also formally show that, compared to "classical" handovers, our aforesaid sensitively-made patches to XN/X2 lower the trust needed and are very competitive communication-wise.

Acknowledgments

This work was partially supported by the PhD-studentship "5GTech-Sec" (funded by UK's NCSC) and the EP/S024565/1 research grant " EP/S024565/1" (funded by UK's EPSRC).

References

[1] Minimum requirements related to technical performance for IMT-2020 radio interface(s). 11 2017. URL https://www.itu.int/pub/R-REP-M.2410-2017.

[2] 3GPP. General packet radio service (gprs) enhancements for evolved universal terrestrial radio access network (e-utran) access. Technical Specification (TS) 23.401, 3GPP, 09 2014. URL https://www.etsi.org/deliver/etsi_ts/123400_123499/123401/12.06.00_60/ts_123401v120600p.pdf. Version 12.6.0.

[3] 3GPP. X2 application protocol (x2ap). Technical Specification (TS) 36.423, 3GPP, 09 2014. URL https://www.etsi.org/deliver/etsi_ts/136400_136499/136423/12.03.00_60/ts_136423v120300p.pdf. Version 12.3.0.

[4] 3GPP. Generic authentication architecture (gaa);generic bootstrapping architecture (gba). Technical Specification (TS) 33.220, 3GPP, 10 2018. URL https://www.etsi.org/deliver/etsi_ts/133200_133299/133220/15.02.00_60/ts_133220v150200p.pdf. Version 15.0.0.

[5] 3GPP. Digital cellular telecommunications system (phase 2+) (gsm); universal mobile telecommunications system (umts); lte. Technical Specification (TS) 33.401, 3GPP, 07 2018. URL https://www.etsi.org/deliver/etsi_ts/133400_133499/133401/15.04.00_60/ts_133401v150400p.pdf. Version 15.4.0.

[6] 3GPP. S1 application protocol (s1ap). Technical Specification (TS) 36.413, 3GPP, 09 2018. URL https://www.etsi.org/deliver/etsi_ts/136400_136499/136413/15.03.00_60/ts_136413v150300p.pdf. Version 15.3.0.

[7] 3GPP. Wireless and wireline onvergence access support for the 5g system (5gs). Technical Specification (TS) 23.316, 3GPP, 10 2020. URL https://www.etsi.org/deliver/etsi_ts/123300_123399/123316/16.04.00_60/ts_123316v160400p.pdf. Version 16.0.0.

[8] 3GPP. System architecture for the 5G System (5GS). Technical Specification (TS) 23.501, 3GPP, 10 2020. URL https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf. Version 16.0.0.

[9] 3GPP. System architecture for the 5G System (5GS). Technical Specification (TS) 23.501, 3GPP, 10 2020. URL https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf. Version 16.0.0.

[10] 3GPP. Xn application protocol (xnap). Technical Specification (TS) 38.423, 3GPP, 07 2020. URL https://www.etsi.org/deliver/etsi_ts/138400_138499/138423/16.02.00_60/ts_138423v160200p.pdf. Version 16.2.0.

[11] 3GPP. Release 16, 2020. URL https://www.3gpp.org/release-16.

[12] 3GPP. Procedures for the 5g system. Technical Specification (TS) 23.502, 3GPP, 10 2021. URL https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/15.02.00_60/ts_123502v150200p.pdf. Version 16.7.0.

[13] 3GPP. Becoming 5G-Advanced: the 3GPP 2025 Roadmap. https://www.5gamericas.org/wp-content/uploads/2022/12/Becoming-5G-Advanced-the-3GPP-2025-Roadmap-InDesign.pdf, 2022.

[14] Konstantinos Alexandris, Navid Nikaein, Raymond Knopp, and Christian Bonnet. Analyzing x2 handover in LTE/LTE-a. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–7. doi: 10.1109/WIOPT.2016.7492906.

[15] Noomene Ben Henda and Karl Norrman. Formal analysis of security procedures in LTE - a feasibility study. In *RAID '14*, LNCS, pages 341–361. Springer. ISBN 978-3-319-11379-1.

[16] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW '14*, pages 82–96. IEEE Computer Society, June 2001.

[17] Olivier Blazy, Ioana Boureanu, Pascal Lafourcade, Cristina Onete, and Léo Robert. How fast do you heal? a taxonomy for post-compromise security in secure-channel establishment. Cryptology ePrint Archive, Paper 2022/1090, 2022. URL https://eprint.iacr.org/2022/1090. https://eprint.iacr.org/2022/1090.

[18] Piergiuseppe Bettassa Copet, Guido Marchetto, Riccardo Sisto, and Luciana Costa. Formal verification of LTE-UMTS handover procedures. In *ISCC '15*, pages 738–744. IEEE, .

[19] Piergiuseppe Bettassa Copet, Guido Marchetto, Riccardo Sisto, and Luciana Costa. Formal verification of LTE-UMTS and LTE-LTE handover procedures. 50:92–106, . ISSN 0920-5489. doi: 10.1016/j.csi.2016.08.009. URL http://www.sciencedirect.com/science/article/pii/S092054891630071X.

[20] C.J.F. Cremers. *Scyther : semantics and verification of security protocols*. PhD thesis, Mathematics and Computer Science, 2006.

[21] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Trans. Inf. Theory 29*, 29(2), 1983.

[22] Fraunhofer. Open5gcore. URL https://www.open5gcore.org/.

[23] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 5greasoner: A property-directed security and privacy analysis framework for 5g cellular network protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 669–684, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354263. URL https://doi.org/10.1145/3319535.3354263.

[24] Volker Jungnickel, Konstantinos Manolakis, Wolfgang Zirwas, Berthold Panzner, Volker Braun, Moritz Lossow, Mikael Sternad, Rikke Apelfröjd, and Tommy Svensson. The role of small cells, coordinated multipoint, and massive mimo in 5g. *IEEE communications magazine*, 52(5):44–51, 2014.

[25] K. Anitha Kumari, G. Sudha Sadasivam, S. Shymala Gowri, Sebastin Arockia Akash, and E.G. Radhika. An approach for end-to-end (e2e) security of 5g applications. In *2018 IEEE 4th International Conference on Big Data Security on Cloud (Big-*

DataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), pages 133–138, 2018. doi: 10.1109/BDS/HPSC/IDS18.2018.00038.

[26] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *CAV '13*, LNCS, pages 696–701. Springer. ISBN 978-3-642-39799-8.

[27] Rhys Miller, Ioana Boureanu, Stephan Wesemeyer, and Christopher J. P. Newton. The 5g key-establishment stack: In-depth formal verification and experimentation. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22, page 237–251, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391405. doi: 10.1145/3488932.3517421. URL https://doi.org/10.1145/3488932.3517421.

[28] Aleksi Peltonen, Ralf Sasse, and David Basin. A comprehensive formal analysis of 5g handover. In *WiSec '21*, WiSec '21, page 1–12, New York, NY, USA, 2021. ACM. ISBN 9781450383493.

[29] The Tamarin Team. Tamarin prover manual, 2016. https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf [Online: accessed 09-April-2019].

[30] Xiaobei Yan and Maode Ma. A lightweight and secure handover authentication scheme for 5g network using neighbour base stations. *Journal of Network and Computer Applications*, 193:103204, 2021. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2021.103204. URL https://www.sciencedirect.com/science/article/pii/S1084804521002095.

A. Figure for Two of Our New XN/X2 Protocols

Due to space constraints, we give the diagrams of two of our new XN/X2 protocols, below.

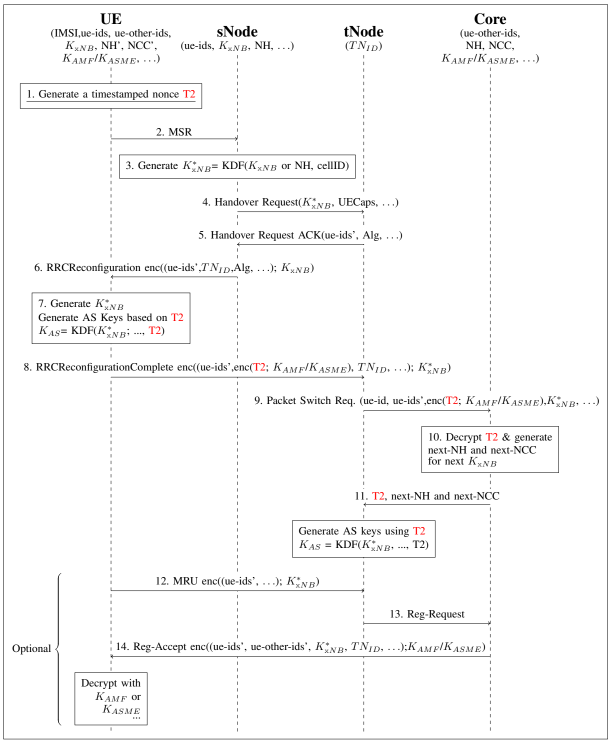

UE
(IMSI,ue-ids, ue-other-ids, K_{xNB}, NH', NCC', $K_{AMF}/K_{ASME}, \ldots$)
sNode
(ue-ids, K_{xNB}, NH, ...)
tNode
(TN_{ID})
Core
(ue-other-ids, NH, NCC, $K_{AMF}/K_{ASME}, \ldots$)
1. Generate a timestamped nonce T2
2. MSR
3. Generate K^*_{xNB}= KDF(K_{xNB} or NH, cellID)
4. Handover Request(K^*_{xNB}, UECaps, ...)
5. Handover Request ACK(ue-ids', Alg, ...)
6. RRCReconfiguration enc((ue-ids',TN_{ID},Alg, ...); K_{xNB})
7. Generate K^*_{xNB}
Generate AS Keys based on T2
K_{AS}= KDF(K^*_{xNB}; ..., T2)
8. RRCReconfigurationComplete enc((ue-ids',enc(T2; K_{AMF}/K_{ASME}), TN_{ID}, ...); K^*_{xNB})
9. Packet Switch Req. (ue-id, ue-ids',enc(T2; K_{AMF}/K_{ASME}),K^*_{xNB}, ...)
10. Decrypt T2 & generate next-NH and next-NCC for next K_{xNB}
11. T2, next-NH and next-NCC
Generate AS keys using T2
K_{AS} = KDF(K^*_{xNB}, ..., T2)
12. MRU enc((ue-ids', ...); K^*_{xNB})
13. Reg-Request
14. Reg-Accept enc((ue-ids', ue-other-ids', K^*_{xNB}, TN_{ID}, ...);K_{AMF}/K_{ASME})
Optional
Decrypt with K_{AMF} or K_{ASME} ...

Figure 7. A New Enhanced XN/X2 with Added AS Backwards Security via UE Operations Protocol: $\texttt{XN/X2}^{UEc}$

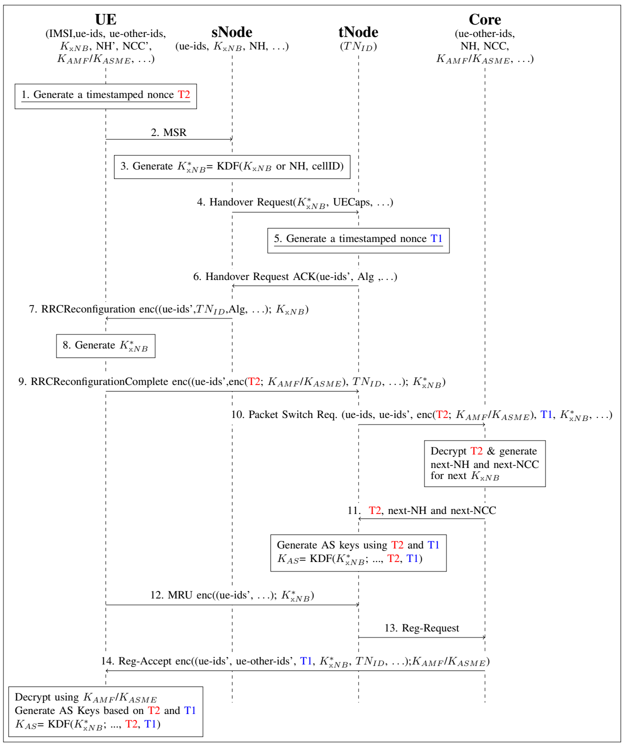

UE
(IMSI,ue-ids, ue-other-ids, K_{xNB}, NH', NCC', $K_{AMF}/K_{ASME}, \ldots$)
sNode
(ue-ids, K_{xNB}, NH, ...)
tNode
(TN_{ID})
Core
(ue-other-ids, NH, NCC, $K_{AMF}/K_{ASME}, \ldots$)
1. Generate a timestamped nonce T2
2. MSR
3. Generate K^*_{xNB}= KDF(K_{xNB} or NH, cellID)
4. Handover Request(K^*_{xNB}, UECaps, ...)
5. Generate a timestamped nonce T1
6. Handover Request ACK(ue-ids', Alg ,...)
7. RRCReconfiguration enc((ue-ids',TN_{ID},Alg, ...); K_{xNB})
8. Generate K^*_{xNB}
9. RRCReconfigurationComplete enc((ue-ids',enc(T2; K_{AMF}/K_{ASME}), TN_{ID}, ...); K^*_{xNB})
10. Packet Switch Req. (ue-ids, ue-ids', enc(T2; K_{AMF}/K_{ASME}), T1, K^*_{xNB}, ...)
Decrypt T2 & generate next-NH and next-NCC for next K_{xNB}
11. T2, next-NH and next-NCC
Generate AS keys using T2 and T1
K_{AS}= KDF(K^*_{xNB}; ..., T2, T1)
12. MRU enc((ue-ids', ...); K^*_{xNB})
13. Reg-Request
14. Reg-Accept enc((ue-ids', ue-other-ids', T1, K^*_{xNB}, TN_{ID}, ...);K_{AMF}/K_{ASME})
Decrypt using K_{AMF}/K_{ASME}
Generate AS Keys based on T2 and T1
K_{AS}= KDF(K^*_{xNB}; ..., T2, T1)

Figure 8. A New Enhanced XN/X2 with Added AS Backwards Security via UE-RAN Shared OperationsProtocol: $\texttt{XN/X2}^{RANc-UEc}$