Fine-Grained Trackability in Protocol Executions

Ksenia Budykho,
Ioana Boureanu,
Stephan Wesemeyer
Surrey Centre for Cyber Security, University of Surrey, UK
sccs@surrey.ac.uk

Daniel Romero,
Matt Lewis
NCC Group, UK
daniel.romero@nccgroup.com, matt.lewis@nccgroup.com,

Yogaratnam Rahulan
5G/6G Innovation Centre, University of Surrey, UK
y.rahulan@surrey.ac.uk

Fortunat Rajaona,
and Steve Schneider
Surrey Centre for Cyber Security, University of Surrey, UK
sccs@surrey.ac.uk

***Abstract*—We introduce a new framework, `TrackDev`, for encoding and analysing what we call the "tracking" of an entity via its executions of a protocol or its usages of a system. `TrackDev` considers multiple dimensions combined: whether the attacker is active or passive, whether an entity is trackable in its every single appearance on the network or just in a compound set thereof, and whether the entity can be explicitly or implicitly identified. `TrackDev` can be applied to most identification-based systems, and, interestingly, in practice, i.e., over actual executions of systems. To this end, we test `TrackDev` on real-life traffic for two well-known protocols, the LoRaWAN Join and 5G handovers, showing new trackability/privacy attacks on these and proposing countermeasures. We study the strength of `TrackDev`'s various trackability properties and show that many of our notions are incomparable amongst each other, thus justifying the fine-grained nature of `TrackDev`. Finally, we detail how the main thrust of `TrackDev` can be mechanised in formal-verification tools. We exemplify this fully on the LoRaWAN Join, in the Tamarin prover. We also uncover and discuss within two important aspects: (a) `TrackDev`'s separation between "explicit" and "implicit" trackability offers new formal-verification insights; (b) our analyses of the LoRaWAN Join protocol in Tamarin against `TrackDev`'s privacy notions, as well as against existing approximations of unlinkability by Baelde et al., concretely show that the latter approximations can be coarser than our notions.**

I. Introduction

Privacy is an umbrella term for a number of notions [40]. But –in most cases– it refers to entities not being linked to their actions. In the strictest case, in a privacy-preserving environment, an entity may wish that no individual action of theirs could ever be linked back to them. Or, in weaker privacy-preserving cases, no group of such actions should be collectively identifiable as being theirs. Meanwhile, if any such discerning can occur, then it is possible to track entities via their actions. Broadly speaking, we refer to this type of privacy breach as "trackability" and its absence (in a system or context) as "non-trackability". Clearly, if entities/devices can be tracked, then they can become the subject of targeted attacks: e.g., their traffic be selectively blocked, their systems be injected with personalised, malicious code, their physical whereabouts be known; notorious examples of this kind were the attacks in the style of "IMSI-catchers" [17], [8], whereby mobile-phone users have been traced around the network based on different identifiers[1], or in the tracking of RFID-tagged goods in the supply-chain [45]. In this work, we place ourselves in the domain of secure and privacy-sensitive systems, and aim to systematically define and analyse meaningful variations notions of such "(non)-trackability" in a generic, not domain-specific manner.

Pfitzmann and Hansen offer a consolidated report [40] of distinct privacy notions such as undetectability, unobservability, pseudonymity, anonymity, identity management, unlinkability and untraceability. We are most interested in notions of privacy closest to "un-traceability" and/or "un-linkability", and specifically in considering these over multiple, concurrent executions of security protocols. Whether we call our notions of interest "(non)-trackability", "(un)linkability' or "(non)-traceability", we intend to capture one aspect: an adversary's (in)ability to link a long-term identifier (or a pseudonym of it) to *secured* application-level traffic in such a way that the adversary would infer "who" issued a given message or took a given action in executions of the systems. In privacy-aware applications, such inferences by attackers are clearly undesirable. Our adversary may attempt this by looking at the whole protocol execution, including the exchanges done before the secure-channel establishment and/or by comparing several executions (i.e., sessions). We choose to use the word "(non)-trackability", to distinguish this line from prior work. Clearly, "non-trackability" is a positive notion (i.e., a privacy requirement), whereas "trackability" denotes a negative aspect that is an attack (i.e., the refutation of a privacy requirement).

Fine-Grained Trackability Notions. We define a framework called `TrackDev` that encompasses several distinct trackability notions. There are privacy notions close to ours, such as (non)-traceability [31], [45] or (un)linkability [19], whereby the latter was studied particularly as part of formal verification in, e.g, [10], [15], [16], [19], [5], [33]. But, as per Pfitzmann's report [40], there are many subtle differences amongst notions related to this. Thus, we believe that a framework that can

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23xxx
www.ndss-symposium.org

[1] We show a different way to track mobile user, even if IMSIs are now obfuscated via changeable/ephemeral identifiers; our tracking is also outside of the Registration procedure, a la [8].

systematise multiple nuances of (un)linkability-related notions, potentially both for practical and formal-methods purposes, is required. `TrackDev` is such a framework and provides increased expressivity and finesse, as it has four *orthogonal* dimensions: (1) whether the attacker is active or passive; (2) whether the attacker is free to chose whom to track or this is imposed by someone/something else (i.e., static vs. adaptive); (3) whether an entity is trackable in its every single appearances or just in a compound set thereof (session trackability vs. session-insensitive trackability); (4) whether the entity can be tracked via its real identity or via some temporary one (explicit vs. implicit trackability).

Generic & Widely-Applicable Trackability Notions. Moreover, our trackability notions are not limited to specific domains (such as RFID [45]), but aim to be applicable to tracking over *any* secure application-level executions/messages.

Realistic & Practical Trackability Notions. Finally, we define our trackability notions in such a way that are also easy to assess in practice, e.g., with tools that capture network traffic, rather than it be just a theoretic/formal notion. We do so in 5G and in IoT traffic. The former means that your whereabouts are known by parties you may not suspect, and the latter that someone can specifically target a precise IoT device to block their access to services. Such is the real-life impact of our findings that the upcoming LoRaWAN specifications v1.2 will contain our countermeasures to the trackability attacks we show in IoT.

Our Contributions.

(1) **New Trackability Framework.** In Section III we define our trackability framework `TrackDev`, which systematically combines protocol-dependent types of tracking (*session-sensitive* ($SesTrack$) vs *session-insensitive* ($Track$)), threat types (passive vs active attackers), and two natures of identifiability (*explicit* vs *implicit*). In `TrackDev`, we also distinguish between *static* (`St-Track`) and *adaptive* (`Ad-Track`) attacks (if the attacker freely chooses their victim, or if this is imposed on them).

(2) **Fine-Grained Trackability Notions.** In Section V we give a full characterisation of the strengths of our trackability notions over their orthogonal dimensions. For example, we show that varying the threats' nature from a *passive* to an *active* attacker does not necessarily increase the trackability powers from *implicit* to *explicit*, or from *session-insensitive* to *session-sensitive*.

(3) **Trackability in Real-life Traffic.** In Section VI we exhibit `TrackDev`'s main attacks in practice on the LoRaWAN Join v1.0 protocol, and on 5G handover procedures.

(4) **Mechanisable Trackability.** In Section VII we show explicit (session) trackability is mechanisable in formal verification tools without any loss, and discuss the gaps and impossibilities in mechanising our other trackability notions. Our mechanisation of trackability is in the Tamarin tool [39], on the LoRaWAN Join v1.1 protocol.

(5) **Value-Added Unlinkability Analysis.** In Section VII-E we align `TrackDev` with previous relevant privacy notions, and show that `TrackDev`'s distinction between *implicit* and *explicit* trackability is also useful in formal methods. We exemplify concretely, in Tamarin [39], that prior approximations of unlinkability [5] have been coarse and `TrackDev`'s explicit trackability can be finer.

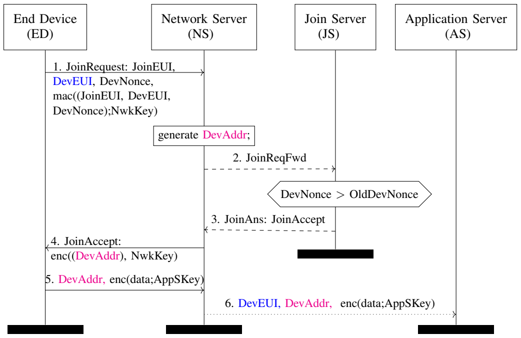


Fig. 1. LoRaWAN Join v1.1[2]

(6) **Novel Attacks, Both in Practice & in Formal Verification.** The three main attacks we show are new. Moreover, they were found via mechanisations of `TrackDev`, be it in practical, traffic-capturing tools (for one attack on the LoRaWAN v1.0 protocol and one on 5G handovers), or in the formal-verification Tamarin tool [39] (for one attack on LoRaWAN v1.1). Finally, we disclosed our privacy flaws to the Lora Alliance and the upcoming LoRaWAN specifications v1.2 will contain our countermeasures.

II. Preliminaries

Now, we briefly describe the authenticated key-establishment (AKE) protocol called the "LoRaWAN Join v1.1". We use this protocol as a case study: we find new trackability attacks on it by using practical tools, as well as formally verifying our and others' trackability notions on it.

The LoRaWAN Join protocol [12], [13] is used by Internet of Things (IoT) devices in order to (re-)connect to a LoRaWAN network. After this (re)connection, these devices can run the IoT application stored on them, and, most importantly to us, they can securely send and receive application-level messages over the LoRaWAN network.

A. LoRaWAN Join v1.1

Fig. 1 gives a simplified description of the LoRaWAN Join v1.1 protocol, and we now provide details on that:

(1) The End Device (ED) sends a plaintext *JoinRequest* message, comprised of a *DevNonce*, a *DevEUI*, a *JoinEUI*, and a Message Authentication Code (MAC) to the NS. The *DevEUI* is the long-term identifier of the ED, while the *DevNonce* is based on a counter starting at 0 and incremented with every *JoinRequest*. The *JoinEUI* is the identifier of the LoRaWAN Join Server (JS) with which the device is registered. The MAC authenticates the *DevNonce*, *DevEUI*, *JoinEUI*, using a long-term key, *NwkKey*, which devices share with the JS.

(2) The NS generates a *DevAddr* (an ephemeral identifier) to be used until a next Join/Re-Join message. The NS appends this to the *JoinRequest* and forwards the result to the JS, identified via the *JoinEUI* inside the *JoinRequest*.

(3) The JS verifies the MAC using its copy of *NwkKey* and checks that the *DevNonce* value is larger than the

[2] Dashes vs. lines: secure vs. insecure channels; dots: down to implementers.

DevNonce value previously associated with the *DevEUI*. If these checks fail, no response is given. If they pass, then the JS creates a *JoinAns*, which contains (amongst other data) a *JoinAccept* message. This message is comprised of the *DevAddr* and a *JoinNonce* generated by the JS, all encrypted with *NwkKey*. The *JoinAns* is sent to the NS.

(4) From the *JoinAns*, the NS extracts and forwards the *JoinAccept* to the ED. The ED decrypts the *JoinAccept* (using *NwkKey*), checks it against its local view, and uses the values inside to compute an *AppSKey* session-key. At this point, the Join procedure is complete and the ED is able to send encrypted application-level messages to an Application Server (AS) via the NS proxy.

(5) The ED constructs an application-level data-packet encrypted with *AppSKey*, and places in its header its new *DevAddr*. This is sent to the NS.

(6) The NS appends the *DevEUI* and forwards the encrypted data packet to the AS. The AS retrieves the associated *AppSKey* for the *DevEUI* and decrypts the data packet.

— After the Join and until a ReJoin (possibly for some days), the ED will send messages as per step 5.

This description of the LoRaWAN Join Protocol [12], [46] is sufficiently detailed for our subsequent discussions of it.

III. The Trackability Framework TrackDev

In this section, we define our TrackDev framework for trackability analysis in secure, networked applications.

A. *Identification-Based Functionality* IDnServe*, Protocols &* TrackDev *Terminology*

Def. 1 gives a generic *identification functionality* called IDnServe. This functionality generalises and idealises the case where a server identifies a client, after which the former provides some application-level service to the latter. In the actual implementation of such an ideal functionality, this application-level service can be as simple as providing access to a network or granting a permission; the functionality operates at the idealised level, so we are not concerned with specifying what the actual service is.

Definition 1. IDnServe ***Functionality.*** *An* IDnServe functionality Π *is described via:*

- *Two entities A and B running probabilistic polynomial time algorithms interacting on channels, such that id_A is a long-term identifier which uniquely identifies the entity A, and which is known to, or verifiable by, B;*
- *The algorithms run by the two parties A and B encode the following exchange:*

$1.\ A \ \rightarrow B : m_1$ *//pre-application msg.*

$\ldots$

$[p.\ B \rightarrow A : m_2]$ *//optional pre-application msg.*

$\ldots$

$q.\ A \ \rightarrow B : m_3$ *// application msg.*

$\ldots,$

where the above notation is as follows:

(1) “$\ldots$” denotes there can be more (non-essential) exchanges, while “[ ]” denotes an optional step in the functionality;

(2) 1, p (if it is used), and q are steps in the cross-interaction of A and B. Due to the optional nature of steps it is possible that in certain realisations of IDnServe*, q=2, i.e., B sends no pre-application message to A;*

(3) application message *m_3 delivers application-dependant functionality from A to B, whereas* pre-application messages *m_1, m_2 do not (i.e., can be viewed as initialisation steps);*

(4) the channels/communications between parties A and B are public, yet messages m_1, m_2 and m_3 may be encrypted and authenticating,

and the following two functional requirements hold:

Req 1. *As part of steps 1 to q, B identified A via id_A.*

Req 2. *In step q and/or thereafter, A is accessing the application-level service that B is facilitating, based on the identification that took place in steps 1 through to q.*

Def. 1 above fully defines IDnServe. But, other aspects w.r.t. IDnServe are notable: (i) Optionally, B confirms A's identification back to A, via the message in step p; (ii) Whilst entity B is identifying A as being the sender of message m_3, the message m_3 (or those thereafter) may not contain id_A in clear-text; (iii) Messages in step q or other steps may contain individually or collectively, in an explicit or implicit form, an *ephemeral identifier of* A, i.e., a temporary pseudonym for id_A.

Functionalities similar to IDnServe exist and model authenticated key-exchange, access-control, etc. However, we aim for IDnServe to be more generic. (1) We only need to express that unique IDs (e.g., id_A) exist in protocols, without differentiating if they are cryptographic keys, serial numbers, etc.; (2) We do not need the cryptographic machinery be modelled. We need just a means to say messages are linkable to real or ephemeral IDs.

Wide Realisation of IDnServe***. The*** IDed-Prot ***Protocols****:* We use the term *identification-based security protocol/system/application* IDed-Prot to refer to any protocol/system/application π that realise/implement[3] IDnServe, i.e., it conforms with statements 1–4 in Def. 1 and satisfies requirements *Req*. 1 and *Req*. 2; this realisation is purely functional and the threat model does not matter. Henceforth, we use *“protocol”* to denote any protocol/system/application realising IDnServe. The part of the functionality pertaining to entity A, in a protocol realising IDnServe, is called the *“role of A”*.

An IDed-Prot protocol may contain “conditionals”, ı tests in its logic (e.g., to check counters or other stateful data). We refer to such instances of IDed-Prot protocols as *identification-based protocols with conditionals*.

Applicability of TrackDev*:* There are numerous security protocols that realise identification functionality. In fact, steps 1 to q are realised by most AKE protocols: Bluetooth [30], key-establishment procedures in mobile networks, e.g., [1], and IoT protocols [12], [13]. Indeed, all our trackability definitions (e.g., Defs. 4, 5) apply to any traffic generated by such protocols. Thus, TrackDev is widely applicable, in fact,

[3] We do not formalise “realisation” here; it is as per usual [38].

it aims to track a party over encrypted messages it sends in the case where this party has been implicitly/explicitly identified in this traffic via long-term or ephemeral identifiers.

The LoRaWAN Join is an* `IDed-Prot` *Protocol: The `IDnServe` entity A is LoRaWAN's ED, while LoRaWAN's NS, JS, and AS form the `IDnServe` entity B. Mapping the LoRaWAN Join further to the `IDnServe` terminology, we have:

- `IDnServe`'s message m_1 in step 1 is LoRaWAN's *Join-Request* message;
- `IDnServe`'s message m_2 in step p is LoRaWAN's *JoinAccept* message;
- `IDnServe`'s message m_3 is LoraWAN's message 5: $(DevAddr, enc(data; AppSKey))$.

As aforementioned, like most `IDed-Prot` protocols, the LoRaWAN Join protocol has additional messages but they are immaterial to the realisation of `IDnServe`.

Parties/Entities' Identification in `IDnServe`*:* We do not specify, on purpose, if this id_A is a serial number, a public key, or a virtual identifier that B can extract out of the pre-application messages; all we require is id_A uniquely identifies the actual entity, a.k.a. the *party*, sending messages m_1 and m_3. At the level of the identification functionality, we also do not need to mandate how B gets this id_A: e.g., id_A could be sent either in clear-text, encrypted, in code, or otherwise.

B. Adversary & Execution Model

Our proposed framework `TrackDev` considers a powerful adversary akin to a Dolev-Yao (DY) adversary [20]. Like DY adversaries, our attacker is not concerned with breaking the cryptographic primitives used in the protocols. In terms of actions, our adversary can passively capture, actively block and inject messages on/from channels. Consequently, we define the *type* $\mathcal{T}$ of our adversary by their actions. If the adversary only listens and captures messages, then their type $\mathcal{T}$ is *passive*. If the adversary acts on the channels, e.g., by blocking or injecting messages, then their type $\mathcal{T}$ is *active*.

C. Preview on `TrackDev`*'s Trackability Notions*

For `IDed-Prot` protocols, we will formalise a set of privacy requirements called *"non-trackability (*`NoTrack`*)", broadly* encoding that an ID in such protocols' executions is not trackable by the attacker. We also specialise this to non-tracking over individual protocol sessions/executions: *"session non-trackability (*`SesNoTrack`*)"*. Both `NoTrack` and `SesNoTrack` can be qualified further as: *static (*`St-`*)* vs. *adaptive (*`Ad-`*)* denoting if the attacker is free to choose whom to track or not; *explicit* vs. *implicit*, denoting the nature of the ID tracked; both can be acted upon by *active* vs. *passive* attackers. Finally, each notion can be *universal* – if no ID in the protocol executions is non-trackable (∀-), or *existential* – if just some IDs in the protocol executions are non-trackable (∃-), If we lay out all our non-trackability notions, over all these dimensions and flavours, we get Table I.

	∃-St-NoTrack	∃-St-SesNoTrack	∀-St-NoTrack	∀-St-SesNoTrack	∃-Ad-NoTrack	∃-Ad-SesNoTrack	∀-Ad-NoTrack	∀-Ad-SesNoTrack
implicit, passive	1	2	3	4	5	6	7	8
implicit, active	9	10	11	12	13	14	15	16
explicit, passive	17	18	19	20	21	22	23	24
explicit, active	25	26	27	28	29	30	31	32

TABLE I. NON-TRACKABILITY NOTIONS WITH ALL `TrackDev` DIMENSIONS FIXED

Clearly, Table I shows that we have 32 notions of non-trackability where each notion has all our trackability dimensions fixed/prescribed. However, there are many more notions of non-trackability if we leave one or more dimensions "free": e.g., static non-trackability (`St-NoTrack`) is a valid notion in its own right (and subsumes all its flavours of "implicit/explicit", "active/passive", "universal/existential"); similarly, non-trackability (`NoTrack`) is a valid notion in `TrackDev` which subsumes all possible flavours of more precise non-trackability notions.

Finally, for any *non-trackability requirement (*`NoTrack`*)*, we also have the corresponding notion of a *trackability attack (*`Track`*)*, e.g., static non-trackability (`St-NoTrack`) is refuted by a static trackability (`St-Track`) attack. We often write *"*`NoTrack`*"* and *"*`Track`*"* in the generic sense, and we add precise detail when needed: e.g., adding "`Ses`" making it session-sensitive, as in *"*`SesNoTrack`*"* and *"*`SesTrack`*"*.

One may already anticipate that some non-trackability notions are weaker than others. Importantly, many non-trackability notions are incomparable, meaning that there is no guarantee that breaking one leads to breaking the other: e.g., *implicit, active* `NoTrack` vs. *explicit, active* `SesNoTrack`. All of the hierarchy and separation results will be discussed in Section V.

Note: All the measures that follow are polynomially quantified; i.e., if we formally considered a security parameter, then the number of executions (available to the adversary) would be polynomial in this parameter, as would the adversary's execution time to refute our privacy definitions.

D. Formalising Non-Trackability

1) ***Attack Setups****:* All our (non-)trackability definitions use the notion of an *attack setup* in Def. 2. This models a "handle" to protocol executions, to which the attacker has access to in order to track IDs/parties in the protocol.

Definition 2. ***Explicit/Implicit Non-Trackability Attack Setups.*** *Let π be an* `IDed-Prot` *protocol. Let $\mathcal{I}d$ be the set of all identifiers of A parties associated with π, known to all parties (including adversaries). Let $\mathcal{E}$ be the set of executions of π such that they contain the pre-application and application-level messages coming from* at least two[4] *parties A_1, A_2, …, A_n of role A with* real *identifiers id_{A_1}, id_{A_2}, id_{A_3}, …, $id_{A_n} \in \mathcal{I}d$.*

A non-trackability attack setup for π's executions $\mathcal{E}$ *is a tuple $S(\mathcal{E})$=$(\mathcal{M}_{id}, \mathcal{M}_{app})$ as follows:*

(a) $\mathcal{M}_{id}$ is an arbitrarily chosen*, non-empty set of pre-application identification messages appearing in $\mathcal{E}$;*

(b) $\mathcal{M}_{app}$ is an arbitrarily chosen*, non-empty set of application messages appearing in $\mathcal{E}$.*

The non-trackability attack setup S is called explicit *if the real identifiers id_{A_1}, id_{A_2}, … are present in the executions*

[4] Otherwise, our next non-trackability definitions can be refuted trivially due to the fact that all application-level messages come from one single party.

$\mathcal{E}$ of protocol π by each party, and so are recorded in the sets $\mathcal{M}_{id}$ and $\mathcal{M}_{app}$.

The non-trackability attack setup S is called implicit *if the real identifiers id_{A_1}, id_{A_2}, … are not present in the executions $\mathcal{E}$ of protocol π by each party, and as such are not recorded in the sets $\mathcal{M}_{id}$ and $\mathcal{M}_{app}$. Bijections of the real identifiers id_{A_1}, id_{A_2}, … to their* implicit identifiers *can be associated with said executions by honest parties who know the full data of these executions*[5].

In subsequent trackability notions, our *attack setups* $S(\mathcal{E})$ (in Def. 2) model an attacker interacting with a series of executions $\mathcal{E}$ of an IDed-Prot protocol observing, interacting with, and ultimately recording all or part of these executions. Such *an attack setup* $S(\mathcal{E})$ constitutes the "material" the attacker uses to mount their attack *onto the executions* $\mathcal{E}$.

a) **Explicit vs. Implicit Attack Setups:** Def. 2 distinguishes between explicit and implicit attack setups. An explicit attack setup will contain long-term, real IDs of parties and as such, an attacker will be able to track them. An implicit attack setup corresponds to executions of an IDed-Prot protocol where the long-term IDs exist, but the executions hide them (from the attacker).

*2) **Trackability Relations**:* Def. 3 below gives another notion which our trackability definitions will use; it encapsulates what the attacker has to output at the end of a successful trackability attack.

Definition 3. ***Explicit/Implicit Trackability Relations.*** *Let π be an* IDed-Prot *protocol.*

Given an explicit attack setup $S(\mathcal{E})$=$(\mathcal{M}_{id}, \mathcal{M}_{app})$ for a set $\mathcal{E}$ of π's executions, an explicit trackability relation *for $S(\mathcal{E})$ is a non-empty relation $Tr \subseteq \mathcal{I}d \times \mathcal{P}(\mathcal{M}_{app})$*[6] *such that for some $id_A \in \mathcal{I}d$, we have $(id_A, M_A) \in Tr$ if:*
(i) the set $M_A \subseteq \mathcal{M}_{app}$ is non-empty, i.e., $M_A \neq \emptyset$,
(ii) $\forall m \in M_A$, m was sent by the entity A with id_A.

Given an implicit attack setup $S(\mathcal{E})$=$(\mathcal{M}_{id}, \mathcal{M}_{app})$ for a set $\mathcal{E}$ of π's executions, an implicit trackability relation *is a non-empty relation $Tr \subseteq \phi(\mathcal{I}d) \times \mathcal{P}(\mathcal{M}_{app})$, such that:*
(iii) ϕ is a bijection from real id-s in $\mathcal{I}d$ to ephemeral id-s,
(iv) for some $\phi(id_A) \in \phi(\mathcal{I}d)$, we have $(\phi(id_A), M_A) \in Tr$ if conditions (i), (ii) above hold.

Def. 3 gives relations that index application-level messages in a protocol execution by IDs, be it real ones or aliases. In the first case, the relation is called *explicit* and in the second case *implicit*. These are the mappings that a trackability attacker will be asked (via subsequent definitions) to create in order to "demonstrate" that they have tracked some party via some protocol executions $\mathcal{E}$ "encapsulated" in an attack setup $S(\mathcal{E})$ as per the above.

*3) **Who Is Tracked & How**:*

a) **Who Is Tracked:** For clarity, we reiterate some aspects around parties and roles. To simplify the notation, in the definitions for attack setups (Def. 2) and trackability relations (Def. 3), "A" is overloaded to mean a role as well as party/entity playing that role. For instance, the *role* A denotes a software/algorithmic client, whereas the *parties* $A_1, A_2, A_3, \ldots$ playing this role are actual machines/computers. Then, id_{A_1}, $id_{A_2}, \ldots$ denote e.g., the Media Access Control (MAC) addresses id_{alice}, $id_{alison}, \ldots$, of these machines. In our definitions when the attacker is asked to track one specific id_A in the series id_{A_1}, $id_{A_2}, \ldots$, that would be to track $alice$, $alison$, … via the messages they sent, and index these messages correctly as coming from id_{alice}, id_{alison}.

b) **How One Is Tracked:** If $alice$ has multiple IDs, for instance id_{alice1} and id_{alice2}, then one trackability relation can be that id_{alice1} is linked to a message, and another relation where both $(id_{alice1}, \cdot)$ and $(id_{alice2}, \cdot)$ appear linked to some message each. In other words, our trackability relations allow for parties with multiple IDs to be tracked via each ID, via some IDs, or via all IDs. Meanwhile, $alice$ can also be tracked not just by their one/all long-terms IDs, but also by ephemeral IDs they may have, e.g., session data, pseudonyms, etc. Finally, the trackability relation defined thus far in Def. 3 does not distinguish between tracking a party via their execution i vs. via their execution j. Def. 7 refines that: i.e., our notions also capture distinct tracking per each session/execution. Put simply, in TrackDev, $alice$ can be tracked via one or all of their real, long-term identifiers, or via one or all of their ephemeral pseudonyms, in each of their executions/sessions or indistinctly in their overall traffic.

E. Non-Trackability Flavours

Having given the notions of attack setups and trackability relations, we will continue with our non-trackability definitions, which will mainly depend on four aspects:

(1) the adversary chooses themselves whom to track (*static trackability*) vs. the adversary is given an attack setup and a specific identifier to track[7] (*adaptive trackability*);
(2) the adversary is able to track A but is unable to single out the execution of the IDed-Prot protocol that A has run (*non-session trackability*) vs. the adversary being able to track A as well as identify every execution of the IDed-Prot protocol that A runs (*session trackability*);
(3) the adversary intervenes in the executions (*trackability with active attacker*) vs. the adversary just observes the executions (*trackability with passive attacker*);
(4) the IDs tracked are the real, long-term IDs appearing in the protocol (*explicit trackability*) vs. the IDs tracked are some ephemeral versions of the real IDs, produced "on the fly" during the protocol executions or by the attacker for the purposes of tracking (*implicit trackability*).

*1) **Static Non-Trackability**:* We first look at a type of trackability we call *"static"* in Def. 4 below. The IDs, which are tracked or the attack setup under which they are tracked, can be chosen by the adversary, i.e., are not pre-imposed.

[5] An honest party id_{A_1} will know they executed under some alias. Yet, there may not even exist a polynomial algorithm D could retrieve these aliases when D is not part of honest executions, even if D has observed the executions.

[6] $\mathcal{P}$ denotes the power set of a set.

[7] We use the terminology of "static" and "adaptive" attackers as per the usual ways in security [18]. Therein, adaptive attackers can change behaviour during their attack. Similarly, herein adaptive attackers have to "adapt" to track an identity not of their choice, by imposed by someone/something else.

Definition 4. ***Static Explicit/Implicit Non-Trackability With Adversary of Type*** $\mathcal{T}$ **(**$\texttt{St-NoTrack}^{\mathcal{T}}$**).** *Let* π *be an* $\texttt{IDed-Prot}$.

We say that π *achieves* static explicit/implicit non-trackability w.r.t. an adversary of type $\mathcal{T}$ ($\texttt{St-NoTrack}^{expl,\mathcal{T}}$) *if given that:*
(i) $\forall \mathcal{A}dv$ *adversaries of type* $\mathcal{T}$*,*
(ii) $\forall$ *sets* $\mathcal{E}$ *of* π*'s execution and the resulting non-trackability attack setup* $S(\mathcal{E})$*,*
adversary $\mathcal{A}dv$ *cannot produce an explicit/implicit trackability relation for* $S(\mathcal{E})$*.*

Def. 4 states that a protocol is statically non-trackable (explicitly/implicitly) if there is no attacker that can find the right executions to observe or interact with, such that it would allow them to pinpoint the application-level messages of at least one explicit/implicit ID in these executions.

To indicate that static non-trackability is explicit or implicit, we write $\texttt{St-NoTrack}^{expl.,\mathcal{T}}$ or $\texttt{St-NoTrack}^{impl.\mathcal{T}}$, respectively. When the type $\mathcal{T}$ of adversary is clear or unimportant, we omit it from the notation.

Universal vs. Existential (Static) Non-Trackability: Def. 4 has an implicit universal quantification inside its statement: i.e., the attacker is free to choose *any* identifier to track. So, we sometimes refer to static non-trackability as *universal static non-trackability* and we use the "$\forall$" symbol in its shorthand $\forall$-$\texttt{St-NoTrack}$. To specifically say which identifier id cannot be statically tracked, we write $\texttt{St-NoTrack}^{\mathcal{T}}(id)$.

Non-Trackability vs. Trackability: Def. 4 and its notion $\texttt{St-NoTrack}$ speak of a privacy requirement dubbed as ***non***-trackability. In general, to denote the refutation of such requirement, we drop the "non" and speak of *trackability* attacks. In such refutations, the quantifications would naturally change: for instance, a universal static non-trackability requirement can yield an *existential static trackability attack*, written $\exists$-$\texttt{St-Track}^{\mathcal{T}}(id)$, meaning that some identifier id can be tracked under the conditions of Def. 4.

2) ***Adaptive Non-Trackability****:* We define another notion of non-trackability called *"adaptive non-trackability"*. In this case, the adversary is challenged as to which identifier they are supposed to track, thus having to "adapt" their behaviour.

Definition 5. ***Adaptive Explicit/Implicit Non-Trackability With Adversary of Type*** $\mathcal{T}$ ***for*** id_A **(**$\texttt{Ad-NoTrack}^{\mathcal{T}}(id_A)$**).** *Let* π *be an* $\texttt{IDed-Prot}$ *protocol.*

We say that π *achieves* adaptive explicit/implicit non-trackability of id_A w.r.t. an adversary of type $\mathcal{T}$ ($\texttt{Ad-NoTrack}^{\mathcal{T}}(id_A)$) *if given that:*
(i) for an arbitrarily chosen set $\mathcal{E}$ *of* π*'s executions,*
(ii) for any arbitrarily chosen $id_A \in \mathcal{I}d$*, such that* $\mathcal{E}$ *contains pre-application and application messages by* id_A*,*
(iii) $\forall$ *adversaries* $\mathcal{A}dv$ *of type* $\mathcal{T}$*,*
(iv) $\forall$ *explicit/implicit attack setups* $S(\mathcal{E})$*,*
adversary $\mathcal{A}dv$ *cannot produce an explicit/implicit trackability relation for* $S(\mathcal{E})$ *containing a tuple* $(id_A, \cdot)$ *in the explicit case, or* $(\phi(id_A), \cdot)$ *in the implicit case for some bijection* ϕ*.*

To see an example of refuting trackability (be it static or adaptive) in our LoRaWAN case study, refer to Example 1 in Section III-G.

Universal vs. Existential (Adaptive) Non-Trackability: Def. 5, via $\texttt{Ad-NoTrack}^{\mathcal{T}}$, has an implicit universal quantifier in it; so, we also call it *"universal adaptive non-trackability (*$\forall$-$\texttt{Ad-NoTrack}^{\mathcal{T}}(id_A)$*)"*. As with the static case, we also employ the dual notion too: *existential adaptive non-trackability (*$\exists$-$\texttt{Ad-NoTrack}^{\mathcal{T}}(id_A)$*)* for not being able to adaptively track *some* given id_A.

Existential trackability $\exists$-$\texttt{Ad-NoTrack}^{\mathcal{T}}(id_A)$ may appear a weak guarantee: protocol π would still achieve adaptive non-trackability $\exists$-$\texttt{Ad-NoTrack}^{\mathcal{T}}(id_A)$, even if the adversary could track all identifiers except for one id_A. However, this is strengthened by the fact that id_A is named by a challenger, who also imposed the attack setup onto the adversary.

Static vs. Adaptive Non-Trackability: The most important difference between static and adaptive trackability is the two different orders of quantification between $\mathcal{A}dv$ and the executions $\mathcal{E}$. In the adaptive case, any adversary that mounts an attack would depend on the set $\mathcal{E}$, whereas in the static case such an adversary would be entirely free to chose the executions to attack.

F. Session Non-Trackability

We now look at trackability that explicitly pins IDs not only to application messages, but also to individual protocol runs. In Defs. 4 and 5, the adversary is not concerned in saying something as specific as "this message belongs to id_A as a consequence of id_A executing this/a specific run of the protocol". Up to now, these notions of trackability do not look at when the party with id_A created the message; Defs. 4 and 5 indiscriminately consider all runs of π, i.e., trackability does not separate id_A's application messages in the i^{th} run of π from id_A's application messages in j^{th} run of π. To achieve this notion, we introduce *session trackability* by strengthening our previous definitions with an indexing of the sets $\mathcal{M}_{id}$ and $\mathcal{M}_{app}$ of messages by the corresponding i^{th} run of the protocol π. We then require the adversary to not only say if a given $m \in \mathcal{M}_{app}$ belongs to a given id_A, but also whether id_A has produced it in the i^{th} run of π from the set of runs that underpins the messages $\mathcal{M}_{app}$.

Sessions*:* As per usual, identification-based security protocols would run multiple times for the same entities. As normal, we refer to one such run of the protocol as a *session*. If multiple sessions for one entity A with id_A occur, then id_A would be identified by entity B more than once over this period of time. In Def. 6, we extend attack setups (Def. 2) to operate over sessions.

Definition 6. ***Session Non-Trackability Attack Setups.*** *Let* π*,* $\mathcal{I}d$ *and* $\mathcal{E}$ *be as per non-trackability attack setups (Def. 2). Let* $Sess$ *be an arbitrarily chosen set of sessions associated with all* $id_A \in \mathcal{I}d$ *where a session* s *for* id_A *is defined as above.*

A non-trackability attack setup for π's executions $\mathcal{E}$ and sessions $Sess$ *is a tuple* $S(\mathcal{E}, Sess)$=($\mathcal{M}_{id}^{Sess}, \mathcal{M}_{app}^{Sess}$) *with:*
(i) $\mathcal{M}_{id}^{Sess}$ *is an arbitrarily chosen family of pre-application messages appearing in* $Sess$*;*
(ii) $\mathcal{M}_{app}^{Sess}$ *is an arbitrarily chosen family of sets of application messages appearing in* $Sess$*.*

We extend trackability relations in Def. 3 over sessions:

Definition 7. ***Session Trackability Relations.*** *Let π be an* `IDed-Prot` *protocol.*

Given a session attack setup $S(\mathcal{E}, Sess)$=$(\mathcal{M}_{id}^{Sess}, \mathcal{M}_{app}^{Sess})$ for a set of π's executions $\mathcal{E}$ and a set of π's sessions $Sess$, a session trackability relation for *$S(\mathcal{E}, Sess)$ is a non-empty relation $Tr \subseteq \mathcal{I}d \times Sess \times \mathcal{P}(\mathcal{M}_{app}^{Sess})$ defined as follows:*
$(id_A, i, M_A^i) \in Tr$ if:
(i) $M_A^i \neq \emptyset$,
(ii) $M_A^i \subset \mathcal{M}_{app}^{Sess}$, $i \in S$,
(iii) $\forall m \in M_A^i$, m was sent by the entity A with id_A in its i^{th} session.

We lift the static and adaptive trackability notions in Defs. 4 and 5 to their finer, session-based counterparts. Thus, we respectively get: *static session non-trackability* (`St-SesNoTrack`) and *adaptive session non-trackability* (`Ad-SesNoTrack`). These lifts make trackability operate in a session-sensitive manner: they both ask that the adversary produce not binary, but ternary relations which include session identifiers in their tuples.

To see an example refuting session trackability on our LoRaWAN case study, refer to Example 2 in Section III-G.

Adaptive, Static, Universal and Existential Session Non-Trackability Flavours: All the concepts of "adaptive", "static", "universal", "existential" transfer identically from the case of session-insensitive non-trackability (`St-NoTrack`) to the case of session-sensitive non-trackability (`St-SesNoTrack`). We thus have the full range of properties: from strong positive requirements like *universal static session non-trackability* ($\forall$-`St-SesNoTrack`) to the weaker classes of attacks such as *existential adaptive session trackability attacks* ($\exists$-`Ad-SesTrack`), etc. All notations also transfer from the session-insensitive case with "*Ses*" before "*NoTrack*" or "*Track*" being added to denote the session-sensitive case.

G. Illustration of Refuting Non-Trackability

We now give examples on how to refute non-trackability as per Def. 4, Def. 5, and their session-based variants in Def. 7.

Example 1. ***How trackability attacks could look on LoRaWAN Join v1.1.*** *In this example, we do not distinguish between adaptive and static cases, but only between universal and existential. For illustration purposes, we exclude the pre-application messages observed by the attacker; this impacts the example in no way.*

Assume the adversary obtained the following traffic:

- *$(DevAddr_1, data_1), \ldots, (DevAddr_n, data_n)$, where $(DevAddr_i, data_i) \in \mathcal{M}_{app}$ for $1 \leq i \leq n$ and these originated from the same $DevEUI_1$ after different executions of the LoRaWAN Join v1.1 by this $DevEUI_1$.*
- *$(DevAddr'_1, data'_1), \ldots, (DevAddr'_n, data'_n) \in \mathcal{M}_{app}$ and these originated from a DevEUI different from $DevEUI_1$, after different executions of the LoRaWAN Join v1.1 of that DevEUI.*

Any of the following relations form a valid existential trackability attack ($\exists$-`Track`) on the LoRaWAN Join:

$$\{(DevEUI_1, \{(DevAddr_{i_1},\ data_{i_1}),\ \ldots,\ (DevAddr_{i_k},\ data_{i_k})\})\}$$

where $0 < k \leq n$ and $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$.

In other words, there would be an existential trackability attack against the LoRaWAN Join v1.1, if the adversary could track just one of the 2 devices, e.g., $DevEUI_1$, to some of its application data: one message, 2 messages, or all n messages.

Similarly, consider the adversary observed/obtained:

$(DevAddr_1, data_1), \ldots, (DevAddr_n, data_n)$, where $(DevAddr_i, data_i) \in \mathcal{M}_{app}$ for $1 \leq i \leq n$, and they originated from n distinct EDs. Assume (for simplicity) that each $DevAddr_i$ is associated with $DevEUI_i$ in the Join.

A universal trackability attack ($\forall$-`Track`) on this dataset would mean the adversary can produce the following relation:

$$\{(DevEUI_1, \{(DevAddr_1, data_1)\}),\ \ldots,\ (DevEUI_n, \{(DevAddr_n, data_n)\})\}$$

So, there would be a universal trackability attack against the LoRaWAN Join v1.1 if the adversary were able to track not just one, but each $DevEUI$ (i.e., $DevEUI_1, \ldots, DevEUI_n$) to some of its application data.

Example 2. ***How session trackability attacks could look on LoRaWAN Join v1.1.*** *Let $m_{id_A}^s$ denote the instance of the message m in a session s of a protocol π run by party A with id_A.*

For the LoRaWAN Join v1.1 protocol, let us assume a device $DevEUI_1$ ran the Join in two sessions denoted i and j, and device $DevEUI_2$ ran the Join in two sessions denoted k and l; also, we assume both ran more Join sessions and other devices were running concurrently in multiple sessions. Assume that the set $\mathcal{M}_{app}^{Sess}$ contains messages:

−$(DevAddr_1, a_{DevEUI_1}^i)$, from $DevEUI_1$
$(DevAddr_1, b_{DevEUI_1}^i)$ in session i,
−$(DevAddr_2, c_{DevEUI_1}^j)$, from $DevEUI_1$
$(DevAddr_2, c_{DevEUI_1}^j)$ in session j,
−$(DevAddr_3, e_{DevEUI_2}^k)$, from $DevEUI_2$
$(DevAddr_3, f_{DevEUI_2}^k)$ in session k,
− similar from $DevEUI_2$ in session l,
− other such messages.

On an attack setup partly described as above, a relation sufficient to produce an existential session trackability attack ($\exists$-`SesTrack`) on the LoRaWAN Join v1.1 is:

$$(DevEUI_1, i, (DevAddr_1, a_{DevEUI_1}^i))$$

The attacker needs to link at least one $DevEUI$ to at least one message in their mutual session. The relation above does so for $DevEUI_1$, message a, and session i.

For session-based trackability, not only can the attacker identify the messages of an id_A but can partition these by different sessions. In practice, this would lead to more serious and very targeted attacks: an intruder can stop/pollute the messages of a given id_A in a given execution.

Clearly, non-trackability is a weaker property than session non-trackability, meaning that if a party id_A cannot be tracked in general, then they cannot be tracked session-wise, either.

IV. Concrete Trackability Attacks

In this section, we show actual trackability attacks on the LoRaWAN Join v1.1; we found these attacks either in practice, `LoRaWAN-Attack1` (Section VI), or via formal verification, `LoRaWAN-Attack2` (Section VII). These have been acknowledged by the LoRa Alliance and one of our solutions is now being adopted for LoRaWAN v1.2 specifications. In Section VI, we also show trackability attacks on a 5G procedure.

As we said in Section III-G, the trackability challenge in the LoRaWAN Join v1.1 (see Fig. 1) is to link the *DevEUI* to the corresponding $DevAddr$, be it in every Join session in which a $DevAddr$ is issued ($\forall$-`St-SesNoTrack`), or at least in some sessions, indiscriminately ($\forall$-`St-NoTrack`).

A. Explicit Trackability Attacks

1) ***`LoRaWAN-Attack1`*** ***: An*** `St-SesTrack`$^{expl.,passive}$ ***Attack on the LoRaWAN Join v1.1:*** This attack works in the following setting: n devices are already sending application messages and only one new deviceidentified, e.g., via $DevEUI_1$, is now (re)joining the network.

In this scenario, an adversary only has to observe the *(Re)JoinRequest*, see a *JoinAccept* message, and then a soon-to-come application message. Not only is this application message identifiable as it comes in a predictable number of seconds (≈2 seconds) after the *JoinAccept* message, but it is also characterised by its frame counter, *FCntUp*, being at 0 (this counter is reset to 0 after each successful Join/ReJoin). All other n devices will have their *FCntUp* not at 0, as this counter increments with every uplink message. Consequently, this first uplink/application message can be easily observed by a passive adversary, who can thus extract the $DevAddr$ from the message header and then link the $DevAddr$ to the $DevEUI_1$ sent as part of the original *JoinRequest*. They can also continue to track the application messages from this $DevAddr$ with counter *FCntUp* equal to $1, 2, \ldots$, etc.

2) **`LoRaWAN-Attack2`: An** `St-SesTrack`$^{expl.,active}$ **Attack on the LoRaWAN Join v1.1:** This attack works in the following setting: there are at least two devices sending a *JoinRequest* roughly at once (e.g., within a window of ≈1 second, if the response time to the *JoinRequest* is ≈6 seconds).

There are two **attack strategies** for an active adversary, and both result in one of the two *JoinRequest*s not receiving a *JoinAccept* from the JS. These strategies are as follows:

- **Selective Blocking:** Given two devices $DevEUI_1$ and $DevEUI_2$, an active adversary can block the *JoinRequest* message of $DevEUI_1$ while letting the *JoinRequest* message of $DevEUI_2$ through.
- **Forced Condition:** The LoRaWAN Join v1.1 protocol is an *identification-based protocol with conditionals* (see Section III). If an adversary changes a $DevNonce$ in a *JoinRequest* to a certain incorrect value (e.g., 0), the JS will drop the *JoinRequest*; the "conditional" on $DevNonce$s being strictly increasing will fail, as will the verification of the MAC included in the *JoinRequest*.

This is therefore similar to the `LoRaWAN-Attack1`, with only one device, $DevEUI_2$, receiving a successful *JoinAccept*. The adversary can again link it to the first application message containing $DevAddr_2$ as described above.

3) ***Countermeasure to*** ***`LoRaWAN-Attack1`*** ***&*** ***`LoRaWAN-Attack2`*** *:*

Our `Join-EncRequest` Protocol. To counteract `LoRaWAN-Attack1` and `LoRaWAN-Attack2`, we encrypt the *JoinRequest* message using a public key of the NS[8]. That way, the *DevEUI* is not revealed to the adversary, and hence, cannot be linked to the subsequent $DevAddr$.

Our countermeasure via the `Join-EncRequest` protocol only does away with explicit trackability and session trackability. A much weaker, not session-sensitive `St-Track`$^{impl.,active}$ attack is still possible on our `Join-EncRequest` protocol. In our `Join-EncRequest` protocol, the encrypted *JoinRequest* message sent by a *DevEUI* acts as an implicit identifier for it. However, this trackability does not apply at session level, i.e., the next time the device sends an encrypted *JoinRequest*, it will be different to its previous one and thus the two are unlinkable to the same device.

B. Implicit Trackability Attacks

1) ***`LoRaWAN-Attack3`*** ***: An*** `St-SesTrack`$^{impl.,passive}$ ***on LoRaWANJoin v1.1 & Its*** ***`Join-EncRequest`*** ***Enhancement:*** The LoRaWAN Join protocol allows a device to send a *ReJoin-request* message to which the network replies with a *JoinAccept* message that can change the $DevAddr$ of a device [12]. In this attack, we assume the following:

(i) the countermeasure of encrypting the *DevEUI* in the `Join-EncRequest` enhancement of the LoRaWAN Join is in place;
(ii) n devices are transmitting application messages at regular intervals, i.e., within t time ticks, all n devices have sent at least 1 application message;
(iii) one and only one device sends a *Join* or *(Re)JoinRequest* during a given interval.

If one of the n devices, e.g., $DevEUI_i$, sends a successful *ReJoinRequest*, then this results in the reset of i's frame counter to 0 and its address, $DevAddr_i$, be changed to $DevAddr'_i$. Not only is $DevAddr'_i$ linked to a 0-value $FCntUp$, but $DevAddr_i$ also no longer sends another application message when all other n-1 devices do. So, the adversary can link the old $DevAddr_i$ address to the new $DevAddr'_i$ address as belonging to the same device.

2) ***Countermeasures to the*** ***`LoRaWAN-Attack3`*** *:*

Our `Join-Rand_Ad` Protocol. Unless we constantly change the $DevAddr$, which is impractical, one cannot stop this attack. We use `Join-Rand_Ad` to refer to this impractical improvement of the original LoRaWAN Join v1.1 protocol, whereby the $DevAddr$ changes with every uplink message via some function known just to the devices and the NS.

[8] If public-key cryptography is not accepted by the LoRa Alliance, then the SigFox [42] approach can be adopted, in which the *JoinRequest* is encrypted with a symmetric key the device shares with the NS. To decrypt such a request, the NS will cycle through all its known device-specific keys; improvements based on expected times for ReJoins and key rotations can be proposed.

Our `Join-Rand_Ad__NoCounters_InHdrs` Protocol. If we wish to counteract the timing-based failing in `LoRaWAN-Attack1` as well, then we can enhance the `Join-Rand_Ad` protocol further to include random timing delays in the *JoinAccept* replies, as well as hide the *FCntUp* from the header. We call this resulting protocol `Join-Rand_Ad__NoCounters_InHdrs`.

The same "improvements" on $DevAddr$s' generation and unpredictable timing between the *JoinRequest* and the *JoinAccept* can be applied to our `Join-EncRequest` protocol, too.

C. Adoption by the LoRa Alliance & the Relevance of Our LoRaWAN Attacks

We demonstrated the usefulness of `TrackDev` by applying it to the LoRaWAN Join v1.1 protocol. Importantly, we actually found these attacks via a practical mechanisation of `TrackDev` (see Section VI) and via formalm verification mechanisation `TrackDev` too (see Section VII). We present them in this section, explicitly first, for ease of understanding. So, we show that `TrackDev` can find, in practice, a wide range of real attacks: some where the attacker is passive, some where they are active, some session-sensitive and some session-insensitive.

Importantly, the LoRa Alliance was receptive to all our attacks. As a consequence, in version 1.2 of the LoRaWAN specifications (to appear later this year), our countermeasures will be adopted to a great extent: (i) *JoinAccept* will arrive at varying time intervals to thwart passive attackers; (ii) an adaption of our `Join-RandAd-NoHDRCounters` is adopted: i.e., at random times, the NS will send the device (in the encrypted application layer) a new, fresh $DevAddr$ to use starting from randomly-picked $FCntUp$ counter amongst the future $FCntUp$ values of the device.

One of the authors of this manuscript is actively working with the security working group of the LoRa Alliance to incorporate these changes to the LoRaWAN specifications.

On a more generic note, `LoRaWAN-Attack1` and `LoRaWAN-Attack2` are more serious and practical than `LoRaWAN-Attack3`. Thus, one could also use this concrete example to extrapolate a wider understanding of `TrackDev`: implicit session-insensitive attacks (like `LoRaWAN-Attack3`) may be too weak, or, equivalently, that implicit, session-insensitive non-trackability may be too strong a property.

V. Trackability Characterisation

We now look at the strengths of our non-trackability properties, over their orthogonal dimensions.

A. Immediate Comparisons of Trackabilities' Strengths

Active vs Passive Attacks: If a protocol has trackability of a specific type (universal/existential, static/adaptive, session (in)sensitive, explicit/implicit) w.r.t. to an passive adversary, then it also has trackability (of the same type) w.r.t. to an active adversary, but not vice-versa:

$$\texttt{Track}^{passive} \overset{\Longrightarrow}{\not\Longleftarrow} \texttt{Track}^{active}$$

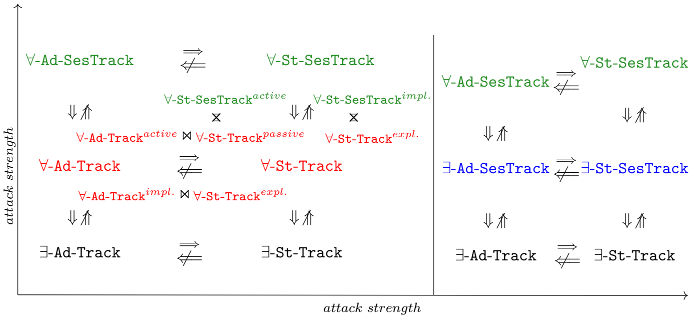


Fig. 2. (In)Comparable (Non-)Trackability over `TrackDev`'s Parameters

Static vs. Adaptive Attacks: In the adaptive cases, any adversary is given access to specific executions to track, as well as being given the target id to track. In the static cases, the adversary's strategies are not conditioned in any way. Apart from this, the attack setups are the same in both cases. So, a successful *adaptive trackability attack* entails a successful *static trackability attack*, but not vice versa:

$$\texttt{Ad-Track} \overset{\Longrightarrow}{\not\Longleftarrow} \texttt{St-Track}$$

Existential vs. Universal Attacks: Any universal attack implies an existential attack but not vice-versa:

$$\forall\texttt{-Ad-Track} \overset{\Longrightarrow}{\not\Longleftarrow} \exists\texttt{-Ad-Track}$$
$$\forall\texttt{-St-Track} \overset{\Longrightarrow}{\not\Longleftarrow} \exists\texttt{-St-Track}$$

Session-sensitive vs. Session-insensitive Attacks*:* If a protocol has session-insensitive non-trackability w.r.t. to an adversary of type $\mathcal{T}$, then it also has session non-trackability w.r.t. to an adversary of type $\mathcal{T}$, but not vice-versa:

$$\texttt{SesTrack}^{\mathcal{T}} \overset{\Longrightarrow}{\not\Longleftarrow} \texttt{Track}^{\mathcal{T}}$$

Implicit vs Explicit Attacks: If a protocol has implicit non-trackability w.r.t. to an adversary of type $\mathcal{T}$, then it also has explicit non-trackability w.r.t. to an adversary of type $\mathcal{T}$, but not vice-versa:

$$\texttt{Track}^{expl.,\mathcal{T}} \overset{\Longrightarrow}{\not\Longleftarrow} \texttt{Track}^{impl.,\mathcal{T}}$$

B. Non-Trivial Comparisons of Trackability Strength

The above compared the strengths of (non-)trackability by varying just one parameter. But, `TrackDev` has multiple parameters (see Section III-E) and varying multiple parameters at once can lead to incomparable (non-)trackability notions. We depict this via the lemmas that follow, as well as in Fig. 2.

We say that two properties are *totally incomparable* (denoted by $\bowtie$) if there exist protocols that satisfy one but not the other. In Fig. 2, "blue" and "red" attacks are incomparable (denoted by "$\bowtie$"), and both are weaker than "green attacks" (denoted by "$\Longrightarrow$"). Indeed, $\forall\texttt{-Ad-Track}^{\mathcal{T}}$ and $\forall\texttt{-St-SesTrack}^{\mathcal{T}}$ are incomparable; i.e., intuitively, if an attacker can, in a session-independent way, track any externally-chosen identifier than this does not necessarily lead to an attacker that can track, session-wise, such/the identifiers even if the latter attacker elects themselves the executions to track.

Now, we state the formal results shown in Fig. 2.

Lemma 1 (Adaptiveness of Tracking vs Attack-Type)**.** *Active adaptive trackability does not imply passive static trackability, i.e.,* $\exists\texttt{-Ad-Track}^{active} \not\Longrightarrow \exists\texttt{-St-Track}^{passive}$.

Lemma 2 (Adaptiveness vs Explicitness of Tracking)**.** *Explicit adaptive trackability and implicit static trackability are totally separable: i.e.,* $\forall\texttt{-Ad-Track}^{expl.} \bowtie \forall\texttt{-St-Track}^{impl.}$.

Lemma 3 (Session-Sensitivity vs Attack-Type)**.** *Session active trackability and session passive trackability are totally separable: i.e.,* $\forall\texttt{-St-SesTrack}^{active} \bowtie \forall\texttt{-St-Track}^{passive}$.

Lemma 4 (Session-Sensitivity vs Explicitness)**.** *Session-sensitive implicit trackability and session-insensitive explicit trackability are totally separable: i.e.,* $\forall\texttt{-St-SesTrack}^{impl.} \bowtie \forall\texttt{-St-Track}^{expl.}$.

The lemmas above are proven similarly to Lemma 5 below.

Amongst our non-trackability notions, there are more incomparable ones that Fig. 2 shows. An example is Lemma 5.

Lemma 5 (Explicitness of Tracking vs Attack-Type)**.** *Let "Track" denote any arbitrarily fixed trackability attack. Then,* $Track^{expl.,active}$ *and* $Track^{impl.,passive}$ *are totally incomparable:* $Track^{expl.,active} \bowtie Track^{impl.,passive}$.

Proof: First, we prove that: $Track^{impl.,passive} \not\Longrightarrow Track^{expl.,active}$.

We need to show that there are protocols that admit a $Track^{impl.,passive}$ attack, but not a $Track^{expl.,active}$ attack.

The `Join-EncRequest` protocol given by us in Section IV-A2 allows for the $Track^{impl.,passive}$ `LoRaWAN-Attack3` in Section IV-B, but not for a $Track^{expl.,active}$ attack. Indeed, the `Join-EncRequest` protocol cannot allow for an explicit attack, since the IDs (i.e., $DevEUI$s) are communicated encrypted.

Second, we prove that: $Track^{expl.,active} \not\Longrightarrow Track^{impl.,passive}$. We need to show that there are protocols that do not admit a $Track^{impl.,passive}$ attack, but do admit a $Track^{expl.,active}$ attack.

Consider the `Join-RandAd-NoHDRCounters` protocol in Section IV-B, which modifies the LoRaWAN Join protocol with a randomised $DevAddr$ and hides *FCntUp* from the header. This protocol allows for a $Track^{expl.,active}$ attack. This attack is variation of $\mathtt{LoRaWAN-Attack1}$: (i) the attacker sees the $DevEUI$s in clear, so the attack is explicit; (ii) the attacker needs to be active and block a *JoinRequest* of one of the $DevEUI$ of the two at their disposal; (iii) this attacker can link the $DevAddr$ seen in the first, subsequent uplink message to the unblocked $DevEUI$.

However, the `Join-RandAd-NoHDRCounters` does not allow for a $Track^{impl.,passive}$. A passive attacker can observe the $DevAddr$s in uplink messages, but they will arrive at randomised intervals from the previous *JoinRequest*s and their associated *FCntUp* will not be visible (it shows 0 initially). The $DevAddr$ will also change frequently outside of the (Re)Join calls, so several uplink messages will not have the same $DevAddr$. A passive attacker has no long-term information in the preamble of uplink messages to link to the $DevEUI$s in the prior *JoinAccept*s. ■

VI. `TrackDev` Applied in Practice

A. *Applying* `TrackDev` *to LoRaWAN*

We now show how we implemented most of `TrackDev` in practice and applied it to LoRaWAN protocols.

At the time of the experiments, no off-the-shelf LoRaWAN v1.1 devices were available. So, we experimented on the LoRaWAN Join v1.0. Note that the LoRaWAN Join v1.0 is not essentially different from the LoRaWAN Join v1.1 discussed in Section II-A. Moreover, the trackability attacks we will show below depend only on features that are the same in both the LoRaWAN Join v1.0 and v1.1.

1) FLoRa: A Packet-Inspector for LoRaWAN: We developed on top of a proprietary product[9] called FLoRa, which is primarily a LoRaWAN traffic sniffer. FLoRa is composed of two RAK831 LoRaWAN gateways[10] which, respectively, intercept up- and downlink LoRaWAN traffic between any LoRa-capable devices (e.g., PyCom devices [41]) within a given LoRa-capable network (e.g., The Things Network (TTN) [34]). The application logic behind the gateways is implemented via the "FLoRa server", which is written in Python. The traffic is stored in a database and can be queried via the command line or a user interface. FLoRa has other capabilities, e.g., replaying packets, but they are not relevant to our attack scenario.

2) ***Collecting LoRaWAN Traffic****:* We used two LoRaWAN v1.0-capable[11] devices, with the $DevEUI$s below:

$DevEUI_1$: `75 C6 00 00 0A CA 25 00`
$DevEUI_2$: `64 7F DA 00 00 00 3F ED`

Time	Type	DevAddr	DevEUI	FCnt
'2021-09-05 11:34:23.87'	'Join-Request'		'b6bf6b860a8b1ea7'	
'2021-09-05 11:34:56.31'	'Join-Request'		**'75c600000aca2500'**	
'2021-09-05 11:35:01.39'	'Join-Accept'			
'2021-09-05 11:35:04.25'	'Uplink Msg (C)'	'260b0523'		'0000'
'2021-09-05 11:35:04.26'	'Uplink Msg (C)'	'260b0523'		'0000'
'2021-09-05 11:35:17.95'	'Join-Request'		'ed3f000000da7f64'	
'2021-09-05 11:35:40.26'	'Uplink Msg (UC)'	'260be271'		'0000'
'2021-09-05 11:35:40.27'	'Uplink Msg (UC)'	'260be271'		'0000'
'2021-09-05 11:36:04.51'	'Uplink Msg (C)'	'260b0523'		'0001'
'2021-09-05 11:37:04.51'	'Uplink Msg (C)'	'260b0523'		'0002'
'2021-09-05 11:37:53.19'	'Uplink Msg (C)'	'e96adc6f'		'4c47'
'2021-09-05 11:38:04.50'	'Uplink Msg (C)'	'260b0523'		'0003'
'2021-09-05 11:38:04.50'	'Uplink Msg (C)'	'260b0523'		'0003'
'2021-09-05 11:39:00.35'	'Join-Request'		'790ef4364d04e6ab'	
'2021-09-05 11:39:02.78'	'Join-Request'		'ed3f000000da7f64'	
...				
'2021-09-05 11:41:31.88'	'Join-Request'		**'75c600000aca2500'**	
'2021-09-05 11:41:36.94'	'Join-Accept'			
'2021-09-05 11:41:39.22'	'Uplink Msg (C)'	'260b24d9'		'0000'
'2021-09-05 11:41:39.23'	'Uplink Msg (C)'	'260b24d9'		'0000'
'2021-09-05 11:42:28.75'	'Join-Request'		'ed3f000000da7f64'	
'2021-09-05 11:42:35.56'	'Join-Accept'			
'2021-09-05 11:42:39.47'	'Uplink Msg (C)'	'260b24d9'		'0001'
'2021-09-05 11:42:39.48'	'Uplink Msg (C)'	'260b24d9'		'0001'
'2021-09-05 11:42:51.07'	'Uplink Msg (UC)'	'260b7bbd'		'0000'
'2021-09-05 11:42:51.09'	'Uplink Msg (UC)'	'260b7bbd'		'0000'
'2021-09-05 11:43:24.07'	'Join-Request'		**'75c600000aca2500'**	
'2021-09-05 11:43:29.15'	'Join-Accept'			
'2021-09-05 11:43:31.39'	'Uplink Msg (C)'	'260b785b'		'0000'

Fig. 3. Snippet of captured FLoRa traffic.

These were configured to join the TTN and send/receive packets. We then randomly disconnected them from the network

[9] After the blind-review process, we will release a version as open source.

[10] https://tinyurl.com/v99v2n7k

[11] Our attacks have been implemented against LoRaWAN v1.0 devices/-traffic, due to no availability of LoRaWAN v1.1 devices at the time of the experiments; but, the trackability attacks shown above reside on features that are the same in both v1.1 and v1.0 LoRaWAN Joins.

to force new sessions, and, hence, $JoinRequest$s. We added additional devices to generate more traffic. FLoRa captured several hours of LoRaWAN traffic[12]. See Fig. 3 for a sample.

*3) **Mounting the `LoRaWAN-Attack1` in Practice**:* The aim was to see if a passive attacker (emulated by FLoRa) can mount a $\texttt{St-SesTrack}^{expl.,passive}$ attack targeting, e.g., $DevEUI_1$. Fig. 3 shows a $JoinRequest$ captured at 11:34:56 and, within 6 seconds, we see a $JoinAccept$ at 11:35:01. At 11:35:04 we can see an $Uplink\ Msg\ (C)$ with a $DevAddr$ of $260b0523$ and an $FCnt$ of 0; recall that the counter $FCnt$ at 0 denotes the very first uplink/application-level message ($\mathcal{M}_{app}$ in `TrackDev` terminology). As per our setup, our devices were randomly disconnected and reconnected to force the sending of $JoinRequest$ messages. After the 11:35:04 message in Fig. 3, we see more uplink and downlink traffic (in $\mathcal{M}_{app}$ as well as in $\mathcal{M}_{id}$). The next $JoinRequest$ message that we see for $DevEUI_1$ is at 11:41:31. Again, within 6 seconds, we see a $JoinAccept$ at 11:41:36 and an $Uplink\ Msg\ (C)$ at 11:41:39 with a $DevAddr$ of $260b24d9$ and an $FCnt$ of 0.

Note: As described in `LoRaWAN-Attack1`, the attacker can gather $DevEUI_1$'s application messages with $FCnt$ being $1, 2, \ldots$, at least until $DevEUI_1$ sends a new $JoinRequest$.

In total, FLoRa captured 225 uplink ($JoinRequest$ and application-level) messages. For all the traffic gathered by FLoRa, out of 39 $JoinRequest$ messages 31 were followed by a $JoinAccept$ within the 6 second timeframe (avg. 6 seconds). Of the remaining 8, 6 were found to be corrupted packets and 2 were found to be a replay of a $JoinRequest$ within 6 seconds of the original message. Within our small sample, there is thus a 93% ($\frac{31}{39-6}$=$\frac{31}{33}$) probability that the `LoRaWAN-Attack1` is able to link a $DevAddr$ appearing in the very first uplink/application-level message after a $JoinAccept$ to the $DevEUI$ contained within the $JoinRequest$ sent just before. If more traffic were to be gathered, then the confidence of the `LoRaWAN-Attack1` would increase, as the overall percentage of corrupted packets is likely to drop.

*4) **Verifying Our Findings**:* FLoRa has a decryption capability which, given the right $NwkKey$, will decrypt all captured $JoinAccept$s. As the devices $DevEUI_1$ and $DevEUI_2$ belong to us, we have their $NwkKey$. To verify that the `LoRaWAN-Attack1` above is correct, we applied the $NwkKey$ for $DevEUI_1$ to our captured traffic and found that, indeed, the $JoinRequest$, $JoinAccept$, $Uplink\ Msg\ (C)$ tuples which came within the 6 second timeframe of each other, allowed for the correct linking between a $DevEUI_i$ and its $DevAddr_i$s. So, the $\texttt{St-SesTrack}^{expl.,passive}$ attack we found on the LoRaWan Join v1.0 is correct.

B. Applying `TrackDev` to LoRaWAN

We now show how `TrackDev` can be applied to 5G (5th generation mobile network) procedures. The 5G procedures we use to demonstrate this are the *5G handover procedures* which we describe first.

1) 5G Handovers – Background: A *handover procedure* is executed whenever a User Equipment (UE) needs to be served by a different radio "base-station" than the one currently serving the UE. Such base-stations in 5G are called *gNBs (next generation nodes)*. A UE may switch to a new gNB for the purposes of load balancing, a change in radio conditions, or a change in the user's location [23].

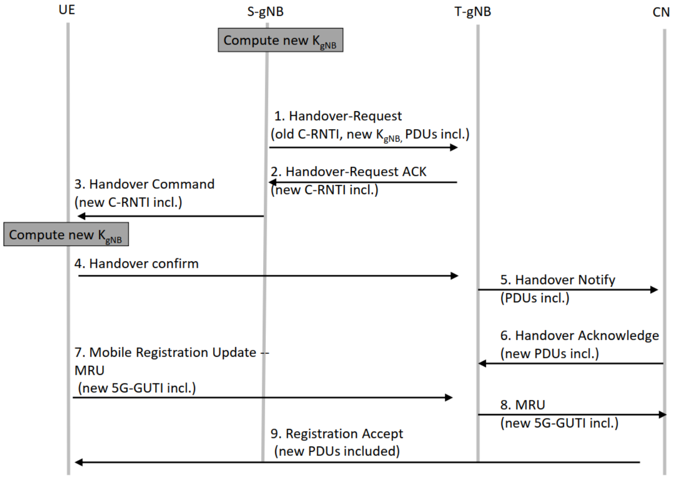


Fig. 4. XN (5G Handover Procedure) – A Sketch

There are two main types of 5G handover procedures, XN and N2 [23]. For `TrackDev`, either procedure would be suitable. Thus, we just focus on the XN procedure. Figure 4 depicts the 5G handover procedure called the XN handover.

In the description that follows, as well as in Figure 4, we simplify the settings of the handovers' execution (i.e., we present it without roaming, whereby the serving network is that of the subscribers' provider), we simplify the network architecture (e.g., we do not distinguish the different parts of the back-end/core of the network as well as leaving out certain entities altogether), and we do not focus on the key-establishment aspects of the protocol, but only on the UE identifying data passed around (be it short-term/ephemeral or long-term identifying data).

The following is a description of the XN handover as depicted in Figure 4.

1) The UE's current gNB, the source gNB (S-gNB), issues a request to transfer the UE to a target gNB (T-gNB). S-gNB provides T-gNB with the current ephemeral *Cell Radio Network Temporary Identifier (C-RNTI)* [25] of the UE and a freshly generated session key K_{gNB}, which will be used by the UE and T-gNB to derive access-stratum keys to encrypt and authenticate the communication link between themselves. Included in this *Handover-Request* message is the *Protocol Data Units (PDU) session-list* for the UE.
   - A *PDU session* is a networking and billing-related data-structure used by the 5G network to book-keep the connections and the associated usage that UEs have. A PDU session will contain several long-term details, such as what a given user is entitled to by its contract (e.g., minimal Quality of Service (QoS) parameters), as well as real-life details such consumption data, actual speed of the connection, network information (i.e., via which User Plane Function (UPF) the connection is

[12] The capture traffic is available at [35] in the files: `flora_lorawan_traffic.db` and `flora_lorawan_traffic.csv`.

routed), all as present on the current connection. An active UE has what is called a *PDU session-list*, which is a series of PDU sessions, each with a unique identifier.

2) The T-gNB responds with an ack which includes fresh ephemeral identifiers to be used to name the UE being handed-over, i.e., new C-RNTIs.
3) The UE receives the new parameters from the S-gNB and computes the same session key, K_{gNB}, issued to the T-gNB in the previous step [25].
4) The UE confirms the handover to the T-gNB.
5) The T-gNB notifies the *Core Network (CN)* that a handover is being carried out for a given UE, and sends the current list of PDU sessions for said UE.
6) The CN acknowledges the handover to the T-gNB, and sends the updated PDU session list to the T-gNB (as well as other parts of the network which we ignore here).
7) Mobile Registration Update (MRU): This is an additional optional step that may be taken during the handover. The UE, T-gNB and the CN exchange and update additional parameters such as the *5G Globally Unique Identifier (5G-GUTI)*, which is another ephemeral identifier for the UE [23].
8) Finally, in the last message in Figure 4, the UE is also updated with all the details from the CN's side, e.g., the new PDU session list.

Note that all these messages in the XN procedure are sent securely, i.e., they are encrypted and authenticated.

2) 5G & the `TrackDev` *Framework:* Let us consider `TrackDev` being applied to the 5G handover procedures. We consider a passive attacker in the case of the attacker not knowing the UE's real long-term 5G identifiers (e.g., their IMSIs [22]). So, let us focus on an $\texttt{St-Track}^{impl.,passive}$ attack on 5G XN handovers.

This is very pertinent, as the handover executions only expose (even to someone who can decrypt the traffic) ephemeral identifiers of the UE. It is the very aim of 5G that the use of these ephemeral identifiers helps improve privacy and non-traceability [24], [11], [9].

At a glance, based on Figure 4, an $\texttt{St-Track}^{impl.,passive}$ attack linking several executions of the handover to the same UE would consider either (a) "bagging" several ephemeral C-RNTIs together as belonging to the same UE, or (b) "bagging" several ephemeral 5G Global Unique Temporary Identifiers (GUTIs) together as belonging to the same UE.

Worryingly, in practice, if an attacker is able to link together several ephemeral identifiers of the same one UE via handover executions, it also implies that this attacker can determine the physical whereabouts of that UE, since the handovers' executions will pertain also to the gNBs executing them.

a) **Threat Sub-Model:** To mount such an attack, traffic needs to be recorded in different places: (a) between the CN and (several) gNBs; (b) between the UE itself and (several) gNBs; (c) between the gNBs. To mount the attack, such traffic also has to be decrypted or collected in decrypted form. This corresponds to a strong threat-model, whereby some (subnet of) gNBs[13] are corrupted and leak decrypted traffic, or are the traffic collectors themselves. That said, since gNBs run software that is provided not by the CN, but by third-party vendors, this threat is realistic. Indeed, it is well-known that rogue gNBs, installed with the very purpose of tracking exist and are active [28].

Thus, we consider two scenarios for our attacker, but of which lead to trackability attacks: (a) a corrupt gNB gathers/leaks traffic in between the UE and itself, or in between itself and another gNB – Section VI-B4; (b) a corrupt gNB gathers/leaks traffic between itself and the CN – Section VI-B5.

3) The Setup for a $\exists\texttt{-St-Track}^{impl.,passive}$ *Attack on 5G XN Handovers:* We explain the setup used to mount the aforementioned $\exists\texttt{-St-Track}^{impl.,passive}$ attack on 5G XN handovers in practice.

Experimental Setup. For our practical attack, we used an experimental but operational 5G network where we have access to the CN. The CN is minimal in the sense that unlike in commercial mobile networks, there are not numerous entities of the same type, e.g., *Access and Mobility Management Functions (AMFs)*. This makes some aspects of the protocol simpler, e.g., the PDU sessions will not contain various AMFs, billing information, etc. The gNBs in this network run Huawei software. The XN protocol is as per the specifications.

Experiment Design. For ease, we focus on the tracking of one single 5G-capable mobile phone generating XN traffic in this network.

We used a Huaweii BLA-L09 in this network. This phone was connected to the network and physically moved between multiple gNBs to trigger the handover procedures for the duration of 1 hour, and other traffic was present in the network. We recorded traffic: (a) at the CN-side between the CN and the gNBs, and (b) at the gNB side, between the UEs and the various gNBs. We used both sets of traffic to mount the aforementioned $\exists\texttt{-St-Track}^{impl.,passive}$. Clearly, we know the long term identifiers as well as the movements of said phone, and we will use that to ascertain the correctness of the attack applied on the collected traffic.

4) Mounting a $\exists\texttt{-St-Track}^{impl.,passive}$ *Attack on 5G Handovers Using gNB-Facing Traffic:* We focus on mounting the attack using traffic collected on the gNB nodes. The traffic we collected is a Huawei proprietary format (called "tmf") and can be found at [35].

We show that by looking at several *Handover-Request* messages (message 1 in Figure 4) between different gNBs, we can tell which handovers belong to the same UE.

That is, even if the same UE has two different C-RNTIs over every two handover executions, we can still say that these two C-RNTIs belong to the same UE. PDU-session data does not change (or changes minimally) across two such Handover-Request messages, which are not far apart in time. It is likely that over two handovers close in time, the billing data, the Internet-traffic routing, etc., has not changed[14] much for the same UE [23].

[13] There would be the alternative of the UE being corrupted, but in this case tracking its identifiers can be done differently.

[14] In our experimental setting, the AMF does not change either.

For clarity, in Figure 5, we show several *Handover-Request* messages (as well as other XN messages) in our captured traffic, with an expanded window on the PDU-session data inside this message. *Handover-Request*s 17 and 25 marked on Figure 5 in green are 13 minutes apart and were sent by different gNBs. The side-by-side comparison can be seen in Figure 6.

The comparison shown in Figure 6 focuses on two aspects: C-RNTIs (in red) and PDU-session data. We see in this (extreme) case that the PDU-session data is identical (the same 30 bytes are illustrated[15]), even if the C-RNTIs differ. This is therefore clearly the same phone doing two handover executions, being at different locations at given points in time.

Note that not all PDU-session data for the same UE was identical across our collected data over two different *Handover-Request*s, but $\approx$ 90% of the bytes match. This allows to unequivocally link C-RNTIs and different handover executions to the same UE. We did this sort of comparison for all our traffic. It was 100% accurate in identifying our target UE.

Note that more sophisticated attack strategies are possible, where one links together *Handover-Request*s and *Handover-Request Acknowledgment* messages. This is not necessary for our illustration purposes here.

5) Mounting a $\exists$-`St-Track`$^{impl.,passive}$ *Attack on 5G Handovers Using Core-Facing Traffic:* We focus on mounting the attack using traffic collected on the CN-side. The traffic we collected is stored in a .pcap file and can be found (zipped) at [35].

The identifiers of interest for our attack are:

- **5G-S-TMSI:** The 5G S-Temporary Mobile Subscriber Identity [21] is the shortened version of the 5G GUTI, and is an ephemeral identifier of UEs, used between UEs and gNBs. The 5G-TMSI is refreshed with each handover and a fresh 5G-TMSI is used between the UE and gNB right after a handover (in the MRU) and (re)registration messages. We will refer to all these messages, by some abuse of naming, as the *InitialUEMessage*.
- **AMF-UE-NGAP-ID:** A unique ID between the UE and the NG interface within the AMF. This ID is refreshed with every new connection, as a UE can only be connected to one AMF at a time [27], [26].
- **RAN-UE-NGAP-ID:** A unique ID between the UE and the NG interface within a gNB node (RAN). This ID is refreshed with every new connection, as a UE can only be connected to one gNB at a time [27], [26].

To mount this attack, the adversary has to observe several *InitialUEMessage* and *InitialContextSetupRequest* messages (in this setting by capturing them between the gNB and the CN). These messages are sent in a new connection between the UE and a gNB [27]. The attack works because across several *InitialUEMessage* and *InitialContextSetupRequest* messages of the same UE, there will be at least one identifier discussed above that does not change.

In Figure 7, we show a snippet of our captured traffic between the 5G CN and the gNB towers. As per Figure 7, the attacker is able to deduce that the 5G-S-TMSI values of 3623909776 and 3623903776 belong to a single device through the following steps:

1. (5G-S-TMSI,RAN-UE-NGAP-ID) pairing of $(3623909776, 1048819)$ in packet 103987
2. (AMF-UE-NGAP-ID, RAN-UE-NGAP-ID) pairing of $(65, 1048819)$ in packet 103995
3. (AMF-UE-NGAP-ID, RAN-UE-NGAP-ID) pairing of $(65, 14)$ in packet 104159
4. (5G-S-TMSI,RAN-UE-NGAP-ID) pairing of $(3623903776, 14)$ in packet 104166

This attack is largely possible because the same UE is handed over in our experiments between several gNB nodes, but the AMF stays the same. Note this is a setup in our network, but this can happen in real 5G networks, too, as the AMF does not always change. In our setup this attack is very effective, working for 82% of the handovers (whilst some captured traffic is malformed).

However, due to the fact that the AMF would change, this attack is less effective in the reality than the attack in Subsection VI-B4. The attacker can then look at the *PathSwitchRequest* message which is sent in each such execution of handover/registration by the backend of the network (e.g., gateways/user-plane functions (UPFs)) to the gNBs [27]. A *PathSwitchRequest* message contains the details of the PDU session to be switched, such as the ID of the associated device and the new gateway tunnel endpoint. The attacker can then reinforce the first correlation of several 5G-S-TMSIs suspected to be of the same UE (based on the AMF-UE-NGAP-ID, RAN-UE-NGAP-ID) via looking at the PDU session data in the associated *PathSwitchRequest* messages; this is similar to Subsection VI-B4.

6) Final Words on `TrackDev`*'s Application to 5G Handovers:* The investigation we made started from the known premise that PDU-session data carries long-term data and that the AMF-UE-NGAP-IDs are known to be relatively static. On this premise, we wished to use `TrackDev` in a systematic way, to see if these are enough to track UEs based on C-RNTIs and 5G-S-TMSI, and do so in line with our well-defined notions. To this end, we do show that `TrackDev` can indeed be used to systematically capture trackability attacks over 5G handovers.

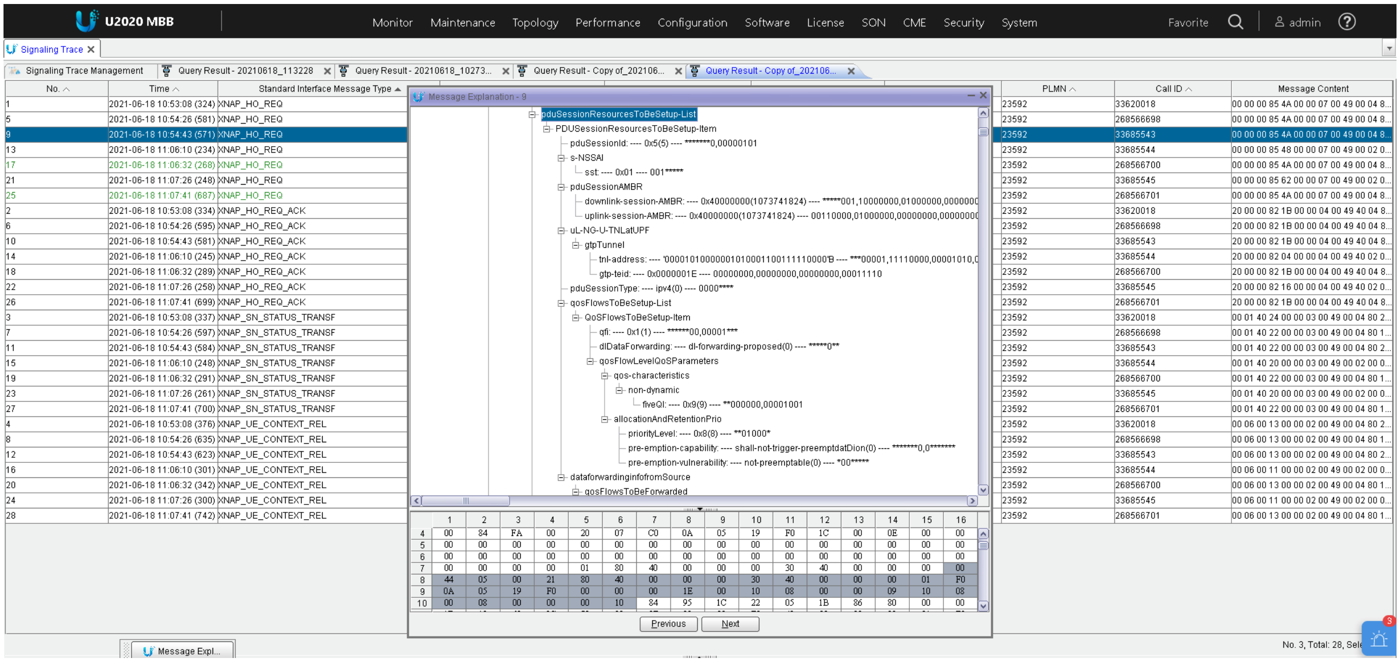
Meaning of the attacks. In practical terms, our findings mean that "honest but curious" gNBs can look at all the traffic they see and link/track together multiple C-RNTIs (which are ephemeral, privacy-driven identifiers) and data associated to these (e.g., PDUs) to determine which C-RNTIs pertain to the same UE, and thus track their traffic/movements long-term.

Since our threat model is arguably strong[16] (i.e., gNBs have to leak decrypted traffic), our findings do not say that C-RNTIs and 5G-S-TMSIs are necessarily/always not useful in masking long-term identifiers such as the IMSIs.

Disclosure. We discussed with 3GPP, and they find the premise of corrupt gNBs of a low risk. We will pursue the

[15] This is also true on field-by-field inspection.

[16] Yet, our threat-model is totally plausible if someone has access to, e.g., (one or more) gNBs that did handovers of UEs, as per shown to be the practice in some cases [28].



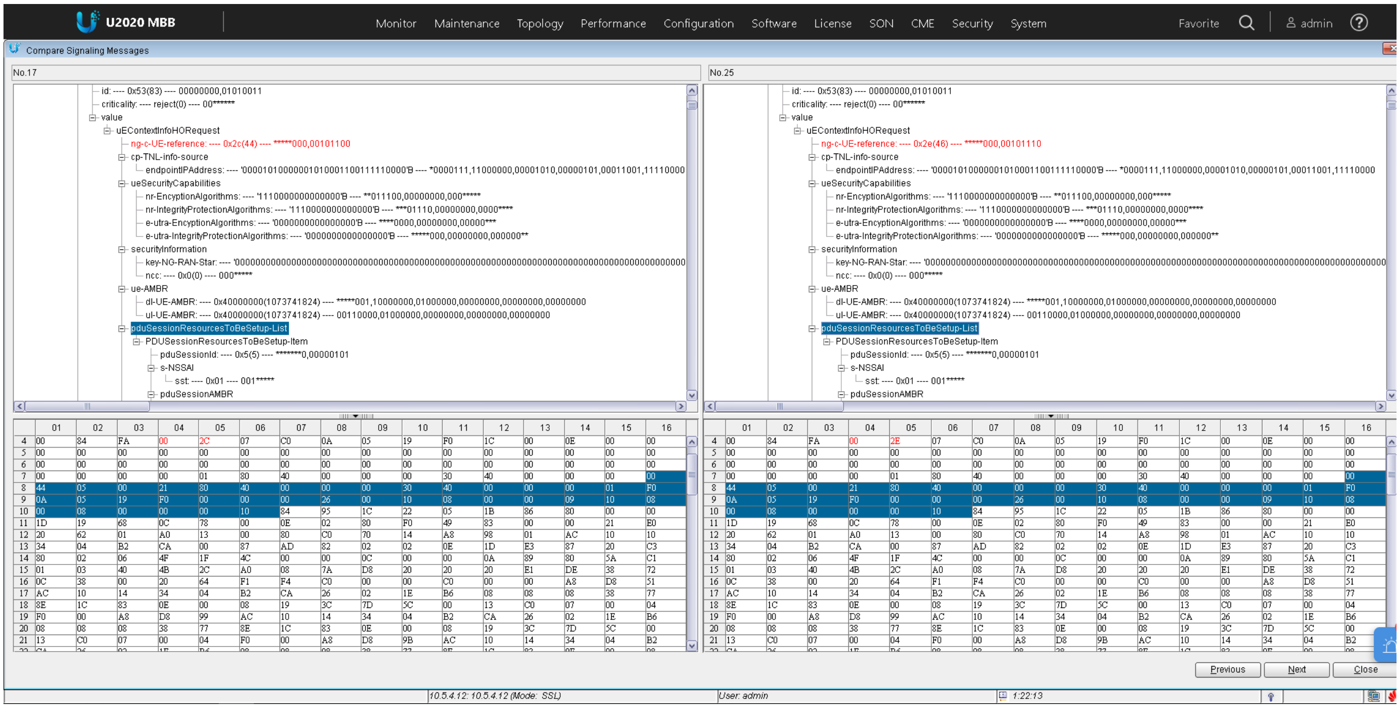
No.	Time	Standard Interface Message Type	PLMN	Call ID	Message Content
1	2021-06-18 10:53:08 (324)	XNAP_HO_REQ	23592	33620018	00 00 00 85 4A 00 00 07 00 49 00 04 8...
5	2021-06-18 10:54:26 (581)	XNAP_HO_REQ	23592	268566698	00 00 00 85 4A 00 00 07 00 49 00 04 8...
9	2021-06-18 10:54:43 (571)	XNAP_HO_REQ	23592	33685543	00 00 00 85 4A 00 00 07 00 49 00 04 8...
13	2021-06-18 11:06:10 (234)	XNAP_HO_REQ	23592	33685544	00 00 00 85 48 00 00 07 00 49 00 02 0...
17	2021-06-18 11:06:32 (268)	XNAP_HO_REQ	23592	268566700	00 00 00 85 4A 00 00 07 00 49 00 04 8...
21	2021-06-18 11:07:26 (248)	XNAP_HO_REQ	23592	33685545	00 00 00 85 62 00 00 07 00 49 00 02 0...
25	2021-06-18 11:07:41 (687)	XNAP_HO_REQ	23592	268566701	00 00 00 85 4A 00 00 07 00 49 00 04 8...
2	2021-06-18 10:53:08 (334)	XNAP_HO_REQ_ACK	23592	33620018	20 00 00 82 1B 00 00 04 00 49 40 04 8...
6	2021-06-18 10:54:26 (595)	XNAP_HO_REQ_ACK	23592	268566698	20 00 00 82 1B 00 00 04 00 49 40 04 8...
10	2021-06-18 10:54:43 (581)	XNAP_HO_REQ_ACK	23592	33685543	20 00 00 82 1B 00 00 04 00 49 40 04 8...
14	2021-06-18 11:06:10 (245)	XNAP_HO_REQ_ACK	23592	33685544	20 00 00 82 04 00 00 04 00 49 40 02 0...
18	2021-06-18 11:06:32 (289)	XNAP_HO_REQ_ACK	23592	268566700	20 00 00 82 1B 00 00 04 00 49 40 04 8...
22	2021-06-18 11:07:26 (258)	XNAP_HO_REQ_ACK	23592	33685545	20 00 00 82 16 00 00 04 00 49 40 02 0...
26	2021-06-18 11:07:41 (699)	XNAP_HO_REQ_ACK	23592	268566701	20 00 00 82 1B 00 00 04 00 49 40 04 8...
3	2021-06-18 10:53:08 (337)	XNAP_SN_STATUS_TRANSF	23592	33620018	00 01 40 24 00 00 03 00 49 00 04 80 2...
7	2021-06-18 10:54:26 (597)	XNAP_SN_STATUS_TRANSF	23592	268566698	00 01 40 22 00 00 03 00 49 00 04 80 1...
11	2021-06-18 10:54:43 (584)	XNAP_SN_STATUS_TRANSF	23592	33685543	00 01 40 22 00 00 03 00 49 00 04 80 2...
15	2021-06-18 11:06:10 (248)	XNAP_SN_STATUS_TRANSF	23592	33685544	00 01 40 20 00 00 03 00 49 00 02 00 0...
19	2021-06-18 11:06:32 (291)	XNAP_SN_STATUS_TRANSF	23592	268566700	00 01 40 22 00 00 03 00 49 00 04 80 1...
23	2021-06-18 11:07:26 (261)	XNAP_SN_STATUS_TRANSF	23592	33685545	00 01 40 20 00 00 03 00 49 00 02 00 0...
27	2021-06-18 11:07:41 (700)	XNAP_SN_STATUS_TRANSF	23592	268566701	00 01 40 22 00 00 03 00 49 00 04 80 1...
4	2021-06-18 10:53:08 (376)	XNAP_UE_CONTEXT_REL	23592	33620018	00 06 00 13 00 00 02 00 49 00 04 80 2...
8	2021-06-18 10:54:26 (635)	XNAP_UE_CONTEXT_REL	23592	268566698	00 06 00 13 00 00 02 00 49 00 04 80 1...
12	2021-06-18 10:54:43 (623)	XNAP_UE_CONTEXT_REL	23592	33685543	00 06 00 13 00 00 02 00 49 00 04 80 2...
16	2021-06-18 11:06:10 (301)	XNAP_UE_CONTEXT_REL	23592	33685544	00 06 00 11 00 00 02 00 49 00 02 00 0...
20	2021-06-18 11:06:32 (342)	XNAP_UE_CONTEXT_REL	23592	268566700	00 06 00 13 00 00 02 00 49 00 04 80 1...
24	2021-06-18 11:07:26 (300)	XNAP_UE_CONTEXT_REL	23592	33685545	00 06 00 11 00 00 02 00 49 00 02 00 0...
28	2021-06-18 11:07:41 (742)	XNAP_UE_CONTEXT_REL	23592	268566701	00 06 00 13 00 00 02 00 49 00 04 80 1...

```
pduSessionResourcesToBeSetup-List
  PDUSessionResourcesToBeSetup-Item
    pduSessionId: ---- 0x5(5) ---- *******0,00000101
    s-NSSAI
      sst: ---- 0x01 ---- 001*****
    pduSessionAMBR
      downlink-session-AMBR: ---- 0x40000000(1073741824) ---- *****001,10000000,01000000,0000000
      uplink-session-AMBR: ---- 0x40000000(1073741824) ---- 00110000,01000000,00000000,0000000
    uL-NG-U-TNLatUPF
      gtpTunnel
        tnl-address: ---- '00001010000001010001100111110000'B ---- ***00001,11110000,00001010,0
        gtp-teid: ---- 0x0000001E ---- 00000000,00000000,00000000,00011110
    pduSessionType: ---- ipv4(0) ---- 0000****
    qosFlowsToBeSetup-List
      QoSFlowsToBeSetup-Item
        qfi: ---- 0x1(1) ---- ******00,00001***
        dlDataForwarding: ---- dl-forwarding-proposed(0) ---- *****0**
        qosFlowLevelQoSParameters
          qos-characteristics
            non-dynamic
              fiveQI: ---- 0x9(9) ---- **000000,00001001
          allocationAndRetentionPrio
            priorityLevel: ---- 0x8(8) ---- **01000*
            pre-emption-capability: ---- shall-not-trigger-preemptdatDion(0) ---- *******0,0*******
            pre-emption-vulnerability: ---- not-preemptable(0) ---- *00*****
    dataforwardinginfofromSource
      qosFlowsToBeForwarded
```

```
     1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
4    00  84  FA  00  20  07  C0  0A  05  19  F0  1C  00  0E  00  00
5    00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
6    00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
7    00  00  00  00  01  80  40  00  00  00  30  40  00  00  00  00
8    44  05  00  21  80  40  00  00  00  30  40  00  00  00  01  F0
9    0A  05  19  F0  00  00  00  1E  00  10  08  00  00  09  10  08
10   00  08  00  00  00  10  84  95  1C  22  05  1B  86  80  00  00
```

No. 3, Total: 28, Sele

Fig. 5. XN Data on gNBs: Focus on Handover-Request & Inner PDU-session List



No.17

```
id: ---- 0x53(83) ---- 00000000,01010011
criticality: ---- reject(0) ---- 00******
value
  uEContextInfoHORequest
    ng-c-UE-reference: ---- 0x2c(44) ---- *****000,00101100
    cp-TNL-info-source
      endpointIPAddress: ---- '00001010000001010001100111110000'B ---- *0000111,11000000,00001010,00000101,00011001,11110000
    ueSecurityCapabilities
      nr-EncyptionAlgorithms: ---- '1110000000000000'B ---- **011100,00000000,000*****
      nr-IntegrityProtectionAlgorithms: ---- '1110000000000000'B ---- ***01110,00000000,0000****
      e-utra-EncyptionAlgorithms: ---- '0000000000000000'B ---- ****0000,00000000,00000***
      e-utra-IntegrityProtectionAlgorithms: ---- '0000000000000000'B ---- *****000,00000000,000000**
    securityInformation
      key-NG-RAN-Star: ---- '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
      ncc: ---- 0x0(0) ---- 000*****
    ue-AMBR
      dl-UE-AMBR: ---- 0x40000000(1073741824) ---- *****001,10000000,01000000,00000000,00000000,00000000
      ul-UE-AMBR: ---- 0x40000000(1073741824) ---- 00110000,01000000,00000000,00000000,00000000
    pduSessionResourcesToBeSetup-List
      PDUSessionResourcesToBeSetup-Item
        pduSessionId: ---- 0x5(5) ---- *******0,00000101
        s-NSSAI
          sst: ---- 0x01 ---- 001*****
        pduSessionAMBR
```

```
     01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16
4    00  84  FA  00  2C  07  C0  0A  05  19  F0  1C  00  0E  00  00
5    00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
6    00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
7    00  00  00  00  01  80  40  00  00  00  30  40  00  00  00  00
8    44  05  00  21  80  40  00  00  00  30  40  00  00  00  01  F0
9    0A  05  19  F0  00  00  00  26  00  10  08  00  00  09  10  08
10   00  08  00  00  00  10  84  95  1C  22  05  1B  86  80  00  00
11   1D  19  68  0C  78  00  0E  02  80  F0  49  83  00  00  21  E0
12   20  62  01  A0  13  00  80  C0  70  14  A8  98  01  AC  10  10
13   34  04  B2  CA  00  87  AD  82  02  02  0E  1D  E3  87  20  C3
14   80  02  06  4F  1F  4C  00  00  0C  00  00  0A  89  80  5A  C1
15   01  03  40  4B  2C  A0  08  7A  D8  20  20  20  E1  DE  38  72
16   0C  38  00  20  64  F1  F4  C0  00  00  C0  00  00  A8  D8  51
17   AC  10  14  34  04  B2  CA  26  02  1E  B6  08  08  08  38  77
18   8E  1C  83  0E  00  08  19  3C  7D  5C  00  13  C0  07  00  04
19   F0  00  A8  D8  99  AC  10  14  34  04  B2  CA  26  02  1E  B6
20   08  08  08  38  77  8E  1C  83  0E  00  08  19  3C  7D  5C  00
21   13  C0  07  00  04  F0  00  A8  D8  9B  AC  10  14  34  04  B2
```

No.25

```
id: ---- 0x53(83) ---- 00000000,01010011
criticality: ---- reject(0) ---- 00******
value
  uEContextInfoHORequest
    ng-c-UE-reference: ---- 0x2e(46) ---- *****000,00101110
    cp-TNL-info-source
      endpointIPAddress: ---- '00001010000001010001100111110000'B ---- *0000111,11000000,00001010,00000101,00011001,11110000
    ueSecurityCapabilities
      nr-EncyptionAlgorithms: ---- '1110000000000000'B ---- **011100,00000000,000*****
      nr-IntegrityProtectionAlgorithms: ---- '1110000000000000'B ---- ***01110,00000000,0000****
      e-utra-EncyptionAlgorithms: ---- '0000000000000000'B ---- ****0000,00000000,00000***
      e-utra-IntegrityProtectionAlgorithms: ---- '0000000000000000'B ---- *****000,00000000,000000**
    securityInformation
      key-NG-RAN-Star: ---- '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
      ncc: ---- 0x0(0) ---- 000*****
    ue-AMBR
      dl-UE-AMBR: ---- 0x40000000(1073741824) ---- *****001,10000000,01000000,00000000,00000000,00000000
      ul-UE-AMBR: ---- 0x40000000(1073741824) ---- 00110000,01000000,00000000,00000000,00000000
    pduSessionResourcesToBeSetup-List
      PDUSessionResourcesToBeSetup-Item
        pduSessionId: ---- 0x5(5) ---- *******0,00000101
        s-NSSAI
          sst: ---- 0x01 ---- 001*****
        pduSessionAMBR
```

```
     01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16
4    00  84  FA  00  2E  07  C0  0A  05  19  F0  1C  00  0E  00  00
5    00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
6    00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
7    00  00  00  00  01  80  40  00  00  00  30  40  00  00  00  00
8    44  05  00  21  80  40  00  00  00  30  40  00  00  00  01  F0
9    0A  05  19  F0  00  00  00  26  00  10  08  00  00  09  10  08
10   00  08  00  00  00  10  84  95  1C  22  05  1B  86  80  00  00
11   1D  19  68  0C  78  00  0E  02  80  F0  49  83  00  00  21  E0
12   20  62  01  A0  13  00  80  C0  70  14  A8  98  01  AC  10  10
13   34  04  B2  CA  00  87  AD  82  02  02  0E  1D  E3  87  20  C3
14   80  02  06  4F  1F  4C  00  00  0C  00  00  0A  89  80  5A  C1
15   01  03  40  4B  2C  A0  08  7A  D8  20  20  20  E1  DE  38  72
16   0C  38  00  20  64  F1  F4  C0  00  00  C0  00  00  A8  D8  51
17   AC  10  14  34  04  B2  CA  26  02  1E  B6  08  08  08  38  77
18   8E  1C  83  0E  00  08  19  3C  7D  5C  00  13  C0  07  00  04
19   F0  00  A8  D8  99  AC  10  14  34  04  B2  CA  26  02  1E  B6
20   08  08  08  38  77  8E  1C  83  0E  00  08  19  3C  7D  5C  00
21   13  C0  07  00  04  F0  00  A8  D8  9B  AC  10  14  34  04  B2
```

10.5.4.12: 10.5.4.12 (Mode: SSL) User: admin 1:22:13

Fig. 6. Two Handover-Requests: Two C-RNTIs & The Same PDU-session List Data

Packet	Source	Destination	Message	5G-S-TMSI	AMF-UE-NGAP-ID	RAN-UE-NGAP-ID
103987	Tower 1	5G-Core	InitialUEMessage	3623909776		1048819
103995	5G-Core	Tower 1	InitialcontextSetupRequest		65	1048819
			...			
104159	Tower 2	5G-Core	InitialUEMessage	3623903776		14
104166	5G-Core	Tower 2	InitialcontextSetupRequest		65	14

Fig. 7. Snippet of Captured Traffic between the gNB and the Core

conversation further based on certain on-going experiments at our end, in which we aim to make our trackability simpler and more resolute, i.e., track UEs with more certain based on more data.

VII. TrackDev for Formal Verification: TrackDev Mechanised in Tamarin

We discuss how our TrackDev notions are mechanisable in formal verification tools, how it relates to existing privacy properties in formal analysis, and how we add to that.

A. Preliminaries on Related Formal-Verification Notions

1) ***Arapinis' Unlinkability [2]*** *($unlink.^{Arapinis}$):* Trackability as encapsulated by `TrackDev` transcends privacy properties encoded as a trace-to-trace equivalence and verified in tools such as Deepsec [10] or SAT-Equiv [15], [16]. Instead, it is closest to unlinkability as presented by Arapinis et al. in [2]; unlinkability denotes an equivalence property between (any) two sets of traces, not just between two traces. We refer to this notion as $unlink.^{Arapinis}$. One flavour of $unlink.^{Arapinis}$ is as follows: a protocol-execution in which $alice$ has just one session should be indistinguishable to an attacker from the protocol-execution in which $alice$ has two or more sessions.

2) ***The Verification of*** *$unlink.^{Arapinis}$:* Currently, no tool exists that can verify $unlink.^{Arapinis}$, in general. However, the authors of [5], [33] showed that, for two-party, stateful protocols, $unlink.^{Arapinis}$ can be reduced to verifying if the protocol achieves the following three properties:

- **Well-Authentication (WA):** Whenever the outcome of a conditional is positive, the corresponding agent has an honest interaction with a counterpart of the expected identity. It thus ensures that the attacker cannot learn anything about identities by (indirectly) observing the outcome of conditionals;
- **No-Desynchronisation (ND):** *Honest* sessions are not desynchronised in stateful protocols;
- **Frame Opacity (FO):** The attacker cannot distinguish (ephemeral or long-term) identities via any relations over run-time messages[17].

The first two properties (WA, ND) are trace properties that can be verified in well-established protocol/authentication/secrecy verification tools such as Tamarin [39] and Proverif [6]. Moreover, ND is only needed for stateful protocols. The third property, FO, is not a trace property (as it refers to messages across traces) and can be mechanised as an extension of diff-equivalence [7], which is implemented in Tamarin [39] and Proverif [6]; if it is does not hold, the verification is trackable, but if it does hold, it is harder to verify with these tools.

In the next section, we model our (non-)trackability properties on the LoRaWAN Join v1.1 as well as $unlink.^{Arapinis}$ (as WA, ND and FO) in Tamarin [39]. We show that WA holds while FO does not, which means that $unlink.^{Arapinis}$ does not hold for LoRaWAN Join v1.1. We also compare the results of verifying $unlink.^{Arapinis}$ in this way vs. verifying our non-trackability (in LoRaWAN Join v1.1).

B. Mechanising Explicit Trackability

1) ***Explicit Trackability as Trace Properties:*** We now explain how static explicit session-sensitive and session-insensitive trackabilities $\forall$-`St-NoTrack`$^{expl.,\cdot}$ and $\forall$-`St-SesNoTrack`$^{expl.,\cdot}$ can be reduced soundly to reachability checking. We also explain any gaps in mechanising other types of trackability.

The idea of this mechanisation hinges on the `LoRaWAN-Attack2` attack given in Section IV-A2 on the LoRaWAN Join v1.1, which is a $\forall$-`St-SesNoTrack`$^{expl.,active}$ attack. Recall that for this attack, we presented two "strategies": (i) selective blocking and (ii) forced condition. In fact, these strategies work for active trackability attackers against any `IDed-Prot` protocol as follows:

- **Selective blocking** applies to all explicit attack setups and active attackers: i.e., any active attacker can block messages with specific IDs;
- **Forced conditions** are a general exploitation technique by an attacker in all protocols with conditionals: i.e., any active attacker can force the protocol to take the `if` as well as the `else` branch of a test to "see" the results.

Moreover, our notion of the forced condition strategy is equivalent to the failure of WA by Baelde et al. [5]. Concretely, WA encodes that whenever a protocol-condition is positively evaluated, the agents involved up to then are, in fact, honest. In LoRaWAN Join v1.1, it means that if a party was authenticated, then there was no forced condition applied.

The mechanisation is done in two steps. Firstly, one of the above trackability attack strategies has to be implemented in a model in a trace-based verification tool. On the one hand, selective blocking comes almost for free in any DY [20] tool: the attacker can and will try to block any message, and the "selective" nature is achieved by writing a model with a set of rules, restrictions and/or predicates, which quantify over a particular ID whose messages were "selectively" blocked[18]. On the other hand, forced conditions are implicitly implemented in any DY tool: the DY attacker will try arbitrary inputs forcing both the `if` and the `else` branch of all protocol tests. Explicitly forcing a condition thus reduces to quantifying over the right predicates in the model.

Secondly, $\forall$-`St-NoTrack`$^{expl.}$ or $\forall$-`St-SesNoTrack`$^{expl.}$ can be checked with an "exist-trace" lemma that quantifies over the aforementioned implementation of strategies and expresses the trackability relations in the definitions $\forall$-`St-NoTrack`$^{\cdot,expl.}$ or $\forall$-`St-SesNoTrack`$^{\cdot,expl.}$. Clearly, the quantification in our definitions allows for this to be expressed as a trace. However, the chosen strategy must be built in the model first, so the lemmas encapsulates not just simply a relation that (always) exists, but one the attacker can build.

This mechanisation may be prescriptive, but it is arguably similar to the approximation [5] of unlinkability by a series of (trace-)properties. Our illustration of the mechanisation is protocol specific, as is the approximation in [5]. Yet, in our case, the above provides a clear and generic recipe to formulate the necessary rules and lemmas to demonstrate static explicit trackability in general.

2) ***Explicit Trackability Concretely Encoded in Tamarin:*** We implement our mechanisations in Tamarin [39], a well-known DY protocol prover [39]. Yet, any trace-based verification tool, such as Proverif [6] or AVISPA [4], can be used.

[17] This is clearly a very strong requirement: to achieve this, with a universal quantification over messages, a protocol would need to have its run-time messages be idealised/random, i.e., all messages look the same to an attacker.

[18] Selective blocking can also be implemented explicitly: via a simple fact restricting the number of communicating parties; we show this in Section VII-B2a (using the LoRaWAN Join) and in Fig. 8 in the long version of our paper [35].

We now concretely illustrate the mechanisation sketched above in Section VII-B1 in Tamarin [39] for the LoRaWAN Join v1.1 protocol, as shown in Fig. 1. In our model, we collapse the NS and JS servers into one entity as we are not concerned with the messages sent between them. Indeed, our framework is about two parties A and B, where A corresponds to the device in the LoRaWAN Join and B represents the whole of the backend network (NS, JS and AS). In fact, the messages sent between the NS and JS are sent over a secure channel [14] (i.e., confidential, integral, authenticated) and are thus immaterial to Tamarin's DY attacker[19].

a) **Our Selective Blocking in Tamarin:** Within the DY model, we model an active attacker who implements the selective blocking strategy: i.e., blocks messages from all but one party. Without loss of generality, we can restrict our attention to two devices, $DevEUI_1$ and $DevEUI_2$, sending a *JoinRequest* at roughly the same time. As shown in Section IV-A2, the `LoRaWAN-Attack2` consists of blocking the *JoinRequest* of, say, $DevEUI_1$ and then linking the first subsequent application message containing $DevAddr_2$ to the $DevEUI_2$ which appears in the unblocked *JoinRequest*.

Selective Blocking Implemented via Action Facts. To implement selective blocking, we use Tamarin's *action facts* [43]. These conditions are essentially fairness constraints: predicates that restrict the set of executable traces which are analysed. To this end, we use a predicate called $OnlyOnce()$ which restricts the analysed traces to those in which only one $JoinRequest$ goes through while the model allows several $JoinRequest$s to be sent (see Fig. 8 in the long version of our paper [35]). This corresponds to the behaviour where the DY attacker, underpinned in the tool, blocks all but one device's requests.

Selective Blocking Leading to the Trackability Lemma. In order to demonstrate the ability of the attacker to link the unblocked $DevEUI$ to the $DevAddr$, we show that the model presents a trace in which one device is blocked while the other one completes its run of the LoRaWAN Join v1.1 protocol and sends its first application message and receives a response. In fact, the Tamarin lemma in Fig. 8 in the long version of our paper [35] captures the query for such a trace. This lemma holds.

b) **Our Forced Conditions In Tamarin:** As per Section IV-A2, forced conditions are about forcing a test in a protocol. In the LoRaWAN Join v1.1 protocol we consider, for the attacker to mount the `LoRaWAN-Attack2` they have to force the test on the incremental $DevNonce$ to fail on the JS side by changing its value in-transit for a device-issued $JoinRequest$.

Forced Condition as Well-Authentication [5]. As we said, our forced condition strategy is equivalent to the failure of WA by Baelde et al. [5]. We encoded a lemma (see Fig. 9 in the long version of our paper [35]) for the WA-property in our Tamarin model for the LoRaWAN Join v1.1. This lemma holds, as expected, showing that if no forced condition is applied, devices authenticate correctly. The WA-property holds as the *JoinRequest* is subject to an integrity check, which ensures that the values inside cannot be manipulated without detection, despite the message being sent in cleartext.

We also show that there exists a trace where the attacker inflicts a change in a $DevEUI$'s $DevNonce$ to $DevNonce'$, thus making $DevEUI$'s authentication fail and emulating the forced condition strategy. We can then mechanise the trackability of this $DevEUI$ by exhibiting a trace with $DevEUI$ and $DevNonce'$ in it, in the right order. To this end, see Fig. 10 in the long version of our paper [35]. This means that trackability can be shown simply by the existence of a trace that proves that a property linked to WA fails, i.e., that a forced condition exists.

We now give snippets of the Tamarin code we used to illustrate in a more didactic way the aforesaid mechanisation of the trackability verification for the LoRaWAN Join v1.1.

Fig. 8 shows the Tamarin code that implements the selective blocking strategy by applying the $OnlyOnce()$ action fact. The blocking strategy is used by an active attacker in order to mount `LoRaWAN-Attack2`, which is a $\forall$-`St-SesNoTrack`$^{expl.,active}$ attack on the LoRaWAN Join v1.1, attack illustrated first in Section IV-A2.

```
/**********
EXPLICIT TRACKABILITY LEMMA
 *****/
lemma linked_deveui_devaddr_onlyonce: exists-trace
"
        Ex DevEUI DevEUI2 NS JoinEUI DevNonce DevNonce2
    tau_c1 tau_c2 DevAddr2  #t01 #t02 #t03
        .
          DeviceJoinRequest(DevEUI, NS, JoinEUI, DevNonce,
    tau_c1)@ t01

        & DeviceJoinRequest(DevEUI2, NS, JoinEUI, DevNonce2,
     tau_c2)@ t02

        & NSReceivedReKeyInd(NS, DevEUI2, DevAddr2) @t03 //
    link

        &#t01<#t02
        &#t02<#t03
        & not(DevNonce=DevNonce2)
        & not(DevEUI =DevEUI2)

       // But only 1 JoinRequest makes it
        & (All #i #j . OnlyOnce('
    NetworkServer_Receive_JoinRequest_Forward_To_JS') @ i &
     OnlyOnce('
    NetworkServer_Receive_JoinRequest_Forward_To_JS') @ j
    ==> #i=#j)
        //and no key reveal
        & not(Ex Entity Type Key #kr . KeyReveal(Entity,
    Type, Key) @ kr)
"
```

Fig. 8. Trackability Lemma with Selective Blocking of $DeviceJoinRequest$

Moreover, Fig. 8 also shows the Tamarin code that expresses the search for the actual trackability-attack `LoRaWAN-Attack2`. That is, the lemma states that there is a trace (fully observable by the attacker which is a given in Tamarin), in which the only application messages seen can be unequivocally attributable to a given (unblocked) device; $DevEUI2$ in this case.

Fig. 9 shows a well-authentication (WA) lemma, written in Tamarin code; a WA property [5] encodes honest parties in a

[19] Our Tamarin files can be found at [35]: `LoRaWan_v1_1_PrivacyModel.spthy` for Section VII-B2a and VII-B2b, and `LoRaWan_v1_1_PrivacyModel_diff.spthy` for Section VII-D.

```
/************************************
WELL-AUTHENTICATION (WA)
Lemma from Solene, Delaune and Baelde CSF 2020,
applied here to LoRaWAN Join v1.1, showing
that whenever a conditional is positively evaluated,
the agents involved are having so far an honest interaction.
************************************/

lemma WA_NS:
  "All NS JoinEUI DevEUI DevNonce tau_c #t02.
    ReadyToOutJoinAccept(NS, JoinEUI, DevEUI, DevNonce,
     tau_c)@t02
    &
     //no forced condition applied
    TestPassed_SurelyNoForcedNonceTest...() @ t02
    ==>
     (
       ( Ex #t01.
        (
           DeviceJoinRequest(DevEUI, NS, JoinEUI, DevNonce,
     tau_c) @t01 &
           t01<t02
        )
      )
      |  ( Ex Entity Type Key #k1 . KeyReveal(Entity, Type,
      Key) @ k1)
     )
  "
```

Fig. 9. Lemma for Well-Authentication (WA)

```
// this lemma is expected to fail and exhibit an attack
    trace
/************************************
FAILURE OF "WELL-AUTHENTICATION" WITH DISHONEST PARTIES
******************/
lemma negation_WA_spellingOutTheForcedCondition:
  "All NS JoinEUI DevEUI DevNonce1 DevNonce2 tau_c1 #t01 #
     t03.
    // there is a JoinRequest
    DeviceJoinRequest(DevEUI, NS, JoinEUI, DevNonce1, tau_c1
     ) @t01
    &
    //and it gets to the backend
    ReadyToOutJoinAccept(NS, JoinEUI, DevEUI, DevNonce2,
     tau_c1)@ t03
    &
    #t01< #t03
    //but the Dev Nonces disagree
    & not(DevNonce1=DevNonce2)
    ==>
    //there must have been a second JoinRequest by the
     device with DevNonce2
    (Ex #t02 tau_c2 . DeviceJoinRequest(DevEUI, NS, JoinEUI,
      DevNonce2, tau_c2) @ t02   & #t02<#t03    )
  "
```

Fig. 10. Lemma for WA-Failure via Forced Condition

protocol authenticated correctly in a strong sense: whenever a condition is positively evaluated, the agents involved up to then are interacting honestly. Here, it encodes that an honest device (i.e., $DevEUI$) is authenticated via its $JoinRequest$ to the NS/JS. We use this lemma in our Tamarin model to make a point via the "forced condition" strategy in `LoRaWAN-Attack2` and in order to discuss notions related to trackability.

Figure 10 shows the lemma encoding our forced conditions. It explicitly forces a trace in which the attacker manipulates a device's $JoinRequest$ by changing the $DevNonce$, thus ensuring the NS rejects the request.

*c) **Statistics of Our Tamarin Models:*** In the models we provide, for simplicity/speed of the proof and without loss of generality, we limit the number of devices to 2 in some of the lemmas. Thus, when also using the provided oracle, `LoraWanPrivacy.py`, the verification times are under 1 min for all lemmas. Our timings were obtained using a laptop with an Intel i7-1065G7 CPU @ 1.3GHz/3.9GHz with turbo boost, 4 cores/8 threads and 16GB RAM. Note that no special proof techniques are needed (i.e., , one can use the "automatic" mode in Tamarin); so anyone can replicate these results easily, even on modest hardware.

C. Gaps in Mechanising Adaptive & Implicit Trackability

We do not show a generic mechanisation method for implicit attacks. Whilst the selective blocking strategy does not apply for implicit attacks (as the IDs are not visible), the forced condition strategy does. A lemma could be written quantifying over implicit IDs in a meaningful way, but this would not be generalisable as in the above case of explicit tracking (i.e., it would be highly protocol specific) and thus we cannot express this is in a generic way.

Adaptive attackers can be mechanised in tools such as Tamarin via restrictions (i.e., fairness constraints [36]), but that would depend on what adaptiveness one wishes to model (i.e., which executions to "cull" from the attacker's visibility).

D. Verifying Our Trackability vs. Arapinis' Unlinkability

As we said above, for unlinkability to fail, at least one of Baelde et al. [5]'s "well-authentication" (WA), "no-desynchornisation" (ND), or "frame opacity" (FO) has to fail. These should be checked in the order stated, as that is the order of their respective strength. In the LoRAWAN Join v1.1, WA does not fail while it is easy to see that FO does. Therefore, in our example, different *JoinRequest* messages would need to be indistinguishable. However, this is obviously not true since each *JoinRequest* contains the $DevEUI$ in plain text, thus clearly labelling each message, i.e., two *JoinRequest* messages from the same $DevEUI$ are easily "linkable". So, FO trivially fails and the LoRaWAN Join v1.1 is therefore deemed linkable by Baelde et al. [5]. While the failure is "obvious" in the LoRaWan Join protocol, we also modelled FO and checked its failure using Tamarin's diff-equivalence mode[20].

Our analysis highlights a gap between practice and the formalisation in the method by Baelde et al. [5]. We show a more fine-grained reason why LoRaWAN Join v1.1 lacks privacy which is not due to the lack of the very idealised FO, but because of the trackability attacks we found. This is a concrete example that making protocols idealised, so that they respect a property as strong as FO, may not be necessary. Indeed, we refer the reader to Section IV in which we give simpler countermeasures to the lack of privacy in the LoRAWAN Join, guided by our trackability attacks.

E. Expressions of Our Trackability vs. Unlinkability Beyond Formal Verification

We now compare our framework with the notion $unlink.^{Arapinis}$, beyond formal verification (which is covered in Section VII-B).

[20] See `LoRaWan_v1_1_PrivacyModel_diff.spthy` available at [35].

1) Our $\forall$-St-SesNoTrack^{impl,active} notion implies $unlink.^{Arapinis}$ and vice-versa. Intuitively, this is because both notions speak of an attacker pinning down sessions to entities.
2) Our $\forall$-St-SesNoTrack^{expl,active} notion is stronger than $unlink.^{Arapinis}$. Intuitively, $unlink.^{Arapinis}$ does not ask that there is a pinning-down to an explicit long-term identifier id of the entity tracked.
3) Our $\forall$-St-NoTrack^{impl,active} notion is weaker than $unlink.^{Arapinis}$. Intuitively, this is because $unlink.^{Arapinis}$ asks to pin down the sessions to entities, and not just to a collection of messages. $\forall$-St-NoTrack^{impl,active} asks just for the latter.
4) Our $\forall$-St-NoTrack$^{\cdot,passive}$ notion and our $\forall$-St-SesNoTrack$^{\cdot,passive}$ notion do not relate to $unlink.^{Arapinis}$. Intuitively, this is because $unlink.^{Arapinis}$ does not consider passive (i.e., non-active) attackers separately from active ones.

The above implications, present in both directions, show our framework is overall more fine-grained than $unlink.^{Arapinis}$. Points 3 and 4 show that our session-insensitive trackability notions (i.e., $\forall$-St-NoTrack) have been largely overlooked by formal unlinkability analysis and, as Section IV shows, these notions carry meaningful threats. In fact, session-insensitive trackability notions (i.e., $\forall$-St-NoTrack) are sufficient in most cases; at the intuitive level, one may simply wish to know where you were yesterday, without caring that it was you who was there exactly at 2am or at 5pm during the day.

Final Words on Trackability vs. Unlinkability. The above shows that our distinction between explicit and implicit trackability is useful; also, Section VII-B showed that explicit trackability can be expressed as a reachability property without any gap. But, this is not true for implicit trackability, in general; in those cases, one ought to rely (for now) on using the approximations in [5], [33].

VIII. Related Work

We cover the most relevant pre-existing work in the field.

A. Domain-Specific Privacy

We aim for a generic framework, applicable to a variety of protocols that contain an identification element, explicit or implicit, cryptographic or non-cryptographic, unlike domain-specific traceability notions recounted below.

Vaudenay proposed a formal model that addresses privacy in RFID in [45]. This is totally focused on the ability to manipulate RFID tags. It considers notions of weak and strong privacy with different powers of attackers (e.g., narrow, insiders, etc.). Hermans et al. [31] propose a similar RFID privacy model to that of Vaudenay's, but theirs is based on different formal definitions for the same notions. Arfaoui et al. [3] use a similar method to create a privacy model specifically for the TLS 1.3 protocol. Koustos in [37] introduces a new privacy property inspired by Vaudenay's work, the *σ-unlinkability* property, for the purposes of privacy verification specifically in the 5G AKA protocol; others AKA-based tracing of users [17], [8], not necessarily via systematic privacy frameworks.

Vaudenay's non-narrow, insider privacy attackers do not correspond to any adversary in our framework, as they would corrupt the parties and not just actively block messages. Yet, the trackability attacks we present on 5G handovers in Section VI relate to his insider attackers to some extent (i.e., the gNBs are not corrupted to have arbitrary behaviour, but they leak the decrypted traffic or the channel keys they share with the network).

Other works looked LoRaWAN Join but not at its privacy, only its security, formally [46] or in practice [32].

B. On Formal Analysis of Privacy

This is discussed in detail in Section VII. Therein, (for LoRaWAN Join), concrete verification comparisons are shown between our mechanisation in Tamarin for our non-trackability properties vs. the approximation of unlinkability put forward by [33], [5], and also implemented by us for the LoRaWAN Join in Tamarin; our mechanisation trackability proved finer for the case of tracking over the LoRaWAN Join.

C. On the Terminology Around Trackability

Some of these aspects, such as the seminal report of Pfitzmann and Hansen [40], were discussed in the introduction when we contextualised our work. Trackability is closest to unlinkability in these senses. Tsukada et al. investigate and organise some of these concepts further [44], particularly relating the concept of unlinkability to that of minimal anonymity. Goriac [29] expands on the work by Pfitzmann and Hansen by adding definitions for *involvement* and *unobservability*, and investigates privacy in terms of *behavioural equivalence* between local states to define indistinguishability.

IX. Conclusions

We have defined a new, multi-parameter framework called TrackDev, to reason about tracking protocol participants in their protocol executions. We show TrackDev is worthwhile: it has several new privacy notions, separable concepts, adds to existing formal analysis' capabilities, it is mechanisable in traffic-capturing *practical* tools, as well as formal-verification tools, and has yielded new, realistic attacks, some of which already lead to the edition of existing standards. In the future, we will further mechanise TrackDev, as well as apply it to more protocols.

References

[1] 3GPP. System architecture for the 5G System (5GS). Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), 10 2020. Version 16.0.0.

[2] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 107–121, Edinburgh, UK, 2010. IEEE, IEEE Computer Society.

[3] G. Arfaoui, X. Bultel, P.-A. Fouque, A. Nedelcu, and C. Onete. The privacy of the TLS 1.3 protocol. *Proceedings on Privacy Enhancing Technologies*, 2019(4):190–210, 2019.

[4] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification*, pages 281–285, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[5] D. Baelde, S. Delaune, and S. Moreau. A method for proving unlinkability of stateful protocols. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 169–183, Los Alamitos, CA, USA, June 2020. IEEE Computer Society.

[6] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001.*, pages 82–96, Cape Breton, NS, Canada, 2001.

[7] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[8] R. Borgaonkar, L. Hirschi, S. Park, and A. Shaik. New privacy threat on 3g, 4g, and upcoming 5g aka protocols. *Proceedings on Privacy Enhancing Technologies*, 2019(3):108–127, 2019.

[9] R. Borgaonkar, L. Hirschi, S. Park, and A. Shaik. New privacy threat on 3g, 4g, and upcoming 5g AKA protocols. *Proc. Priv. Enhancing Technol.*, 2019(3):108–127, 2019.

[10] V. Cheval, S. Kremer, and I. Rakotonirina. The DEEPSEC prover. In H. Chockler and G. Weissenbacher, editors, *International Conference on Computer Aided Verification*, pages 28–36. Springer International Publishing, 2018.

[11] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz. 5g suci-catchers: Still catching them all? In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 359–364, New York, NY, USA, 2021. Association for Computing Machinery.

[12] L. A. T. Committee. *LoRaWAN Specification v1.0*. LoRa Alliance, January 2015.

[13] L. A. T. Committee. *LoRaWAN Specification v1.0*. LoRa Alliance, October 2017.

[14] L. A. T. Committee. *LoRaWAN Backend Interfaces Technical Specification (TS002-1.1.0)*. LoRa Alliance, October 2020.

[15] V. Cortier, A. Dallon, and S. Delaune. SAT-Equiv: An efficient tool for equivalence properties. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 481–494, Santa Barbara, CA, USA, 2017.

[16] V. Cortier, A. Dallon, and S. Delaune. Efficiently deciding equivalence for standard primitives and phases. In J. Lopez, J. Zhou, and M. Soriano, editors, *Computer Security*, pages 491–511. Springer International Publishing, 2018.

[17] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl. Imsi-catch me if you can: Imsi-catcher-catchers. In *Proceedings of the 30th annual computer security applications Conference*, pages 246–255, 2014.

[18] I. Damgard and J. B. Nielsen. Adaptive versus static security in the uc model. Cryptology ePrint Archive, Report 2014/601, 2014. https://ia.cr/2014/601.

[19] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols: A taster. In *Towards trustworthy elections*, pages 289–309. Springer, 2010.

[20] D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactionson Information Theory*, 29(2):198–208, 1983.

[21] ETSI. *ETSI TS 124 501 V16.5.1*. ETSI, August 2020.

[22] ETSI. *ETSI TS 123 003 V16.7.0*. ETSI, August 2021.

[23] ETSI. *ETSI TS 123 502 V16.9.0*. ETSI, July 2021.

[24] ETSI. *ETSI TS 133 501 V16.7.1*. ETSI, August 2021.

[25] ETSI. *ETSI TS 138 300 V16.5.0*. ETSI, April 2021.

[26] ETSI. *ETSI TS 138 401 V16.5.0*. ETSI, April 2021.

[27] ETSI. *ETSI TS 138 413 V16.5.0*. ETSI, April 2021.

[28] H. Gee. Almost gone: The vanishing fourth amendment's allowance of stingray surveillance in a post-carpenter age. https://gould.usc.edu/students/journals/rlsj/issues/assets/docs/volume28/summer2019/3-1-Gee.pdf, 2019.

[29] I. Goriac. An epistemic logic based framework for reasoning about information hiding. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 286–293, Vienna, Austria, 2011. IEEE.

[30] J. C. Haartsen. The bluetooth radio system. *IEEE personal communications*, 7(1):28–36, 2000.

[31] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel. A new RFID privacy model. In *European symposium on research in computer security*, pages 568–587, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[32] F. Hessel, L. Almon, and F. Álvarez. ChirpOTLE. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, jul 2020.

[33] L. Hirschi, D. Baelde, and S. Delaune. A method for unbounded verification of privacy-type properties. *Journal of Computer Security*, 27(3):277–342, 2019.

[34] T. T. Industries. The Things Network. https://www.thethingsnetwork.org/, 2021.

[35] K. B. S. U. o. S. D. R. N. G. Y. R. U. o. S. Ioana Boureanu, Stephan Wesemeyer. Files for trackdev – formal and practical tracking with trackdev. https://github.com/fmsec/trackdev, 2022.

[36] Y. Kesten, A. Pnueli, L.-o. Raviv, and E. Shahar. Model checking with strong fairness. *Formal Methods in System Design*, 28:57–84, 01 2006.

[37] A. Koutsos. The 5G-AKA authentication protocol privacy. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 464–479, Stockholm, Sweden, 2019. IEEE.

[38] Y. Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pages 277–346. Springer International Publishing, Cham, 2017.

[39] S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN Prover for the symbolic analysis of security protocols. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, pages 696–701, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[40] A. Pfitzmann and M. Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. *Version v0*, 31:15, 2008.

[41] Pycom. Pycom. https://pycom.io/, 2020.

[42] Sigfox. *Sigfox connected objects: Radio specifications*. Sigfox, February 2020.

[43] T. T. Team. *Tamarin prover manual*. The Tamarin Team, 2016.

[44] Y. Tsukada, K. Mano, H. Sakurada, and Y. Kawabe. Anonymity, privacy, onymity, and identity: A modal logic approach. In *2009 International Conference on Computational Science and Engineering*, volume 3, pages 42–51, Bellaterra, Catalonia, Spain, January 2009.

[45] S. Vaudenay. On privacy models for RFID. In *International conference on the theory and application of cryptology and information security*, pages 68–87, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[46] S. Wesemeyer, I. Boureanu, Z. Smith, and H. Treharne. Extensive security verification of the LoRaWAN key-establishment: Insecurities & patches. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 425–444, Genoa, Italy, 2020. IEEE.