A Formal Treatment of Multi-key Channels

Felix Günther and Sogol Mazaheri

Cryptoplexity, Technische Universität Darmstadt, Germany
`guenther@cs.tu-darmstadt.de` `sogol.mazaheri@cryptoplexity.de`

Abstract. Secure channel protocols protect data transmission over a network from being overheard or tampered with. In the common abstraction, cryptographic models for channels involve a single key for ensuring the central security notions of confidentiality and integrity. The currently developed next version of the Transport Layer Security protocol, TLS 1.3, however introduces a key updating mechanism in order to deploy a sequence of multiple, possibly independent encryption keys in its channel sub-protocol. This design aims at achieving forward security, protecting prior communication after long-term key corruption, as well as security of individual channel phases even if the key in other phases is leaked (a property we denote as phase-key insulation). Neither of these security aspects has been treated formally in the context of cryptographic channels so far, leading to a current lack of techniques to evaluate such channel designs cryptographically.

We approach this gap by introducing the first formal model of multi-key channels, where sender and receiver can update their shared secret key during the lifetime of the channel without interrupting the communication. We present modular, game-based notions for confidentiality and integrity, integrating forward security and phase-key insulation as two advanced security aspects. As we show, our framework of notions on the lower end of its hierarchy naturally connects to the existing notions of stateful encryption established for single-key channels. Like for classical channels, it further allows for generically composing chosen-ciphertext confidentiality from chosen-plaintext confidentiality and ciphertext integrity. We instantiate the strongest security notions in our model with a construction based on authenticated encryption with associated data and a pseudorandom function. Being comparatively close, our construction additionally enables us to discuss the TLS 1.3 record protocol design.

1 Introduction

Secure channel protocols are at the heart of today's communication infrastructure, protecting data in transit in countless connections each day. Major examples include the Transport Layer Security (TLS) protocol [22] securing the Web, the Secure Shell (SSH) protocol [48] enabling secure remote logins, and the Internet Protocol Security (IPsec) protocol [34] protecting, e.g., tunneled network-to-network connections.

1.1 Secure Cryptographic Channels

In the cryptographic realm, the established game-based abstraction of secure channels is that of *stateful encryption*, introduced by Bellare, Kohno, and Namprempre [9]. Stateful encryption first of all inherits the classical security requirements of (non-stateful) encryption: confidentiality and integrity. Confidentiality of encryption, first formalized by Goldwasser and Micali [31], intuitively demands that the content of transmitted messages remains secret. Integrity, in parts concurrently introduced by Katz and Yung [33], Bellare and Rogaway [11], and Bellare and Namprempre [10], in contrast ensures that an adversary cannot forge ciphertexts that, on decryption, lead to (meaningful) messages. In order to provide secure communication through a sequence of messages, stateful encryption schemes go beyond these standard requirements and moreover protect against reordering, dropping, and replays of messages transmitted in a channel. On a constructive level, channels to this extend incorporate authenticated encryption with associated data (AEAD) schemes [44] as an essential cryptographic building block, integrated with message-order and error handling.

Starting from and partially building upon the work by Bellare, Kohno, and Namprempre, various extensions and adaptations of (game-based) channel models have been proposed. For example, Kohno, Palacio, and Black [35] define a hierarchy of channels with varying resilience against replays, reordering, or message dropping. In order to capture potential padding of messages before encryption, Paterson, Ristenpart, and Shrimpton [42] introduce the notion of length-hiding authenticated encryption. Motivated by practical attacks due to implicit information leakage through different error messages or different timings of an error message, e.g., caused by either a MAC or a decryption failure, Boldyreva et al. [17] discuss decryption algorithms that distinguish more than a single error message. They also study the effects of multiple error messages on the generic relation between confidentiality and integrity established earlier by Bellare and Namprempre [10]. In order to capture fragmented delivery of ciphertexts as it arises in real-world attacks on secure channels (cf. [3]), Boldyreva et al. [16] and Albrecht et al. [2] consider stateful encryption with ciphertext fragmentation. Going one step further, Fischlin et al. [29] additionally study plaintext fragmentation to capture scenarios where channels are required to process a stream of data. Finally, protocols in practice usually establish a bi-directional communication channel, a setting whose security was recently studied by Marson and Poettering [39].

1.2 Multi-key Channels

In all cryptographic models of secure channels established so far, security originates from a single, symmetric key shared between the two endpoints of the channel. The upcoming version of the TLS protocol, TLS 1.3 [43], whose specification is currently being developed, however deviates from this paradigm and instead deploys a sequential series of multiple keys. The TLS 1.3 channel (the so-called record protocol) as usual begins with deriving an initial key for encryption and decryption of messages. As a novel component, both parties are further able to trigger key updates, leading to a key switch according to a pre-defined schedule while maintaining channel's operation. One particular motivation for this approach is that long-lived TLS connections may exhaust the cryptographic limits of some algorithms on how much data can be safely encrypted under a single key (cf. [43, Section 5.5], [38]).

A more general, major reason for refreshing the key used in a secure channel and specifically TLS 1.3 is *forward security*, a notion primarily known from and well-established in the context of key exchange protocols [32,23,19]. When using the same key throughout the lifetime of a channel, an attacker that learns this key (e.g., through cryptanalysis or even temporary break-in into the system) immediately compromises the confidentiality of previous and the integrity of future communication. In contrast, forward security demands that even if key material is leaked at some point, previous communication remains secure. Forward-secure symmetric encryption in the non-stateful setting is considered understood and in particular can be built from forward-secure pseudorandom bit generator [13] or, more generally, through re-keying [1]. In the context of secure channels, a formal treatment of forward security is however lacking so far.

Beyond forward security, a second security property arises for secure channels (in particular in the design of TLS 1.3) which we refer to as *phase-key insulation*. While forward security targets a full compromise (and prior security), phase-key insulation is concerned with the *temporary* compromise of a channel in the form of leaking the key used in a certain time period (phase), but not in others. Such temporary compromise might, e.g., result from differing strengths of key material used to derive some of the phase keys (as is the case for keys established in the TLS 1.3 key exchange [37,27,28]) or from storing the currently active key in less secure memory for efficiency reasons. A secure channel with phase-key insulation should then uphold confidentiality and integrity in uncompromised phases, even if the key of prior or later phases is revealed. Moreover, security should be retained even if the attacker learned a phase's key while that phase was still active.

As we will see, phase-key insulation orthogonally complements the notion of forward security, which is only concerned with a posteriori leakage of keys. Requiring it furthermore introduces new pitfalls in the design of secure channels. For example, the initial draft design of the TLS 1.3 record protocol with key updates enabled truncation attacks in non-compromised phases that would go unnoticed during the further execution of the protocol, as Fournet and the miTLS [40] team discovered [30]. We hence consider it being crucial to establish a formal understanding of channels using multiple keys, which is lacking at this point, in order to allow thorough analyses of proposed protocols and means for evaluating their provable security guarantees.

1.3 Our Contributions

In this work we initiate the study of channels that employ a sequence of multiple keys. To this end, we introduce a formalization of such *multi-key channels* and set up an according framework of game-based security notions. We then analyze the relations between our security notions as well as connections to the established notions for stateful encryption and finally provide a generic construction of a provably secure multi-key channel.

Following the game-based tradition in modeling channels, our formalism builds upon and extends that of Bellare, Kohno, and Namprempre [9] and Bellare and Yee [13]. More specifically, our notion of multi-key channels augments that of regular stateful encryption in three aspects. Obviously, we first of all consider a sequence of keys to be used for encryption and decryption. Secondly, switches between these keys are initiated through a specific key-update algorithm which makes the channel proceed from one phase to the next. Lastly, we separate two hierarchies of keys by additionally considering a level of master secret keys which, also evolving over time, are used to derive the channel key for each phase. As we will discuss, this carefully crafted syntax and key hierarchy in particular allows us to quite closely model the key schedule of the TLS 1.3 record protocol draft [43].

We then define security of multi-key channels via a a framework of notions. Beyond capturing the classical requirements of confidentiality and integrity, our notions modularly integrate the advanced security properties of forward security and phase-key insulation arising in the context of multi-key channels. The core technical challenge here is to appropriately capture the desired security properties while excluding trivial attacks in the stateful multi-key setting. We furthermore modularize the adversary's capability to proceed a channel to a next phase through key updates. Thereby, our framework elegantly also captures the single-key variants of our security notions, i.e., the cases where a multi-key channel only operates in a single phase.

Our single-key security notions enable us to provide a formal link to the established stateful-encryption notions for regular channels. We show that analogous notions in both models are essentially equivalent (modulo the differences in syntax) by providing natural, generic transforms between each pair of corresponding confidentiality and integrity notions. Furthermore, we establish separations that give rise to a hierarchy of our security notions and in particular establish forward security and phase-key insulation as independent security properties. To complete the picture of relations, we also translate the classical composition result for symmetric encryption by Bellare and Namprempre [10] to the setting of multi-key channels, showing that chosen-plaintext confidentiality combined with ciphertext integrity implies the stronger chosen-ciphertext notion of confidentiality.

Finally, we instantiate our model by providing a construction of a multi-key channel from a nonce-based authenticated encryption with associated data (AEAD) scheme and a pseudorandom function. To ensure both forward security and phase-key insulation, we match suitable techniques established for forward-secure key generation and for ensuring causal integrity. Leveraging our composition theorem, we then prove that our construction meets our strongest confidentiality and integrity notions for multi-key channels. Coming back to the initial motivation from real-world protocol design, we compare our construction with the draft design of the TLS 1.3 record protocol.

1.4 Related Work

Beyond the preceding works on secure channels discussed earlier, there has been substantial work on mostly the handshake but also the record protocol of the TLS 1.3 drafts; see Paterson and van der Merwe [41] for an overview. Badertscher et al. [4] analyze an early draft of the TLS 1.3 record protocol without key updates in the constructive cryptography setting. Bellare and Tackmann [12] analyze the multi-user security of the AES-GCM as authenticated-encryption building block of TLS 1.3. Bhargavan et al. [14,15] provide verified implementations of the TLS 1.3 record protocol.

Our notion of phase-key insulation is similar in spirit to, and hence borrows its name from, the notion of key insulation introduced in the public-key setting [24,25] and also transferred to (non-stateful) symmetric encryption [26]. Beyond treating (phase-)key insulation in the different context of secure channels, our notion permits more fine-grained corruption of keys. It thereby enables studying the interaction of forward secrecy and phase-key insulation in a single, modular framework.

2 Multi-key Channels

We begin with defining the syntax and correctness of multi-key channels, focusing on their functionality in this section; we will treat their security in Section 3. In Figure 1 we exemplify the operations of a multi-key channel and already hint at their expected security.

Like a regular, single-key channel (abstractly modeled as stateful encryption [9]), a multi-key channel is used by a sender to transform a sequence of messages $m_1, m_2, \ldots \in \{0,1\}^*$ into a corresponding sequence of ciphertexts c_1, $c_2, \ldots \in \{0,1\}^*$ using a sending algorithm Send.[1] The receiver then sequentially uses a corresponding Recv algorithm on each transmitted ciphertext to recover the sent message sequence.

In addition to regular channels, both sender and receiver can decide to update their keys used for sending and receiving, thereby switching to the next *phase* of the multi-key channel. In our model, we consider a two-level hierarchy for key derivation. On the first level, the complete multi-key channel is bootstrapped from a single, initial *master secret key* generated upon initialization of the channel. Master secret keys are furthermore evolved when switching to the

[1] In order to make explicit that a secure multi-key channel might only provide integrity but no confidentiality, we choose to make use of the more general terms “sending” and “receiving” instead of “encryption” and “decryption”.

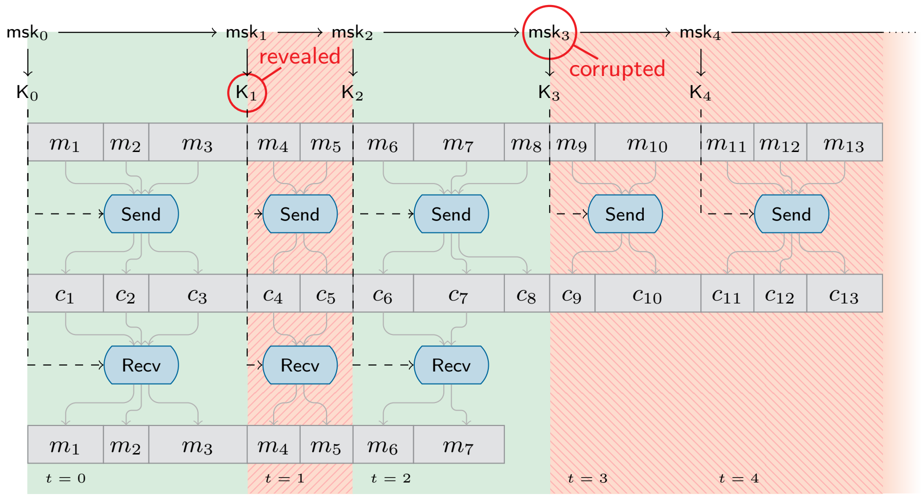
msk$_0$
msk$_1$
msk$_2$
msk$_3$
msk$_4$
revealed
corrupted
K$_0$
K$_1$
K$_2$
K$_3$
K$_4$
m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 m_{10} m_{11} m_{12} m_{13}
Send
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{13}
Recv
m_1 m_2 m_3 m_4 m_5 m_6 m_7
$t = 0$
$t = 1$
$t = 2$
$t = 3$
$t = 4$

Fig. 1: Illustration of the behavior of a multi-key channel (cf. Definition 1). The beginning of a new phase t is indicated by the derivation of a phase key K_t from the corresponding master secret key msk_t. The phase key K_t is then used to send and receive in-order messages resp. ciphertexts via algorithms Send and Recv in this phase.
In this example, the phase key K_1 of phase 1 is revealed and the master secret key msk_3 is corrupted. The affected phases 1 resp. 3 and following are marked in hatched-pattern red (with lines towards top right for the effects of the revealed K_1 and toward bottom right for the effects of the corrupted msk_3). For security (cf. Section 3), a forward-secure and phase-key–insulated multi-key channel is demanded to provide security in the non-affected phases 0 and 2, marked by non-hatched green areas.

next phase, following a deterministic key schedule to derive the master secret key msk_{t+1} for phase $t+1$ from the master secret key msk_t of the previous phase. On the second level, the actual *phase key* K_t used in the channel for sending and receiving messages in a phase t is derived (again deterministically) from that phase's master secret key msk_t.

Although Figure 1 depicts only a single key schedule with the phase keys forwarded to both the Send and Recv algorithms of that phase, in a real execution of the channel, the key updates and derivations are invoked independently on the sending and receiving side. For correct functionality, the key updates need to be aligned in order to process sent and received ciphertexts under matching keys on both sides. In practice, key updates may be either delivered alongside of the messages transmitted in a channel (and hence potentially authenticated) or in an out-of-band manner, e.g., via a separate control channel, and with their position in the channel's ciphertext sequence not being explicitly authenticated.[2] In our abstraction of multi-key channels, we do not rely on the authenticity of the key-update signaling (in particular, we will later allow adversaries to tamper with the timing of key updates) but leave it up to the channel to ensure their correct position with respect to the transmitted ciphertexts.

We now define the syntax and correctness of multi-key channels capturing the given intuition.

Definition 1 (Syntax of multi-key channels). *A* multi-key channel $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Update})$ *with associated sending and receiving state space $\mathcal{S}_S$ resp. $\mathcal{S}_R$, master secret key space $\mathcal{MSK}$, phase key space $\mathcal{K}$, error space $\mathcal{E}$ with $\mathcal{E} \cap \{0,1\}^* = \emptyset$, and maximum number* $\mathsf{maxmsg} \in \mathbb{N} \cup \{\infty\}$ *of messages supported per phase consists of four efficient algorithms defined as follows.*

– $\mathsf{Init}(1^\lambda) \xrightarrow{\$} (\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_{S,0}, \mathsf{st}_{R,0})$. *This probabilistic algorithm is composed of three algorithms:*
  - $\mathsf{MasterKeyGen}(1^\lambda) \xrightarrow{\$} \mathsf{msk}_0$. *On input security parameter 1^λ, this probabilistic algorithm outputs an initial master secret key $\mathsf{msk}_0 \in \mathcal{MSK}$.*
  - $\mathsf{KeyDerive}(\mathsf{msk}) \to \mathsf{K}$. *On input a master secret key* msk*, this deterministic algorithm outputs a phase key $\mathsf{K} \in \mathcal{K}$. The initial phase key is derived as $\mathsf{K}_0 \leftarrow \mathsf{KeyDerive}(\mathsf{msk}_0)$.*
  - $\mathsf{StateGen}(1^\lambda) \to (\mathsf{st}_{S,0}, \mathsf{st}_{R,0})$. *On input 1^λ, this deterministic algorithm outputs initial sending and receiving states $\mathsf{st}_{S,0} \in \mathcal{S}_S$ resp. $\mathsf{st}_{R,0} \in \mathcal{S}_R$.*

– $\mathsf{Send}(\mathsf{st}_{S,t}, \mathsf{K}_t, m) \xrightarrow{\$} (\mathsf{st}'_{S,t}, c)$. *On input of a sending state $\mathsf{st}_{S,t} \in \mathcal{S}_S$, a key $\mathsf{K}_t \in \mathcal{K}$, and a message $m \in \{0,1\}^*$, this (possibly) probabilistic algorithm outputs an updated state $\mathsf{st}'_{S,t} \in \mathcal{S}_S$ and a ciphertext (or error symbol) $c \in \{0,1\}^* \cup \mathcal{E}$.*

– $\mathsf{Recv}(\mathsf{st}_{R,t}, \mathsf{K}_t, c) \to (\mathsf{st}'_{R,t}, m)$. *On input of a receiving state $\mathsf{st}_{R,t} \in \mathcal{S}_R$, a key $\mathsf{K}_t \in \mathcal{K}$, and a ciphertext $c \in \{0,1\}^*$, this deterministic algorithm outputs an updated state $\mathsf{st}'_{R,t} \in \mathcal{S}_R$ and a message (or error symbol) $m \in \{0,1\}^* \cup \mathcal{E}$.*

– $\mathsf{Update}(\mathsf{msk}_t, \mathsf{st}_{S,t}/\mathsf{st}_{R,t}) \to (\mathsf{msk}_{t+1}, \mathsf{K}_{t+1}, \mathsf{st}_{S,t+1}/\mathsf{st}_{R,t+1})$. *This deterministic algorithm is composed of the following two algorithms:*
  - $\mathsf{MasterKeyUp}(\mathsf{msk}_t) \to \mathsf{msk}_{t+1}$. *On input of a master secret key $\mathsf{msk}_t \in \mathcal{MSK}$, this deterministic algorithm outputs a master secret key $\mathsf{msk}_{t+1} \in \mathcal{MSK}$ for the next phase.*
  - $\mathsf{StateUp}(\mathsf{st}_{S,t}/\mathsf{st}_{R,t}) \to \mathsf{st}_{S,t+1}/\mathsf{st}_{R,t+1}$. *On input of a sending or receiving state $\mathsf{st}_{S,t} \in \mathcal{S}_S$ resp. $\mathsf{st}_{R,t} \in \mathcal{S}_R$, this deterministic algorithm derives the next phase's state $\mathsf{st}_{S,t+1} \in \mathcal{S}_S$, resp. $\mathsf{st}_{R,t+1} \in \mathcal{S}_R$.*

  It further employs the (same) deterministic algorithm $\mathsf{KeyDerive}$ *as given for* Init *to derive an updated phase key $\mathsf{K}_{t+1} \in \mathcal{K}$ as $\mathsf{K}_{t+1} \leftarrow \mathsf{KeyDerive}(\mathsf{msk}_{t+1})$.*

We call a channel with a deterministic Send algorithm a *deterministic* multi-key channel.

[2] In the context of TLS 1.3, for example, both variants have been discussed. The current draft design [43] specifies that key update notifications are transmitted (and authenticated) within the data channel.

Shorthand notation. Given a sending state $\mathsf{st}_S \in \mathcal{S}_S$, a phase key $\mathsf{K} \in \mathcal{K}$, an integer $\ell \geq 0$, and a vector of messages $\mathbf{m} = (m_1, \ldots, m_\ell) \in (\{0,1\}^*)^\ell$, let $(\mathsf{st}'_S, \mathbf{c}) \xleftarrow{\$} \mathsf{Send}(\mathsf{st}_S, \mathsf{K}, \mathbf{m})$ be shorthand for the sequential execution $(\mathsf{st}^1_S, c_1) \xleftarrow{\$} \mathsf{Send}(\mathsf{st}^0_S, \mathsf{K}, m_1), \ldots, (\mathsf{st}^\ell_S, c_\ell) \xleftarrow{\$} \mathsf{Send}(\mathsf{st}^{\ell-1}_S, \mathsf{K}, m_\ell)$ with $\mathbf{c} = (c_1, \ldots, c_\ell)$, $\mathsf{st}^0_S = \mathsf{st}_S$, and $\mathsf{st}'_S = \mathsf{st}^\ell_S$. For $\ell = 0$ we define $\mathbf{c}$ to be the empty vector and the final state $\mathsf{st}^\ell_S = \mathsf{st}'_S$ to be the initial state st_S. We use an analogous notation for the Recv algorithm.

Correctness of multi-key channels intuitively guarantees that if at the receiver side the keys are updated only after having received all messages sent in the previous phase, then the received messages are equal to those sent in the entire communication.

Definition 2 (Correctness of multi-key channels). *Let $t \in \mathbb{N}$ and $(\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_{S,0}, \mathsf{st}_{R,0}) \xleftarrow{\$} \mathsf{Init}(1^\lambda)$. Let $\mathbf{m}_0, \ldots, \mathbf{m}_t \in \{0,1\}^{**}$ be $t+1$ vectors of messages of lengths $|\mathbf{m}_i| \leq \mathsf{maxmsg}$ (for $i \in \{0, \ldots, t\}$). Let $\mathbf{c_0}, \ldots, \mathbf{c_t} \in \{0,1\}^{**}$ be the corresponding ciphertext vectors output by* Send *given that* Update *is invoked between each sending of two subsequent message sequences, i.e., such that for $k = 0, \ldots, t$, $(\mathsf{st}'_{S,k}, \mathbf{c_k}) \xleftarrow{\$} \mathsf{Send}(\mathsf{st}_{S,k}, \mathsf{K}_k, \mathbf{m}_k)$ and for $k = 0, \ldots, t-1$, $(\mathsf{msk}_{k+1}, \mathsf{K}_{k+1}, \mathsf{st}_{S,k+1}) \leftarrow \mathsf{Update}(\mathsf{msk}_k, \mathsf{st}'_{S,k})$.*

Now let $\mathbf{m}'_0, \ldots, \mathbf{m}'_t \in \{0,1\}^{**}$ be the results of receiving these ciphertexts with likewise interleaved Update *invocations on the receiver's side, i.e., for $k = 0, \ldots, t$, let $(\mathsf{st}'_{R,k}, \mathbf{m}'_k) \leftarrow \mathsf{Recv}(\mathsf{st}_{R,k}, \mathsf{K}_k, \mathbf{c_k})$ and for $k = 0, \ldots, t-1$, let $(\mathsf{msk}_{k+1}, \mathsf{K}_{k+1}, \mathsf{st}_{R,k+1}) \leftarrow \mathsf{Update}(\mathsf{msk}_k, \mathsf{st}'_{R,k})$.*

We say that a multi-key channel Ch *is* correct *if for any choice of t, $\mathbf{m}_0, \ldots, \mathbf{m}_t$, and all choices of the randomness in the channel algorithm it holds that $\mathbf{m}_0 = \mathbf{m}'_0, \ldots, \mathbf{m}_t = \mathbf{m}'_t$.*

2.1 Syntax Rationale

The syntax of a cryptographic component defines its design space and also drives the security properties it may achieve. Before we continue with defining security for multi-key channels, let us pause to provide some rationale for our choices in the given syntax.

Probabilistic vs. deterministic Send. At first glance, the modeling of secure channels in form of stateful encryption [9] may appear as merely a stateful variant of authenticated encryption. For authenticated encryption (optionally with associated data), the established notion is a deterministic one [44], where encryption instead of fresh randomness takes a (unique) nonce. One major motivation for this approach is that (good) randomness may be hard to obtain in practice, e.g., due to design flaws or implementation bugs in random number generators, or limited system entropy available. Ideally, one hence bootstraps an encryption scheme from a (short) random key and then only relies on a unique nonce (e.g., a counter) for message encryption.[3]

[3] See the work originating from [46] on (nonce-misuse) resistance to non-unique nonces.

The same argument in principle applies to secure channels, yielding the question whether the Send algorithm should be fixed as deterministic. As we will see next, our security model allows us to seamlessly capture the desired security properties for channels with probabilistic and deterministic Send at the same time. We hence decided to stay in line with previous formalizations of channels (including [9,42,16,29]) and use the more generic syntax with (possibly) probabilistic Send. Nevertheless, we deem a *deterministic* multi-key channel to be the more desirable variant in practice. Indeed, the generic construction we provide in Section 4 is deterministic.

Inputs to key updates. We define updates of master secret and phase keys (via $\mathsf{MasterKeyUp}$ and $\mathsf{KeyDerive}$) to be deterministically derived from the initial master secret key msk_0. They are hence necessarily equivalent (in each phase) on the sender and receiver side.

A design alternative would be to also include the current state in the derivation, enabling keys to be influenced by, e.g., the message history. We however decided to focus on deterministic updates from msk_0, for mainly two reasons (besides significantly reducing the security model's complexity). First, this approach captures the concept of separating key derivation from message sending, in particular if master secrets are kept in more secure memory. Second, the syntax is compliant with both theoretical concepts for forward-secret encryption [13] as well as the practical key schedule employed in TLS 1.3 [43]. Note that, still, channels can for example take the message history into account within the Send and Recv algorithms.

3 Security Notions for Multi-key Channels

Classically, two security properties are expected from a secure channel. *Confidentiality* aims at protecting the content of transported messages from being read by eavesdroppers or active adversaries on the network. In contrast, *integrity* ensures that messages are received unmodified and in correct order, i.e., without messages being reordered or intermediate messages being dropped. We take up these notions in the context of multi-key channels and extend them to capture two more advanced security aspects arising in this scenario which we denote as *forward security* and *phase-key insulation*.

Forward security, as established also in other settings, is concerned with the effects of leaking a channel's master secret key on prior communication. The notion aims at situations where all key material of a communication partner becomes known to an attacker, e.g., through a break-in into a system or exfiltration of secrets. Following common terminology, we demand that a forward-secure multi-key channel upholds both confidentiality and integrity for messages sent in phases *before* corruption of a master secret key took place, even if one endpoint of the channel is still processing data in these phases when the corruption happens. Naturally, as the deterministic key schedule implies that the current and any future phase's key can be derived from a master secret key, we however cannot expect confidentiality or integrity for messages sent from the point of corruption on.

Phase-key insulation in contrast captures the selective leakage of some phases' keys while the master secret key remains uncompromised. Such leakage may be due to cryptanalysis of some of these keys, partial misuse of the key material, or temporary compromise. In particular, it reflects that the master secret key of a channel may be stored in more secure memory (e.g., trusted hardware) while the current phase key potentially resides in lesser secured memory for performance reasons. From a phase-key–insulated multi-key channel we demand, on a high level, that confidentiality and integrity in a certain phase is not endangered by the leakage of keys in prior or later phases.

3.1 Confidentiality

The established way of modeling confidentiality for channels is by demanding that the encryptions of two (left and right) sequences of messages are indistinguishable [31,9]. Formally, an adversary sequentially inputs pairs of messages m_0, m_1 of its choice to a sending oracle $\mathcal{O}_{\mathsf{Send}}$ and is given the encryption c_b of always either the first or the second message depending on an initially fixed, random challenge bit $b \xleftarrow{\$} \{0,1\}$. The adversary's task is to finally determine b. Hence, the corresponding security notion is established under the name of indistinguishability under chosen-plaintext attacks ($\mathsf{IND\text{-}CPA}$). In the stronger setting of chosen-ciphertext attacks ($\mathsf{IND\text{-}CCA}$), the adversary is additionally given a receiving oracle $\mathcal{O}_{\mathsf{Recv}}$ with the limitation that it may not query it on challenge ciphertexts, in a way to be defined later.

In the multi-key setting however, the advanced security aspects of forward security and particularly phase-key insulation render it impossible to use a single challenge bit throughout all phases. An adversary that adaptively learns keys for some phases is immediately able to learn whether the left or the right messages were encrypted in these phases. If this would be a fixed choice for all phases, the adversary could also tell which messages were encrypted in all other phases. In our formalization of multi-key confidentiality we hence deploy a separate challenge bit b_i for each phase i, chosen independently at random. This allows us to capture the expected insulation of phases against compromises in other phases and, ultimately, later corruption.

We define confidentiality in a modular notion s-IND-$k\mathsf{ATK}$ through the experiment $\mathsf{Expt}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}$ given in Figure 2. The experiment is parameterized with s, k, and ATK.

- The parameter s specifies the advanced security aspects captured in the notion and can be either empty or take one of the values ki, fs, or fski. As expected, fs indicates that the notion ensures forward security and ki denotes that the notion demands phase-key insulation; for fski both properties are integrated. Forward security is modeled through allowing the adversary to corrupt the master secret key at some point through a corruption oracle $\mathcal{O}_{\mathsf{Corrupt}}$. When ensuring phase-key insulation, the adversary is given a

```
$\mathsf{Expt}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}(1^\lambda)$:
$(\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_S, \mathsf{st}_R) \xleftarrow{\$} \mathsf{Init}(1^\lambda)$
$t_S \leftarrow 0,\ t_R \leftarrow 0$
$b_0 \xleftarrow{\$} \{0,1\}$
$i_0 \leftarrow 0,\ j_0 \leftarrow 0$
$\mathsf{sync} \leftarrow 1$
$t_{corr} \leftarrow +\infty$
$Rev \leftarrow \emptyset$
$(t, b) \xleftarrow{\$} \mathcal{A}(1^\lambda)^{\mathcal{O}_{\mathsf{LoR}}, [\mathcal{O}_{\mathsf{Recv}}]_{\mathsf{ATK}=\mathsf{CCA}}, [\mathcal{O}_{\mathsf{Update}}]_{k=\mathsf{mk}}, [\mathcal{O}_{\mathsf{Reveal}}]_{s\in\{\mathsf{ki},\mathsf{fski}\}}, [\mathcal{O}_{\mathsf{Corrupt}}]_{s\in\{\mathsf{fs},\mathsf{fski}\}}}$
if $t > \max(t_S, t_R)$ then
    return 0
return $((b_t = b) \wedge (t \notin Rev) \wedge (t < t_{corr}))$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{LoR}}(m_0, m_1)$:
if $|m_0| \neq |m_1|$ then
    return $\lightning$
$i_{t_S} \leftarrow i_{t_S} + 1$
$(\mathsf{st}_S, \mathbf{C}[t_S][i_{t_S}]) \xleftarrow{\$}$
        $\mathsf{Send}(\mathsf{st}_S, \mathsf{K}_{t_S}, m_{b_{t_S}})$
if $t_R > t_S$ and $t_S \notin Rev$ then
    $\mathsf{sync} \leftarrow 0$
return $\mathbf{C}[t_S][i_{t_S}]$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Recv}}(c)$:
$j_{t_R} \leftarrow j_{t_R} + 1$
$(\mathsf{st}_R, m) \leftarrow \mathsf{Recv}(\mathsf{st}_R, \mathsf{K}_{t_R}, c)$
if $(t_R > t_S$ or $j_{t_R} > i_{t_R}$
        or $c \neq \mathbf{C}[t_R][j_{t_R}])$
        and $t_R \notin Rev$ then
    $\mathsf{sync} \leftarrow 0$
if $\mathsf{sync} = 0$ then
    return $m$
else
    return $\lightning$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Update}}(role)$:
$(\mathsf{msk}_{t_{role}+1}, \mathsf{K}_{t_{role}+1}, \mathsf{st}_{role}) \leftarrow$
        $\mathsf{Update}(\mathsf{msk}_{t_{role}}, \mathsf{st}_{role})$
if $role = R$ and $t_S \geq t_R$ and $j_{t_R} < i_{t_R}$
        and $t_R \notin Rev$ then
    $\mathsf{sync} \leftarrow 0$
$t_{role} \leftarrow t_{role} + 1$
$\mathsf{st}^{begin}_{role,t_{role}} \leftarrow \mathsf{st}_{role}$
if $role = S$ then
    $b_{t_S} \xleftarrow{\$} \{0,1\}$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Reveal}}(t, role)$:
if $t > t_{role}$ then
    return $\lightning$
$Rev \leftarrow Rev \cup \{t\}$
return $(\mathsf{st}^{begin}_{role,t}, \mathsf{K}_t)$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Corrupt}}(role)$:
if $t_{corr} < +\infty$ then
    return $(\mathsf{st}^{begin}_{role,t_{corr}}, \mathsf{msk}_{t_{corr}})$
$t_{corr} \leftarrow t_{role}$
return $(\mathsf{st}^{begin}_{role,t_{role}}, \mathsf{msk}_{t_{role}})$
```

Fig. 2: Security experiment for *confidentiality* ($s\mathsf{IND}\text{-}k\mathsf{ATK}$) of a multi-key channel Ch. An adversary $\mathcal{A}$ has only access to an oracle $[\mathcal{O}_\mathsf{X}]_c$ if the condition c is satisfied.

reveal oracle $\mathcal{O}_{\mathsf{Reveal}}$ which allows it to selectively learn the keys of some phases.

– Via the parameter k, we capture both single-key (sk) and multi-key (mk) security notions in a single experiment. To model the single-key setting, we simply drop the adversary's capability to proceed to a next phase via an $\mathcal{O}_{\mathsf{Update}}$ oracle, essentially restricting it to a single phase (and hence key).
– Finally, the parameter ATK distinguishes between chosen-plaintext ($\mathsf{ATK} = \mathsf{CPA}$) and chosen-ciphertext ($\mathsf{ATK} = \mathsf{CCA}$) attacks. While the adversary always has access to a left-or-right encryption oracle $\mathcal{O}_{\mathsf{LoR}}$, the receiving oracle $\mathcal{O}_{\mathsf{Recv}}$ is only available for notions with CCA attacks.

The adversary finally has to output a phase t and a bit guess b and wins if the challenge bit used in phase t by the left-or-right oracle $\mathcal{O}_{\mathsf{LoR}}$ is equal to b and the targeted challenge phase t is neither revealed nor affected by corruption (i.e., $t < t_{corr}$, where t_{corr} is the corrupted phase, initialized to infinity).

In order to prevent trivial attacks, we have to restrict the output of adversarial queries to the receiving oracle $\mathcal{O}_{\mathsf{Recv}}$ in the setting of chosen-ciphertext attacks. Obviously, if $\mathcal{O}_{\mathsf{Recv}}$ outputs the message decrypted on input the unmodified challenge ciphertext sequence, the challenge bit used in $\mathcal{O}_{\mathsf{LoR}}$ would be immediately distinguishable. Still, as the Recv algorithm is stateful, we must allow the adversary to first make this algorithm proceed to a certain, potentially vulnerable state, before mounting its attack. For this purpose, we follow Bellare et al. [9] in suppressing the output of the Recv algorithm as long as the adversary's inputs to $\mathcal{O}_{\mathsf{Recv}}$ are *in sync* with the challenge ciphertext sequence output by $\mathcal{O}_{\mathsf{LoR}}$. As soon as synchronization is lost though, $\mathcal{O}_{\mathsf{Recv}}$ returns the output of the receiving algorithm Recv to the adversary.

Defining what it means to be in sync now becomes the crucial task in defining CCA security: we want to make the security notion as strong as possible without allowing trivial attacks. Intuitively, $\mathcal{O}_{\mathsf{Recv}}$ stays in sync (denoted by a flag $\mathsf{sync} = 1$) and decryptions are suppressed as long as the adversary forwards ciphertexts to $\mathcal{O}_{\mathsf{Recv}}$ that are obtained from $\mathcal{O}_{\mathsf{LoR}}$ in the same phase. So far, this is essentially a transcription of the stateful encryption definition of CCA security ($\mathsf{IND\text{-}sfCCA}$ [9]) to the multi-key setting with multiple phases. When targeting forward security and phase-key insulation, we however also need to consider how to define synchronization in phases where the adversary knows the key. Obviously, in such phases we cannot demand that a channel can strictly distinguish adversarial encryptions from the honest ciphertext sequence generated in $\mathcal{O}_{\mathsf{LoR}}$ as the adversary may simply replicate the latter's behavior. We accordingly do not consider synchronization to become lost in revealed phases. Still, we demand that a secure channel notices modifications later in uncompromised phases. Moreover, it should even detect truncations at the end of an uncompromised phase if the next phase's key is revealed, latest when the channel recovers from temporary compromise and enters the next, uncompromised phase.[4] We hence, additionally to the regular stateful encryption setting, define

[4] Recall that we consider key updates to be unauthenticated, possibly transmitted out-of-band.

synchronization to be lost if the receiver proceeds from an uncompromised phase to the next phase without having received all sent ciphertexts, or if the sender issues a ciphertext in a phase when the receiver already proceeded to the next phase.

In the following we describe the functionality and purpose of the oracles in the multi-key confidentiality experiment in Figure 2 in detail.

- The $\mathcal{O}_{\mathsf{LoR}}$ oracle can be queried with a pair of messages (m_0, m_1) of equal length. It responds with the output of Send on message $m_{b_{t_S}}$, where b_{t_S} is the challenge bit for the current sending phase t_S.
  If the receiver already proceeded to a later phase, the sent message cannot be received correctly anymore. As long as the key of the sender's phase is unrevealed, we hence declare synchronization to be lost (setting $\mathsf{sync} \leftarrow 0$). The restriction to uncompromised phases is necessary to prevent trivial attacks where the adversary leverages the phase key to, e.g., make the receiver process more messages than sent earlier to cover up the mismatch.
- The $\mathcal{O}_{\mathsf{Recv}}$ oracle can only be queried if $\mathsf{ATK} = \mathsf{CCA}$. On input a ciphertext c, $\mathcal{O}_{\mathsf{Recv}}$ computes the corresponding messages obtained under Recv. In case the receiving oracle is ahead in phase, has received more messages than sent, or c deviates from the corresponding sent ciphertext, synchronization is lost (again, to ignore trivial forgeries, as long the receiver's current phase is unrevealed). Finally, if still in sync, $\mathcal{O}_{\mathsf{Recv}}$ suppresses the message output and returns an according flag $\lightning$ to the adversary $\mathcal{A}$. Otherwise it provides $\mathcal{A}$ with the obtained message m.
- The $\mathcal{O}_{\mathsf{Update}}$ oracle is only available if $k = \mathsf{mk}$. Using the oracle, the adversary can separately make both the sender or receiver proceed to the next phase, updating their master secret, phase key, and state. If the sender side is updated, a new challenge bit for the new phase is chosen at random. Moreover, the experiment goes out of sync if the receiver side is updated too soon, i.e., without having received all sent ciphertexts, and the receiver's phase is not revealed.
- The $\mathcal{O}_{\mathsf{Reveal}}$ oracle can be used by the adversary to obtain the key of any phase t (along with this phase's initial sender resp. receiver state) and is accessible if $s \in \{\mathsf{ki}, \mathsf{fski}\}$. Phase t is then added to a set of revealed phases Rev.
- The $\mathcal{O}_{\mathsf{Corrupt}}$ oracle is provided if $s \in \{\mathsf{fs}, \mathsf{fski}\}$. Upon the first call, the adversary obtains for a chosen role *role* the current phase's master secret key and initial state. This phase is then recorded as the phase of corruption t_{corr} for later comparison. If a corruption has already taken place (i.e., $t_{corr} < +\infty$), the adversary can obtain the other role's initial state in the corrupted phase via a further $\mathcal{O}_{\mathsf{Corrupt}}$ call. For simplicity, we assume the state to be empty in phases not yet entered. Observe that it suffices to consider a single point in time for corruption, as later master keys are deterministically derived from the corrupted one.

Definition 3 (s-IND-kATK Security). *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Update})$ *be a multi-key channel and experiment* $\mathsf{Expt}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}(1^\lambda)$ *for an adversary* $\mathcal{A}$ *be defined as in Figure 2.*

The security experiment is parameterized in three directions: s, k, and ATK*. The parameter s indicates the advanced security aspects and can take one of the values* ki *(phase-key–insulated),* fs *(forward-secure),* fski *(forward-secure and phase-key–insulated), or the empty string*[5] *(plain / neither forward-secure nor phase-key–insulated). The parameter k integrates both single-key (*sk*) and multi-key (*mk*) security notions in a single experiment. Finally, the parameter* ATK *distinguishes between chosen-plaintext (*$\mathsf{ATK} = \mathsf{CPA}$*) and chosen-ciphertext (*$\mathsf{ATK} = \mathsf{CCA}$*) security.*

Within the experiment the adversary $\mathcal{A}$ always has access to a left-or-right sending oracle $\mathcal{O}_{\mathsf{LoR}}$. Moreover, $\mathcal{A}$ has access to a receiving oracle $\mathcal{O}_{\mathsf{Recv}}$ if $\mathsf{ATK} = \mathsf{CCA}$*, an update oracle $\mathcal{O}_{\mathsf{Update}}$ if $k = \mathsf{mk}$, a key-reveal oracle $\mathcal{O}_{\mathsf{Reveal}}$ if $s \in \{\mathsf{ki}, \mathsf{fski}\}$, and finally a corruption oracle $\mathcal{O}_{\mathsf{Corrupt}}$ if $s \in \{\mathsf{fs}, \mathsf{fski}\}$.*

We say that the channel Ch *provides* indistinguishability under multi-key (resp. single-key) chosen-plaintext (resp. chosen-ciphertext) attacks *(s-IND-$k\mathsf{CPA}$ resp. s-IND-$k\mathsf{CCA}$ for $k = \mathsf{mk}$ resp. $k = \mathsf{sk}$), potentially* with forward security *(if $s \in \{\mathsf{fs}, \mathsf{fski}\}$)* and/or phase-key insulation *(if $s \in \{\mathsf{ki}, \mathsf{fski}\}$) if for all PPT adversaries $\mathcal{A}$ the following advantage function is negligible in the security parameter:*

$$\mathsf{Adv}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Expt}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2}.$$

Our generic confidentiality notion in Definition 3 captures as its weakest variant indistinguishability under single-key chosen-plaintext attacks ($\mathsf{IND\text{-}skCPA}$) and as its strongest variant indistinguishability under multi-key chosen-ciphertext attacks with forward security and phase-key insulation ($\mathsf{fski\text{-}IND\text{-}mkCCA}$). We discuss the relations among these notions in more detail in Section 3.4.

3.2 Integrity

Integrity is traditionally defined in two flavors: integrity of plaintexts ($\mathsf{INT\text{-}PTXT}$) and integrity of ciphertexts ($\mathsf{INT\text{-}CTXT}$) [10], with according stateful-encryption analogs $\mathsf{INT\text{-}sfPTXT}$ [18] and $\mathsf{INT\text{-}sfCTXT}$ [9]. Integrity of plaintexts intuitively ensures that no adversary is able to make the receiver output a valid message that differs from the previously sent (sequence of) messages. The stronger notion of ciphertext integrity ensures that no adversary can make the receiver output any valid, even recurring message by inputting a forged or modified ciphertext.

Similarly to confidentiality, we define a modular multi-key integrity notion s-INT-$k\mathsf{ATK}$, given through the experiment $\mathsf{Expt}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}$ in Figure 3. Again, the notion is parameterized to integrate forward security and phase-key insulation (via s), the single- and multi-key setting (via k), as well as the two attack targets, $\mathsf{ATK} = \mathsf{PTXT}$ and $\mathsf{ATK} = \mathsf{CTXT}$. An adversary $\mathcal{A}$ against the experiment $\mathsf{Expt}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}$ has access to a sending oracle $\mathcal{O}_{\mathsf{Send}}$ (in contrast to confidentiality without left-or-right functionality), one of two receiving oracles $\mathcal{O}^{\mathsf{ATK}}_{\mathsf{Recv}}$ depending on ATK, and—depending on the advanced security properties and

[5] For legibility, we also drop the leading dash in a notion s-IND-$k\mathsf{ATK}$ if s is the empty string and simply write IND-$k\mathsf{ATK}$ in this case.

```
$\mathsf{Expt}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{ATK}}_{\mathsf{Ch},\mathcal{A}}(1^\lambda)$:
  $(\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_S, \mathsf{st}_R) \xleftarrow{\$} \mathsf{Init}(1^\lambda)$
  $t_S \leftarrow 0$, $t_R \leftarrow 0$
  $i_0 \leftarrow 0$, $j_0 \leftarrow 0$
  $\mathsf{sync} \leftarrow 1$
  $\mathsf{win} \leftarrow 0$
  $t_{corr} \leftarrow +\infty$
  $Rev \leftarrow \emptyset$
  $\mathcal{A}(1^\lambda)^{\mathcal{O}_{\mathsf{Send}}, \mathcal{O}^{\mathsf{ATK}}_{\mathsf{Recv}}, [\mathcal{O}_{\mathsf{Update}}]_{k=\mathsf{mk}}, [\mathcal{O}_{\mathsf{Reveal}}]_{s\in\{\mathsf{ki},\mathsf{fski}\}}, [\mathcal{O}_{\mathsf{Corrupt}}]_{s\in\{\mathsf{fs},\mathsf{fski}\}}}$
  return $\mathsf{win}$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Send}}(m)$:
  $i_{t_S} \leftarrow i_{t_S} + 1$
  $(\mathsf{st}_S, \mathbf{C}[t_S][i_{t_S}]) \xleftarrow{\$} \mathsf{Send}(\mathsf{st}_S, \mathsf{K}_{t_S}, m)$
  $\mathbf{M}[t_S][i_{t_S}] \leftarrow m$
  if $t_R > t_S$ and $t_S \notin Rev$ then
    $\mathsf{sync} \leftarrow 0$
  return $\mathbf{C}[t_S][i_{t_S}]$

If $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{PTXT}}_{\mathsf{Recv}}(c)$:
  $j_{t_R} \leftarrow j_{t_R} + 1$
  $(\mathsf{st}_R, m) \leftarrow \mathsf{Recv}(\mathsf{st}_R, \mathsf{K}_{t_R}, c)$
  if $m \neq \mathbf{M}[t_R][j_{t_R}]$ and $m \notin \mathcal{E}$
        and $t_R \notin Rev$
        and $t_R < t_{corr}$ then
    $\mathsf{win} \leftarrow 1$
  return $m$

If $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{CTXT}}_{\mathsf{Recv}}(c)$:
  $j_{t_R} \leftarrow j_{t_R} + 1$
  $(\mathsf{st}_R, m) \leftarrow \mathsf{Recv}(\mathsf{st}_R, \mathsf{K}_{t_R}, c)$
  if $(t_R > t_S$ or $j_{t_R} > i_{t_R}$
        or $c \neq \mathbf{C}[t_R][j_{t_R}])$
        and $t_R \notin Rev$ then
    $\mathsf{sync} \leftarrow 0$
  if $\mathsf{sync} = 0$ and $m \notin \mathcal{E}$
        and $t_R \notin Rev$
        and $t_R < t_{corr}$ then
    $\mathsf{win} \leftarrow 1$
  return $m$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Update}}(role)$:
  $(\mathsf{msk}_{t_{role}+1}, \mathsf{K}_{t_{role}+1}, \mathsf{st}_{role}) \leftarrow$
        $\mathsf{Update}(\mathsf{msk}_{t_{role}}, \mathsf{st}_{role})$
  if $role = R$ and $t_S \geq t_R$ and $j_{t_R} < i_{t_R}$
        and $t_R \notin Rev$ then
    $\mathsf{sync} \leftarrow 0$
  $t_{role} \leftarrow t_{role} + 1$
  $\mathsf{st}^{begin}_{role,t_{role}} \leftarrow \mathsf{st}_{role}$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Reveal}}(t, role)$:
  if $t > t_{role}$ then
    return $\lightning$
  $Rev \leftarrow Rev \cup \{t\}$
  return $(\mathsf{st}^{begin}_{role,t}, \mathsf{K}_t)$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Corrupt}}(role)$:
  if $t_{corr} < +\infty$ then
    return $(\mathsf{st}^{begin}_{role,t_{corr}}, \mathsf{msk}_{t_{corr}})$
  $t_{corr} \leftarrow t_{role}$
  return $(\mathsf{st}^{begin}_{role,t_{role}}, \mathsf{msk}_{t_{role}})$
```

Fig. 3: Security experiment for *integrity* ($s\mathsf{INT}\text{-}k\mathsf{ATK}$) of a multi-key channel Ch. An adversary $\mathcal{A}$ has only access to an oracle $[\mathcal{O}_{\mathsf{X}}]_c$ if the condition c is satisfied.

key setting captured—oracles $\mathcal{O}_{\mathsf{Update}}$ (without setting a new challenge bit), and $\mathcal{O}_{\mathsf{Reveal}}$ and $\mathcal{O}_{\mathsf{Corrupt}}$, identical to those for confidentiality. In the integrity experiment, the adversary does not provide a particular challenge output, but instead needs to trigger a winning flag win to be set within the experiment run.

Beyond the sending oracle $\mathcal{O}_{\mathsf{Send}}$ only taking and encrypting a single message, the major difference to the confidentiality setting lies in the definition of the $\mathcal{O}_{\mathsf{Recv}}^{\mathsf{ATK}}$ oracle, which in particular comprises the winning condition check. Depending on the attack target, the adversary has access to either the $\mathcal{O}_{\mathsf{Recv}}^{\mathsf{PTXT}}$ or the $\mathcal{O}_{\mathsf{Recv}}^{\mathsf{CTXT}}$ variant of the receiving oracle. Both oracles first of all obtain a ciphertext c and provide the adversary $\mathcal{A}$ with the decrypted message m output by Recv on that ciphertext. Beyond this, they differ in assessing whether $\mathcal{A}$ has succeeded in breaking plaintext resp. ciphertext integrity (in which case they set $\mathsf{win} \leftarrow 1$):

– The $\mathcal{O}_{\mathsf{Recv}}^{\mathsf{PTXT}}$ oracle declares the adversary successful if the received message m differs from the corresponding sent message in this phase and position, given that the current receiving phase is neither revealed nor corrupted.
– The $\mathcal{O}_{\mathsf{Recv}}^{\mathsf{CTXT}}$ in contrast for winning requires that, on input an out-of-sync ciphertext in a phase neither revealed nor corrupted, Recv outputs a valid message m, i.e., $m \notin \mathcal{E}$ is not an error message.
  In the same way as for confidentiality, synchronization is considered to be lost on an $\mathcal{O}_{\mathsf{Recv}}$ oracle call if the receiving oracle, in a non-revealed phase, is ahead of the sending oracle in phase or message count, or if c deviates from the corresponding sent message. Furthermore, synchronization may be lost by non-aligned key updates on both sides of the channel, captured in $\mathcal{O}_{\mathsf{Send}}$ and $\mathcal{O}_{\mathsf{Update}}$ as in the confidentiality experiment (cf. Figure 2).

Definition 4 (s-INT-$k\mathsf{ATK}$ Security). *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Update})$ *be a multi-key channel and experiment* $\mathsf{Expt}_{\mathsf{Ch},\mathcal{A}}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{ATK}}(1^\lambda)$ *for an adversary* $\mathcal{A}$ *be defined as in Figure 3. The security experiment is parameterized via s, k, and* ATK*. Parameters s and k are as for confidentiality in Definition 3. The parameter* ATK *distinguishes between plaintext integrity (*$\mathsf{ATK} = \mathsf{PTXT}$*) and ciphertext integrity (*$\mathsf{ATK} = \mathsf{CTXT}$*).*

Within the experiment the adversary $\mathcal{A}$ has always access to a sending oracle $\mathcal{O}_{\mathsf{Send}}$ *and a receiving oracle* $\mathcal{O}_{\mathsf{Recv}}^{\mathsf{ATK}}$ *(the latter differs depending on* ATK*). Moreover, $\mathcal{A}$ has access to an update oracle* $\mathcal{O}_{\mathsf{Update}}$ *if* $k = \mathsf{mk}$*, a key-reveal oracle* $\mathcal{O}_{\mathsf{Reveal}}$ *if* $s \in \{\mathsf{ki}, \mathsf{fski}\}$*, and finally a corruption oracle* $\mathcal{O}_{\mathsf{Corrupt}}$ *if* $s \in \{\mathsf{fs}, \mathsf{fski}\}$*.*

We say that Ch *provides* multi-key (resp. single-key) integrity of plaintexts (resp. ciphertexts) *(*s-INT-$k\mathsf{PTXT}$ *resp.* s-INT-$k\mathsf{CTXT}$ *for* $k = \mathsf{mk}$ *resp.* $k = \mathsf{sk}$*), potentially* with forward security *(if* $s \in \{\mathsf{fs}, \mathsf{fski}\}$*)* and/or phase-key insulation *(if* $s \in \{\mathsf{ki}, \mathsf{fski}\}$*) if for all PPT adversaries $\mathcal{A}$ the following advantage function is negligible in the security parameter:*

$$\mathsf{Adv}_{\mathsf{Ch},\mathcal{A}}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{ATK}}(\lambda) := \Pr\left[\mathsf{Expt}_{\mathsf{Ch},\mathcal{A}}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{ATK}}(1^\lambda) = 1\right].$$

Remark 1. Note that the advanced properties of forward security and phase-key insulation are only reasonable to consider in the multi-key setting ($k = \mathsf{mk}$).

Indeed, for the single-key setting ($k = \mathsf{sk}$), the plain, fs, ki, and fski flavors of each notion collapse to being equivalent. For this, observe that an adversary in the single-key setting, lacking access to the $\mathcal{O}_{\mathsf{Update}}$ oracle, is restricted to the initial phase $t_S = t_R = 0$. At the same time, in order to win in this phase (by outputting a confidentiality guess resp. breaking integrity), it must neither reveal nor corrupt either of the parties. Hence, it effectively cannot make use of the $\mathcal{O}_{\mathsf{Reveal}}$ and $\mathcal{O}_{\mathsf{Corrupt}}$ queries, rendering both non-effective. Consequently, we can focus on only the plain version of our single-key security notions.

3.3 Modeling Rationale

As for the definition of syntax, there are choices to make when defining security for multi-key channels. Before further studying the relations among the confidentiality and integrity notions just set up, let us hence provide some rationale for aspects of our security model.

LoR vs. IND\$. In our confidentiality experiment, the adversary is challenged to (be unable to) distinguish encryptions of left-or-right (LoR) messages. In the stateless authenticated-encryption setting particularly for AEAD schemes [44], the established notion for defining confidentiality instead is the stronger indistinguishability from random strings (IND\$) [45].[6]

It might seem natural to adopt the strong IND\$ confidentiality for channels from its common building block AEAD. On second thought, however, this notion turns out to be inappropriate for secure channels. While AEAD is an invaluable building block, a channel is a higher-layer object in a more complex setting, aiming not only at confidentiality and integrity, but also at replay and reordering protection [9,35] as well as further aspects such as data processing [16,29]. For this purpose, channel protocols regularly include header information like length or content type fields within the output ciphertexts, rendering them clearly distinguishable from random strings. In our security definition, we hence stick to the left-or-right indistinguishability notion rightfully established through previous channel models including [9,42,16,29].

Multiple challenge bits. As pointed out earlier, using a single challenge bit across all phases in the confidentiality experiment is infeasible: an adaptive Reveal query for some phase would in this case also disclose the challenge phase's (same) bit. We hence deploy multiple, independent challenge bits for each phase.

Alternative options would be to employ a single challenge bit in one phase and provide regular (non–LoR) encryption oracles for all other phases, or to have the adversary choose whether to compromise a phase at its beginning. We however deem these approaches not only more complex, but most importantly less adaptive, as they prevent the adversary from retrospectively choosing (non-)challenge phases.

[6] A third variant, real-or-random (RoR) indistinguishability is equivalent to LoR indistinguishability [8]. See also Barwell et al. [5] for an (historical) overview of the security notions established for authenticated encryption.

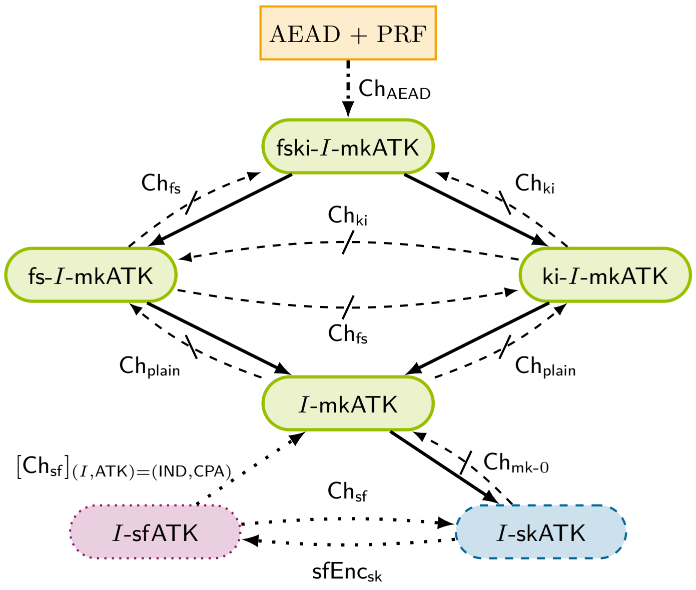

Fig. 4: Illustration of the relations between different flavors of confidentiality and integrity in our multi-key and single-key settings as well as for stateful encryption [9]. The variables I and ATK are placeholders for confidentiality notions ($I = \mathsf{IND}$ with $\mathsf{ATK} = \mathsf{CPA}/\mathsf{CCA}$) and integrity notions ($I = \mathsf{INT}$ with $\mathsf{ATK} = \mathsf{PTXT}/\mathsf{CTXT}$).
Rounded rectangles indicate multi-key (solid-line, green), single-key (dashed-line, blue), or stateful-encryption notions (dotted-line, purple); regular (orange) rectangles indicate building blocks. Solid arrows indicate trivial implications. Dashed, stroke-out arrows indicate separations and dotted arrows generic transforms we establish, both provided in Section 3.4. The dash-dotted arrow indicates the generic construction we provide in Section 4. Labels refer to the respective construction, with brackets $[\mathsf{X}]_c$ restricting a relation to condition c.

3.4 Relations Between Multi- and Single-key Notions

The modularity of our notions for multi-key confidentiality and integrity, parameterized by forward security and phase-key insulation, leads to a set of notions of varying strength. In the following, we establish that forward security and phase-key insulation are orthogonal properties; expectedly both adding to the strength of a security notion. Furthermore, we show that without forward security and phase-key insulation the single-key security notions of our framework are essentially equivalent to the respective established stateful encryption notions: we give generic, pure syntactical transforms to translate secure single-key schemes between the two realms. Figure 4 illustrates the relations we establish.

Trivial implications. First of all, let us observe the trivial implications between the security notions of our framework, indicated by solid arrows in Figure 4. Those implications arise by restricting the access to one (or multiple) oracles in the security experiments: a notion with access to a certain oracle immediately implies an otherwise identical notion without this oracle access. For instance, a $\mathsf{fski\text{-}IND\text{-}mkCPA}$-secure channel is also $\mathsf{ki\text{-}IND\text{-}mkCPA}$-secure, since if no adversary can distinguish left-or-right ciphertexts when being able to corrupt the master secret key, then doing so does not become easier when corruption is not a possibility.

Separations. We discuss the separations between notions possibly providing forward security and phase-key insulation starting from a multi-key channel that provides both properties at the example of indistinguishability under chosen-plaintext attacks. The cases of integrity and indistinguishability under chosen-ciphertext attacks are analogous. More precisely, let $\mathsf{Ch}_{\mathsf{fski}} := (\mathsf{Init}_{\mathsf{fski}}, \mathsf{Send}_{\mathsf{fski}}, \mathsf{Recv}_{\mathsf{fski}}, \mathsf{Update}_{\mathsf{fski}})$ be a multi-key channel which provides $\mathsf{fski\text{-}IND\text{-}mkCPA}$ security. Recall that master secret and phase keys are computed using two deterministic sub-algorithms $\mathsf{MasterKeyUp}_{\mathsf{fski}}$ and $\mathsf{KeyDerive}_{\mathsf{fski}}$, respectively.

Now we construct a new channel $\mathsf{Ch}_{\mathsf{fs}}$ which differs from $\mathsf{Ch}_{\mathsf{fski}}$ only in its key derivation algorithm, which we replace by the identity function, i.e., we define $\mathsf{KeyDerive}_{\mathsf{fs}}(\mathsf{msk}_i) := \mathsf{msk}_i$ for all phases $i \in \mathbb{N}$. As $\mathsf{MasterKeyUp}$ remains unmodified, $\mathsf{Ch}_{\mathsf{fs}}$ inherits the forward security of $\mathsf{Ch}_{\mathsf{fski}}$. Furthermore, observe that a revealed phase key (equal to the master secret key $\mathsf{K}_i = \mathsf{msk}_i$) can be iteratively used to compute the next master secret keys $\mathsf{msk}_{i+1} = \mathsf{MasterKeyUp}_{\mathsf{fs}}(\mathsf{msk}_i)$ and therefore also the next phase keys $\mathsf{K}_{i+1} = \mathsf{KeyDerive}_{\mathsf{fs}}(\mathsf{msk}_{i+1})$. As a result, $\mathsf{Ch}_{\mathsf{fs}}$ has dependent phase keys and hence only provides $\mathsf{fs\text{-}IND\text{-}mkCPA}$ security, but not $\mathsf{fski\text{-}IND\text{-}mkCPA}$ security, separating the two notions.

Next we build a channel $\mathsf{Ch}_{\mathsf{ki}}$ from $\mathsf{Ch}_{\mathsf{fski}}$ which has a master secret key space $\mathcal{MSK}_{\mathsf{ki}} = \mathcal{MSK}^{*}_{\mathsf{fski}}$ and updates its master secret keys using a function $\mathsf{MasterKeyUp}_{\mathsf{ki}}(\mathbf{msk}_i) := (\mathbf{msk}_i, \mathsf{MasterKeyUp}_{\mathsf{fski}}(\mathbf{msk}_i[i]))$, where $\mathbf{msk}_0 = (\mathsf{MasterKeyGen}_{\mathsf{fski}}(1^{\lambda}))$. In other words, $\mathsf{Ch}_{\mathsf{ki}}$ keeps a copy of all master secret keys generated so far in the current master secret key, and uses the last entry to derive the next master secret key. The phase keys are then derived from the last master secret key entry, i.e., we define $\mathsf{KeyDerive}_{\mathsf{ki}}(\mathbf{msk}_i) := \mathsf{KeyDerive}_{\mathsf{fski}}(\mathbf{msk}_i[i])$. While $\mathsf{Ch}_{\mathsf{ki}}$ provides the phase-key insulation of $\mathsf{Ch}_{\mathsf{fski}}$, forward security is lost. On corruption in any phase, all previous master secret keys are leaked, allowing an adversary to derive any previous phase key. Therefore $\mathsf{Ch}_{\mathsf{ki}}$ only provides $\mathsf{ki\text{-}IND\text{-}mkCPA}$ security, but not $\mathsf{fski\text{-}IND\text{-}mkCPA}$ security.

Combining the two modifications above leads to a channel $\mathsf{Ch}_{\mathsf{plain}}$ which only satisfies plain $\mathsf{IND\text{-}mkCPA}$ security, but neither $\mathsf{ki\text{-}IND\text{-}mkCPA}$ nor $\mathsf{fs\text{-}IND\text{-}mkCPA}$ security.

Finally, we consider the separation between the single-key notions and their corresponding multi-key notions, both without forward security and phase-key insulation. Again, we only discuss the notions $\mathsf{IND\text{-}skCPA}$ and $\mathsf{IND\text{-}mkCPA}$ as an example; the other cases follow identically. We build from an $\mathsf{IND\text{-}skCPA}$ secure single-key channel $\mathsf{Ch}_{\mathsf{sk}}$ a multi-key channel $\mathsf{Ch}_{\mathsf{mk}\text{-}0}$ which uses the single-key channel's key for the initial phase both as master secret and phase key. As the master secret key for the second and all following phases it then uses the zero-string, i.e., $\mathsf{MasterKeyUp}_{\mathsf{mk}}(\mathsf{msk}_i) := 0^\lambda$. Clearly the security is not preserved by $\mathsf{Ch}_{\mathsf{mk}\text{-}0}$ in any phase other than the initial one, in which it behaves exactly like $\mathsf{Ch}_{\mathsf{sk}}$. Hence, $\mathsf{Ch}_{\mathsf{mk}\text{-}0}$ is $\mathsf{IND\text{-}skCPA}$-secure, but not $\mathsf{IND\text{-}mkCPA}$-secure.

Generic Transforms Between Stateful Encryption and Multi-key Channels. To complete the picture, we finally study the relations between the established notions for secure channels, stateful authenticated encryption, and our notion of multi-key channels.

For this purpose, let us first briefly recall the notation for stateful encryption schemes as introduced by Bellare, Kohno, and Namprempre [9]. A stateful encryption scheme $\mathsf{sfEnc} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of the following three efficient algorithms. The randomized key generation algorithm $\mathsf{KGen}(1^\lambda) \xrightarrow{\$} (K, \mathsf{st}_\mathsf{E}, \mathsf{st}_\mathsf{D})$ outputs a key $K \in \mathcal{K}$ and initial encryption and decryption states st_E, st_D. The randomized, stateful encryption algorithm $\mathsf{Enc}(\mathsf{st}_\mathsf{E}, K, m) \xrightarrow{\$} (\mathsf{st}_\mathsf{E}', c)$ takes state, key, and a message m and outputs an updated state and ciphertext c. The deterministic, stateful decryption algorithm $\mathsf{Dec}(\mathsf{st}_\mathsf{D}, K, c) \xrightarrow{\$} (\mathsf{st}_\mathsf{D}', m)$ conversely maps state, key, and a ciphertext to an updated state and either a message or special error symbol $\perp$.

Clearly, stateful encryption does not aim at achieving the advanced security properties we consider in this work, forward security and phase-key insulation. In the comparison, we hence focus on the plain confidentiality and integrity notions, i.e., $\mathsf{IND}\text{-}k\mathsf{ATK}$ and $\mathsf{INT}\text{-}k\mathsf{ATK}$ (for both $k \in \{\mathsf{mk}, \mathsf{sk}\}$ and variants $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$ resp. $\mathsf{ATK} \in \{\mathsf{PTXT}, \mathsf{CTXT}\}$) in our framework as well as the stateful-encryption notions $\mathsf{IND\text{-}sfCPA}$ resp. $\mathsf{IND\text{-}sfCCA}$ and $\mathsf{INT\text{-}sfPTXT}$ resp. $\mathsf{INT\text{-}sfCTXT}$.

The relations we establish are twofold. First, our single-key security notions which allow an adversary to access a multi-key channel only in its initial phase are indeed equivalent in strength to the corresponding stateful-encryption notions, beyond syntactical differences. For this, consider the following natural and generic transforms for constructing a multi-key channel $\mathsf{Ch}_{\mathsf{sf}}$ from any stateful encryption scheme sfEnc and, conversely, a stateful encryption scheme $\mathsf{sfEnc}_{\mathsf{sk}}$ from any multi-key channel with single-entry error space $\mathcal{E} = \{\perp\}$.

– $\mathsf{Ch}_{\mathsf{sf}}(\mathsf{Init}_{\mathsf{sf}}, \mathsf{Send}_{\mathsf{sf}}, \mathsf{Recv}_{\mathsf{sf}}, \mathsf{Update}_{\mathsf{sf}})$.
  For initialization, derive $(K, \mathsf{st}_\mathsf{E}, \mathsf{st}_\mathsf{D}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$ and set $\mathsf{msk}_0 = \mathsf{K}_0 = K$, $\mathsf{st}_{S,0} = \mathsf{st}_\mathsf{E}$, and $\mathsf{st}_{R,0} = \mathsf{st}_\mathsf{D}$. For sending and receiving, use Enc and Dec as direct replacements. Finally, the Update algorithm does nothing; i.e., $\mathsf{StateUp}$, $\mathsf{MasterKeyUp}$, and $\mathsf{KeyDerive}$ are defined to be the identity function.

– $\mathsf{sfEnc}_{\mathsf{sk}}(\mathsf{KGen}_{\mathsf{sk}}, \mathsf{Enc}_{\mathsf{sk}}, \mathsf{Dec}_{\mathsf{sk}})$.
  For key generation, derive $(\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_{S,0}, \mathsf{st}_{R,0}) \xleftarrow{\$} \mathsf{Init}(1^\lambda)$ and set $K = \mathsf{msk}_0$, $\mathsf{st}_\mathsf{E} = \mathsf{st}_{S,0}$, and $\mathsf{st}_\mathsf{D} = \mathsf{st}_{R,0}$. Encryption and decryption is directly replaced by Send resp. Recv.

Careful inspection of the single-key ($k = \mathsf{sk}$) notions in our framework and those defined for stateful encryption [9,18][7] readily establishes that each two corresponding notions (i.e., I-skATK and I-sfATK for same I and ATK) are preserved by the generic transforms given above. That is, if the underlying stateful encryption scheme sfEnc achieves, e.g., IND-sfCCA security then the transformed multi-key channel $\mathsf{Ch}_{\mathsf{sf}}$ satisfies the corresponding IND-skCCA notion.

Finally, and perhaps surprisingly at first glance, our generic transform $\mathsf{Ch}_{\mathsf{sf}}$ of a stateful encryption scheme into a multi-key channel also achieves (plain) multi-key IND-mkCPA security. The reason for this is that the degenerated Update algorithm does not alter the key which hence also makes the $\mathcal{O}_{\mathsf{Send}}$ oracle not alter its behavior across different phases. On the other hand, the message resp. ciphertext vectors $\mathbf{M}$ resp. $\mathbf{C}$ in the $\mathcal{O}_{\mathsf{Recv}}$ oracle can be easily set out-of-sync by invoking Update at different positions in the ciphertext sequence on the sender and receiver side. As a result, an adversary can make challenge ciphertexts to be considered as valid forgery in a "different" phase (in the multi-key integrity game) or force challenge messages to be output by $\mathcal{O}_{\mathsf{Recv}}$ (in the IND-mkCCA game). Hence, $\mathsf{Ch}_{\mathsf{sf}}$ achieves neither IND-mkCCA nor INT-mkPTXT or INT-mkCTXT security.

3.5 Generic Composition

We round up the discussion of our framework of multi-key security notions by lifting the classical composition theorem by Bellare and Namprempre [10] for symmetric encryption, namely that IND-CPA and INT-CTXT security imply IND-CCA security, to the setting of multi-key channels. As noted by Boldyreva et al. [17], this result is not directly applicable in settings where the decryption algorithm may output multiple, distinguishable errors, an observation that also applies to our setting. Boldyreva et al. re-establish composition in the multiple-error setting by requiring that with overwhelming probability an adversary is only able to produce a single error (a notion they call *error invariance*). Here, we instead make use of the more versatile approach introduced as *error predictability* in the context of stream-based channels by Fischlin et al. [29]. Error predictability roughly requires that there exists an efficient *predictor* algorithm Pred that, given the ciphertexts sent and received so far, can with overwhelming probability predict the error message caused by receiving a certain next ciphertext (if that ciphertext produces at all an error).

In comparison, error predictability is a milder assumption than error invariance [17] as it allows for channels outputting multiple distinguishable and non-negligible errors. For stateless authenticated encryption, Barwell et al. [5] considered the alternative notion of *error simulatability* in which error leakage is simulated under an independent key. Their notion seems incomparable to error predictability in the stateful setting, where the history of ciphertexts needs to

[7] As a technical side-remark, we here consider a slight variant of stateful integrity where the adversary in the decryption oracle is given the decrypted message instead of only a bit telling whether decryption resulted in an error or not.

```
$\mathsf{Expt}^{s\text{-}k\mathsf{ERR\text{-}PRE}}_{\mathsf{Ch},\mathsf{Pred},\mathcal{A}}(1^\lambda)$:
  $(\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_S, \mathsf{st}_R) \xleftarrow{\$} \mathsf{Init}(1^\lambda)$
  $t_S \leftarrow 0$, $t_R \leftarrow 0$
  $i_0 \leftarrow 0$, $j_0 \leftarrow 0$
  $\mathcal{A}(1^\lambda)^{\mathcal{O}_{\mathsf{Send}}, \mathcal{O}_{\mathsf{Recv}}, [\mathcal{O}_{\mathsf{Update}}]_{k=\mathsf{mk}}, [\mathcal{O}_{\mathsf{Reveal}}]_{s \in \{\mathsf{ki},\mathsf{fski}\}}, [\mathcal{O}_{\mathsf{Corrupt}}]_{s \in \{\mathsf{fs},\mathsf{fski}\}}}$
  return $\mathsf{win}$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Send}}(m)$:
  $i_{t_S} \leftarrow i_{t_S} + 1$
  $(\mathsf{st}_S, \mathbf{C}_S[t_S][i_{t_S}]) \xleftarrow{\$}$
        $\mathsf{Send}(\mathsf{st}_S, \mathsf{K}_{t_S}, m)$
  return $\mathbf{C}_S[t_S][i_{t_S}]$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Recv}}(c)$:
  $j_{t_R} \leftarrow j_{t_R} + 1$
  $(\mathsf{st}_R, m) \leftarrow \mathsf{Recv}(\mathsf{st}_R, \mathsf{K}_{t_R}, c)$
  if $m \in \mathcal{E}$ and
        $m \neq \mathsf{Pred}(\mathbf{C}_S, \mathbf{C}_R, c)$ then
    $\mathsf{win} \leftarrow 1$
  $\mathbf{C}_R[t_R][j_{t_R}] \leftarrow c$
  return $m$ to $\mathcal{A}$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Update}}(role)$:
  $(\mathsf{msk}_{t_{role}+1}, \mathsf{K}_{t_{role}+1}, \mathsf{st}_{role}) \leftarrow$
        $\mathsf{Update}(\mathsf{msk}_{t_{role}}, \mathsf{st}_{role})$
  $t_{role} \leftarrow t_{role} + 1$
  $\mathsf{st}^{begin}_{role,t_{role}} \leftarrow \mathsf{st}_{role}$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Reveal}}(t, role)$:
  if $t > t_{role}$ then
    return $\lightning$
  return $(\mathsf{st}^{begin}_{role,t}, \mathsf{K}_t)$

If $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Corrupt}}(role)$:
  return $(\mathsf{st}^{begin}_{role,t_{role}}, \mathsf{msk}_{t_{role}})$
```

Fig. 5: Security experiment for *error predictability* (s-kERR-PRE) with respect to error predictor Pred of a multi-key channel Ch. An adversary $\mathcal{A}$ has only access to an oracle $[\mathcal{O}_{\mathsf{X}}]_c$ if the condition c is satisfied.

be taken into account and it is less clear how to define an independent receiver's internal state.

We translate the notion of error predictability to the multi-key setting, parameterized as s-kERR-PRE with forward security and phase-key insulation, and in a single- and multi-key variant. This enables us to show the following composition result: for any advanced security property $s \in \{\varepsilon, \mathsf{fs}, \mathsf{ki}, \mathsf{fski}\}$ and key setting $k \in \{\mathsf{sk}, \mathsf{mk}\}$, if a multi-key channel provides the according notion of ciphertext integrity (s-INT-kCTXT), chosen-plaintext confidentiality (s-IND-kCPA), and error predictability (s-kERR-PRE), then it also provides chosen-ciphertext confidentiality (s-IND-kCCA).

We formalize the parameterized, multi-key version of error predictability, s-kERR-PRE, in Definition 5 below through the experiment $\mathsf{Expt}^{s\text{-}k\mathsf{ERR\text{-}PRE}}_{\mathsf{Ch},\mathcal{A}}$ in Figure 5. An adversary wins against this experiment if it can ever cause the Recv algorithm to output an error message that differs from the output of the predictor algorithm. Meanwhile, when forward security or phase-key insulation is demanded, the adversary is even allowed to corrupt the master secret key resp. reveal phase keys at will.

Definition 5 (Error predictability of multi-key channels (s-kERR-PRE)). *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Update})$ *be a multi-key channel with error space* $\mathcal{E}$*, advanced security aspects* $s \in \{\varepsilon, \mathsf{fs}, \mathsf{ki}, \mathsf{fski}\}$ *and key setting* $k \in \{\mathsf{sk}, \mathsf{mk}\}$*. We say that* Ch *provides* error predictability *(s-kERR-PRE) with respect to an efficient probabilistic algorithm* $\mathsf{Pred}\colon \{0,1\}^{**} \times \{0,1\}^{**} \times \{0,1\}^{*} \xrightarrow{\$} \mathcal{E}$*, called the* error predictor*, if, for every PPT adversary* $\mathcal{A}$ *playing in the experiment* s-kERR-PRE *defined in Figure 5 against channel* Ch*, the following advantage function is negligible:*

$$\mathsf{Adv}^{s\text{-}k\mathsf{ERR\text{-}PRE}}_{\mathsf{Ch},\mathsf{Pred},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Expt}^{s\text{-}k\mathsf{ERR\text{-}PRE}}_{\mathsf{Ch},\mathsf{Pred},\mathcal{A}}(1^\lambda) = 1\right].$$

We are now ready to state our generic composition theorem for the setting of multi-key channels. The proof of the multi-key composition theorem follows along the lines of the classical result [10] adapted to the stateful setting and making use of the error predictor in the simulation of multiple errors in the receiving oracle as in [17,29]. Due to space limitations, we provide the proof in the full version of this work.

Theorem 1 (s-INT-kCTXT $\wedge$ s-IND-kCPA $\wedge$ s-kERR-PRE $\implies$ s-IND-kCCA). *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Update})$ *be a correct multi-key channel with error space* $\mathcal{E}$*. If* Ch *provides indistinguishability under chosen-plaintext attacks, integrity of ciphertexts, and error predictability (wrt. some predictor* Pred*) with advanced security aspects* $s \in \{\varepsilon, \mathsf{fs}, \mathsf{ki}, \mathsf{fski}\}$ *for a key setting* $k \in \{\mathsf{sk}, \mathsf{mk}\}$*, then it also provides indistinguishability under chosen-ciphertext attacks for s and k. Formally, for every efficient* s-IND-kCCA *adversary* $\mathcal{A}$ *there exist an efficient* s-INT-kCTXT *adversary* $\mathcal{B}_1$*,* s-kERR-PRE *adversary* $\mathcal{B}_2$*, and* s-IND-kCPA *adversary* $\mathcal{B}_3$ *such that*

$$\mathsf{Adv}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{CCA}}_{\mathsf{Ch},\mathcal{A}} \leq \mathsf{Adv}^{s\text{-}\mathsf{INT}\text{-}k\mathsf{CTXT}}_{\mathsf{Ch},\mathcal{B}_1} + \mathsf{Adv}^{s\text{-}k\mathsf{ERR\text{-}PRE}}_{\mathsf{Ch},\mathsf{Pred},\mathcal{B}_2} + \mathsf{Adv}^{s\text{-}\mathsf{IND}\text{-}k\mathsf{CPA}}_{\mathsf{Ch},\mathcal{B}_3}.$$

4 AEAD-based Construction of a Multi-key Channel

In this section we generically construct a (deterministic) multi-key channel $\mathsf{Ch}_{\mathsf{AEAD}}$ from on a nonce-based AEAD scheme AEAD and a pseudorandom function f. We then prove that our construction provides the strongest security notions for both confidentiality and integrity in our model, namely indistinguishability under multi-key chosen-ciphertext attacks and multi-key integrity of ciphertexts, both with forward security and phase-key insulation (fski-IND-mkCCA and fski-INT-mkCTXT).

Our generic construction $\mathsf{Ch}_{\mathsf{AEAD}} = (\mathsf{Init}, \mathsf{Send}, \mathsf{Recv}, \mathsf{Update})$ is defined via the algorithms given in Figure 6. It uses a nonce-based AEAD scheme $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$ with key space $\mathcal{K} = \{0,1\}^\lambda$, message and ciphertext space $\{0,1\}^*$, nonce space $\{0,1\}^n$, associated data space $\{0,1\}^*$, and an error symbol $\perp$. Furthermore, it employs a pseudorandom function $f\colon \{0,1\}^\lambda \times \{0,1\} \to \{0,1\}^\lambda$. The deterministic AEAD encryption algorithm maps a key $K \in \{0,1\}^\lambda$ (which we write in subscript), a nonce $N \in \{0,1\}^n$, an associated data value $ad \in \{0,1\}^*$, and a message $m \in \{0,1\}^*$ to a ciphertext $c \in \{0,1\}^*$. The deterministic decryption algorithm conversely maps a key, nonce, associated data value, and ciphertext to either a message or the error symbol $\perp$.

Our construction supports a maximum number of $\mathsf{maxmsg} = 2^n$ messages per phase, where n is the AEAD nonce length. The master-secret-key and phase-key space in our construction are equal to the AEAD and PRF key space, $\mathcal{MSK} = \mathcal{K} = \{0,1\}^\lambda$. The error space $\{\perp, \perp'\}$ consists of the error symbol $\perp$ of the AEAD scheme and a second symbol $\perp'$ indicating exceedance of maxmsg. The sending and receiving state space is $\mathcal{S}_S = \mathcal{S}_R = \mathbb{N} \times \mathbb{N}^* \times \{0,1\}$, encoding a message sequence number, a list of the message counts in all previous phases, and a failure flag indicating a previously occurred error.

On a high level, $\mathsf{Ch}_{\mathsf{AEAD}}$ derives master secret and phase keys via the (domain-separated) PRF f, an established technique ensuring forward security and separation of the keys derived; see, e.g., [13]. For encryption, it ensures reorder protection via a sequence number used as nonce. It further authenticates the number of messages seen in previous phases via the associated data field, borrowing established concepts from distributed computing to ensure causality.[8] In detail, our construction operates as follows.

- The Init algorithm uses $\mathsf{StateGen}$ to initialize the sending and receiving states as tuples containing a message sequence number $\mathsf{seqno} = 0$, a list of the number of messages sent in all previous phases $\mathsf{prevnos} = ()$, and a failure flag $\mathsf{fail} = 0$. Via $\mathsf{MasterKeyGen}$, the Init algorithm then samples an initial master secret key $\mathsf{msk}_0 \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random. Finally it derives the initial phase key $\mathsf{K}_0 \leftarrow f(\mathsf{msk}_0, 1)$ via $\mathsf{KeyDerive}$ as the output of the PRF f keyed with the initial master secret key and on input 1.
- The Send algorithm immediately outputs an error $\perp'$ in case the maximum number $\mathsf{maxmsg} = 2^n$ of messages has been reached in this or a prior call (indicated by $\mathsf{fail} = 1$). Otherwise, it increases the message sequence number in its state by one. It then invokes the deterministic AEAD encryption algorithm on the message m to obtain the ciphertext c. Here, the sequence number is used as the nonce $N = \mathsf{seqno}$ and the previous phases' message count as the associated data $ad = \mathsf{prevnos}$. The output of Send is the new state and the ciphertext c.
- The Recv algorithm immediately outputs an error $\perp$ in case the failure flag has been set ($\mathsf{fail} = 1$) in an earlier invocation, indicating that a previous AEAD decryption algorithm has failed. Otherwise it increases the message sequence number contained in the receiving state by one. It then uses the nonce $N = \mathsf{seqno}$ and associated data $\mathsf{prevnos}$ in the AEAD decryption algorithm on the ciphertext c to obtain m. In case the decryption fails and

[8] Note that, for a more efficient construction, one can get similar authenticity guarantees by storing a chained hash value of the number of messages received in previous phases using a collision-resistant hash function. For the sake of simplicity we omit this hash-chain optimization here and focus on demonstrating the feasibility of our security notions.

```
$\mathsf{Init}(1^\lambda)$:
  $(\mathsf{st}_{S,0}, \mathsf{st}_{R,0}) \leftarrow \mathsf{StateGen}(1^\lambda)$
  $\mathsf{msk}_0 \xleftarrow{\$} \mathsf{MasterKeyGen}(1^\lambda)$
  $\mathsf{K}_0 \leftarrow \mathsf{KeyDerive}(\mathsf{msk}_0)$
  return $(\mathsf{msk}_0, \mathsf{K}_0, \mathsf{st}_{S,0}, \mathsf{st}_{R,0})$

$\mathsf{Send}(\mathsf{st}_S, \mathsf{K}, m)$:
  parse $\mathsf{st}_S$ as $(\mathsf{seqno}, \mathsf{prevnos}, \mathsf{fail})$
  if $\mathsf{seqno} = \mathsf{maxmsg}$ or $\mathsf{fail} = 1$ then
    $\mathsf{fail} \leftarrow 1$
    $\mathsf{st}_S \leftarrow (\mathsf{seqno}, \mathsf{prevnos}, \mathsf{fail})$
    return $(\mathsf{st}_S, \perp')$
  $\mathsf{seqno} \leftarrow \mathsf{seqno} + 1$
  $c \leftarrow \mathsf{Enc}_\mathsf{K}(\mathsf{seqno}, \mathsf{prevnos}, m)$
  $\mathsf{st}_S \leftarrow (\mathsf{seqno}, \mathsf{prevnos})$
  return $(\mathsf{st}_S, c)$

$\mathsf{Recv}(\mathsf{st}_R, \mathsf{K}, c)$:
  parse $\mathsf{st}_R$ as $(\mathsf{seqno}, \mathsf{prevnos}, \mathsf{fail})$
  if $\mathsf{fail} = 1$ then
    return $(\mathsf{st}_R, \perp)$
  $\mathsf{seqno} \leftarrow \mathsf{seqno} + 1$
  $m \leftarrow \mathsf{Dec}_\mathsf{K}(\mathsf{seqno}, \mathsf{prevnos}, c)$
  if $m = \perp$ then
    $\mathsf{fail} \leftarrow 1$
    $\mathsf{st}_R \leftarrow (\mathsf{seqno}, \mathsf{prevnos}, \mathsf{fail})$
  return $(\mathsf{st}_R, m)$

$\mathsf{StateGen}(1^\lambda)$:
  $\mathsf{st}_{S,0} = (0, (), 0)$
  $\mathsf{st}_{R,0} = (0, (), 0)$
  return $(\mathsf{st}_{S,0}, \mathsf{st}_{R,0})$

$\mathsf{MasterKeyGen}(1^\lambda)$:
  $\mathsf{msk}_0 \xleftarrow{\$} \{0,1\}^\lambda$
  return $\mathsf{msk}_0$

$\mathsf{KeyDerive}(\mathsf{msk})$:
  return $f(\mathsf{msk}, 1)$

$\mathsf{Update}(\mathsf{msk}, \mathsf{st})$:
  $\mathsf{msk} \leftarrow \mathsf{MasterKeyUp}(\mathsf{msk})$
  $\mathsf{K} \leftarrow \mathsf{KeyDerive}(\mathsf{msk})$
  $\mathsf{st} \leftarrow \mathsf{StateUp}(\mathsf{st})$
  return $(\mathsf{msk}, \mathsf{K}, \mathsf{st})$

$\mathsf{StateUp}(\mathsf{st})$:
  parse $\mathsf{st}$ as $(\mathsf{seqno}, \mathsf{prevnos}, \mathsf{fail})$
  $\mathsf{st} \leftarrow (0, (\mathsf{prevnos}, \mathsf{seqno}), \mathsf{fail})$
  return $\mathsf{st}$

$\mathsf{MasterKeyUp}(\mathsf{msk})$:
  return $f(\mathsf{msk}, 0)$
```

Fig. 6: Our generic construction of a deterministic multi-key channel $\mathsf{Ch}_\mathsf{AEAD}$.

$m = \perp$, the failure flag is set to 1. The output of Recv is the new state and the message (or error) m.

- The Update algorithm uses StateUp to reset the new message sequence number to 0, and appends the previous message sequence number to the list of previous phases' message counts, i.e., $\mathsf{prevnos} \leftarrow (\mathsf{prevnos}, \mathsf{seqno})$. Then it invokes MasterKeyUp to derive a new master secret key as the output of f keyed with the previous master secret key and on input 0. Finally, it uses KeyDerive to compute a new phase key from the new master secret key.

Correctness. Correctness of our $\mathsf{Ch}_\mathsf{AEAD}$ construction follows immediately from correctness of the underlying AEAD scheme. In particular, observe that both receiver and sender compute their master secret and phase keys via the same, deterministic key schedule. Moreover, whenever both sides process the same number—not exceeding maxmsg—of messages per phase (as is a precondition in the correctness definition), they will also use the same associated data values for encryption and decryption, thus rendering the receiver to derive the correct messages as required.

Remark 2. At first glance, it might seem counter-intuitive that the sequence number in our $\mathsf{Ch}_{\mathsf{AEAD}}$ construction is reset to 0 at the start of a new phase. Would it not be more natural to have the sequence number running over all phases in order to ensure at the start of a phase that all messages of the previous phase were received, and to prevent reordering of messages across phases?

As surfaced by Fournet and the miTLS [40] team in the discussion around TLS 1.3 [30], this approach would however enable truncation attacks if the leakage of phase keys is considered in the security definition, as we do for phase-key insulation.[9] If sequence numbers are continued, an adversary holding the key of some phase t can truncate a prefix of the messages (with sequence numbers $i, \ldots, i+j$) in phase $t+1$ by providing the receiver with $j+1$ self-generated messages at the end of t. Dropping the first $j+1$ messages in phase $t+1$, the receiver's sequence number matches again the one of the sender (for message $i+j+1$), so the truncation would go unnoticed. Resetting the sequence numbers to 0 when switching phases prevents this attack, though additional care needs to be taken to prevent suffix truncation at the end of a phase. In our construction, we ensure the latter through authenticating the number of messages sent in all previous phases. We note that this mechanism would even allow to not reset the sequence number, but we decided to keep the reset in order to stay closer to the channel design of TLS 1.3 (cf. the discussion in Section 4.2).

4.1 Security Analysis

We now show that our generic $\mathsf{Ch}_{\mathsf{AEAD}}$ construction achieves the strongest multi-key security notions for confidentiality and integrity, namely forward-secure and phase-key–insulated indistinguishability under multi-key chosen-ciphertext attacks ($\mathsf{fski\text{-}IND\text{-}mkCCA}$) and integrity of ciphertexts ($\mathsf{fski\text{-}INT\text{-}mkCTXT}$). For proving the former notion we proceed via first showing the corresponding CPA confidentiality variant as well as that our construction provides error predictability (for multiple keys and with forward security and phase-key insulation), and then leverage our generic composition theorem (Theorem 1). Our results hold under the assumption that the underlying nonce-based AEAD scheme AEAD provides confidentiality in the sense of $\mathsf{IND\text{-}CPA}$ security and integrity in terms of AUTH security as defined by Rogaway [44][10], as well as that the employed pseudorandom function f meets the standard notion of PRF security.

We begin with stating the multi-key chosen-plaintext confidentiality with forward security and phase-key insulation. The according proof, provided in the full version, proceeds in three steps. First, we guess the challenge phase t the adversary will select, introducing a loss of n_t. Second, we gradually replace all

[9] In our framework, the weakest integrity property broken through this attack is phase-key–insulated integrity of plaintexts ($\mathsf{ki\text{-}INT\text{-}mkPTXT}$).

[10] While Rogaway defines confidentiality via the stronger $\mathsf{IND\$\text{-}CPA}$ notion, it suffices for our result that AEAD provides regular indistinguishability of encryptions.

master secret keys up to msk_{t+1} and phase keys up to K_t with independent random values, bounding the advantage difference by n_t times the PRF security of f via a hybrid argument. In the last step, we show that the remaining advantage can be bounded by reducing the challenge phase's operations to the $\mathsf{IND\text{-}CPA}$ security of the employed AEAD scheme.

Theorem 2 **($\mathsf{Ch}_{\mathsf{AEAD}}$ is $\mathsf{fski\text{-}IND\text{-}mkCPA}$-secure).** *The* $\mathsf{Ch}_{\mathsf{AEAD}}$ *construction from Figure 6 provides forward-secure and phase-key–insulated indistinguishability under multi-key chosen-plaintext attacks (*$\mathsf{fski\text{-}IND\text{-}mkCPA}$*) if the employed authenticated encryption with associated data scheme* AEAD *provides indistinguishability under chosen-plaintext attacks (*$\mathsf{IND\text{-}CPA}$*) and the employed pseudorandom function f is* PRF*-secure.*

Formally, for every efficient $\mathsf{fski\text{-}IND\text{-}mkCPA}$ *adversary $\mathcal{A}$ against* $\mathsf{Ch}_{\mathsf{AEAD}}$ *there exists efficient algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{fski\text{-}IND\text{-}mkCPA}}_{\mathsf{Ch}_{\mathsf{AEAD}},\mathcal{A}}(\lambda) \leq n_t \cdot \Big(n_t \cdot \mathsf{Adv}^{\mathsf{PRF}}_{f,\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AEAD},\mathcal{B}_2}(\lambda)\Big),$$

where $n_t = \max(t_S, t_R) + 1$ is the maximum number of phases active in the $\mathsf{fski\text{-}IND\text{-}mkCPA}$ *experiment.*

We now turn to the multi-key integrity of ciphertexts with forward security and phase-key insulation of $\mathsf{Ch}_{\mathsf{AEAD}}$. In the proof, the first two steps follow closely the proof of $\mathsf{fski\text{-}IND\text{-}mkCPA}$ security. For the last step, a careful case analysis of the situations where synchronization is lost in the integrity experiment and how this is reflected in the $\mathsf{Ch}_{\mathsf{AEAD}}$ construction establishes the reduction to the underlying AEAD scheme's authenticity. We provide the proof of integrity for our $\mathsf{Ch}_{\mathsf{AEAD}}$ construction in the full version.

Theorem 3 **($\mathsf{Ch}_{\mathsf{AEAD}}$ is $\mathsf{fski\text{-}INT\text{-}mkCTXT}$-secure).** *The* $\mathsf{Ch}_{\mathsf{AEAD}}$ *construction from Figure 6 provides forward-secure and phase-key–insulated multi-key integrity of ciphertexts (*$\mathsf{fski\text{-}INT\text{-}mkCTXT}$*) if the employed authenticated encryption with associated data scheme* AEAD *provides authenticity (*AUTH*) and the employed pseudorandom function f is* PRF*-secure.*

Formally, for every efficient $\mathsf{fski\text{-}INT\text{-}mkCTXT}$ *adversary $\mathcal{A}$ against* $\mathsf{Ch}_{\mathsf{AEAD}}$ *there exists efficient algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathsf{Adv}^{\mathsf{fski\text{-}INT\text{-}mkCTXT}}_{\mathsf{Ch}_{\mathsf{AEAD}},\mathcal{A}}(\lambda) \leq n_t \cdot \Big(n_t \cdot \mathsf{Adv}^{\mathsf{PRF}}_{f,\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{AUTH}}_{\mathsf{AEAD},\mathcal{B}_2}(\lambda)\Big),$$

where $n_t = \max(t_S, t_R) + 1$ is the maximum number of phases active in the $\mathsf{fski\text{-}INT\text{-}mkCTXT}$ *experiment.*

Finally, in the full version, we show that our $\mathsf{Ch}_{\mathsf{AEAD}}$ provides multi-key error predictability with forward security and phase-key insulation ($\mathsf{fski\text{-}mkERR\text{-}PRE}$). We can then conclude from Theorem 1 that it also achieves strong $\mathsf{fski\text{-}IND\text{-}mkCCA}$ confidentiality.

4.2 Comparison to the TLS 1.3 Record Protocol

Our notion of multi-key channels is particularly inspired by the ongoing developments of the upcoming Transport Layer Security (TLS) protocol version 1.3 [43]. It is hence insightful to compare our generic construction with the design of the TLS 1.3 record protocol (cf. [43, Section 5]).

First of all note that, in contrast to previous TLS versions, TLS 1.3 mandates the use of AEAD schemes as encryption and authentication mechanisms for the record protocol. It follows the basic secure-channel design principle to include a sequence number for protecting against reordering attacks; as in our construction. Both in TLS 1.3 and our construction, the sequence number enters the AEAD's nonce field and is reset to 0 at the start of each new phase. Also identically to our construction, the TLS 1.3 record protocol keys are derived via a deterministic key schedule in which, starting from an initial master secret key (denoted `client/server_traffic_secret_0` in TLS 1.3) the current phase's key as well as the next phase's master secret key are derived via independent applications of a pseudorandom function (TLS 1.3 uses HMAC [7,36] for this purpose). Beyond enabling key switches to allow secure encryption of large amounts of data, the TLS 1.3 design in particular names forward security (combined with insulation of phase keys) as a security goal [43, Appendix E.2]. In this sense, our generic $\mathsf{Ch}_{\mathsf{AEAD}}$ construction is comparatively close to the internal channel design of the TLS 1.3 record protocol in both techniques and security goals.

Still, there are some notable differences between the two designs, both in technical details as well as in the practically achieved security and its underlying assumptions. On the technical side, the TLS 1.3 record protocol additionally includes a content-type field in ciphertexts to enable multiplexing of messages from multiple sources. Furthermore, TLS 1.3 does not explicitly authenticate the numbers of seen ciphertexts in previous phases (as our construction does via the $\mathsf{prevnos}$ field), but instead relies on the authenticated transmission of key update messages. To be precise, key update messages are encoded as a specific control ("post-handshake") message and sent within the data channel. Thereby associated with a sequence number, they serve as an authenticated "end-of-phase indicator" that allows the record protocol to infer in unrevealed phases that all messages in a phase have been correctly received when the key update message arrives.

In contrast, our model does not rely on the authenticity of key updates but captures more generic settings where key update notifications may be send out-of-band and without being authenticated. Our construction hence cannot rely on key updates as indicators that a phase was gracefully completed, but instead needs to leverage the next uncompromised phase to detect truncations in an earlier phase. Nevertheless, our generic $\mathsf{Ch}_{\mathsf{AEAD}}$ scheme serves as proof-of-concept construction that strong confidentiality and integrity can be achieved in the multi-key setting with forward security and phase-key insulation even with unauthenticated, out-of-band key updates.

5 Conclusions and Future Work

In this work we initiate the study of multi-key channels, providing a game-based formalization, a framework of security notions and their relations, as well as a provably secure construction based on authenticated encryption with associated data and a pseudorandom function. Motivated by the channel design of the upcoming version 1.3 of the Transport Layer Security (TLS) protocol involving key updates and thus multiple keys, our work casts a formal light on the design criteria for multi-key channels and their achievable security guarantees.

Being a first step towards the understanding of, in particular, real-world designs of multi-key channels, our work also gives rise to further research questions. A natural next step is to analyze the exact security guarantees achieved by the multi-key TLS 1.3 record protocol. In this context, a question of independent interest lies in analyzing the trade-offs between relying on authenticated key updates versus not authenticating them, both with respect to the security properties achievable as well as potential functional and efficiency impacts. In a different direction, Fischlin et al. [29] observed that TLS and other channels deviate on the API level from the classical cryptographic abstraction of channels by providing a streaming interface rather than an atomic-message interface. Hence, their notion of stream-based channels is a natural candidate to blend with our multi-key notions in order to investigate the interplay of discrete key updates with a non-discrete stream of message data. Finally, it would be interesting to extend the notion of multi-key channels to capture more complex, non-deterministic key schedules, e.g., those employed in secure messaging protocols like Signal [47] aiming at extended security properties [21,20,6].

Acknowledgments

We thank Giorgia Azzurra Marson for helpful discussions in the early phase of this work. We furthermore thank the anonymous reviewers of EUROCRYPT 2017 and CRYPTO 2017 for their valuable comments. This work has been funded by the DFG as part of projects P2, S4 within the CRC 1119 CROSSING.

References

1. Abdalla, M., Bellare, M.: Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 546–559. Springer, Heidelberg (Dec 2000)
2. Albrecht, M.R., Degabriele, J.P., Hansen, T.B., Paterson, K.G.: A surfeit of SSH cipher suites. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16. pp. 1480–1491. ACM Press (Oct 2016)
3. Albrecht, M.R., Paterson, K.G., Watson, G.J.: Plaintext recovery attacks against SSH. In: 2009 IEEE Symposium on Security and Privacy. pp. 16–26. IEEE Computer Society Press (May 2009)

4. Badertscher, C., Matt, C., Maurer, U., Rogaway, P., Tackmann, B.: Augmented secure channels and the goal of the TLS 1.3 record layer. In: Au, M.H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 85–104. Springer, Heidelberg (Nov 2015)
5. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: Reconciling AE robustness notions. In: Groth, J. (ed.) 15th IMA International Conference on Cryptography and Coding. LNCS, vol. 9496, pp. 94–111. Springer, Heidelberg (Dec 2015)
6. Bellare, M., Camper Singh, A., Jaeger, J., Nyayapati, M., Stepanovs, I.: Ratcheted encryption and key exchange: The security of messaging. In: CRYPTO 2017. LNCS, Springer, Heidelberg (Aug 2017)
7. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (Aug 1996)
8. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press (Oct 1997)
9. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the encode-then-encrypt-and-MAC paradigm. ACM Trans. Inf. Syst. Secur. 7(2), 206–241 (2004)
10. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (Dec 2000)
11. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer, Heidelberg (Dec 2000)
12. Bellare, M., Tackmann, B.: The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 247–276. Springer, Heidelberg (Aug 2016)
13. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (Apr 2003)
14. Bhargavan, K., Blanchet, B., Kobeissi, N.: Verified models and reference implementations for the TLS 1.3 standard candidate. In: 2017 IEEE Symposium on Security and Privacy (S&P 2017). pp. 483–503. IEEE (May 2017)
15. Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Pan, J., Protzenko, J., Rastogi, A., Swamy, N., Zanella-Béguelin, S., Zinzindohoué, J.K.: Implementing and proving the TLS 1.3 record layer. In: 2017 IEEE Symposium on Security and Privacy (S&P 2017) (2017)
16. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: Security of symmetric encryption in the presence of ciphertext fragmentation. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 682–699. Springer, Heidelberg (Apr 2012)
17. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 367–390. Springer, Heidelberg (Mar 2014)
18. Brzuska, C., Smart, N.P., Warinschi, B., Watson, G.J.: An analysis of the EMV channel establishment protocol. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13. pp. 373–386. ACM Press (Nov 2013)
19. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (May 2001)

20. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the Signal messaging protocol. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P 2017). IEEE (Apr 2017)
21. Cohn-Gordon, K., Cremers, C.J.F., Garratt, L.: On Post-compromise Security. In: IEEE 29th Computer Security Foundations Symposium (CSF 2016). pp. 164–178 (2016)
22. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (Aug 2008), `http://www.ietf.org/rfc/rfc5246.txt`, updated by RFCs 5746, 5878, 6176
23. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes and Cryptography 2(2), 107–125 (1992)
24. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (Apr / May 2002)
25. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong key-insulated signature schemes. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (Jan 2003)
26. Dodis, Y., Luo, W., Xu, S., Yung, M.: Key-insulated symmetric key cryptography and mitigating attacks against cryptographic cloud software. In: Youm, H.Y., Won, Y. (eds.) ASIACCS 12. pp. 57–58. ACM Press (May 2012)
27. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In: Ray, I., Li, N., Kruegel:, C. (eds.) ACM CCS 15. pp. 1197–1210. ACM Press (Oct 2015)
28. Fischlin, M., Günther, F.: Replay attacks on zero round-trip time: The case of the TLS 1.3 handshake candidates. In: 2017 IEEE European Symposium on Security and Privacy. IEEE (Apr 2017)
29. Fischlin, M., Günther, F., Marson, G.A., Paterson, K.G.: Data is a stream: Security of stream-based channels. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 545–564. Springer, Heidelberg (Aug 2015)
30. Fournet, C.: Re: [TLS] [tls13-spec] resetting the sequence number to zero for each record key. (#379). `https://mailarchive.ietf.org/arch/msg/tls/exto09ETJLnEm3MRDT023x70DFM` (Dec 2015)
31. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
32. Günther, C.G.: An identity-based key-exchange protocol. In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT'89. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (Apr 1990)
33. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (Apr 2001)
34. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard) (Dec 2005), `http://www.ietf.org/rfc/rfc4301.txt`, updated by RFC 6040
35. Kohno, T., Palacio, A., Black, J.: Building secure cryptographic transforms, or how to encrypt and MAC. Cryptology ePrint Archive, Report 2003/177 (2003), `http://eprint.iacr.org/2003/177`
36. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational) (Feb 1997), `http://www.ietf.org/rfc/rfc2104.txt`, updated by RFC 6151
37. Krawczyk, H., Wee, H.: The OPTLS protocol and TLS 1.3. In: 2016 IEEE European Symposium on Security and Privacy. pp. 81–96. IEEE (Mar 2016)

38. Luykx, A., Paterson, K.: Limits on authenticated encryption use in TLS. `http://www.isg.rhul.ac.uk/~kp/TLS-AEbounds.pdf` (2016)
39. Marson, G.A., Poettering, B.: Security notions for bidirectional channels. IACR Trans. Symm. Cryptol. 2017(1), 405–426 (2017)
40. miTLS: A Verified Reference Implementation of TLS, `http://mitls.org/`
41. Paterson, K.G., van der Merwe, T.: Reactive and proactive standardisation of TLS. In: Chen, L., McGrew, D., Mitchell, C. (eds.) SSR 2016. Lecture Notes in Computer Science, vol. 10074, pp. 160–186. Springer (Dec 2016)
42. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag size does matter: Attacks and proofs for the TLS record protocol. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 372–389. Springer, Heidelberg (Dec 2011)
43. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 – draft-ietf-tls-tls13-20. `https://tools.ietf.org/html/draft-ietf-tls-tls13-20` (Apr 2017)
44. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 02. pp. 98–107. ACM Press (Nov 2002)
45. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS 01. pp. 196–205. ACM Press (Nov 2001)
46. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (May / Jun 2006)
47. Signal protocol: Advanced cryptographic ratcheting. `https://whispersystems.org/blog/advanced-ratcheting/`
48. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard) (Jan 2006), `http://www.ietf.org/rfc/rfc4251.txt`