Secure Anycast Channels with Applications to 4G and 5G Handovers

Karl Norrman[0000−0003−0164−1478]

KTH Royal Institute of Technology, Stockholm, Sweden
Ericsson Research, Security, Stockholm, Sweden karl.norrman@ericsson.com

Abstract. In 3GPP mobile networks, application data is transferred between the phone and an access point over a wireless link. The mobile network wireless link is special since one channel endpoint is handed over from one access point to another as the phone physically moves. Key evolution during handover has been analyzed in various works, but these do not combine the analysis with analysis of the wireless-link application-data encryption protocol that uses the keys.

To enable formal analysis of the 4G/5G wireless link, we develop a game-based security framework for such channels and define flexible key insulation security notions for application data transfer, including forward and backward security in the given adversary model. Our notions are modular and combine a bidirectional application data transfer channel with a generic framework for multiparty channel-evolution protocols. These two components interact, and the security of the channel-evolution protocol may rely on the security of the data transfer channel for some or all its messages.

We also develop the first formal model of 4G/5G wireless link security including both handover key evolution and application data transfer, in the complexity theoretic setting. We prove the model secure w.r.t. our security notions. As a byproduct, we identify recommendations for improving the security of future mobile network standards to achieve key insulation. Specifically, we show that the current standards do not achieve forward secure encryption, even though this appears to be an explicit goal. We show how this can be rectified.

Keywords: Anycast · Handover · 3GPP · 5G · Cryptographic Channel.

1 Introduction

Mobile networks, providing more than 5 billion subscribers[1] with wireless internet access, are a cornerstone of modern society. As a part of critical infrastructure, their security is essential. A core functionality is secure transmission of application data between the mobile phone and the network while the phone physically moves. To the best of our knowledge, this secure transmission in combination with its key evolution schemes has not yet been formally analyzed and even lack a clear security model.

[1] https://www.gsma.com

1.1 Motivation

Mobile Networks. Mobile phones and IoT devices obtain wireless access from mobile networks, which are divided into several parts, see Fig. 1. The two main parts are the serving network and the home network. The former provides wireless access, and the latter controls the phone's subscription and its authentication. Roaming occurs when the serving network is controlled by a different operator than the home network. The serving network is further divided into a *core network* (CN) and a *radio access network* (RAN). In 4G and 5G mobile networks [1], the RAN consists of a set of *access points* connected to the CN. The phone connects to one of the access points, over what we will call the *wireless link*. This name is chosen for its place in the architecture, not because we consider special radio characteristics. We focus on the interactions between the phone and the serving network, i.e., roaming is out of scope.

While key establishment and identifier-privacy aspects of mobile networks have been formally analyzed, data transmission confidentiality and integrity have not. One important reason is lack of appropriate cryptographic channel models for the wireless link, which is the problem we address in this paper.

The wireless link is maintained as the phone moves out of one access point's radio coverage and into coverage of another. This ostensible continuity of the wireless link is achieved through so-called handovers. During handover, the phone, the access points and other support functions in the mobile network coordinate, via a set of protocols, to transfer control of the network's endpoint of the wireless link from one access point to another. This coordination derives new keys and protocol states for the phone and the target access point. The coordination relies on that the links between access points and links between access points and the core network is secured (using IPsec). For our purposes it suffices to treat the core network as a single entity as shown in Fig. 1.

Ultimately, we wish to analyze the security of application messages passed between the phone and the network in both directions over this wireless link. That is, we wish to analyze this communication in the secure channel setting, including how a *Channel Evolution (CE)* protocol updates the channel state at handovers. Prior works exist which focus on establishment of keys for the wireless link, and how these keys evolve at handover, but these works leave the security of the sent application messages, i.e., the secure channel aspect, out of scope. Composing key establishment and encryption protocols is known to be intricate.

We view the wireless link as a special case of the abstract concept stateful anycast [42], but where endpoints of the link can be re-selected during the link's lifetime. As far as we know, this abstract setting has not been analyzed through the lens of secure channels earlier. Anycast considers access points equivalent from a service delivery perspective. It may therefore be tempting to consider them equivalent also from a security perspective, so that a multi-key channel [32] composed with state transfer between access points suffices to model the secure channel. However, mobile networks require a more complicated threat model and some access points may be in physically exposed locations such as shopping malls or outdoors, where adversaries may have easy access. That is, not all access points

share the same degree of protection. Therefore, it is prudent to consider more elaborate threat models, where an adversary may compromise some access points but not all. We define security notions incorporating a kind of recovery from state compromise after a certain number of endpoint transfers. Such recovery may at first appear contradicting the impossibility result of Bellare et al. which states that key insulation is unachievable under active attacks when it makes use of a *public channel* between the helper and the primitive [6]. However, we make use of the fact that each access point has its own public channel to the helper function and that compromising an access point compromises its public channel, but not public channels between the helper function and other access points.

Running a complete secure channel (re-)establishment protocol at handover straightforwardly ensures strict key insulation. However, that may be prohibited by network topology, or may be infeasible due to the service's real-time nature. 4G/5G networks therefore require security notions which are flexible in terms of how many handovers are allowed before achieving forward and backward security.

The 4G and 5G specifications TS 33.401 and TS 33.501 [1] are unfortunately not accompanied by an explicit security model. Neither are they written for the purpose of formal verification, and security notions and claims are often vague and implicit. Although the specifications define two concepts, *forward security* and *backward security*, these do not directly translate into the traditional notions of similar names [12,21,31]. [2] Therefore, one of our aims is to formally capture what secure handover perhaps *ought* to mean, in terms of the more traditional meaning of forward and backward secrecy. Another aim is to investigate what improvements 4G/5G require to reach this notion of security.

We therefore construct a game-based framework for analyzing these types of links. It captures both key evolution and the security of the encryption protocol, seen as a channel for sending application data securely between the phone and the network. The task is made challenging by the fact that mobile network key-evolution protocols may rely on the security of the encryption protocol, which in turn relies on the key-evolution protocol to provide encryption keys. Further complications arise from that the network endpoint of the channel is transferred between access points, some of which may be compromised.

Secure Channels. Although key establishment and key evolution may be part of secure channel functionality, the core purpose is transfer of sequences of arbitrary messages between parties. There are many variations on the theme, handling complex use cases, e.g., multi-key channels [32]and unreliable transmission [27,40,16,47]. Some cryptographic channel formalizations include initial channel and key establishment [19,4,37,17,24], whereas others assume the channel is already securely established, and focus on application data transmission security properties [8,16,47,41,32]. There are even formalizations modeling channel re-establishment with an endpoint under a different name [29]. None of these

[2] We discuss the differences in a remark in Section 5.2. The differences give an interesting indication of opposing views on security definitions.

formalizations, however, model handovers, where one party seamlessly transfers control of its endpoint to another party.

The process algebra and protocol logic communities have formally studied 4G/5G handover security [43,20,34]. There is informal analysis [28]. There is also work in the computational model, some focusing on Small Cell Networks (SCN). SCN is a collective term for systems integrating low-power access points in the mobile network architecture. This is an extended architecture that not necessarily corresponds to the basic setting of the 4G/5G standard, and calls for *new* schemes rather than analysis of the standardized schemes. Many works propose schemes for authentication and key establishment for handovers from one region to another in SCN. They consider the evolution of keys and authentication, but disregard security of the data transmission at handover. Therefore, they often analyze their schemes in derivatives of Bellare-Rogaway's framework for key establishment [10], which does not treat subsequent message encryption using the established key. Recent works include ReHand, focusing on extended architectures where small-cell access points are connected a regular 5G access point, and the phone is handed over between the small-cell access points, for which they propose a scheme [26]. Alnashwan et al. propose new authentication and key establishment schemes for SCN handovers [3]. They prove key indistinguishability, anonymity, and authentication properties. Gupta et al. [33] propose a key establishment scheme for 4G/5G handover based on bilinear pairings.

Blazy et al. [13] analyze Post Compromise Security in a multi-stage key exchange model and consider key indistinguishability for basic 5G handovers. While they consider a vast array of adversary models, their 5G model is much more abstract than ours. Specifically, they disregard application data security, they consider all access points as one, and they assume secure channels between the access point(s) and the core network. Their model is too abstract to capture the forward security weakness we uncovered. We stress again that these proposals consider key establishment and evolution, but not application data transfer.

1.2 Contributions

We initiate the study of secure anycast channels capable of channel endpoint transfer between parties and use this to analyze 4G/5G wireless link handovers.

- We define a general notion of secure bidirectional anycast channels, featuring endpoint control transfer from one party to another. Our adversary model include (restricted) endpoint corruption and session key reveal. Our work fills a gap in techniques required to analyze real-world systems, e.g., 4G and 5G. It is to the best of our knowledge the first to address secure anycast channels as such, which may be of independent interest.
- Our game-based notions capture a flexible version of key insulation, achieving backward and forward security after n phases under the adversary model Fig. 2. This enables analysis of realistic 4G/5G network topologies.
- Our notions are modular. Endpoint control transfer is managed by a channel evolution (CE) protocol. Importantly, CE-protocols may send messages over the secure channel itself, and crucially rely on its security. A generic interface

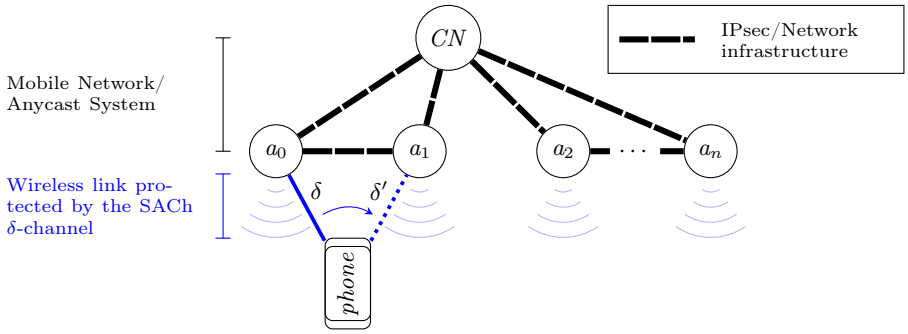

Fig. 1. System model. The *phone* exchanges application data with access point a_0 before the handover and with a_1 afterwards. In our generic SACh model, the phone is called a client, the (serving) core network (CN) is called the orchestrator (o), and the wireless link is called the δ-channel. Links between CN and access points, and links between access points, is in mobile networks protected by IPsec when cryptographically protected. The channel evolution protocol updates keys and other channel state at handover to ensure that messages sent over the wireless link/δ-channel are secured.

for interaction between the secure channel and a wide range of CE protocols. These can be individually designed for specific topologies.
– We provide the first complexity-theoretic channel-based model of the 4G/5G wireless link data transmission security, which includes handovers. It is rather close to the 3GPP standard. We show that it fulfills our security notions.
– We show that 4G/5G do not achieve forward security and how future mobile network generations can be improved to do.

1.3 Related Work

Our work sprung from realizing that Günther's and Mazaheri's multi-key channels [32] (GM henceforth) do not suffice for modeling 4G or 5G wireless link security. GM focuses on key-usage and cannot handle endpoint transfer, whereas we take spatial aspects into account: access point break-ins should be compartmentalized. GM requires deterministic key evolution and it cannot rely on the security of the secure channel. However, our δ-channel share structure with their phase-based model and we highlight differences and similarities throughout.

Dodis et al. proposed Key Insulation (KI) for the public key setting [22]. They consider a primitive, assisted by a secure helper entity that evolves keys, for the purpose of temporally compartmentalizing attacks against the primitive. This concept was later extended to the symmetric key setting by Dodis et al. [23].

Bellare et al. [6] considered communication between the helper and the key insulated primitive in the KI setting. In GM, these two entities are co-located and this is less important. In our case, they are separated. Bellare et al. showed that their communication cannot be secured against an active adversary when

the primitive is corruptible. We show how this problem is present in mobile networks, but may be practically less severe due to the endpoint transfers.

Secure channels, including GM, are often unidirectional for simplicity. Marson and Poettering [41] showed that combining two such channels does not trivially result in a secure bidirectional channel. We adopt their conclusions.

Our domain is similar to secure messaging, where double-ratchet designs are often used. However, many such designs, e.g., [4,25], use asymmetric keys and only focus on two-party message exchange; hence they are inapplicable here.

2 Secure Anycast Channels

2.1 Preliminaries

Sets and Sequences. We write the disjoint union of sets S and T as $S \uplus T$. We write $|S|$ for the size of S and $|s|$ for the length of $s \in S$ for some metric on S. We write $S^{|s|}$ for the subset of elements in S that are of length $|s|$. A sequence $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ over S is an ordered list of possibly repeating elements $\sigma_i \in S$. Slightly abusing notation, we denote the length of the sequence σ by $|\sigma|$, and the fact that an element σ_i is in σ by $\sigma_i \in \sigma$. The set of all finite sequences over S is denoted by S^*. We write $S^\perp$ for $S \cup \{\perp\}$, where $\perp \notin S$. We write $\sigma \| e$ when appending an element e to σ, and write $S \xleftarrow{\cup} e$ instead of $S \leftarrow S \cup \{e\}$.

Algorithms. Let S be a finite set, and let u and v be variables. We write $v \xleftarrow{R} S$ to denote uniform random sampling of an element from S and assigning it to v. We use $\leftarrow$ to denote value assignment and $\leftarrowtail$ to denote reference assignment. For example, if V is a reference, then $v \leftarrow 1;\ V \leftarrowtail v;\ v \leftarrow 2;$ implies that $V = 2$. Tuples are uniquely encoded. This enables multiple simultaneous assignments when decoded, e.g., $(u, v) \leftarrow (1, 2)$ implies that $u = 1 \wedge v = 2$.

We model stateful (randomized) algorithms using conventional monadic sequencing. Let $A(s, \sigma_i, \cdot)$ be an algorithm taking as input at least a state s and data item σ_i, and outputting at least an updated state s' and a processed data item τ_i. The entity invoking A maintains a state variable s, invokes A with s and other inputs, and assigns the output state s' to s when A returns, i.e., $(s', \tau_i, \cdot) \leftarrow A(s, \sigma_i, \cdot);\ s \leftarrow s'$. Suppose that s_0 is a state for A and that $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$ is a sequence of inputs. We write $(s_{n-1}, \tau, \cdot) \leftarrow\!\!\!\!\rightarrow A(s_0, \sigma, \cdot)$ for the n-fold monadic sequence of invocations of A over σ starting in state s_0.

Computational Model. We consider efficient algorithms in a uniform probabilistic polynomial time (PPT) Random Access Memory model. Definitions, theorems and reductions are given in asymptotic black-box constructive form for a security parameter λ [46]. As usual, a function $f(\lambda)$ is negligible in λ if $\forall c \in \mathbb{N}.\ \exists \lambda_c \in \mathbb{N}.\ \forall \lambda \in \mathbb{N}.\ \lambda \geq \lambda_c \Rightarrow f(\lambda) \leq \lambda^{-c}$. We assume a uniform distribution for sampling and random objects unless otherwise specified.

2.2 Anycast Systems as an Abstraction of Mobile Networks

We now define what we mean by anycast systems, their architecture and system dynamics. Our model generalizes and abstracts wireless-link communication in 3GPP mobile networks, subject to handovers.

System Architecture. Consider a *client* c accessing a service provided by an *anycast system* S consisting of an *orchestrator* o and a set of (service) *access points* (see Fig. 1). The client (the phone in mobile networks) is connected to, and receives service from, one of the access points in S. The orchestrator represents the serving core network in mobile networks. Formally, let $\mathcal{U} = \mathcal{U}_O \uplus \mathcal{U}_A \uplus \mathcal{U}_C$ be a set of identities, where $\mathcal{U}_O = \{o\}$, $\mathcal{U}_A = \{a_k\}_{k\in\mathbb{N}}$ and $\mathcal{U}_C = \{c\}$. Identities represent unique parties and we use them as identifiers for the parties. The orchestrator may encompass many subfunctions, executed by different logical or physical servers, but we treat it as a single entity. The orchestrator and access points can all be connected by communication channels but do not need to be.

We call the channel between the client and one of the access points the δ-*channel*. The δ-channel models the secure transmission of application data over the wireless link in mobile networks. An access point may transfer control of its δ-channel endpoint to another access point without terminating the δ-channel. We use endpoint transfer to model handover. The multiparty protocol executed by S to transfer its endpoint is a Channel Evolution (CE) protocol. Together, the CE protocol and the δ-channel constitute a *Secure Anycast Channel*, a SACh.

System Dynamics. At any point in time, exactly one access point controls the δ-channel endpoint on the system side. The damage compartmentalization goal suggests establishing fresh keys and state for the δ-channel at endpoint transfer. This naturally leads to phase-divided processing; the δ-channel connects the client c and a single access point a^t in each phase t, and endpoint transfer using a CE-protocol progresses the SACh to the next phase (see Fig. 6). Let t_u denote the phase that party u is in. Parties progress to the next phase when they receive or send a message using the next δ-channel state, regardless of whether this is an application message or a CE message. Since c may return to a previously visited access point, we may have $a^t = a^{t+n}$ for some integer n.

The δ-channel's global state δ consists of the client substate and the controlling access point's substate, i.e., $\delta = (\delta_c, \delta_a)$. For $u \in \{c, a\}$ and phase t_u, the variables $i_u^{t_u}$ and $j_u^{t_u}$ range over the number of δ-channel messages u sent and received, respectively, in phase t_u. We write $\delta_u^{t_u}$ for u's δ-state in phase t_u, and $\delta_u^{(t_u, i_u^{t_u}, j_u^{t_u})}$ when the number of processed messages matter. Indices quickly become unwieldy, so we define a special notation for δ-substates. Let f and g be expressions in the number of sent and received messages, respectively. Now define $(u, h(t) \mid f(i), g(j)) \triangleq (h(t_u), f(i_u^{t_u}), g(j_u^{t_u}))$. Using this notation, we may write, e.g., $\delta_u^{(v,t \mid i+1,j)}$ instead of $\delta_u^{(t_v, i_v^{t_v}+1, j_v^{t_v})}$ to indicate party u's δ-substate when party v is in phase t_v, and v has sent $i_v^{t_v}+1$ messages and has received $j_v^{t_v}$ messages. The notation is not fully generic, but covers the cases we need.

CE protocols may send messages over the δ-channel, and may rely on its security. This interplay allows more faithful handover models, but is simultaneously our model's main source of complexity. It appears that this complexity is often purposely avoided, favoring simplicity over expressiveness [32,6].

The CE protocol's global state γ consists of the substates of all parties, i.e., $\gamma = (\gamma_o, \gamma_c, \{\gamma_a\}_{a \in \mathcal{U}_A})$. Intuitively, session-keys in the δ-states are derived from secret information shared by only γ_o and γ_c, but we impose no special restrictions on the δ-states or γ-states because we want flexibility to handle a large class of CE protocols.

Our goal is to define the meaning of security for application data sent over the δ-channel during a sequence of phases, given an already initialized system.

2.3 Syntax

The SACh syntax is composed of the CE syntax and the δ-channel syntax. We call the CE protocol the CE component when emphasizing the formalism. States are explicit algorithm inputs, but are implicitly assumed here in the description.

Inspired by Brzuska et al. [18], we use an algorithm Next, passing messages between the entity invoking Next and the underlying CE component (see Definition 1). Invoking $\mathsf{NextInit}$ sets the target access point ν for the next endpoint transfer. Execution starts by invoking Next with the message $\perp$ and the CE-protocol initiator u. Next returns the first CE message m generated by u. Continuing, Next is invoked again with m and its intended receiver, and so on until Next returns the message $\perp$, signaling completion.

Interactions with the δ-channel works as follows. Next takes an input κ, indicating the δ-state used to decrypt the message (or $\perp$ for no decryption). Failed decryption implies that $m = \perp$. Next returns a value κ', indicating which δ-state the SACh must encrypt the returned message with (or $\perp$ for no encryption).

Next also returns a tuple E indicating whether a party has progressed to the next phase during *this* invocation. Specifically, $E_a = 1$ implies that the target access point ν has taken control over the δ-channel.

Definition 1 (Syntax for CE Component). *A* Channel Evolution (CE) Component *is a tuple of distributed algorithms* $\mathsf{CE} = (\mathsf{NextInit}, \mathsf{Next})$ *for parties with identities* $\mathcal{U} = \mathcal{U}_O \uplus \mathcal{U}_C \uplus \mathcal{U}_A$*, s.t.* $o \in \mathcal{U}_O$ *is an orchestrator,* $c \in \mathcal{U}_C$ *is a client, and* $\mathcal{U}_A$ *is a set of access points.* CE *is associated with a* γ*-state space* $\mathcal{S}_\gamma$*, a* δ*-state space* $\mathcal{S}_\delta$ *and a message space* $\mathcal{M}$*.*

- $\mathsf{NextInit}\colon \mathcal{S}_\gamma \times \mathcal{U}_A \longrightarrow \mathcal{S}_\gamma$ *The algorithm sets the target access point for the next phase progression to* u *for the given* γ*-state.*
- $\mathsf{Next}\colon \mathcal{S}_\gamma \times \mathcal{S}_\delta \times \mathcal{U} \times (\mathcal{U}_C \cup \mathcal{U}_A)^\perp \times \mathcal{M}^\perp \longrightarrow \mathcal{S}_\gamma \times \mathcal{S}_\delta \times \mathbb{Z}_2^2 \times (\mathcal{U}_C \cup \mathcal{U}_A)^\perp \times \mathcal{M}^\perp$ *The algorithm takes a* γ*-state, a* δ*-state, a recipient* $u \in \mathcal{U}$ *and message for* u *as input.* Next *returns, possibly updated,* γ*-state and* δ*-state, a tuple* $E = (E_c, E_a)$ *of indicator variables (telling whether the corresponding party progressed to the next phase), a* δ*-channel endpoint* $\kappa \in \{c, a\}$ *if the returned message must be sent over the* δ*-channel from that endpoint (and* $\perp$ *otherwise), and finally the actual message itself (or* $\perp$ *if there is no message).*

An example CE protocol is the following. Parties o and c share a root key and a monotonic counter, from which they derive keys for the next δ-state using a PRF. A message from c to o informs o that c progressed to the next phase and has derived the next key. In response, o sends the next δ-state key to the target access point, which then progresses the other end of the δ-channel to the next phase. We do not define a proper scheme-class for CE components to avoid restricting their design space or complicating their syntax.

It may be tempting to require that the CE component returns a key indistinguishable from random key, similar to [6]. However, that requirement disqualifies key evolution protocols that require more than one progression to achieve KI. As discussed, natural examples of such protocols exist. The price for this flexibility is twofold; CE protocols may be difficult to compare and, generic composition theorems for CE components and δ-channels are likely complicated.

Secure δ-channels. Our δ-channel is a bidirectional channel, close in spirit to the work of Marson and Poettering [41], but with significant differences. First, we internalize the associated data parameter *ad* into Send and Recv. While Authenticated Encryption with Associated Data (AEAD) syntax [44] commonly exposes this parameter, we consider nonce management to be a natural internal task for the channel abstraction itself. One could imagine combining external associated data with internally generated associated data. This gives the adversary some control over the associated data and makes the syntax more flexible. We opt not to do this since it increases complexity and distracts from our anycast focus. Second, we use a Real-or-Random (RoR) flavor instead of a Left-or-Right (LoR) flavor for the CPA experiment. Finally, a stylistic difference is that we set a flag *win* instead of aborting when the adversary succeeds in the integrity experiment.

The δ-channel is initialized with a configuration that contains secret information, e.g., keys, and information that may be public, e.g., which encryption algorithm to use. Note that we do not distinguish between the messages space and the ciphertext space.

Definition 2 (Syntax for δ-channel Scheme). *A δ-*channel Ch *is a tuple of algorithms* $(\mathsf{Init}_\delta, \mathsf{Send}, \mathsf{Recv})$ *implementing a bidirectional secure channel for a client and an access point.* Ch *is associated with a configuration space $\mathcal{CFG}$ and a state space $\mathcal{S}_\delta$. The* Send *and* Recv *algorithms are further associated with a message space $\mathcal{M}^\perp$. The syntax of the algorithms are as follows.*

- $\mathsf{Init}_\delta \colon \mathcal{CFG} \longrightarrow \mathcal{S}_\delta$ *Given a configuration as input, it returns an initial δ-state $\delta = (\delta_c, \delta_a)$, where δ_c is the substate for the client and δ_a is the substate for the access point currently controlling the δ-channel.*
- $\mathsf{Send} \colon \mathcal{S}_{\delta_u} \times \mathcal{M} \longrightarrow \mathcal{S}_{\delta_u} \times \mathcal{M}$ *Given a δ-substate $\delta_u^{(u,t\,|\,i,j)}$ and a message $m^{(u,t\,|\,i,j)}$,* Send *returns a next state $\delta_u^{(u,t\,|\,i+1,j)}$ and a ciphertext $c^{(u,t\,|\,i,j)}$.*
- $\mathsf{Recv} \colon \mathcal{S}_{\delta_u} \times \mathcal{M} \longrightarrow \mathcal{S}_{\delta_u} \times \mathcal{M}^\perp$ *Given a δ-substate $\delta_u^{(u,t\,|\,i,j)}$ and a ciphertext $c^{(u,t\,|\,i,j)}$,* Recv *returns a next state $\delta_u^{(u,t\,|\,i,j+1)}$, and a message $m^{(u,t\,|\,i,j)}$ or $\perp$.*

Definition 3 (Syntax for Secure Anycast Channels (SACh)). *A* secure anycast channel (SACh) Ch *is a tuple of algorithms* $(\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{NextInit}, \mathsf{Next})$ *for a set of parties with identities* $\mathcal{U}$*, partitioned into an* orchestrator $o \in \mathcal{U}_O$*, a set of* access points $\mathcal{U}_A$ *and a* client $c \in \mathcal{U}_C$*.* Ch *is associated with state spaces* $\mathcal{S}_\gamma$ *and* $\mathcal{S}_\delta$*. The* Send *and* Recv *algorithms are further associated with a message space* $\mathcal{M}^\perp$*. The syntax of the algorithms are as follows.*

- $\mathsf{Init}\colon \mathbb{N} \longrightarrow \mathcal{S}_\gamma \times \mathcal{S}_\delta$ *Given the security parameter* λ *as input, it returns an initial* γ*-state and an initial* δ*-state* $\delta = (\delta_c, \delta_a)$*, where* δ_a *is the substate for the access point* a *currently controlling the* δ*-channel.*
- $\mathsf{Send}\colon \mathcal{S}_{\delta_u} \times \mathcal{M} \longrightarrow \mathcal{S}_{\delta_u} \times \mathcal{M}$ *As defined for the* δ*-channel (Definition 2).*
- $\mathsf{Recv}\colon \mathcal{S}_{\delta_u} \times \mathcal{M} \longrightarrow \mathcal{S}_{\delta_u} \times \mathcal{M}^\perp$ *As defined for the* δ*-channel (Definition 2).*
- $\mathsf{NextInit}\colon \mathcal{S}_\gamma \times \mathcal{U}_A \longrightarrow \mathcal{S}_\gamma$ *As defined for the CE component (Definition 1).*
- $\mathsf{Next}\colon \mathcal{S}_\gamma \times \mathcal{S}_\delta \times \mathcal{U} \times (\mathcal{U}_C \cup \mathcal{U}_A)^\perp \times \mathcal{M}^\perp \longrightarrow \mathcal{S}_\gamma \times \mathcal{S}_\delta \times \mathbb{Z}_2^3 \times (\mathcal{U}_C \cup \mathcal{U}_A)^\perp \times \mathcal{M}^\perp$ *As defined for the CE component (Definition 1).*

The intuition behind the SACh correctness definition is the now de facto standard: the sequence of received messages must be a prefix of the sequence of sent messages in the presence of a benign adversary who faithfully relays messages [8] in a sequence of phases [32]. An important aspect of the latter work is that correctness is conditioned on the receiver receiving all messages in a phase before progressing to the next. Messages sent in the previous phase cannot be decoded by that receiver, who has disposed of the previous phase's keys.

Definition 4 (SACh Correctness). *Let the tuple* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Next})$ *be a secure anycast channel. Suppose* $\mathsf{Init}(\lambda, \mathcal{U}_A)$ *has been invoked so that the states* $\gamma = (\gamma_o, \gamma_c, \{\gamma_a\}_{a \in \mathcal{U}_A})$ *and* $\delta = (\delta_c, \delta_a)$ *are initialized.*

- *Let* $\mathbf{m}^t_{c\downarrow} = (m^{t,0}_{c\downarrow}, \ldots, m^{t,M^t_c}_{c\downarrow}) \in \mathcal{M}^*$ *and* $\mathbf{m}^t_{a\downarrow} = (m^{t,0}_{a\downarrow}, \ldots, m^{t,M^t_a}_{a\downarrow}) \in \mathcal{M}^*$ *be the message sequence that the client, and the anycast system respectively, sends over the* δ*-channel in phase* t*,*
- *the encodings of* $\mathbf{m}^t_{c\downarrow}$ *and* $\mathbf{m}^t_{a\downarrow}$ *be* $\mathbf{s}^t_c = (s^{t,0}_c, \ldots, s^{t,M^t_c}_c) \hookleftarrow \mathsf{Send}(\delta_c^{(c,t_c \mid 0,j)}, \mathbf{m}^t_{c\downarrow})$ *and* $\mathbf{s}^t_a = (s^{t,0}_a, \ldots, s^{t,M^t_a}_a) \hookleftarrow \mathsf{Send}(\delta_a^{(a,t_a \mid 0,j)}, \mathbf{m}^t_{a\downarrow})$ *respectively, and*
- *the decodings of* $\mathbf{s}^t_a$ *and* $\mathbf{s}^t_c$ *be* $\mathbf{m}^t_{c\uparrow} = (m^{t,0}_{c\uparrow}, \ldots, m^{t,M^t_a}_{c\uparrow}) \hookleftarrow \mathsf{Recv}(\delta_c^{(c,t_c \mid i,0)}, \mathbf{s}^t_a)$ *and* $\mathbf{m}^t_{a\uparrow} = (m^{t,0}_{a\uparrow}, \ldots, m^{t,M^t_c}_{a\uparrow}) \hookleftarrow \mathsf{Recv}(\delta_a^{(a,t_a \mid i,0)}, \mathbf{s}^t_c)$ *respectively.*
- *Let* $\mathbf{m}_{c\downarrow} = (\mathbf{m}^0_{c\downarrow}, \ldots, \mathbf{m}^{N_c}_{c\downarrow})$ *and* $\mathbf{m}_{a\downarrow} = (\mathbf{m}^0_{a\downarrow}, \ldots, \mathbf{m}^{N_a}_{a\downarrow})$ *be all messages sent by the client and the anycast system respectively, in all phases.*
- *Let* $\mathbf{m}_{c\uparrow} = (\mathbf{m}^0_{c\uparrow}, \ldots, \mathbf{m}^{N_a}_{c\uparrow})$ *and* $\mathbf{m}_{a\uparrow} = (\mathbf{m}^0_{a\uparrow}, \ldots, \mathbf{m}^{N_c}_{a\uparrow})$ *be all messages received by the client and the anycast system respectively, in all phases.*

The iterative invocations of Send *and* Recv *may be interleaved with each other, but not with invocations of* Next*. Once* Next *is first invoked after an invocation of* $\mathsf{NextInit}$*,* Next *must be invoked without interleaved* Send*s and* Recv*s for other messages than CE protocol messages until the next phase.*

We say that Ch *is correct if for any* N_c *and* N_a*, and for* $0 \le t_c \le N_c$ *and* $0 \le t_a \le N_a$ *any* $M^{t_c}_c$ *and* $M^{t_a}_a$*, and for* $0 \le i_c \le M^{t_c}_c$ *and* $0 \le i_a \le M^{t_a}_a$ *and any* $m^{t_c,i_c}_{c\downarrow}$ *and* $m^{t_a,i_a}_{a\downarrow}$ *we have* $\mathbf{m}_{c\downarrow} = \mathbf{m}_{a\uparrow}$ *and* $\mathbf{m}_{a\downarrow} = \mathbf{m}_{c\uparrow}$*.*

The definition is complicated by the δ-channel being bidirectional, leading to interleaved invocations of Send and Recv for both endpoints of the channel.

2.4 Rationale

Removing the option to send CE messages over the δ-channel results in simpler and clearer definitions. The downside is that CE components then must protect themselves and cannot offload any of that work to the δ-channel. Consequently, they would become more complicated. Moreover, this interaction is of practical importance since deployed systems, such as 4G and 5G, do use it.

In contrast to GM, who use a deterministic and non-interactive derivation scheme for phase keys, and to Alwen et al.'s CKA [4], who do not allow general (symmetric key) CE protocols, 4G/5G handovers require fairly general multi-party CE protocols. One may expect this to complicate the correctness definition further. However, CE protocols only indirectly affect the δ-channel messages by evolving the δ-state. Therefore, verifying that sent and received δ-channel messages are equal rejects constructions, which do not properly evolve the δ-state.

Bellare et al. [6] model key evolution as an atomic operation. The transcript of a two-party protocol execution is returned together with a fixed set of keys, which must be indistinguishable from randomly sampled keys. While simpler to work with and appropriate for their use case the technique is not general enough for modeling 4G/5G handovers, which involve more than two parties, and where the primitive must interact with individual messages in the transcript.

3 Security Notions for Anycast Channels

Secure communication typically requires confidentiality and integrity. Confidentiality schemes are often divided into those resisting chosen plaintext attacks (CPA) and those also resisting chosen ciphertext attacks (CCA). Both these notions are traditionally based on indistinguishability [30]. In contrast to the more customary LoR indistinguishability, used by many secure channels, we employ RoR plaintext indistinguishability [5]. We motivate this choice later, but the intuition is that it allows a simpler abstraction for the CE protocol. A RoR-CPA adversary queries the encrypt/send oracle with messages m_i. The game samples a bit b uniformly at random and returns ciphertexts c_i, which are encryptions of either m_i, if $b = 0$, or equal length randomly sampled messages if $b = 1$. The adversary wins the game if it can determine from m_i and c_i whether b equals 0 or 1. RoR-CCA adversaries are equipped with an additional decrypt/receive oracle, but cannot forward the output from the encrypt/send oracle to the decrypt/receive oracle since this leads to trivial attacks. Bellare et al. lifted the notion of encrypting individual messages to stateful encryption of sequences of messages [8]. A key point is that the decryption algorithm is stateful and can thus detect replayed and out-of-order delivered ciphertexts. Their notion comes in CPA and CCA versions, which are today core properties of secure channels.

GM [32] later extended this notion to stateful encryption where encryption keys evolve [2,12] over a sequence of phases.

Integrity protection, when combined with encryption, is usually captured by either of the security notions INT-CTXT [39,11] or INT-PTXT [9]. The former covers integrity of ciphertexts and the latter integrity of plaintexts. They have also been lifted to sequences of messages, resulting in INT-sfCTXT and INT-sfPTXT [8,18]. Analogous to the encryption counterparts, these integrity notions were then extended to cover sequences of phases with evolving keys [32].

Most generic secure channel models are unidirectional, but bidirectional exceptions exist [35,15,41]. Recent protocol-specific models are also bidirectional [4].

A somewhat orthogonal consideration is the adversary's capability to reveal keys and corrupt parties. Compared to stateful authenticated encryption, Günther and Mazaheri strengthen the adversary model by allowing the adversary to reveal phase-keys and to corrupt parties in certain situations. Against this stronger adversary, schemes can provide partial protection in the form of KI security [22] and forward security w.r.t. the helper key (see Section 1.3).

3.1 Adversary Model

We equip an adversary $\mathcal{A}$ with standard capabilities to control communications, and advanced capabilities to corrupt parties and reveal δ-state. As usual, we must prevent that $\mathcal{A}$ trivially wins. We handle trivial wins via penalty-style definitions [7]. The experiment records events in a log $\mathcal{L}$, and evaluate a polynomial-time computable validity-predicate $P^{ADV}_{n_B,n_A}$ over $\mathcal{L}$ and other parameters (defined in this section). If the validity-predicate evaluates to false, $\mathcal{A}$ loses the game (or a random bit is returned for decision experiments). The predicate is divided into subpredicates, each one capturing one aspect of the adversary model.

We first give the full predicate and then explain the subpredicates. Quantifiers in $P^{ADV}_{n_B,n_A}$ capture free variables in subpredicates, and the variable t is the phase in which $\mathcal{A}$ attacks. Computing $P^{ADV}_{n_B,n_A}(\mathcal{L},t)$ requires a fixed number of linear passes over $\mathcal{L}$. The length of $\mathcal{L}$ is bounded from above by the number of oracle invocations by $\mathcal{A}$; hence $P^{ADV}_{n_B,n_A}(\mathcal{L},t)$ is efficiently computable.

$$
\begin{aligned}
P^{CPA} &\triangleq \nexists(\mathcal{L}_{\mathrm{CPA}_\perp}) \in \mathcal{L},\\
P^{Ph} &\triangleq t \neq \perp \wedge \nexists(\mathcal{L}_{\mathrm{Next}_\perp}) \in \mathcal{L},\\
P^{\mathrm{Cor}} &\triangleq t_{\mathrm{Cor}} \geq t + n_A \ \vee \ a \notin \{src, trg, c, o\},\\
P^{\mathrm{Rev}} &\triangleq t_{\mathrm{Rev}} \neq \perp \ \wedge \ t_{\mathrm{Rev}} \notin (t - n_B, t + n_A) \ \wedge \ t_{Rev} \leq t_u,\\
P^{ADV}_{n_B,n_A}(\mathcal{L},t) &\triangleq P^{CPA} \wedge P^{Ph} \wedge \ \forall(\mathcal{L}_{\mathrm{Cor}}, t_{\mathrm{Cor}}, a) \in \mathcal{L}, (\mathcal{L}_{\mathrm{HO}}, src, trg) \in \mathcal{L} \,.\, P^{\mathrm{Cor}}\\
&\quad \wedge \ \forall(\mathcal{L}_{\mathrm{Rev}}, t_{\mathrm{Rev}}, t_u, u) \in \mathcal{L} \,.\, P^{\mathrm{Rev}}.
\end{aligned}
$$

The P^{CPA} subpredicate prevents trivial attacks in the CPA game. If $\mathcal{A}$ performs active attacks in the CPA game (and only in this game), an event $\mathcal{L}_{\mathrm{CPA}_\perp}$ is logged to $\mathcal{L}$. P^{CPA} is true if no such events exist in $\mathcal{L}$, and is guaranteed to be vacuously true for all other games.

Challenge Bits and Phases. A single challenge bit b is insufficient to capture security in multi-phase games when $\mathcal{A}$ may reveal individual phases [38,36]. To see this, suppose a single bit is used, that $\mathcal{A}$ obtains a ciphertext in one phase and reveals the corresponding δ-substate. $\mathcal{A}$ then learns b for all phases and can legitimately use this to successfully attack a different, past or future, phase. To handle this, we apply the common technique of using a separate challenge bit b_t in each phase t and require that $\mathcal{A}$ determines b_{t^*} for a specific phase t^*, called the *test phase*. In integrity games, the test phase refers to the first phase that $\mathcal{A}$ attacks. The cost of this design is an index-guessing game-hop in security proofs and a security-loss growing linearly in the number of phases.

The adversary must select a target access point by invoking $\mathcal{O}_{\mathsf{NextInit}}$ before starting the CE protocol execution. This models adversaries that, e.g., selectively jam radio frequencies to lure a phone into connecting to another access point, which possibly is under the adversaries' control. To not complicate the Next algorithm's interface further, we choose to restrict CE-protocol executions in the following sense. Once a target access point is selected, the CE protocol must complete before another handover is allowed to start. Whether a handover is in progress or not is tracked by the variable α. It equals $\top$ if a handover is in progress and $\bot$ otherwise. If $\mathcal{A}$ violates the restriction, an event $\mathcal{L}_{\mathrm{Next}_\bot}$ is logged.

Formally, the P^{Ph} subpredicate captures that the test phase is valid and that $\mathcal{A}$ progressed handovers one at a time.

Corruption and State Reveal. $\mathcal{A}$ is malicious, active and adaptive with respect to communications, δ-state reveals and party corruptions. $\mathcal{A}$ can drop, re-order, modify and inject messages arbitrarily.

$\mathcal{A}$ can corrupt any party except the orchestrator, and then obtains their γ-state. $\mathcal{A}$ controls corrupted parties and can track all changes in their γ-state unless that is affected by the δ-state, in which case $\mathcal{A}$ must reveal that state too. We model corruption using a $\mathcal{O}_{\mathbf{Cor}}$ oracle, which logs the corrupted party u and u's current phase t_u with and event $(\mathcal{L}_{\mathrm{Cor}}, t_u, u)$.

We claimed above that our notions are flexible in terms of how many handovers are allowed before KI must be achieved. We realize this by parametrizing SACh with constants n_A and n_B, which are the upper and lower bound on the *critical region*, used as follows (see Fig. 2). Corruption of any party is allowed from phase $t^* + n_A$ and onward. Any access point other than the source and target access points in phase t^* can be corrupted any time. Corruption of parties active in the test phase t^* is disallowed before phase $t^* + n_A$. While corrupted access points stay corrupt, the orchestrator and client can establish fresh δ-state with an uncorrupted access point based on their own γ-state, i.e., the channel globally self-heals in a sense. GM cannot capture this since it is a two-party channel with deterministic key evolution, so if one party is corrupted, the channel is forever compromised.

Formally, the P^{Cor} subpredicate captures this: if any party, taking part in the CE protocol execution in the critical region, is corrupted, that corruption happened after the critical region ended.

$\mathcal{A}$ may reveal δ-substate of access points and the client for phases up to and including phase $t^* - n_B$, and from phase $t^* + n_A$ and forward. The δ-substate equals $\perp$ for phases where a party have not (yet) derived a δ-substate. We model δ-substate reveal for a phase t using the $\mathcal{O}_{\mathbf{Rev}}$ oracle, which adds the tagged entry $(\mathcal{L}_{\text{Rev}}, t, t_u, u)$ to the log $\mathcal{L}$, where u is the party and t_u is the phase that u is actually in when the reveal query is made. $\mathcal{O}_{\mathbf{Rev}}$ stores all prior δ-states to handle queries for them.

Formally, the P^{Rev} subpredicate captures that $\mathcal{L}$ contains no reveal events for a phase inside the critical region, and that the revealed party indeed had progressed to the revealed phase when the $\mathcal{O}_{\mathbf{Rev}}$ query was made. The latter is required, because $\mathcal{A}$ may ask to reveal a future phase or a past phase of an access point that was not the controlling access point for that phase.

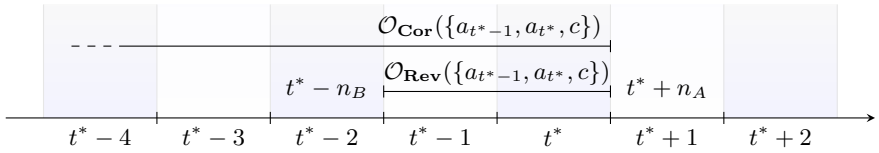

Fig. 2. Disallowed key exposures on a_{t^*-1}, a_{t^*} and the client c for test phase t^* with $n_B = 2$ and $n_A = 1$. Access points a_{t^*-1} and a_{t^*}, and the client can be corrupted earliest when phase t^* completes. Their δ-states cannot be revealed during phases $t^* - 1$ and t^*, but can be revealed afterwards. Other access points can be corrupted and revealed at any time, so the channel has a form of global self-healing capability.

Ciphertext Synchronization. Beginning with Bellare et al. [8], the standard CCA and integrity games for stateful authenticated encryption are split into two stages. First, $\mathcal{A}$ may benignly forward outputs from $\mathcal{O}_{\mathsf{Send}}$ (in their case represented by a LoR oracle) to $\mathcal{O}_{\mathsf{Recv}}$ as is, to progress the oracles' states. During this stage, $\mathcal{O}_{\mathsf{Recv}}$ suppresses its output. Inputs to $\mathcal{O}_{\mathsf{Recv}}$ are said to be *in sync* with outputs from $\mathcal{O}_{\mathsf{Send}}$. The second stage starts when $\mathcal{A}$ first invokes $\mathcal{O}_{\mathsf{Recv}}$ with a ciphertext that is out of sequence or is not produced by an invocation of $\mathcal{O}_{\mathsf{Send}}$. From then on, $\mathcal{O}_{\mathsf{Recv}}$ no longer suppresses output.

GM [32] extends this idea to multi-key channels. Its security experiments include two additional oracles $\mathcal{O}_{\mathbf{Rev}}$ and $\mathcal{O}_{\mathbf{Cor}}$. Through these, $\mathcal{A}$ may reveal phase-keys (corresponding to our δ-state) of other phases than the current one, and may corrupt one or both parties. To avoid trivial attacks, GM considers ciphertexts to be in sync *even* when $\mathcal{A}$ has revealed a phase key and thus clearly can inject ciphertexts that will be accepted by $\mathcal{O}_{\mathsf{Recv}}$ during that phase. However, secure GM schemes must detect these injections when the receiver progresses to the next (uncompromised) phase. Hence synchronization is considered lost if the receiver progress to the next phase without having received all ciphertexts sent in the previous phase. We adopt this approach and extend it.

A δ-channel is bidirectional, so loss of synchronization in one direction may affect the δ-substate used to process messages in the opposite direction [41]. Hence, if a receiving party loses synchronization, so may the other party. This is reflected in Fig. 3 by that **Snd** will no longer add ciphertexts to the validity-table C_u; hence these ciphertext will be considered out of sync by the receiver.

For their CCA notion, Bellare et al. [8] and GM deem synchronization lost when an invalid ciphertext *is input* to the receiver. Marson and Poettering [41], in contrast, deem synchronization lost when the receiver *accepts* the invalid ciphertext, i.e., a failed decryption does count as a received message (see line 44 of Figure 4 in [41]). Receive algorithms rejecting ciphertext adds a strong sense of integrity to the confidentiality notion, which blurs the distinction between the notions, so we follow the first approach.

3.2 Confidentiality

We give CPA and CCA experiments for confidentiality (see Fig. 3). We minimize the textual differences between the two definitions. Therefore, there are superfluous lines in the CPA definition. For instance, variables $sync_c$ and $sync_a$ are never re-assigned after initialization and can be ignored.

Secure channels commonly use LoR CPA and CCA, but we use the RoR plaintext variants since there is no natural way to allow $\mathcal{A}$ to provide a left and a right CE message to be sent over the δ-channel. We do not use the notion indistinguishable from random bit-strings (IND$) because we want to allow ciphertexts to contain metadata, e.g., sequence numbers. When they do, they are easily distinguishable from random strings.

The experiments allow CE protocols to make use of secure channels between parties in the anycast system S, e.g., as instantiated by IPsec/TLS in 4G/5G. The $\mathcal{I}_{\mathsf{Next}}$ algorithm implements operations on these channels that S perform after a CE protocol complete.

Experiment Parametrization. A single experiment, parametrized by ATK $\in \{\text{CPA}, \text{CCA}\}$, captures both CPA and CCA. When ATK = CCA, $\mathcal{A}$ can invoke $\mathcal{O}_{\mathsf{Recv}}$ with arbitrary ciphertexts. When ATK = CPA, $\mathcal{A}$ can only invoke $\mathcal{O}_{\mathsf{Recv}}$ with ciphertexts generated by $\mathcal{O}_{\mathsf{Send}}$, and the output of $\mathcal{O}_{\mathsf{Recv}}$ is then suppressed. $\mathcal{A}$ is given access to $\mathcal{O}_{\mathsf{Recv}}$ in the CPA game for two reasons. First, reception of messages may affect the δ-state for sending in bidirectional channels, and second, it enables fast-forwarding the δ-state to a potentially vulnerable state.

Emphasizing the similarities between the confidentiality and the integrity experiments, we introduce a technical parameter $\mathcal{D}$ for the challenge-bit sample-domain. For the confidentiality experiments $\mathcal{D} = \{0, 1\}$. For integrity experiments $\mathcal{D} = \{0\}$, resulting in the experiments only encrypting real messages.

We call our confidentiality notions SACh-IND-CPA and SACh-IND-CCA. Oracles type check their inputs and reject invocations with incorrect argument types. All variables are initialized to $\perp$ at the start of the experiment.

Definition 5 (SACh-IND-ATK Security). *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Next})$ *be a SACh (Definition 3), and let* $\mathbf{Exp}^{\text{sach-ind-atk}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}(\lambda)$ *be the security experiment defined in Fig. 3, parametrized by* $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$ *for an adversary* $\mathcal{A}$ *and challenge domain* $\mathcal{D} = \{0, 1\}$*. When* $\text{ATK} = \text{CCA}$ *boxed code is included in* $\mathbf{Exp}^{\text{sach-ind-atk}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}(\lambda)$*, but not shaded code. When* $\text{ATK} = \text{CPA}$ *the opposite holds. Let* $\mathcal{O}_{\text{IND}} = \{\mathcal{O}_{\mathsf{Send}}, \mathcal{O}_{\mathsf{Recv}}, \mathcal{O}_{\mathsf{NextInit}}, \mathcal{O}_{\mathsf{Next}}, \mathcal{O}_{\mathbf{Cor}}, \mathcal{O}_{\mathbf{Rev}}\}$*.* $\mathcal{A}$*'s advantage for the experiment is defined as*

$$\mathbf{Adv}^{\text{sach-ind-atk}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = 2 \cdot \left| \Pr\left[\mathbf{Exp}^{\text{sach-ind-atk}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}(\lambda) = 1\right] - 1/2 \right|.$$

Ch *is said to be* SACh-IND-ATK*-secure, or* (n_B, n_A)-SACh-IND-ATK*-secure to emphasize the parameters, when* $\mathbf{Adv}^{\text{sach-ind-atk}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

The definition rejects constructions vulnerable to reflection attacks where $\mathcal{A}$ replays ciphertexts back to the party that generated them.

3.3 Integrity

We now define SACh-INT-PTXT and SACh-INT-CTXT integrity notions. The experiments are almost identical to $\mathbf{Exp}^{\text{sach-ind-cca}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}$ (see Fig. 3). The key difference is that $\mathcal{A}$'s goal in the integrity games is to force one party of the δ-channel out of sync and to make them accept a plaintext (ciphertext) for SACh-INT-PTXT (SACh-INT-CTXT) not generated in sync by the other party. Due to the similarity with $\mathbf{Exp}^{\text{sach-ind-cca}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}$, we only show the main experiments and the $\mathcal{O}_{\mathsf{Recv}}$ oracles in Fig. 4. Remaining oracles are identical to their counterparts in $\mathbf{Exp}^{\text{sach-ind-cca}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}$. The integrity experiments do not include the shaded code in 3. Therefore, no $\mathcal{L}_{\text{CPA}_\perp}$ events are generated and the subpredicate P^{CPA} is vacuously true.

Definition 6 (SACh-INT-ATK Security). *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Next})$ *be a SACh (Definition 3), and let* $\mathbf{Exp}^{\text{sach-int-atk}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}(\lambda)$ *be the security experiment defined in Fig. 4, parametrized by* $\text{ATK} \in \{\text{PTXT}, \text{CTXT}\}$ *for an adversary* $\mathcal{A}$ *and challenge domain* $\mathcal{D} = \{0\}$*. The parameter* ATK *decides whether* $\mathcal{O}^{\text{PTXT}}_{\mathsf{Recv}}$ *or* $\mathcal{O}^{\text{CTXT}}_{\mathsf{Recv}}$ *is available to* $\mathcal{A}$*. Let* $\mathcal{O}_{\text{INT}} = \{\mathcal{O}_{\mathsf{Send}}, \mathcal{O}^{\text{ATK}}_{\mathsf{Recv}}, \mathcal{O}_{\mathsf{NextInit}}, \mathcal{O}_{\mathsf{Next}}, \mathcal{O}_{\mathbf{Cor}}, \mathcal{O}_{\mathbf{Rev}}\}$*, where* $\mathcal{O}_{\mathsf{Send}}$*,* $\mathcal{O}_{\mathsf{NextInit}}$*,* $\mathcal{O}_{\mathsf{Next}}$*,* $\mathcal{O}_{\mathbf{Cor}}$ *and* $\mathcal{O}_{\mathbf{Rev}}$ *are as defined in Fig. 3.* $\mathcal{A}$*'s advantage for the experiment is defined as*

$$\mathbf{Adv}^{\text{sach-int-atk}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = \Pr\left[\mathbf{Exp}^{\text{sach-int-atk}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}(\lambda) = 1\right].$$

Ch *is said to be* SACh-INT-ATK*-secure, or* (n_B, n_A)-SACh-INT-ATK*-secure to emphasize the parameters, when* $\mathbf{Adv}^{\text{sach-int-atk}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

3.4 Generic Composition

Authenticated Encryption (IND-CPA + INT-CTXT) is today accepted as the appropriate security notion for symmetric-key schemes, partially because, if appropriately defined, it implies IND-CCA security [14,9]. We now proceed to

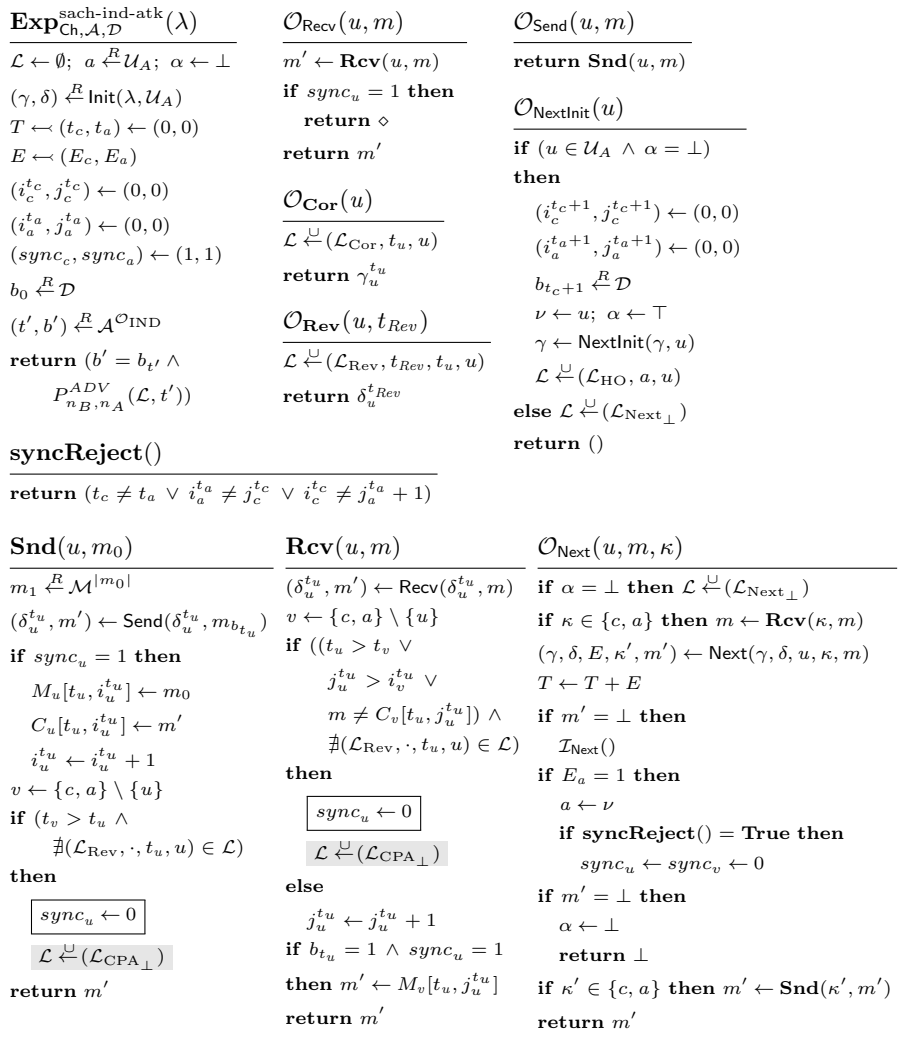

Fig. 3. SACh-IND-CPA and SACh-IND-CCA experiments for anycast channels. Boxed code is only used for SACh-IND-CCA. Shaded code is only used for SACh-IND-CPA.

show the corresponding relation between SACh-IND-CPA, SACh-IND-CCA and SACh-INT-CTXT. We assume that the single error Recv-algorithm can return is $\perp$, and we are hence in a single-error setting [14].

Theorem 1 (Generic Composition). *An* (n_B, n_A)-SACh-IND-CPA-*secure and* (n_B, n_A)-SACh-INT-CTXT-*secure SACh* Ch *is* (n_B, n_A)-SACh-IND-CCA-*secure. Formally, for every* (n_B, n_A)-SACh-IND-CCA *adversary* $\mathcal{A}$ *against* Ch *there are two adversaries, an* (n_B, n_A)-SACh-INT-CTXT *adversary* $\mathcal{B}_1$ *and an*

(n_B, n_A)-SACh-IND-CPA *adversary* $\mathcal{B}_2$ *against* Ch *s.t.*

$$\mathbf{Adv}^{\text{sach-ind-cca}}_{\mathsf{Ch},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{sach-int-ctxt}}_{\mathsf{Ch},\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\text{sach-ind-cpa}}_{\mathsf{Ch},\mathcal{B}_2}(\lambda)$$

The proof of Theorem 1 is provided in Appendix A.1.

$\mathbf{Exp}^{\text{int-atk}}_{\mathsf{Ch},\mathcal{A},\mathcal{D}}(\lambda)$

$\mathcal{L} \leftarrow \emptyset;\ a \xleftarrow{R} \mathcal{U}_A;\ t' \leftarrow \bot;\ \alpha \leftarrow \bot$
$(\gamma, \delta) \xleftarrow{R} \mathsf{Init}(\lambda, \mathcal{U}_A)$
$T \leftarrow\!\!\prec (t_c, t_a) \leftarrow (0,0)$
$E \leftarrow\!\!\prec (E_c, E_a)$
$(i_c^{t_c}, j_c^{t_c}) \leftarrow (0,0)$
$(i_a^{t_a}, j_a^{t_a}) \leftarrow (0,0)$
$(sync_c, sync_a) \leftarrow (0,0)$
$b_0 \xleftarrow{R} \mathcal{D}$
$win \leftarrow 0;\ \mathcal{A}^{\mathcal{O}_{\text{INT}}}$
return $win = 1 \wedge P^{ADV}_{n_B,n_A}(\mathcal{L}, t')$

$\mathcal{O}^{\text{ATK}}_{\mathsf{Recv}}(u, m)$

$m' \leftarrow \mathbf{Rcv}^{\text{ATK}}(u, m)$
if $(sync_u = 0 \wedge m' \neq \bot)$ **then**
  $win \leftarrow 1$
  if $t' = \bot$ **then** $t' \leftarrow t_u$
return $(m' = \bot)$

$\mathbf{Rcv}^{\text{CTXT}}(u, m)$

return $\mathbf{Rcv}(u, m)$

$\mathbf{Rcv}^{\text{PTXT}}(u, m)$

$(\delta_u^{t_u}, m') \leftarrow \mathsf{Recv}(\delta_u^{t_u}, m)$
$v \leftarrow \{c, a\} \setminus \{u\}$
if $(t_u > t_v \vee j_u^{t_u} > i_v^{t_v} \vee m \neq M_v[t_u, j_u^{t_u}])$ **then**
  $sync_u \leftarrow 0$
else
  $j_u^{t_u} \leftarrow j_u^{t_u} + 1$
if $b_{t_u} = 1 \wedge sync_u = 1$ **then**
  $m' \leftarrow M_v[t_u, j_u^{t_u}]$
return m'

Fig. 4. SACh-INT-PTXT and SACh-INT-CTXT experiments. Oracles and functions not defined here are as defined in the SACh-IND-CCA experiment (see Fig. 3). Shading in this figure indicates differences compared to the SACh-IND-CCA experiment.

4 Relations to Other Secure Channel Notions

The SACh syntax and the multi-key channel syntax [32] differ. Nonetheless, SACh can essentially simulate multi-key channels by restricting its functionality as follows. Let the anycast system consist of a single party taking on both the access point and the orchestrator roles, and let the CE protocol be deterministic. Since the orchestrator and the access point are collocated CE messages do not traverse other untrusted public channels than the δ-channel.

By additionally removing the $\mathsf{NextInit}$ and Next algorithms, phase progression is impossible and the functionality coincides with stateful encryption schemes [8].

If there is a single access point, but the orchestrator is a separate party, the impossibility result for key insulation [6] applies as is. Once the access point is compromised the secure channel to the orchestrator cannot be re-established since the adversary knows the access point's full γ-state. With several access points in the anycast system, each uncompromised access point has a separate secure channel to the orchestrator, which can be used to re-establish δ-channel security for future phases.

5 Construction of 4G and 5G Wireless Link

We now give a model of the 4G/5G wireless link including handovers and prove its security in the SACh meaning. The model closely follows the cryptographic core of 5G Xn handovers, which is also the core of 4G X2 handovers, and we call it XnCh. We need to make some improvements to the core to achieve forward security, which is necessary for SACh security. These improvements are computational, and do not require changes to architecture or distribution of shared secrets. We do not model S1/N2 handover of 4G/5G.

We first describe a model, XnCh^*, without our enhancements and which exposes the lack of forward security in 4G/5G. We then give our modifications and prove that, with these, the model is secure in the SACh sense.

XnCh^* Structure and State. Let S be a mobile network in the form of a SACh, consisting of access points $\mathcal{U}_A$ and all other functions collapsed into a single orchestrator o Fig. 6. All key spaces equal $\{0,1\}^\lambda$. Message spaces and ciphertext spaces equal $\{0,1\}^*$. Let $\mathsf{F}_C\colon\ \{0,1\}^\lambda \times \{0,1\}^* \longrightarrow \{0,1\}^\lambda$, be a domain-separated PRF. The domain-separation constant C is a fixed-length injective integer encoding of C that is prepended to the arbitrary length argument of F_C. When C is irrelevant we write F. We discard parenthesis of tuples input to F.

Between each pair of access points, and between each access point and o, there are secure private channels in both directions, marked XnIP in Fig. 6. 4G/5G instantiate these channels with IPsec/TLS. XnCh^* assumes that they are ideal, but that the algorithm $\mathcal{I}_{\mathsf{Next}}$ does nothing. States for sending messages from $u \in S$ to $v \in S$ over a secure infrastructure channel is denoted $\mathcal{I}^{u,v}_{\mathsf{st_s}}$ for the sender state and $\mathcal{I}^{u,v}_{\mathsf{st_r}}$ for the receiver state. Using these states, the algorithms $\mathcal{I}_{\mathsf{Tx}}$ and $\mathcal{I}_{\mathsf{Rx}}$ sends and receives, respectively, messages over the channels.

We assume that the SACh is initialized. This means that, on top of all infrastructure protection states being initialized, the orchestrator o and the client c share a key $\mathsf{K_f}$ and a key counter cnt initialized to 0; the initial access point a_s and c share a δ-channel key k for the current phase t. In addition, o has computed a key $\mathsf{nk} = \mathsf{F}_1(\mathsf{K_f}, cnt)$, and sent this to a_s.

The wireless link, i.e., the δ-channel, between the client and a_s is in 4G/5G secured by the Packet Data Convergence Protocol (PDCP). It carries application data and control signaling, including CE messages. 4G PDCP and 5G PDCP are similar enough to both fit our single δ-channel model PDCP_δ, which is a bidirectional stateful AEAD. A δ-substate $(\mathsf{id}, \mathsf{k}, \mathsf{sq}_\downarrow, \mathsf{sq}_\uparrow, \mathsf{sq}_{\mathsf{fin}})$ consists of the party's identity id, the encryption key k, the sequence number for the next sent $(\mathsf{sq}_\downarrow)$ and received $(\mathsf{sq}_\uparrow)$ message, and a variable $\mathsf{sq}_{\mathsf{fin}}$ for handling phase progressions.

The global γ-state for S is $\gamma = (\gamma_o, \gamma_c, \{\gamma_a\}_{a \in \mathcal{U}_A})$. The orchestrator's γ-substate γ_o contains the key $\mathsf{K_f}$, the counter cnt, and infrastructure protection states $\mathcal{I}^{o,a}_{\mathsf{st_s}}$ and $\mathcal{I}^{a,o}_{\mathsf{st_r}}$ for each $a \in \mathcal{U}_A$. The client's γ-substate γ_c contains the key $\mathsf{K_f}$ and corresponding counter cnt. The γ-substate of an access point u contains the infrastructure protection states $\mathcal{I}^{u,a}_{\mathsf{st_s}}$ and $\mathcal{I}^{a,u}_{\mathsf{st_r}}$ for each $a \in S$.

We now describe the δ-channel and the CE protocol XnCE in more detail.

5.1 δ-channel PDCP_δ

Our bidirectional stateful AEAD-based model PDCP_δ of PDCP is given in Fig. 5. PDCP provides authenticated encryption with associated data, but does not follow the established AEAD syntax [44]; our model on the other hand does. Integrity protected message sequence numbers, $\mathsf{sq}_\downarrow$ for sent messages and $\mathsf{sq}_\uparrow$ for received messages, ensure detection of out of order delivery or dropped messages. To counter truncation attacks (see Section 5.3), we chain the last sequence numbers of each phase into the variable $\mathsf{sq}_{\mathsf{fin}}$, which is input as AEAD associated data (ad) in the next phase, c.f. [32]. We add also the sender's identity to ad to avoid collisions in $\mathsf{sq}_{\mathsf{fin}}$ between the two channel directions. Sequence numbers are insufficient to fully counter reordering and packet-drop attacks; reflection attacks are still possible since the δ-channel is bidirectional. PDCP counters this by inputting the channel direction as associated data to the AEAD. We instead, similarly to how we handle $\mathsf{sq}_{\mathsf{fin}}$, input the sender's identity, which necessarily is different for each direction. Doing so avoids additional model variables. The sequence numbers also serve as nonces (nc) for the AEAD. Again, we add the sender identity to resolve collisions.

PDCP tolerates message loss. To cope with this, messages are accompanied by the lower order bits of the sequence numbers. The receiver estimates the higher order bits. Analysis of such constructions requires special care [27]. Our PDCP_δ model instead guarantees the stronger, but less flexible, security notion requiring all messages to be securely received. The mechanism used is standard: parties only accept the next expected sequence number, which is added to the nonce nc. If more than $\mathsf{maxMsg} = 2^\lambda$ messages are sent in one direction in one phase, we abort.

$\mathsf{pdcpInit}(\mathsf{k})$	$\mathsf{pdcpSend}(\delta_u, m)$	$\mathsf{pdcpRecv}(\delta_u, m)$
$\delta_{\mathsf{c}} \leftarrow (\mathsf{c}, \mathsf{k}, 0, 0, \emptyset)$	**assert** $\delta_u \neq \bot$	**assert** $\delta_u \neq \bot$
$\delta_{\mathsf{ap}} \leftarrow (\mathsf{ap}, \mathsf{k}, 0, 0, \emptyset)$	$(\mathsf{id}, \mathsf{k}, \mathsf{sq}_\downarrow, \mathsf{sq}_\uparrow, \mathsf{sq}_{\mathsf{fin}}) \leftarrow \delta_u$	$(\mathsf{id}, \mathsf{k}, \mathsf{sq}_\downarrow, \mathsf{sq}_\uparrow, \mathsf{sq}_{\mathsf{fin}}) \leftarrow \delta_u$
return $(\delta_{\mathsf{c}}, \delta_{\mathsf{ap}})$	**assert** $\mathsf{sq}_\downarrow < \mathsf{maxMsg}$	**assert** $\mathsf{sq}_\uparrow < \mathsf{maxMsg}$
	$nc \leftarrow (\mathsf{id}, \mathsf{sq}_\downarrow)$	$v \leftarrow \{\mathsf{c}, \mathsf{ap}\} \setminus \{\mathsf{id}\}$
	$ad \leftarrow (\mathsf{id}, \mathsf{sq}_{\mathsf{fin}})$	$nc \leftarrow (v, \mathsf{sq}_\uparrow)$
	$m' \leftarrow \mathsf{AEnc}(\mathsf{k}, nc, ad, m)$	$ad \leftarrow (\mathsf{id}, \mathsf{sq}_{\mathsf{fin}})$
	$\delta_u \leftarrow (\mathsf{id}, \mathsf{k}, \mathsf{sq}_\downarrow + 1, \mathsf{sq}_\uparrow, \mathsf{sq}_{\mathsf{fin}})$	$m' \leftarrow \mathsf{ADec}(\mathsf{k}, nc, ad, m)$
	return (δ_u, m')	**if** $m' = \bot$ **then return** $(\bot, \bot)$
		$\delta_u \leftarrow (\mathsf{id}, \mathsf{k}, \mathsf{sq}_\downarrow, \mathsf{sq}_\uparrow + 1, \mathsf{sq}_{\mathsf{fin}})$
		return (δ_u, m')

Fig. 5. δ-channel model PDCP_δ of enhanced 4G and 5G PDCP.

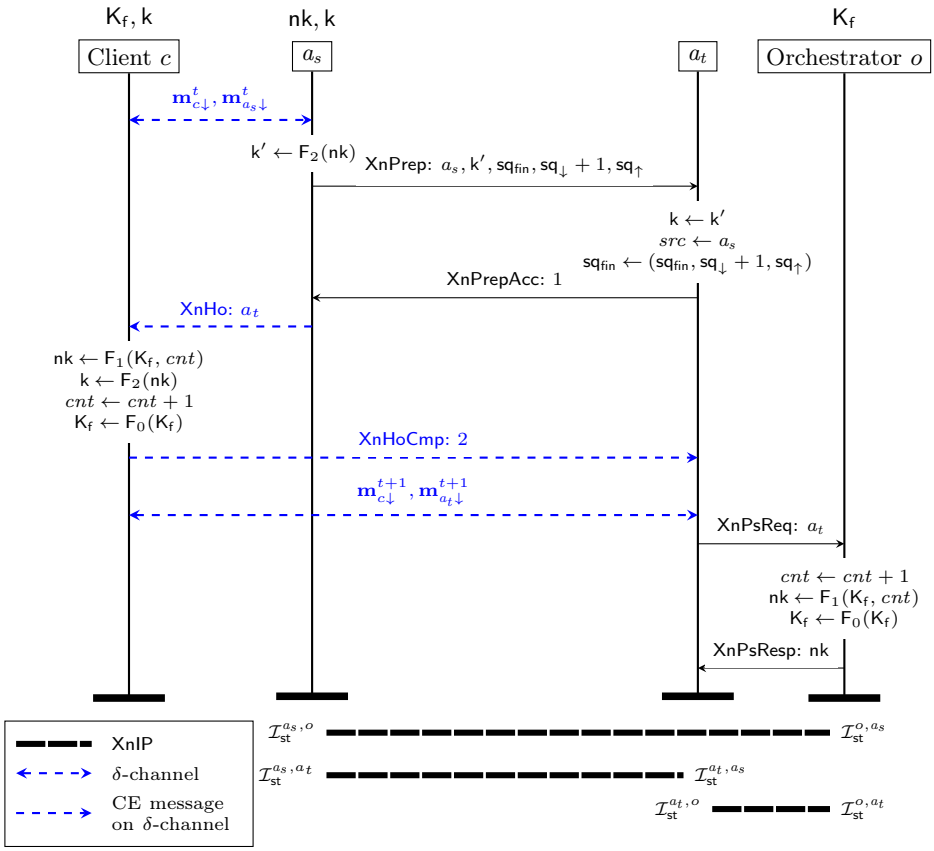

Fig. 6. Execution of a progression between phases t and $t+1$ for XnCh. Thick dashed lines indicate secure channels instantiated by the infrastructure protection layer XnIP. Dashed double-arrows represent the bidirectional δ-channel as instantiated by the PDCP protocol, protecting application message sequences between the client and an access point over the wireless link using the key k. The δ-channel also protects CE messages XnHo and XnHoCmp. Messages $\mathbf{m}^t_{c\downarrow}$ are sent from the client to access point a_s and $\mathbf{m}^t_{a_t\downarrow}$ are sent from a_s to client before the progression. After the progression, control of the δ-channel endpoint on the network side has been transferred from a_s to a_t. At this point the message sequences $\mathbf{m}^{t+1}_{c\downarrow}$ and $\mathbf{m}^{t+1}_{a_t\downarrow}$ are protected by the evolved δ-channel and simultaneously, the XnCE protocol prepares a_t to act as source access point in the next progression by involving the orchestrator.

5.2 CE Protocol XnCE

Our XnCE model for handover of an PDCP_δ endpoint is given in Fig. 7, and an overview of its use when combined with the infrastructure protection and PDCP_δ is shown in Fig. 6. XnCE uses the key $\mathsf{K_f}$, shared only by the client c and the orchestrator o, to establish a fresh k for PDCP_δ in the next phase.

4G/5G allow handovers from an access point back to itself and cancellation of partially completed handovers. We exclude both cases. [3]

The heart of CE protocols is the message dispatcher Next. It invokes an appropriate message handler and provides it with γ-state and δ-state corresponding to the party that is to execute the handler. The auxiliary algorithm $\mathcal{H}$ returns the handler to invoke for the given message type, and the party that should execute it. Using κ, indicating whether the message has passed through pdcpRecv, $\mathcal{H}$ aborts if an inbound CE message was either rejected by pdcpRecv, i.e., it equals $\perp$, or if the receiver is not equal to κ. Moreover, $\mathcal{H}(\kappa, \perp)$ returns ap, i.e., the access point currently controlling PDCP_δ, and the handler XnStart, which starts the CE protocol run. Message handlers receive the incoming message (an underscore _ indicates that they ignore it), process it and return another message (or $\perp$). They also return an indicator $\mathsf{E} = (E_c, E_a)$ of whether a or c has progressed to the next phase, and also an indication of whether the CE component shall process the returned message with pdcpSend or not. Handlers update the γ-state and δ-state as needed, and may process outgoing and incoming messages with the algorithms $\mathcal{I}_{\mathsf{Tx}}$ and $\mathcal{I}_{\mathsf{Rx}}$ when messages are to be sent/received over the secure infrastructure channels. Handlers tag their returned message with its type (c.f., message types shown over arrows in Fig. 6). The Next algorithm finally returns the updated state information and the output message. When **assert** fails in a handler, the handler returns the tuple $((0,0), \perp, \perp)$, representing $(\mathsf{E}, \mathsf{O_{ch}}, \mathsf{m}')$, i.e., the parties that progressed to the next phase, the direction of the δ-channel to use if any, and the next CE-message. When **assert** fails in NextInit it returns the input state γ as is, and when it fails in Next, it returns $(\gamma, \delta, \mathsf{E}, \mathsf{O_{ch}}, \mathsf{m}')$. The special identifier *self* refers to the identity of the party executing the handler.

On a high level, the CE protocol works as follows. The XnStart handler derives the key k', which will become the key k for the next phase, and includes this and the sequence number information of the current phase in a message for a_t. The XnPrep handler models reception of that message by a_t, which stores the information and returns an acknowledgment XnPrepAck. Upon reception of this, the XnPrepAck handler creates a security mode command message XnHo, which is sent to the client over the PDCP_δ channel *in the current phase*. The client processing this message, modeled by the XnHo handler, derives the nk key and, from that, the key k', *reconfigures* PDCP_δ *for the next phase* and sends the handover complete message XnHoCmp to a_t. Now, a_t processes that message using the XnHoCmp handler, which returns a path switch request message XnPsReq for the orchestrator. The XnPsReq handler receives the message, computes a new nk for a_t to use in the next handover and returns it in the XnPsResp message.

[3] The model would have to handle situations similar to lost messages if one end of the δ-channel considers the handover complete, but the other does not (see Section 3.1).

The XnPsResp handler stores nk and marks the handover complete by letting $\alpha \leftarrow \bot$. Every time a new nk is derived, a new base key $\mathsf{K_f}$ is derived from the previous $\mathsf{K_f}$ using a PRF F_0. While this is possible in 5G (not in 4G), it is not required. We require it for every handover to ensure forward security.

Correctness follows from the correctness of the AEAD, deterministic key derivations and straight forward but careful verification.

Init(λ)

$\mathsf{K_f} \xleftarrow{R} \{0,1\}^\lambda$
$(\gamma_o, \gamma_c, \{\gamma_a\}_{a \in \mathcal{U}_A}) \leftarrow$
  γ-Init($\mathsf{K_f}$)
$\mathsf{k} \leftarrow \gamma_c.\mathsf{nk}$
$(\delta_\mathsf{c}, \delta_\mathsf{ap}) \leftarrow$ pdcpInit(k)
return $(\delta_\mathsf{c}, \delta_\mathsf{ap})$

NextInit(γ, trg)

assert $\alpha \neq \top$
$\alpha \leftarrow \top$; $\gamma_a.\mathsf{trg} \leftarrow \mathsf{trg}$
return γ

Next($\gamma, \delta, \mathsf{u}, \kappa, \mathsf{m}$)

assert $\alpha \neq \bot$
$(\mathsf{p}, \mathsf{h}) \leftarrow \mathcal{H}(\kappa, \mathsf{m})$
assert $\mathsf{h} \neq \bot \wedge \mathsf{m} \neq \bot$
$(\mathsf{E}, \mathsf{O_{ch}}, \mathsf{m}') \leftarrow$
  $\mathsf{h}(\gamma_\mathsf{p}, \delta_\mathsf{p}, \mathsf{m})$
return $(\gamma, \delta, \mathsf{E}, \mathsf{O_{ch}}, \mathsf{m}')$

XnStart(γ_s, δ_s, _)

$\gamma_s.\mathsf{k}' \leftarrow \mathsf{F}_2(\gamma_s.\mathsf{nk})$
$\mathsf{m}' \leftarrow (self, \gamma_s.\mathsf{k}', \delta_s.\mathsf{sq_{fin}},$
  $\delta_s.\mathsf{sq}_\downarrow + 1, \delta_s.\mathsf{sq}_\uparrow)$
$\mathsf{m}' \leftarrow \mathcal{I}_{\mathsf{Tx}}(\mathcal{I}^{self,\gamma_s.\mathsf{trg}}_{\mathsf{st_s}}, \mathsf{m}')$
$\mathsf{m}' \leftarrow (\mathsf{T_{Prep}}, \mathsf{m}')$
return $((0,0), \bot, \mathsf{m}')$

XnHo($\gamma_s, \delta_s, \mathsf{m}$)

$trg \leftarrow \mathsf{m}$
$\delta_s.\mathsf{sq_{fin}} \leftarrow (\delta_s.\mathsf{sq_{fin}},$
  $\delta_s.\mathsf{sq}_\downarrow, \delta_s.\mathsf{sq}_\uparrow)$
$\mathsf{nk} \leftarrow \mathsf{F}_1(\gamma_s.\mathsf{K_f}, \gamma_s.cnt)$
$\delta_s.\mathsf{k} \leftarrow \mathsf{F}_2(\mathsf{nk})$
$\gamma_s.cnt \leftarrow \gamma_s.cnt + 1$
$\gamma_s.\mathsf{K_f} \leftarrow \mathsf{F}_0(\gamma_s.\mathsf{K_f})$
$\mathsf{m}' \leftarrow (\mathsf{T_{HoCmp}}, 2)$
return $((1,0), \mathsf{c}, \mathsf{m}')$

XnPrep($\gamma_s, \delta_s, \mathsf{m}$)

$\mathsf{m} \leftarrow \mathcal{I}_{\mathsf{Rx}}(\mathcal{I}^{self,-}_{\mathsf{st_r}}, \mathsf{m})$
$(src, \delta_s.\mathsf{k}, \delta_s.\mathsf{sq_{fin}},$
  $\delta_s.\mathsf{sq}_\downarrow, \delta_s.\mathsf{sq}_\uparrow) \leftarrow \mathsf{m}$
$\delta_s.\mathsf{sq_{fin}} \leftarrow (\delta_s.\mathsf{sq_{fin}},$
  $\delta_s.\mathsf{sq}_\downarrow, \delta_s.\mathsf{sq}_\uparrow)$
$\mathsf{m}' \leftarrow \mathcal{I}_{\mathsf{Tx}}(\mathcal{I}^{self,src}_{\mathsf{st_s}}, 1)$
$\mathsf{m}' \leftarrow (\mathsf{T_{PrepAcc}}, \mathsf{m}')$
return $((0,0), \bot, \mathsf{m}')$

XnPrepAcc($\gamma_s, \delta_s, \mathsf{m}$)

$\mathsf{m} \leftarrow \mathcal{I}_{\mathsf{Rx}}(\mathcal{I}^{self,-}_{\mathsf{st_r}}, \mathsf{m})$
$\mathsf{m}' \leftarrow (\mathsf{T_{Ho}}, \gamma_s.\mathsf{trg})$
return $((0,1), \mathsf{ap}, \mathsf{m}')$

γ-Init($\mathsf{K_f}$)

$\gamma_o.\mathsf{K_f} \leftarrow \gamma_c.\mathsf{K_f} \leftarrow \mathsf{K_f}$
$\gamma_a.\mathsf{nk} \leftarrow \mathsf{F}_1(\mathsf{K_f}, 0)$
$\gamma_\mathsf{o}.cnt \leftarrow \gamma_\mathsf{c}.cnt \leftarrow 0$
return γ

XnPsReq($\gamma_s, \delta_s, \mathsf{m}$)

$trg \leftarrow \mathcal{I}_{\mathsf{Rx}}(\mathcal{I}^{self,-}_{\mathsf{st_r}}, \mathsf{m})$
$\gamma_s.cnt \leftarrow \gamma_s.cnt + 1$
$\mathsf{nk} \leftarrow \mathsf{F}_1(\gamma_s.\mathsf{K_f}, \gamma_s.cnt)$
$\gamma_s.\mathsf{K_f} \leftarrow \mathsf{F}_0(\gamma_s.\mathsf{K_f})$
$\mathsf{m}' \leftarrow \mathcal{I}_{\mathsf{Tx}}(\mathcal{I}^{self,trg}_{\mathsf{st_s}}, \mathsf{nk})$
$\mathsf{m}' \leftarrow (\mathsf{T_{PsResp}}, \mathsf{m}')$
return $((0,0), \bot, \mathsf{m}')$

XnPsResp($\gamma_s, \delta_s, \mathsf{m}$)

$\gamma_s.\mathsf{nk} \leftarrow \mathcal{I}_{\mathsf{Rx}}(\mathcal{I}^{self,-}_{\mathsf{st_r}}, \mathsf{m})$
$\alpha \leftarrow \bot$
return $((0,0), \bot, \bot)$

XnHoCmp($\gamma_s, \delta_s, \mathsf{m}$)

$\mathsf{m}' \leftarrow \mathcal{I}_{\mathsf{Tx}}(\mathcal{I}^{self,\circ}_{\mathsf{st_s}}, self)$
$\mathsf{m}' \leftarrow (\mathsf{T_{PsReq}}, \mathsf{m}')$
return $((0,0), \bot, \mathsf{m}')$

$\mathcal{H}(\kappa, m)$ returns $(\mathsf{p}, \bot)$ for messages XnHo and XnHoCmp unless κ equals the expected δ-channel (c.f. Fig. 6).

Fig. 7. The XnCE CE protocol and main SACh model.

We have mentioned deviations from the standard throughout. Some parts, not necessary to capture the essence of an endpoint-transferring channel have also been excluded. As a result, XnCE may seem over-engineered. For example, key k' is not strictly needed. However, 4G/5G allow preparation of many potential target access points, ultimately one being chosen. Therefore, the standard include the identity of the target access point in the derivation of k', making it unique per access point. We leave it as future work to extend the model in this direction.

Lack of Forward Security in 4G and 5G, and a Fix. The SACh we have just defined, XnCh^* is fairly close to the 3GPP standard, and sufficient to illustrate the lack of forward security. Consider Fig. 6. Suppose $\mathcal{A}$ records message

XnPrep in phase t, and in phase $t+1$ also some δ-channel traffic, protected by the key k' carried in XnPrep. Unless the IPsec/TLS tunnel is re-keyed after each handover, $\mathcal{A}$ can in a later phase, e.g., $t+2$, corrupt access point a_t, obtain its IPsec/TLS keys (which are still the same as in phase t), and decrypt the stored XnPrep message to obtain the key k' inside. Using k', $\mathcal{A}$ can now decrypt the δ-channel traffic recorded in phase $t+1$ even though access point a_{t+1} and c have deleted k'. Thus, the 4G/5G δ-channel encryption is not forward secure. A similar attack is possible by recording the XnPsResp message and obtaining the key nk. These attacks work on any deployment of the standard if the adversary can break in to the access point's secure execution environment.

To counter this, we propose a new forward secure encryption layer, XnIP, for CE messages as a replacement for IPsec/TLS for this purpose. XnIP is independent of the transport and tighter coupled to CE protocol. It allows efficient key evolution at every handover without additional messages sent. Note, IPsec/TLS may still be needed for other functionalities in the mobile network. We believe adding such a layer to future 3GPP standards should be the long-term goal. However, a more pragmatic approach, working already with current standards, may be to re-establish IPsec/TLS security associations sufficiently often.

Remark The 3GPP specifications use different definitions of forward- and backward security (see Section 3 of TS 33.501 [1]) compared to the usual [12]. Paraphrased, 4G/5G define forward security by *an access point knowing a (δ-channel) key shall not be able to derive any future (δ-channel) keys*. Backward security is defined analogously. There are four noteworthy differences between the 4G/5G terms and the usual interpretations: 1) the 4G/5G terms define a property of a fail-condition (if an attack happens, that should not help in future attacks), whereas the usual definitions capture a positive property of derived keys, which subsumes the 4G/5G terms (if the derived key is secure, it will remain secure); 2) the 4G/5G terms forward and backward have opposite "directions" compared to what is usual; 3) the 4G/5G terms only relate one key to another and ignore other secret information, specifically IPsec/TLS keys are left out; and 4) the 4G/5G terms only consider compromise of the access point, not the client, or (partial) key recovery attacks from ciphertext. The second difference may be a consequence of the first: when considering a fail-condition that should not propagate *forward* into the future, it is natural to connect the term to the forward direction. However, the usual interpretation of forward security is that if long-term security state is compromised, *past* keys (and encrypted traffic) should still be secure. Peltonen et al. [43] noted that the 4G/5G term forward security is temporally reversed compared to *(perfect) forward secrecy*, but we stress that it is even reversed compared to its much closer related namesake *forward security* by Abdalla and Bellare [2]. Since the 4G/5G definitions consider a specific fail-condition and disregard IPsec/TLS, they do not capture the attack described above. It seems as if the intention was to protect against physical break-in to the access points (see Section 7.4.13.2 of TR 33.821 [1]), so arguably they should.

We now present a modification in the form of a new component XnIP, providing forward secure infrastructure protection. We call the resulting SACh XnCh.

5.3 Infrastructure Protection XnIP

We first define the syntax, and then the its two security notions. Intuitively, the infrastructure protection is a set of secure channels connecting parties in S. These channels are built from an authenticated encryption scheme $\mathsf{AE} = (\mathsf{AEnc}, \mathsf{ADec})$ and the PRF F. Parties involved in handovers use F to evolve their AE encryption keys and ensure forward security for sent XnCE messages.

Definition 7 (Infrastructure Protection Syntax). *Let S be an anycast system. Each pair $(u, v) \in S \times S$ is connected by two unidirectional communication-channels, one in each direction.* Infrastructure Protection XnIP *for S is a tuple of algorithms $(\mathcal{I}_{\mathsf{Init}}, \mathcal{I}_{\mathsf{Tx}}, \mathcal{I}_{\mathsf{Rx}}, \mathcal{I}_{\mathsf{Next}})$. A channel from u to v, denoted (u, v), is associated with two states from a state space $\mathcal{I}_{\mathsf{st}}$. The first state is the sender state $\mathcal{I}_{\mathsf{st_s}}^{u,v}$, maintained by u and input to the sending algorithm $\mathcal{I}_{\mathsf{Tx}}$. The second is the receiver state $\mathcal{I}_{\mathsf{st_r}}^{u,v}$, maintained by v and input to the receiving algorithm $\mathcal{I}_{\mathsf{Rx}}$. The syntax for XnIP is given by the following algorithms.*

- $\mathcal{I}_{\mathsf{Init}} \colon \mathbb{N} \longrightarrow \mathcal{I}_{\mathsf{st}}^*$ *Given the security parameter λ as input, it returns two paired states for each channel in $\mathcal{I}$.*
- $\mathcal{I}_{\mathsf{Tx}} \colon \mathcal{I}_{\mathsf{st}} \times \{0,1\}^* \longrightarrow \mathcal{I}_{\mathsf{st}} \times \{0,1\}^*$ *Given a state and a message, it returns an updated state and a protected message.*
- $\mathcal{I}_{\mathsf{Rx}} \colon \mathcal{I}_{\mathsf{st}} \times \{0,1\}^* \longrightarrow (\mathcal{I}_{\mathsf{st}} \times \{0,1\}^*) \cup \{(\bot, \bot)\}$ *Given a state and a protected message, it returns an updated state and a message. On failure, the special pair $(\bot, \bot)$ is returned.*
- $\mathcal{I}_{\mathsf{Next}} \colon \mathcal{I}_{\mathsf{st}} \times \mathcal{I}_{\mathsf{st}} \longrightarrow \mathcal{I}_{\mathsf{st}} \times \mathcal{I}_{\mathsf{st}}$ *Given two states, it progresses them to the next phase and returns the updated states.*

An infrastructure protection scheme is correct if for every channel (u, v) and message m it holds that $\mathcal{I}_{\mathsf{Rx}}(\mathcal{I}_{\mathsf{st_r}}^{u,v}, \mathcal{I}_{\mathsf{Tx}}(\mathcal{I}_{\mathsf{st_s}}^{u,v}, m)) = m$.

Definition 8 (Infrastructure Protection I-ATK Security). *Let S be an anycast system and let XnIP be its infrastructure protection. Let $\mathbf{Exp}^{\text{i-auth}}_{\mathsf{XnIP},\mathcal{A},\{0\}}(\lambda)$ and $\mathbf{Exp}^{\text{i-cpa}}_{\mathsf{XnIP},\mathcal{A},\{0,1\}}(\lambda)$ be security experiments as defined in Fig. 9 for attacks* ATK = AUTH *and* ATK = CPA*, and for challenge domains $\mathcal{D} = \{0\}$ and $\mathcal{D} = \{0,1\}$ respectively. Let $\mathcal{O}_{\mathcal{I}} = \{\mathcal{O}_{\mathcal{I}_{\mathsf{Tx}}}, \mathcal{O}_{\mathcal{I}_{\mathsf{Rx}}}, \mathcal{O}_{\mathcal{I}_{\mathsf{Next}}}, \mathcal{O}_{\mathcal{I}_{\mathsf{Expose}}}\}$, where $\mathcal{O}_{\mathcal{I}_{\mathsf{Tx}}}$, $\mathcal{O}_{\mathcal{I}_{\mathsf{Rx}}}$, $\mathcal{O}_{\mathcal{I}_{\mathsf{Next}}}$, $\mathcal{O}_{\mathcal{I}_{\mathsf{Expose}}}$ are defined in Fig. 9. An adversary $\mathcal{A}$'s advantages for these experiments are defined as*

$$\mathbf{Adv}^{\text{i-auth}}_{\mathsf{XnIP},\mathcal{A}}(\lambda) = \Pr\left[\mathbf{Exp}^{\text{i-auth}}_{\mathsf{XnIP},\mathcal{A},\{0\}}(\lambda) = 1\right]$$
$$\mathbf{Adv}^{\text{i-cpa}}_{\mathsf{XnIP},\mathcal{A}}(\lambda) = 2 \cdot \left|\Pr\left[\mathbf{Exp}^{\text{i-cpa}}_{\mathsf{XnIP},\mathcal{A},\{0,1\}}(\lambda) = 1\right] - 1/2\right|.$$

The intuition for the predicate $P^{\mathcal{I}}$ in the I-ATK definition in Fig. 9 is the same as for predicate P^{Cor} (see Section 3.1). Consequently, if $\mathcal{A}$ attacks phase t^*, it is disallowed to expose a_{t^*} or a_{t^*+1}.

$\mathcal{I}_{\mathsf{Init}}(\lambda)$

$\ell \leftarrow \emptyset$
$\forall (u, v) \in \mathcal{I} \times \mathcal{I}$ **do**
  $\mathsf{rik} \xleftarrow{R} \{0,1\}^{\lambda}$
  $\ell \xleftarrow{\cup} \mathsf{chInit}(\mathsf{rik}, u, v)$
return ℓ

$\mathsf{chInit}(\mathsf{rik}, u, v)$

$\mathsf{ik} \leftarrow \mathsf{F}_{\mathcal{I}}(1, \mathsf{rik})$
$\mathcal{I}^{u,v}_{\mathsf{sts}} \leftarrow (\mathsf{rik}, \mathsf{ik}, 0)$
$\mathcal{I}^{u,v}_{\mathsf{str}} \leftarrow (v, \mathsf{rik}, \mathsf{ik}, 0)$
return $\{\mathcal{I}^{u,v}_{\mathsf{sts}}, \mathcal{I}^{u,v}_{\mathsf{str}}\}$

$\mathcal{I}_{\mathsf{Tx}}(\mathcal{I}^{\mathsf{u},\mathsf{v}}_{\mathsf{sts}}, \mathsf{m})$

$(\mathsf{rik}, \mathsf{ik}, \mathsf{sq}_{\downarrow}) \leftarrow \mathcal{I}^{\mathsf{u},\mathsf{v}}_{\mathsf{sts}}$
$\mathsf{m}' \leftarrow \mathsf{AEnc}(\mathsf{ik}, \mathsf{sq}_{\downarrow}, \mathsf{m})$
$\mathsf{sq}'_{\downarrow} \leftarrow \mathsf{sq}_{\downarrow} + 1$
$\mathcal{I}^{\mathsf{u},\mathsf{v}\prime}_{\mathsf{sts}} \leftarrow (\mathsf{rik}, \mathsf{ik}, \mathsf{sq}'_{\downarrow})$
return $(\mathcal{I}^{\mathsf{u},\mathsf{v}\prime}_{\mathsf{sts}}, \mathsf{m}')$

$\mathcal{I}_{\mathsf{Rx}}(\mathcal{I}^{\mathsf{u},\mathsf{v}}_{\mathsf{str}}, \mathsf{m})$

assert $\mathcal{I}^{\mathsf{u},\mathsf{v}}_{\mathsf{str}} \neq \bot$
$(\mathsf{id}, \mathsf{rik}, \mathsf{ik}, \mathsf{sq}_{\uparrow}) \leftarrow \mathcal{I}^{\mathsf{u},\mathsf{v}}_{\mathsf{str}}$
$v \leftarrow \{\mathsf{u}, \mathsf{v}\} \setminus \{\mathsf{id}\}$
$\mathsf{m}' \leftarrow \mathsf{ADec}(\mathsf{ik}, \mathsf{sq}_{\uparrow}, \mathsf{m})$
if $\mathsf{m}' = \bot$ **then**
  return $(\bot, \bot)$
$\mathsf{sq}'_{\uparrow} \leftarrow \mathsf{sq}_{\uparrow} + 1$
$\mathcal{I}^{\mathsf{u},\mathsf{v}\prime}_{\mathsf{str}} \leftarrow (\mathsf{id}, \mathsf{rik}, \mathsf{ik}, \mathsf{sq}'_{\uparrow})$
return $(\mathcal{I}^{\mathsf{u},\mathsf{v}\prime}_{\mathsf{str}}, \mathsf{m}')$

$\mathcal{I}_{\mathsf{Next}}(\mathcal{I}^{u,v}_{\mathsf{sts}}, \mathcal{I}^{u,v}_{\mathsf{str}})$

$(\mathsf{rik}, \mathsf{ik}, \mathsf{sq}) \leftarrow \mathcal{I}^{u,\mathsf{v}}_{\mathsf{sts}}$
$\mathsf{rik}' \leftarrow \mathsf{F}_{\mathcal{I}}(0, \mathsf{rik})$
$\mathsf{ik}' \leftarrow \mathsf{F}_{\mathcal{I}}(1, \mathsf{rik}')$
$\mathcal{I}^{u,\mathsf{v}\prime}_{\mathsf{sts}} \leftarrow (\mathsf{rik}', \mathsf{ik}', 0)$
$\mathcal{I}^{u,\mathsf{v}\prime}_{\mathsf{str}} \leftarrow (\mathsf{id}, \mathsf{rik}', \mathsf{ik}', 0)$
return $(\mathcal{I}^{u,\mathsf{v}\prime}_{\mathsf{sts}}, \mathcal{I}^{u,\mathsf{v}\prime}_{\mathsf{str}})$

Fig. 8. Infrastructure protection construction.

XnIP, shown in Fig. 8, is similar in design to PDCP_{δ}. Algorithms $\mathcal{I}_{\mathsf{Tx}}$ and $\mathcal{I}_{\mathsf{Rx}}$ correspond to pdcpSend and pdcpRecv, respectively. They use increasing sequence numbers as nonces, and to ensure message order. Unlike the bidirectional PDCP_{δ}, XnIP uses pairs of unidirectional channels Channel initialization comprises sampling a root key rik and deriving an initial encryption key ik from rik. $\mathcal{I}_{\mathsf{Next}}$ evolves a channel to the next phase, derives a new rik from the previous one, and a new ik from the new rik. It sets sequence numbers to 0, and so prevents prefix-truncation attacks [32]. These attacks work when sequence numbers continue across phases. An adversary exposes the key of a phase t in phase t, injects k messages at the end of phase t, and then discards k messages at the start of phase $t+1$. The message loss in phase $t+1$ is undetected.

An exposed access point reveals all states it shares with other access points to the adversary. The corresponding states at those access points are also exposed.

Lemma 1 (XnIP Infrastructure Protection Security). *Let S be an anycast system. Let the infrastructure protection for S be as defined in Fig. 8, where $\mathsf{AE} = (\mathsf{AEnc}, \mathsf{ADec})$ is an* IND-CPA *and* AUTH *secure nonce based authenticated encryption scheme [45] operating under a randomly sampled key.* [4] *Let F be the domain-separated pseudorandom function employed by XnIP. Let* I-CPA *and* I-AUTH *be the security experiments defined in Fig. 9. Let n_{ph} be the number of phases appearing in the execution of the security experiment. Then, for every* I-CPA *adversary $\mathcal{A}_1$ against XnIP there are adversaries $\mathcal{B}_2$ and $\mathcal{B}_4$ s.t.*

$$\mathbf{Adv}^{\text{i-cpa}}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda) \leq n_{ph} \cdot \left(n_{ph} \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{B}_2}(\lambda) + 4 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},\mathcal{B}_4}(\lambda)\right).$$

For each I-AUTH *adversary $\mathcal{A}_2$ against XnIP there are adversaries $\mathcal{B}_7$ and $\mathcal{B}_9$ s.t.*

$$\mathbf{Adv}^{\text{i-auth}}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda) \leq n_{ph} \cdot \left(n_{ph} \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{B}_7}(\lambda) + 4 \cdot \mathbf{Adv}^{\text{auth}}_{\mathsf{AE},\mathcal{B}_9}(\lambda)\right).$$

[4] As in the rest of the paper, we use is the RoR *plaintext* variant of IND-CPA whereas Rogaway [45] uses RoR *ciphertext*, a.k.a. IND$.

$$P^{\mathcal{I}}(\mathcal{L}_E, \mathcal{L}_H) \triangleq \forall (t_E, u, v) \in \mathcal{L}_E.\ \forall (t_H, src, dst) \in \mathcal{L}_H.\ (t_E \leq t_H - n_B \ \wedge \ u, v \notin \{src, dst\}) \ \vee \ t_E \geq t_H + n_A.$$

$\mathbf{Exp}^{\text{i-atk}}_{\mathsf{XnIP},\mathcal{A},\mathcal{D}}(\lambda)$

$\mathcal{L}_E \leftarrow \emptyset$
$a \xleftarrow{R} \mathcal{I} \setminus \{o\}$
$(\mathcal{I}^{u,v}_{\mathsf{st}})^* \leftarrow \mathcal{I}_{\mathsf{Init}}(\lambda)$
$\forall (u, v) \in \mathcal{I} \times \mathcal{I}$ **do**
  $C_{u,v} \leftarrow [\,]$
  $b_{t,(u,v)} \xleftarrow{R} \mathcal{D}$
$\boxed{(t', ch', b') \xleftarrow{R} \mathcal{A}^{\mathcal{O}_{\mathcal{I}}}}$
$win \leftarrow 0$
$\mathcal{A}^{\mathcal{O}_{\mathcal{I}}}$
return (
  $\boxed{b' = b_{t',ch'}}$
  $\wedge\ win = 1$
  $\wedge\ P^{\mathcal{I}}(\mathcal{L}_E, \mathcal{L}_H)$)

$\mathcal{O}_{\mathcal{I}_{\mathsf{Tx}}}(u, v, m_0)$

$m_1 \xleftarrow{R} \{0,1\}^{|m_0|}$
$(\mathcal{I}^{u,v\prime}_{\mathsf{st_s}}, m') \leftarrow \mathcal{I}_{\mathsf{Tx}}(\mathcal{I}^{u,v}_{\mathsf{st_s}}, m_{b_t})$
$C_{u,v} \leftarrow C_{u,v} \| m'$
return m'

$\mathcal{O}_{\mathcal{I}_{\mathsf{Next}}}(v)$

$\mathcal{L}_H \xleftarrow{\cup} (t, a, v)$
$\ell \leftarrow \{(\mathsf{ap}, v), (v, \mathsf{ap}), (o, v), (v, o)\}$
$\forall (u, w) \in \ell$ **do**
  $b_{t,(u,w)} \xleftarrow{R} \mathcal{D}$
  $(\mathcal{I}^{u,w\prime}_{\mathsf{st_s}}, \mathcal{I}^{u,w\prime}_{\mathsf{st_r}}) \leftarrow \mathcal{I}_{\mathsf{Next}}(\mathcal{I}^{u,w}_{\mathsf{st_s}}, \mathcal{I}^{u,w}_{\mathsf{st_r}})$

$\mathcal{O}_{\mathcal{I}_{\mathsf{Expose}}}(u)$

$\ell \leftarrow \emptyset$
$\forall v \in \mathcal{I}$ **do**
  $\mathcal{L}_E \xleftarrow{\cup} (t, u, v)$
  $\ell \xleftarrow{\cup} \{\mathcal{I}^{u,v}_{\mathsf{st_s}}, \mathcal{I}^{u,v}_{\mathsf{st_r}}\}$
return ℓ

$\mathcal{O}_{\mathcal{I}_{\mathsf{Rx}}}(u, v, m)$

$(\mathcal{I}^{u,v\prime}_{\mathsf{st_r}}, m') \leftarrow \mathcal{I}_{\mathsf{Rx}}(\mathcal{I}^{u,v}_{\mathsf{st_r}}, m)$
$m_e \leftarrow \mathbf{head}(C_{u,v})$
if $m' \neq \perp \wedge (|C_{u,v}| = 0 \vee (|C_{u,v}| > 0 \wedge m \neq m_e))$ **then**
  $win \leftarrow 1$
  if $t' = \perp$ **then**
    $t' \leftarrow t_u$
return m'

Fig. 9. I-CPA and I-AUTH experiments for infrastructure protection. Shaded code, in particular the $\mathcal{O}_{\mathcal{I}_{\mathsf{Rx}}}$ oracle, is only available in the I-AUTH game. Boxed code is only available in the I-CPA game. $\mathcal{L}_E$ logs party-exposures and $\mathcal{L}_H$ logs handovers.

The proof of Lemma 1 is provided in Appendix A.2.

5.4 Security of the XnCh scheme

We arrive at the main theorems combining the components into a SACh. More precisely, XnCh meets the $(2,1)$-SACh-IND-CPA and $(2,1)$-SACh-INT-CTXT security notions, and thus meets the $(2,1)$-SACh-IND-CCA notion.

Theorem 2 (XnCh SACh-IND-CPA Security). *Let* XnCh *be the SACh defined by the* PDCP *δ-channel in Fig. 5, the infrastructure protection* XnIP *defined in Fig. 8, and the CE component* XnCE *defined in Fig. 6. Let* AEAD $=$ (AEADEnc, AEADDec) *be the AEAD employed by the* PDCP *δ-channel, capable of securely handling* maxMsg *messages. Let* F *the pseudorandom function employed by* XnCh*. Let n_{ph} be the number of phases occurring in the experiment* $\mathbf{Exp}^{\text{sach-ind-cpa}}_{\mathsf{XnCh},\mathcal{A},\{0,1\}}(\lambda)$*. For each $(2,1)$-SACh-IND-CPA adversary $\mathcal{A}$ against* XnCh *there are adversaries $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$ and $\mathcal{B}_5$ such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \leq n_{ph} \cdot \big(\mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_3}(\lambda) + n_{ph} \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{B}_4}(\lambda) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AEAD},\mathcal{B}_5}(\lambda)\big).$$

The proof of Theorem 2 is provided in Appendix A.3.

Theorem 3 (XnCh SACh-INT-CTXT Security). *Let* XnCh *be the SACh defined by the* PDCP *δ-channel in Fig. 5, the infrastructure protection* XnIP *defined in Fig. 8, and the CE component* XnCE *defined in Fig. 6. Let* $\mathsf{AEAD} = (\mathsf{AEADEnc}, \mathsf{AEADDec})$ *be the AEAD employed by the* PDCP *δ-channel, capable of securely handling* maxMsg *messages. Let* F *the pseudorandom function employed by* XnCh*. Let n_{ph} be the number of phases occurring in the experiment* $\mathbf{Exp}^{\text{sach-int-ctxt}}_{\mathsf{XnCh},\mathcal{A},\{0,1\}}(\lambda)$*. For each* $(2,1)$-SACh-INT-CTXT *adversary $\mathcal{A}$ against* XnCh *there are adversaries $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$ and $\mathcal{B}_5$ such that*

$$\begin{aligned}\mathbf{Adv}^{\text{sach-int-ctxt}}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \leq n_{ph} \cdot \big(&\mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_3}(\lambda) \\ &+ n_{ph} \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{B}_4}(\lambda) + \mathbf{Adv}^{\text{auth}}_{\mathsf{AEAD},\mathcal{B}_5}(\lambda)\big).\end{aligned}$$

The proof of Theorem 3 is provided in Appendix A.4.

This final corollary follows immediately from Theorems 1, 2 and 3.

Corollary 1 (XnCh SACh-IND-CCA Security). *The* SACh-IND-CCA *security of* XnCh *is bounded as follows. For each* SACh-IND-CCA *adversary $\mathcal{A}$ against* XnCh *there are adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathbf{Adv}^{\text{sach-ind-cca}}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{sach-int-ctxt}}_{\mathsf{XnCh},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\text{sach-ind-cpa}}_{\mathsf{XnCh},\mathcal{B}_1}(\lambda).$$

6 Conclusions

In this paper we initiated the study of secure anycast channels. We presented the first secure channel model featuring endpoint transfer between parties. We showed its practicality by constructing a model of 3GPP 4G/5G wireless link security and proving the model secure. Our model is rather close to the standards. We demonstrated a lack of forward security in 4G/5G, and proposed a correction, which suggests how future mobile networks can achieve key insulation, and compartmentalize damage to individual access points if they are compromised.

Acknowledgements This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

1. 3GPP/ETSI: 3rd Generation Partnership Project (3GPP): GSM, UMTS, LTE and 5G standards (2022), https://www.3gpp.org
2. Abdalla, M., Bellare, M.: Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In: Advances in Cryptology - ASIACRYPT. pp. 546–559. LNCS, Springer (2000)
3. Alnashwan, R., Gope, P., Dowling, B.: Privacy-aware Secure Region-based Handover for Small Cell Networks in 5G-enabled Mobile Communication. CoRR (2022)

4. Alwen, J., Coretti, S., Dodis, Y.: The double ratchet: Security notions, proofs, and modularization for the signal protocol. In: Advances in Cryptology - EUROCRYPT. pp. 129–158. LNCS, Springer (2019)
5. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS. pp. 394–403. IEEE Computer Society (1997)
6. Bellare, M., Duan, S., Palacio, A.: Key Insulation and Intrusion Resilience over a Public Channel. In: Topics in Cryptology - CT-RSA. pp. 84–99. LNCS (2009)
7. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: when and how should challenge decryption be disallowed? J. Cryptol. (1), 29–48 (2015)
8. Bellare, M., Kohno, T., Namprempre, C.: Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In: ACM CCS. pp. 1–11. ACM (2002)
9. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Advances in Cryptology - ASIACRYPT. pp. 531–545. LNCS, Springer (2000)
10. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Advances in Cryptology - CRYPTO. pp. 232–249 (1993)
11. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Advances in Cryptology - ASIACRYPT. pp. 317–330. LNCS, Springer (2000)
12. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. In: CT-RSA. pp. 1–18. LNCS, Springer (2003)
13. Blazy, O., Boureanu, I., Lafourcade, P., Onete, C., Robert, L.: How fast do you heal? a taxonomy for post-compromise security in secure-channel establishment. Cryptology ePrint Archive (2022), https://eprint.iacr.org/2022/1090
14. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Fast Software Encryption - 20th International Workshop, FSE. pp. 367–390. LNCS, Springer (2013)
15. Boyd, C., Hale, B.: Secure channels and termination: The last word on TLS. In: LATINCRYPT. pp. 44–65. LNCS, Springer (2017)
16. Boyd, C., Hale, B., Mjølsnes, S.F., Stebila, D.: From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS. In: Topics in Cryptology - CT-RSA. pp. 55–71. LNCS, Springer (2016)
17. Brzuska, C., Fischlin, M., Warinschi, B., Williams, S.C.: Composability of bellare-rogaway key exchange protocols. In: ACM CCS. pp. 51–62. ACM (2011)
18. Brzuska, C., Smart, N.P., Warinschi, B., Watson, G.J.: An analysis of the EMV channel establishment protocol. In: ACM CCS. pp. 373–386. ACM (2013)
19. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT. pp. 453–474. LNCS, Springer (2001)
20. Copet, P.B., Marchetto, G., Sisto, R., Costa, L.: Formal verification of LTE-UMTS handover procedures. In: IEEE ISCC. pp. 738–744. IEEE Computer Society (2015)
21. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Cryptography (2), 107–125 (1992)
22. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Advances in Cryptology - EUROCRYPT. pp. 65–82. LNCS, Springer (2002)
23. Dodis, Y., Luo, W., Xu, S., Yung, M.: Key-insulated symmetric key cryptography and mitigating attacks against cryptographic cloud software. In: ACM ASIACCS. pp. 57–58. ACM (2012)
24. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol. J. Cryptol. (4), 37 (2021)
25. Drucker, N., Gueron, S.: Continuous key agreement with reduced bandwidth. In: CSCML. pp. 33–46. LNCS, Springer (2019)

26. Fan, C., Huang, J., Zhong, M., Hsu, R., Chen, W., Lee, J.: ReHand: Secure Region-based Fast Handover with User Anonymity for Small Cell Networks in 5G. CoRR (2018), http://arxiv.org/abs/1806.03406
27. Fischlin, M., Günther, F., Janson, C.: Robust channels: Handling unreliable networks in the record layers of QUIC and DTLS 1.3. IACR Cryptol. ePrint Arch. p. 718 (2020), https://eprint.iacr.org/2020/718
28. Forsberg, D.: LTE key management analysis with session keys context. Comput. Commun. (16), 1907–1915 (2010)
29. Gellert, K., Handirk, T.: A formal security analysis of session resumption across hostnames. In: ESORICS. pp. 44–64. LNCS (2021)
30. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. (2), 270–299 (1984)
31. Günther, C.G.: An identity-based key-exchange protocol. In: Advances in Cryptology - EUROCRYPT. pp. 29–37. LNCS, Springer (1989)
32. Günther, F., Mazaheri, S.: A formal treatment of multi-key channels. In: Advances in Cryptology - CRYPTO. pp. 587–618. LNCS, Springer (2017)
33. Gupta, S., Parne, B.L., Chaudhari, N.S.: PSEH: A provably secure and efficient handover AKA protocol in LTE/LTE-A network. Peer-to-Peer Netw. Appl. (4), 989–1011 (2019)
34. Henda, N.B., Norrman, K.: Formal analysis of security procedures in LTE - A feasibility study. In: RAID. pp. 341–361. LNCS, Springer (2014)
35. Hoepman, J.: The ephemeral pairing problem. In: Financial Cryptography, 8th International Conference, FC. pp. 212–226. LNCS, Springer (2004)
36. Jaeger, J., Stepanovs, I.: Optimal channel security against fine-grained state compromise: The safety of messaging. In: CRYPTO. pp. 33–62. LNCS, Springer (2018)
37. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Advances in Cryptology - CRYPTO. LNCS, Springer (2012)
38. Jager, T., Stam, M., Stanley-Oakes, R., Warinschi, B.: Multi-key Authenticated Encryption with Corruptions: Reductions Are Lossy. In: Theory of Cryptography, TCC. pp. 409–441. LNCS, Springer (2017)
39. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Fast Software Encryption, FSE. pp. 284–299. LNCS (2000)
40. Kohno, T., Palacio, A., Black, J.: Building secure cryptographic transforms, or how to encrypt and MAC. IACR Cryptol. ePrint Arch. p. 177 (2003)
41. Marson, G.A., Poettering, B.: Security notions for bidirectional channels. IACR Trans. Symmetric Cryptol. (1), 405–426 (2017)
42. Milliken, W., Mendez, T., Partridge, D.C.: Host Anycasting Service. RFC 1546 (Nov 1993), https://www.rfc-editor.org/rfc/rfc1546
43. Peltonen, A., Sasse, R., Basin, D.A.: A comprehensive formal analysis of 5G handover. In: WiSec. pp. 1–12. ACM (2021)
44. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM CCS. pp. 98–107. ACM (2002)
45. Rogaway, P.: Nonce-based symmetric encryption. In: Fast Software Encryption, FSE. pp. 348–359. LNCS, Springer (2004)
46. Rogaway, P.: Formalizing human ignorance. In: Progressin Cryptology - VIETCRYPT. pp. 211–228. LNCS, Springer (2006)
47. Rogaway, P., Zhang, Y.: Simplifying game-based definitions - indistinguishability up to correctness and its application to stateful AE. In: Advances in Cryptology - CRYPTO. pp. 3–32. LNCS, Springer (2018)

A Postponed Proofs

A.1 Generic Composition

Proof. The proof essentially follows the structure of [8]. We first derive a probability bound for the advantage conditioned on that $\mathcal{A}$ obtains a valid decryption, and then a second bound conditioned on that $\mathcal{A}$ succeeds without doing so. The sum of these bounds then gives the upper bound on $\mathbf{Adv}^{\text{sach-ind-cca}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ stated in the theorem, by the law of total probability.

We say that a phase t is (n_B, n_A)-*revealed* if any phase in $(t - n_B, t + n_A)$ is revealed. Similarly, we say that t is (n_B)-*corrupted* if the client c was corrupted in t or any prior phase, or, if any $a \in \mathcal{U}_A$ controlling a phase after $t - n_b$ was corrupted.

Let G be the SACh-IND-CCA experiment as per Definition 5. Assume $\mathcal{A}$ is an effective and efficient G adversary attacking phase $t_{\mathcal{A}}$, resulting in an event log $\mathcal{L}$. $\mathcal{A}$ being effective implies that $P^{ADV}_{n_B,n_A}(\mathcal{L}, t_{\mathcal{A}})$ evaluates to true.

Next, let bad_I be the failure event where $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Recv}}$ with a ciphertext for a party r, which is in a non-(n_B, n_A)-revealed and non-(n_B)-corrupted phase t_r, and G goes out of sync and would return a valid message $m \in \mathcal{M}$ as reply. Specifically, $m \neq \bot$. By the difference lemma we get $\Pr[G = 1 \wedge bad_I] \leq \Pr[bad_I]$. We bound $\Pr[bad_I]$ by constructing an INT-CTXT adversary $\mathcal{B}_1$ from $\mathcal{A}$. $\mathcal{B}_1$ simulates G towards $\mathcal{A}$. Since we use RoR based confidentiality notions, the simulator does not need to select one out of two messages provided by $\mathcal{A}$, which LoR based notions require. When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Send}}$, $\mathcal{O}_{\mathsf{NextInit}}$, $\mathcal{O}_{\mathsf{Next}}$, $\mathcal{O}_{\mathbf{Cor}}$ or $\mathcal{O}_{\mathbf{Rev}}$, $\mathcal{B}_1$ forwards the queries verbatim to its INT-CTXT-challenger, and provide $\mathcal{A}$ with the corresponding replies. When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Recv}}$, $\mathcal{B}_1$ forwards the query to $\mathcal{O}^{\text{CTXT}}_{\mathsf{Recv}}$, but returns $\diamond$ to $\mathcal{A}$. $\mathcal{B}_1$ stops when $\mathcal{A}$ first triggers bad_I with a query q_I, and clearly runs in polynomial time.

The simulation is correct. Until bad_I occurs, the simulator is identical to G; all queries are forwarded verbatim to the INT-CTXT-game apart from $\mathcal{O}_{\mathsf{Recv}}$, where the simulator translates replies from **false** to $\diamond$.

$\mathcal{B}_1$ is an effective INT-CTXT adversary. When bad_I occurs, $\mathcal{B}_1$ wins the INT-CTXT game because $\mathcal{A}$ is effective, $P^{ADV}_{n_B,n_A}(\mathcal{L}, t_{\mathcal{A}})$ is true, and win is 1 (see the main experiment of Fig. 4). The flag win is 1 because q_I results in **Rcv** returning a valid message (by bad_I), and the party for which q_I was made is out of sync (see $\mathcal{O}^{\text{ATK}}_{\mathsf{Recv}}$ in Fig. 4). The party is out of sync by the effectiveness of $\mathcal{A}$ and by bad_I (see boxed code in Fig. 3). Therefore, $\mathcal{B}_1$ allows us to conclude $\Pr[bad_I] \leq \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},\mathcal{B}_1}(\lambda)$, and so $\Pr[G = 1 \wedge bad_I] \leq \mathbf{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},\mathcal{B}_1}(\lambda)$.

We now have a bound on $\Pr[G = 1 \wedge bad_I]$ and continue by bounding $\Pr[G = 1 \wedge \overline{bad_I}]$. We assume bad_I does not occur. A simple reduction from the SACh-IND-CPA experiment shows that $\Pr[\overline{bad_I}] = 1/2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{Ch},\mathcal{B}_2}(\lambda) - 1/2$ for an IND-CPA adversary $\mathcal{B}_2$ we construct as follows. $\mathcal{B}_2$ simulates G towards $\mathcal{A}$, relays all $\mathcal{A}$'s oracle queries to the IND-CPA-challenger, and provides the received replies back to $\mathcal{A}$. When $\mathcal{A}$ stops and outputs a test-phase t and bit-guess b_t, $\mathcal{B}_2$ also stops and forwards (t, b_t) to its IND-CPA-challenger.

The simulation is correct because G and the SACh-IND-CPA experiment are identical except for the boxed code in Fig. 3). G loses sync whereas the SACh-IND-CPA game logs ($\mathcal{L}_{\text{CPA}_\perp}$) to $\mathcal{L}$. However, since bad_I never occurs, these lines are never executed in either game.

$\mathcal{B}_2$ is an effective SACh-IND-CPA adversary because $\mathcal{A}$ is effective, the simulation is correct and because $\overline{bad_I}$ occurs. Therefore, $\Pr[\overline{bad_I}] = \Pr[\mathbf{Exp}^{\text{sach-ind-cpa}}_{\mathsf{Ch},\mathcal{B}_2,\{0,1\}}(\lambda) = 1]$, and so $\Pr[G = 1 \wedge \overline{bad_I}] \leq 1/2 \cdot \mathbf{Adv}^{\text{sach-ind-cpa}}_{\mathsf{Ch},\mathcal{B}_2}(\lambda) + 1/2$. Collecting the bounds we get

$$\Pr[G] = \Pr[G = 1 \wedge bad_I] + \Pr[G = 1 \wedge \overline{bad_I}] \leq$$
$$\mathbf{Adv}^{\text{sach-int-ctxt}}_{\mathsf{Ch},\mathcal{B}_1}(\lambda) + 1/2 \cdot \mathbf{Adv}^{\text{sach-ind-cpa}}_{\mathsf{Ch},\mathcal{B}_2}(\lambda) + 1/2$$

from which the theorem follows. □

A.2 XnCh Infrastructure Protection Security

Postponed proof of XnIP security Lemma 1.

Proof. We proceed by game hopping, starting with the claim of I-CPA security. Let G_0 be equal to the I-CPA experiment in Fig. 9.

Game $\mathbf{G_1}$. Let G_1 be equal to G_0 except that G_1 fixes the phase in which $\mathcal{A}_1$ attacks. G_1 samples a phase at random $t' \xleftarrow{R} \{0, \ldots, n_{ph} - 1\}$. Suppose $\mathcal{A}_1$ tries to discern bit $b_{t,ch}$ for phase t. If $t \neq t'$, G_1 aborts and returns a random boolean; $\mathcal{A}_1$ is then considered to have lost the game. Otherwise, if $t = t'$, $\mathcal{A}_1$ wins G_1 iff $\mathcal{A}_1$ wins G_0. The probability that $t = t'$ is $1/n_{ph}$, so $\mathcal{A}_1$'s advantage is bound by

$$\mathbf{Adv}^{\text{G}_0}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda) \leq n_{ph} \cdot \mathbf{Adv}^{\text{G}_1}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda).$$

In succeeding games we can assume that $\mathcal{A}_1$ attacks phase t.

Game $\mathbf{G_2}$. Let G_2 be equal to G_1 except for that G_2 substitutes randomly sampled keys for the symmetric keys rik and ik in all channel states. We argue that if $\mathcal{A}_1$ can distinguish G_1 from G_2, we can construct an adversary $\mathcal{B}_2$ against the standard PRF-security of F. Simplifying notation, we write $PSK^j_{\mathcal{I}}(o, src, trg)$, for the riks and iks in the channel states for the four channels (in phase j) between the target (trg) access point and the orchestrator (o), as well between the source (src) and the target access point.

Define hybrid distributions H^i_2 for $0 \leq i \leq n_{ph}$, bridging G_1 and G_2, as follows. H^i_2 samples the keys in $PSK^j_{\mathcal{I}}(o, a_j, a_{j+1})$ uniformly at random for all phases $j \leq i$. For phases $j > i$, H^i_2 derives the keys in $PSK^j_{\mathcal{I}}(o, a_j, a_{j+1})$, using $\mathsf{F}_{\mathcal{I}}$ as in G_1. Note that $H^0_2 = G_1$ and $H^{n_{ph}}_2 = G_2$.

We now construct a PRF adversary $\mathcal{B}_2$ against F. $\mathcal{B}_2$ simulates the experiment towards $\mathcal{A}_1$ as G_1 does, except for that $\mathcal{B}_2$ first samples a phase $t' \xleftarrow{R} \{0, t\}$, and then samples the keys in $PSK^j_{\mathcal{I}}(o, a_j, a_{j+1})$ uniformly at random for $j < t'$. When time comes to derive the keys in $PSK^{t'}_{\mathcal{I}}(o, a_{t'}, a_{t'+1})$ in phase t', $\mathcal{B}_2$ instead

computes the rik and ik for each state by querying its own PRF oracle on the corresponding inputs. Recall that the domain separation of F_C was done by prefixing the argument with an injective encoding of the constant C. Hence G_2 can encode the prefix $\mathcal{I}$ in the input to its PRF oracle. Suppose $\mathcal{A}_1$ terminates with a guess b' for channel ch' in phase t', and that $\mathcal{B}_2$ sampled $b_{t',ch}$. Then $\mathcal{B}_2$ guesses 0 in its own game if b' equals $b_{t',ch}$, and 0 otherwise.

$\mathcal{B}_2$ simulates correctly because it initially samples all keys at random and then derives all further keys, either using $\mathsf{F}_{\mathcal{I}}$ or by invoking its PRF oracle. $\mathcal{B}_2$ can hence answer all oracle queries as required.

$\mathcal{B}_2$ is effective. If the PRF oracle uses a randomly selected function, $\mathcal{B}_2$ simulates $H_2^{t'}$ for $\mathcal{A}_1$. Otherwise the PRF oracle uses F, and $\mathcal{B}_2$ simulates $H_2^{t'+1}$.

$\mathcal{A}_1$'s advantage in distinguishing between $H_2^{t'}$ and $H_2^{t'+1}$ is hence converted into distinguishing advantage for $\mathcal{B}_2$ in its PRF game. $\mathcal{A}_1$'s distinguishing advantage between $H_2^{t'}$ and $H_2^{t'+1}$ is then bounded from above by $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathcal{B}_2}(\lambda)$. By a hybrid argument we get

$$\mathbf{Adv}^{\mathrm{G}_1}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_2}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda) + n_{ph} \cdot \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathcal{B}_2}(\lambda)$$

Game $\mathbf{G_3}$. Let G_3 be equal to G_2 except that G_3 fixes which of the four channels connecting the source access point, target access point and the orchestrator that $\mathcal{A}_1$ attacks. G_3 samples one of the four channels at random. Call that one channel ch'. Suppose $\mathcal{A}_1$ tries to discern bit $b_{t,ch}$. If $ch \neq ch'$, G_3 aborts and returns a random boolean; $\mathcal{A}_1$ is then considered to have lost the game. Otherwise, if $ch = ch'$, $\mathcal{A}_1$ wins G_3 iff $\mathcal{A}_1$ wins G_2. The probability that $ch = ch'$, is $1/4$, so $\mathcal{A}_1$'s advantage can be bounded by

$$\mathbf{Adv}^{\mathrm{G}_2}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda) \leq 4 \cdot \mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda).$$

We can now assume that $\mathcal{A}_1$ attacks channel ch.

Game $\mathbf{G_4}$. In this game, we directly bound $\mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda)$ by constructing an IND-CPA adversary $\mathcal{B}_4$ against AE.

$\mathcal{B}_4$ simulates game G_3 for $\mathcal{A}_1$ except for that $\mathcal{B}_4$ computes all encryptions and decryptions itself only in phases $j \neq t$. In phase t, $\mathcal{B}_4$ relays the queries for channel ch and corresponding answers to and from its own AE IND-CPA oracle. This means that the encryption key and the bit $b_{t,ch}$ are implicitly set to the values used by the IND-CPA oracle. Because $\mathcal{A}_1$ is effective, it does not expose the key for phase t, so setting the encryption key implicitly is sound even though $\mathcal{B}_4$ does not have access to it. $\mathcal{B}_4$ has access to all other keys, and can answer queries using these as required.

From G_1 we know that $\mathcal{A}_1$ attacks phase t, from G_3 we know that $\mathcal{A}_1$ attacks channel ch, and from G_2 we know that all keys, up to and including phase t, appear to $\mathcal{A}_1$ as independent and indistinguishable from random. In particular the key used by AE in channel ch in phase t is indistinguishable from random. Further, the nonce consists of synchronous monotonic counters, maintained independently by sender and receiver. Nonces are thus unique per message and cannot be influenced by $\mathcal{A}_1$. Therefore, the simulation is correct and directly transfer

$\mathcal{A}_1$'s advantage in G_3 to $\mathcal{B}_4$ in its IND-CPA game. The difference between $\mathcal{A}_1$'s advantage against G_3 and G_4 respectively is bounded by $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},\mathcal{B}_4}(\lambda)$, so we get

$$\mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnIP},\mathcal{A}_1}(\lambda) \;\leq\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},\mathcal{B}_4}(\lambda).$$

Combining the bounds proves the I-CPA claim.

We now go on to show the I-AUTH claim following a similar proof structure. The main differences are that the attack phase is here where integrity is broken, and that the reduction to the AE's AUTH security instead of its IND-CPA security.

Game $\mathbf{G_5}$. Let G_5 equal the original I-AUTH experiment.

Game $\mathbf{G_6}$. Let G_6 be equal to G_5 except that G_6 fixes the phase in which a party in XnIP first accepts an out of sync ciphertext, i.e., either a valid message received out of order, or a message not generated by the valid encryption function. We refer to this phase as the attack phase. The rest of this game is exactly as game G_1, and we obtain the bound

$$\mathbf{Adv}^{\mathrm{G}_5}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda) \leq n_{ph} \cdot \mathbf{Adv}^{\mathrm{G}_6}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda).$$

In succeeding games we can assume that $\mathcal{A}_2$ attacks phase t.

Game $\mathbf{G_7}$. Let G_7 be equal to G_6 except for that G_7 substitutes randomly sampled keys for the XnIP pre-shared keys. With the definition of attack phase as above, this game is exactly as game G_2, and we construct a PRF adversary $\mathcal{B}_7$ against F to obtain

$$\mathbf{Adv}^{\mathrm{G}_6}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_7}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda) + n_{ph} \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{B}_7}(\lambda)$$

Game $\mathbf{G_8}$. Let G_8 be equal to G_7 except that G_8 fixes which of the four channels connecting the source access point, target access point and the orchestrator that $\mathcal{A}_1$ attacks. This game is exactly as game G_3, and we get the corresponding bound

$$\mathbf{Adv}^{\mathrm{G}_7}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda) \leq 4 \cdot \mathbf{Adv}^{\mathrm{G}_8}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda).$$

We can now assume that $\mathcal{A}_2$ attacks channel ch.

Game $\mathbf{G_9}$. In this game, we directly bound $\mathbf{Adv}^{\mathrm{G}_8}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda)$ by constructing an AUTH adversary $\mathcal{B}_9$ against AE.

$\mathcal{B}_9$ simulates game G_8 for $\mathcal{A}_2$ except for that $\mathcal{B}_9$ computes all encryptions and decryptions itself in phases $j \neq t$. In phase t, $\mathcal{B}_9$ relays the queries for channel ch and corresponding answers to and from its own AE AUTH oracle.

From G_6 we know that $\mathcal{A}_2$ attacks phase t, from G_8 we know that $\mathcal{A}_2$ attacks channel ch, and from G_7 we know that all keys, up to and including phase t, appear to $\mathcal{A}_2$ as independent and indistinguishable from random. In particular the key used by AE in channel ch in phase t is indistinguishable from random. Therefore, the simulation is correct. $\mathcal{A}_2$ is effective, and therefore triggers the win condition in the $\mathcal{O}_{\mathcal{I}_{\mathsf{Rx}}}$ oracle in Fig. 9, setting the *win* variable. The trigger must

have that decryption accepted m, i.e., that $m \neq \perp$. In addition, either $|C_{u,v}| = 0$, in which case $\mathcal{A}_1$ has constructed one more message than the sender has sent, and which is accepted by the AUTH oracle, or, $|C_{u,v}| > 0|$ but the message accepted is not the next one according to the message sequence. Because both the sender and receiver use monotonically increasing counters as nonces, the latter case also implies that the m is a valid forgery against AE, breaking its AUTH security.

$\mathcal{A}_2$'s advantage against G_8 is hence bounded by $\mathbf{Adv}^{\text{auth}}_{\mathsf{AE},\mathcal{B}_9}(\lambda)$, so we get

$$\mathbf{Adv}^{\text{G}_8}_{\mathsf{XnIP},\mathcal{A}_2}(\lambda) \;\leq\; \mathbf{Adv}^{\text{auth}}_{\mathsf{AE},\mathcal{B}_9}(\lambda).$$

Combining the bounds from games G_5 to G_9 proves the I-AUTH claim and concludes the proof. □

A.3 XnCh SACh-IND-CPA Security

Postponed proof of SACh-IND-CPA security Theorem 2.

Proof. We proceed by game hopping. Let G_0 be equal to $\mathbf{Exp}^{\text{sach-ind-cpa}}_{\mathsf{XnCh},\mathcal{A},\{0,1\}}(\lambda)$ defined in Fig. 3.

Game $\mathbf{G_1}$. Let G_1 be equal to G_0 except that G_1 fixes the phase in which $\mathcal{A}$ attacks. G_1 samples a phase at random from the phases appearing in the experiment, $t' \xleftarrow{R} \{0, \ldots, n_{ph} - 1\}$. Suppose $\mathcal{A}$ tries to discern bit b_t for phase t. If $t \neq t'$ then G_1 aborts and returns a random boolean; $\mathcal{A}$ is then considered to have lost the game. Otherwise, if $t = t'$, $\mathcal{A}$ wins G_1 iff $\mathcal{A}$ wins G_0. The probability that $t = t'$ is $1/n_{ph}$, so $\mathcal{A}$'s advantage can be bound by

$$\mathbf{Adv}^{\text{G}_0}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \leq n_{ph} \cdot \mathbf{Adv}^{\text{G}_1}_{\mathsf{XnCh},\mathcal{A}}(\lambda).$$

In succeeding games we can assume that $\mathcal{A}$ attacks phase t.

Game $\mathbf{G_2}$. Let G_2 be equal to G_1 except for that G_2 substitutes the key k' in the XnPrep message by a independently and randomly sampled key in phase $t - 1$. If $\mathcal{A}$ can distinguish between G_1 and G_2, we can construct an adversary $\mathcal{B}_2$ against the I-CPA security of XnIP.

$\mathcal{B}_2$ simulates game G_1 towards $\mathcal{A}$, except for the following. $\mathcal{B}_2$ samples a bit b_I, and if $b_I = 1$, $\mathcal{B}_2$ replaces the key k' in XnPrep with a randomly sampled key in phase $t - 1$. $\mathcal{B}_2$ stores the original k', and restores it when when the XnPrep message is processed by the XnPrep handler. When $b_I = 0$, $\mathcal{B}_2$ passes the real XnPrep messages to $\mathcal{O}_{\mathcal{I}_{\mathsf{Tx}}}$ as is, and uses the output from $\mathcal{O}_{\mathcal{I}_{\mathsf{Rx}}}$ as the XnPrep message. When $\mathcal{A}$ halts with a guessed bit b', $\mathcal{B}_2$ guesses the same bit in its own I-CPA game.

The simulation is correct. When $b_I = 0$, $\mathcal{B}_2$ simulates G_1, and when $b_I = 1$ it simulates G_2. $\mathcal{B}_2$ does not affect the keys actually used by the δ-channel and so reveal queries provide $\mathcal{A}$ with no additional information. When $\mathcal{A}$ invokes $\mathcal{O}_{\mathbf{Cor}}(u)$, $\mathcal{B}_2$ invokes $\mathcal{O}_{\mathcal{I}_{\mathsf{Expose}}}(u)$ to obtain the riks and iks used by u in the phase

$\mathcal{O}_{\mathbf{Cor}}(u)$ was invoked. However, because $\mathcal{A}$ is an effective adversary, it does not corrupt either of a_{t-1} and a_t earlier than phase t. XnIP is 1-hop fortified and a session key forward secure (by Lemma 1), and therefore $\mathcal{A}$ does not gain an advantage in the distinguishing task by corruption queries.

The simulation is effective. Since the ik used for encryption on the channel (a_{t-1}, a_t) appear random to $\mathcal{A}$, $\mathcal{A}$ can only distinguish between G_1 and G_2 if $\mathcal{A}$ detect that k' has been replaced with a randomly sampled key by distinguishing whether a real or random message was encrypted by XnIP. We therefore get the following bound.

$$\mathbf{Adv}^{\mathrm{G}_1}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_2}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_2}(\lambda).$$

Game $\mathbf{G_3}$. Let G_3 be equal to G_2 except for that G_3 substitutes the key nk in the $\mathsf{XnPsResp}$ message by a independently and randomly sampled key in phase $t-2$. If $\mathcal{A}$ can distinguish between G_2 and G_3, we can construct an adversary $\mathcal{B}_3$ against the I-CPA security of XnIP in a way analogously to game G_2. The differences are that

- the key nk is replaced in the $\mathsf{XnPsResp}$ message instead of the k' in the XnPrep message as in game G_2;
- $\mathsf{XnPsResp}$ is passed from the orchestrator to a_{t-1} over channel (o, a_{t-1});
- the phase we analyze is $t-2$ instead of $t-1$ as in game G_2. Here we make crucial use of the fact that $\mathcal{A}$ cannot corrupt a_{t-1} in phase $t-2$ per the adversary model. Access point a_{t-1} is not corrupted before phase $t-2$; if it were, the nk delivered to a_{t-1} in the phase $t-2$ $\mathsf{XnPsResp}$ message would be available to $\mathcal{A}$, and therefore also k' in phase $t-1$, and therefore also k in phase t.

By analysis analogous to the one for game G_2, we obtain the bound

$$\mathbf{Adv}^{\mathrm{G}_2}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_3}(\lambda).$$

From here on, we can assume that $\mathcal{A}$ can obtain neither nk in phase $t-2$ nor k' in phase $t-1$ by decrypting the messages in which they are passed between parties.

Game $\mathbf{G_4}$. Let G_4 be equal to G_3 except for that G_4 substitutes independently and randomly sampled keys for $\mathsf{K_f}$, nk, k' and k in all phases. We claim that if $\mathcal{A}$ can distinguish G_3 from G_4, we can construct an adversary $\mathcal{B}_4$ against the standard PRF-security of F.

Define hybrid distributions H_4^i for $0 \leq i \leq n_{ph}$, bridging G_3 and G_4, as follows. H_4^i samples $\mathsf{K_f}$, nk and k' independently and uniformly at random in all phases $j \leq i$. For phases $j > i$, H_4^i derives $\mathsf{K_f}$ using F_0, nk using F_1, and k' using F_2 as in G_3. Additionally, for $i > 0$, H_4^i sets k in phase i equal to k' from phase $i-1$ when the XnHo handler in G_3 derives k (see Fig. 7). Note that $H_4^0 = G_3$ and $H_4^{n_{ph}} = G_4$.

We now construct a PRF adversary $\mathcal{B}_4$ against F. $\mathcal{B}_4$ simulates the experiment towards $\mathcal{A}_1$ as G_3 does, except for that $\mathcal{B}_4$ first samples a phase $t' \xleftarrow{R} \{0, t\}$, and then sets $\mathsf{K_f}$, nk, k' and k according to H_4^i for $i \neq t'$. In phase t', $\mathcal{B}_4$ instead

computes these keys by invoking its own PRF oracle for F on the corresponding inputs and domain separation constants. $\mathcal{B}_4$ samples all challenge bits b_i itself; in particular it samples b_t. When $\mathcal{A}$ halts with a guess b' for phase t, $\mathcal{B}_4$ answers 0 in its own PRF game if $b' = b_t$, and 1 otherwise.

The simulation is correct. The only dependencies between keys are introduced by deriving them from the initial keys using F (or a real randomly selected function in phase t'). $\mathcal{B}_4$ hence has access to keys and data of the game and no inconsistencies are introduced that $\mathcal{A}$ could detect, apart from the possible use of a randomly selected function instead of F.

$\mathcal{B}_4$ is effective. If the PRF oracle uses a randomly selected function, $\mathcal{B}_4$ simulates $H_4^{t'}$ for $\mathcal{A}$. Otherwise the PRF oracle uses F, and $\mathcal{B}_4$ simulates $H_4^{t'+1}$. $\mathcal{A}$'s advantage in distinguishing between $H_4^{t'}$ and $H_4^{t'+1}$ is hence converted into distinguishing advantage for $\mathcal{B}_4$ in its PRF game. $\mathcal{A}$'s distinguishing advantage between $H_4^{t'}$ and $H_4^{t'+1}$ is then bounded from above by $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathcal{B}_4}(\lambda)$. By a hybrid argument, using the fact that $t' \leq t \leq n_{ph}$, we get

$$\mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_4}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + n_{ph} \cdot \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathcal{B}_4}(\lambda).$$

Game $\mathbf{G_5}$. Let G_5 be equal to G_4 except for that G_5 aborts if $\mathcal{A}$ correctly determines the bit b_t for phase t. We claim that any adversary $\mathcal{A}$ that can distinguish between G_4 and G_5 can be used to construct an effective adversary against the RoR IND-CPA security of the underlying AEAD scheme of the δ-channel. We construct such an adversary $\mathcal{B}_5$ as follows.

$\mathcal{B}_5$ simulates the experiment for $\mathcal{A}$ by executing all the steps of G_4, except for phase t, in which $\mathcal{B}_5$ uses modified $\mathcal{O}_{\mathsf{Send}}$ and $\mathcal{O}_{\mathsf{Recv}}$ oracles to extract its AEAD IND-CPA attack from $\mathcal{A}$. When $\mathcal{A}$ invokes $\mathcal{O}_{\mathsf{Send}}(u_1, m)$ in phase t, $\mathcal{B}_5$, instead of simulating AEnc, invokes its own AEAD IND-CPA encryption oracle with nonce $nc = (u_1, \mathsf{sq}_{u_1\downarrow})$, additional data $ad = (\mathsf{u}_1, \mathsf{sq}_{\mathsf{fin}_{u_1}})$, and message m. $\mathcal{B}_5$ then treats the resulting ciphertext as the outgoing encrypted message m'. Analogously, when $\mathcal{A}$ invokes $\mathcal{O}_{\mathsf{Recv}}(u_2, m)$ in phase t, $\mathcal{B}_5$, instead of simulating ADec, invokes its own AEAD IND-CPA decryption oracle with nonce $nc = (u_2, \mathsf{sq}_{u_2\uparrow})$, additional data $ad = (\mathsf{u}_2, \mathsf{sq}_{\mathsf{fin}_{u_2}})$, and message m'. $\mathcal{B}_5$ then treats the resulting plaintext as the received message. Note that we cannot remove the decryption oracle in the AEAD IND-CPA experiment since we use a bidirectional channel. In particular, without the decryption oracle, we could not simulate the δ-channel, and theoretically, decryption of a message could affect the state for the encryption oracle.

When $\mathcal{A}$ halts with a guess b', $\mathcal{B}_5$ makes the same guess in its own AEAD IND-CPA game.

The simulation is correct. The only difference from G_4 is that $\mathcal{B}_5$ uses its AEAD oracle to answer encryption and decryption queries in phase t. Observe that the nonce consists of a user identifier, ensuring that each direction is encrypted with a unique input, and a monotonically increasing sequence number, ensuring that each message in a given direction has a unique input. Both $\mathsf{pdcpSend}$ and $\mathsf{pdcpRecv}$ are guarded by **assert** statements that ensure that processing with unsafely large sequence numbers.

The experiment requires that no involved party is revealed in phase t. From previous games, we have that: $\mathcal{A}$ attacks phase t; no party corruptions expose a key from which k in phase t is derived; k in phase t is from $\mathcal{A}$'s perspective indistinguishable from a randomly sampled key. Hence, it is sound to implicitly set the key k and the bit b_t to the corresponding values of $\mathcal{B}_5$'s AEAD IND-CPA challenger in phase t.

We can conclude that

$$\mathbf{Adv}^{\mathrm{G}_4}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_5}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AEAD},\mathcal{B}_5}(\lambda).$$

Finally, combining the bounds between each game gives the total bound in the theorem. □

A.4 XnCh SACh-INT-CTXT Security

Postponed proof of SACh-INT-CTXT security Theorem 3.

Proof. The first hops are similar in structure to the proof of Theorem 2. Let G_0 be equal to $\mathbf{Exp}^{\text{sach-int-ctxt}}_{\mathsf{XnCh},\mathcal{A},\{0,1\}}(\lambda)$ defined in Fig. 4.

Game $\mathbf{G_1}$. Let G_1 be equal to G_0 except that G_1 fixes the phase in which the client or controlling access point accept the first malicious message received over the δ-channel. A malicious message is a message delivered out of order or which was not generated by the other party of the δ-channel. We refer to this phase as the attack phase. The rest of the game is exactly as game G_1 in the proof of Theorem 2 and we obtain the bound

$$\mathbf{Adv}^{\mathrm{G}_0}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \leq n_{ph} \cdot \mathbf{Adv}^{\mathrm{G}_1}_{\mathsf{XnCh},\mathcal{A}}(\lambda).$$

In succeeding games we can assume that $\mathcal{A}$ attacks phase t.

Game $\mathbf{G_2}$. Let G_2 be equal to G_1 except for that G_2 substitutes the key k' in the XnPrep message by a independently and randomly sampled key in phase $t-1$. If $\mathcal{A}$ can distinguish between G_1 and G_2, we can construct an adversary $\mathcal{B}_2$ against the I-CPA security of $\mathcal{I}$. The difference compared to game G_2 in the proof of Theorem 2 is how $\mathcal{B}_2$ extracts its answer for the I-CPA game from $\mathcal{A}$. In the present game, $\mathcal{B}_2$ answers 0 if $\mathcal{A}$ succeeds to set the variable *win* to 1, and answers 1 otherwise. Any increased advantage $\mathcal{A}$ get from being able to distinguish between G_1 and G_2 gets directly transferred to $\mathcal{B}_2$'s advantage in winning its own I-CPA game. We therefore get the following bound.

$$\mathbf{Adv}^{\mathrm{G}_1}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_2}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_2}(\lambda).$$

Game $\mathbf{G_3}$. Let G_3 be equal to G_2 except for that G_3 substitutes the key nk in the $\mathsf{XnPsResp}$ message by a independently and randomly sampled key in phase $t-2$. The difference compared to game G_3 in the proof of Theorem 2 is how $\mathcal{B}_3$ extracts its answer for the I-CPA game from $\mathcal{A}$. In the present game, $\mathcal{B}_3$ answers 0 if $\mathcal{A}$ succeeds to set the variable *win* to 1, and answers 1 otherwise. Any increased advantage $\mathcal{A}$ get from being able to distinguish between G_2 and

G_3 gets directly transferred to $\mathcal{B}_3$'s advantage in winning its own I-CPA game. We therefore get the following bound.

$$\mathbf{Adv}^{\mathrm{G}_2}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\text{i-cpa}}_{\mathcal{I},\mathcal{B}_3}(\lambda).$$

From here on, we can assume that $\mathcal{A}$ can obtain neither nk in phase $t-2$ nor k' in phase $t-1$ by decrypting the messages in which they are passed between parties.

Game $\mathbf{G_4}$. Let G_4 be equal to G_3 except for that G_4 substitutes independently and randomly sampled keys for $\mathsf{K_f}$, nk, k' and k in all phases. We claim that if $\mathcal{A}$ can distinguish G_3 from G_4, we can construct an adversary $\mathcal{B}_4$ against the standard PRF-security of F. Similarly to the two previous games in the present proof, the main difference compared to the corresponding game in the proof of Theorem 2 is how $\mathcal{B}_4$ extracts its answer from $\mathcal{A}$. The difference between the present game and game G_4 in the proof of Theorem 2 is that in the present game, $\mathcal{B}_4$ answers 0 if $\mathcal{A}$ succeeds setting the variable win to 1, and answers 1 otherwise.

By a hybrid argument analogous to that of game G_4 in Theorem 2, using the fact that $t' \leq t \leq n_{ph}$, we get

$$\mathbf{Adv}^{\mathrm{G}_3}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_4}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + n_{ph} \cdot \mathbf{Adv}^{\text{prf}}_{\mathsf{F},\mathcal{B}_4}(\lambda).$$

Game $\mathbf{G_5}$. Let G_5 be equal to G_4 except that G_5 aborts if the client or controlling access point accept a message m^*, processing this with some δ-state δ^*, when the party's sync flag $sync_{v^*}$ equals 0. We denote this receiving party by $v*$. We claim that any adversary $\mathcal{A}$ against G_5 can be used to construct an effective adversary against the AUTH security of the scheme AEAD underlying the δ-channel. We construct such an adversary $\mathcal{B}_5$ as follows.

First note that the $\mathsf{pdcpSend}$ and $\mathsf{pdcpRecv}$ functions that provide state management on top of AEAD (see Fig. 5) rejects processing of messages — both incoming and outgoing — once $\mathsf{pdcpRecv}$ has rejected a single incoming message. This is remembered by $\mathsf{pdcpRecv}$ setting the δ-state equal to $\perp$. Consequently $\mathcal{A}$ only has this first opportunity to win its game, and processing of the message m^* must therefore also result in that the experiment sets win to 1. From this we conclude that $\mathsf{pdcpRecv}(\delta^*, m^*) \neq \perp$. That is, $\mathsf{pdcpRecv}$ returns a valid decrypted message.

$\mathcal{B}_5$ behaves almost identically to the reduction in game G_5 in the proof of Theorem 2. The main difference is the extraction of the answer from $\mathcal{A}$. $\mathcal{B}_5$ in the present game extracts it as follows. When $\mathcal{B}_5$ is about to set the variable win to 1 in the $\mathcal{O}^{\mathrm{ATK}}_{\mathsf{Recv}}$ oracle in Fig. 4, we know that this is because $\mathcal{A}$ has successfully crafted a message m^*, which was accepted by $\mathsf{AEADDec}$ (out of sync) when given the nonce nc^* and additional data ad^* constructed from the δ-state δ^* as it was before processing m^*. Therefore, $\mathcal{B}_5$ halts at this point and answers with the triplet (nc^*, ad^*, m^*), which is a valid forgery, to its AEAD AUTH challenger.

The simulation is correct. $\mathcal{B}_5$ has access to all keys required to answer corruption and reveal queries, except for in phase t, but since $\mathcal{A}$ is effective, it does not make any such queries.

$\mathcal{B}_5$ is effective. Similar to the proof of Theorem 4.2 in [32], we here do a case analysis split based on whether m^* is the message that brings v^* out of sync, or whether the v^* already was out of sync when the message is received. In each case we must show that at least one element in the triplet (nc^*, ad^*, m^*) differ from what could have been produced by a call to the AUTH oracle when invoked by the $\mathcal{O}_{\mathsf{Send}}$ oracle. Because $\mathcal{A}$ is effective, P^{Cor} evaluates to true, and for all cases we can assume that neither the sender nor the receiver have been compromised. First consider the case when $sync_{v^*} = 0$ before v^* receives m^*. Let m_s be the message that causes $sync_{v^*}$ to be set to 1. Sync is lost under any out of the following five conditions.

- $\mathcal{O}_{\mathsf{Send}}$ oracle :See **Snd** in Fig. 3. $t_{v^*} > t_u$ and sender u is not revealed. Because ad contains the sender identity, the message m_s, if received by v^*, would have a different identity compared to what v^* would use as ad. Because v^* is at least one phase ahead of u, we have that $i_u^{t_u} > j_{v^*}^{t_{u^*}}$ in the experiment, matched by $\mathsf{sq}_{\downarrow_u} > \mathsf{sq}_{\uparrow_{v^*}}$ in the δ-state during phase t_u. The sender's ad contains $\mathsf{sq}_{\downarrow_u}$ and the receiver's ad contains $\mathsf{sq}_{\uparrow_{v^*}}$. Therefore, these parts of the two ads will never match in phase t_u. Once u progresses to the next phase, it will have transmitted at least one more message in phase t_u than v^*, and hence the values of their $\mathsf{sq}_{\mathsf{fin}}$ variables will differ. Since $\mathsf{sq}_{\mathsf{fin}}$ is included in ad, u and v^* will not have equal ads for any future message. Therefore, whatever message encrypted using the AUTH oracle (as invoked from the $\mathcal{O}_{\mathsf{Send}}$ oracle) will have different ad values, compared to what $\mathcal{O}_{\mathsf{Recv}}$ uses, for all future messages. Therefore, whatever message m^* that v^* later receives, it will bring v^* out of sync and if v^* accepts it, it will be a valid forgery, which could not have been generated by $\mathcal{O}_{\mathsf{Send}}$ using its AUTH oracle.
- $\mathcal{O}_{\mathsf{Recv}}$ oracle: See **Rcv** in Fig. 3. $t_{v}* > t_u$ and the receiver $v*$ is not revealed. The message m_s brings the experiment out of sync, but the pdcpRecv invocation must not allow $\mathcal{A}$ to win, because the assumption in this branch of the proof is that v^* is brought out of sync *before* receiving m^*. $\mathcal{A}$ wins iff pdcpRecv does not return $\perp$, since that would mean that pdcpRecv would continue to answer $\perp$ in all future invocations, effectively preventing $\mathcal{A}$ from winning. By similar reasoning as in the previous case, we have that when the receiver is at least one phase ahead of the sender, the AUTH AEADDec oracle invoked by pdcpRecv will use mismatching ad contents. If pdcpRecv accepts such a message, and returns a valid plaintext, it would be a valid forgery that $\mathcal{B}_5$ can use to win its AUTH game. This contradicts that $\mathcal{A}$ does not win during thins invocation, and can therefore not happen.
- $\mathcal{O}_{\mathsf{Recv}}$ oracle: See **Rcv** in Fig. 3. $j_{v^*}^{t_{v^*}} > i_u^{t_u}$ and the receiver v^* is not revealed. We have that in the δ-state this is matched by $\mathsf{sq}_{\uparrow_{v^*}} > \mathsf{sq}_{\downarrow_u}$ during phase t_{v^*}. As in the previous case, we must have that the AUTH oracle AEADDec invoked by pdcpRecv does not return $\perp$. However, because u includes $\mathsf{sq}_{\downarrow_u}$ in ad when invoking AEADEnc and v^* include $\mathsf{sq}_{\downarrow_{v^*}}$ in ad when invoking AEADDec. If the latter accepts the message, it is therefore a valid forgery,

contradicting that $\mathcal{A}$ does not win in this invocation of $\mathcal{O}_{\mathsf{Recv}}$ Therefore, this case cannot occur.

– $\mathcal{O}_{\mathsf{Recv}}$ oracle: See **Rcv** in Fig. 3. The received ciphertext message is not the ciphertext message that u sent. The ciphertext message generated by pdcpSend's invocation of AUTH for this position in the message sequence does not equal the message m^* received and processed by pdcpRecv for that same position. If pdcpRecv's invocation of its AUTH AEADDec oracle accepts the message, it is a valid forgery, contradicting that $\mathcal{A}$ does not win in this invocation of $\mathcal{O}_{\mathsf{Recv}}$ Therefore, this case cannot occur.
– $\mathcal{O}_{\mathsf{Next}}$ oracle: See **syncReject** in Fig. 3. We first argue that $\mathcal{A}$ cannot interfere with the Xn CE protocol execution, and then that the execution results in that the sync-failure condition in **syncReject**cannot be triggered.
The infrastructure protection XnIP ensures that all Xn CE protocol messages are delivered authentically and in order, except for possibly the XnHo message and the XnHoCmp messages delivered over the δ-channel. The authenticity of Xn CE messages delivered over the δ-channel can be modified by $\mathcal{A}$, and $\mathcal{A}$ can inject messages on the δ-channel iff $\mathcal{A}$ can forge a plaintext, additional data or a nonce against AEADDec — regardless, this requires a valid forgery that $\mathcal{B}_5$ can use to win its AEAD AUTH game. Since we only provide a security guarantee when the Xn CE protocol executes to the end between phases, we do not need to consider truncation attacks against it. The global state machine of the Xn CE protocol is lock-step, so if messages are authentic, there can be no replays and each party execute as expected. None of the parties involved in the phase progression are revealed or corrupted by the security definition. We can therefore conclude that the Xn CE protocol executes as expected unless cut short, but when cut short, we make no security claims.
The Xn CE protocol returns an $E_a = 1$ from the Next function only when calling the XnPrep handler. This handler appends the $\mathsf{sq}_{\downarrow}$ and $\mathsf{sq}_{\uparrow}$ received from the source a to the sq_{fin} and is then prepared to accept a XnHoCmp message over the δ-channel. The Xn CE protocol returns an $E_c = 1$ from the Next function only when calling the XnHo handler. This handler does the corresponding actions for the client. As a remark, the sequence number for the δ-channel message delivering the XnHo message to the client increases the sequence number by one, and this is compensated for in the XnStart function. This means that the sequence number space continues to monotonically increase after this point on both sides. The next message accepted by the target a will be the XnHoCmp, unless $\mathcal{A}$ forges a δ-channel message (and thereby provides $\mathcal{B}_5$ with a valid forgery to win its AEAD AUTH game). All in all, we see that both S and the client progress to and agree on the next phase, and they agree on the sequence numbers for the δ-channel in both directions. Therefore, the sync-failure condition in **syncReject**cannot be triggered unless $\mathcal{B}_5$ extracts a valid forgery from $\mathcal{A}$.

This concludes the cases where sync was lost prior to receiving the first forged message. We now consider the case when $sync_{v^*}$ becomes 0 when v^* processes m^*. We need to consider the following subcases.

- $\mathcal{O}_{\mathsf{Recv}}$ oracle: See **Rcv** in Fig. 3. $t_{v^*} > t_u$ and the receiver $v*$ is not revealed. When the receiver is one phase ahead of the sender, the received message m^* must be necessity be a valid forgery, because pdcpSend has not yet been invoked in phase t_{v^*} (and has hence not invoked its AUTH oracle).
- $\mathcal{O}_{\mathsf{Recv}}$ oracle: See **Rcv** in Fig. 3. $j_{v^*}^{t_{v^*}} > i_u^{t_u}$ and the receiver v^* is not revealed. In the δ-state this is matched by $\mathsf{sq}_{\uparrow_{v^*}} > \mathsf{sq}_{\downarrow_u}$ during phase t_{v^*}. Because pdcpSend includes $\mathsf{sq}_{\downarrow_u}$ in ad if it would have generated m^*, and pdcpRecv includes $\mathsf{sq}_{\uparrow_u}$ in ad when processing m^*, we must have that m^* is a valid forgery when $j_{v^*}^{t_{v^*}} > i_u^{t_u}$.
- $\mathcal{O}_{\mathsf{Recv}}$ oracle: See **Rcv** in Fig. 3. The received ciphertext message is not the ciphertext message that u sent. The ciphertext message generated by pdcpSend's invocation of AUTH for this position in the message sequence does not equal the message m^* received and processed by pdcpRecv for that same position. If pdcpRecv's AUTH AEADDec oracle accepts m^*, it must therefore be a valid forgery.

  We can conclude that

$$\mathbf{Adv}^{\mathrm{G}_4}_{\mathsf{XnCh},\mathcal{A}}(\lambda) \;\leq\; \mathbf{Adv}^{\mathrm{G}_5}_{\mathsf{XnCh},\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathrm{auth}}_{\mathsf{AEAD},\mathcal{B}_5}(\lambda).$$

Finally, combining the bounds between each game gives the total bound in the theorem. □