IoT-friendly AKE: Forward Secrecy and Session Resumption Meet Symmetric-key Cryptography*

Gildas Avoine[1,2], Sébastien Canard[3], and Loïc Ferreira[3,1]

[1] Univ Rennes, INSA Rennes, CNRS, IRISA, France
[2] Institut Universitaire de France
gildas.avoine@irisa.fr
[3] Orange Labs, Applied Crypto Group, Caen, France
{sebastien.canard,loic.ferreira}@orange.com

Abstract. With the rise of the Internet of Things and the growing popularity of constrained end-devices, several security protocols are widely deployed or strongly promoted (e.g., Sigfox, LoRaWAN, NB-IoT). Based on symmetric-key functions, these protocols lack in providing security properties usually ensured by asymmetric schemes, in particular forward secrecy. We describe a 3-party authenticated key exchange protocol solely based on symmetric-key functions (regarding the computations done between the end-device and the back-end network) which guarantees forward secrecy. Our protocol enables session resumption (without impairing security). This allows saving communication and computation cost, and is particularly advantageous for low-resource end-devices. Our 3-party protocol can be applied in a real-case IoT deployment (i.e., involving numerous end-devices and servers) such that the latter inherits from the security properties of the protocol. We give a concrete instantiation of our key exchange protocol, and formally prove its security.

Keywords: Security protocols · Authenticated key exchange · Symmetric-key cryptography · Session resumption · Forward secrecy · Security model · Internet of Things.

1 Introduction

1.1 Context

The arising of the Internet of Things (IoT) gives birth to different types of use cases and environments (smart home, smart cities, eHealth, Industrial IoT, etc.). According to several reports, "*the Industrial Internet of Things is the biggest and most important part of the Internet of Things*" [17] and "*the biggest driver of productivity and growth in the next decade*" [1]. The Industrial IoT (IIoT) covers sensitive applications since it aims at managing networks that provide valuable resources (e.g., energy, water, etc.). Contrary to the smart home case, where

* Extended version of the paper accepted at ESORICS 2019.

a network is localised to the house perimeter and implies merely a domestic management of the network, the IIoT context may require a large coverage zone where connected objects (e.g., sensors, actuators, etc.) are widespread all over an urban area. This implies the involvement of, at least, two players: the application provider (which exploits the connected objects to get some valuable data and provide some service), and the communication provider whose network is used by the application provider to communicate with its connected objects (see Figure 1).

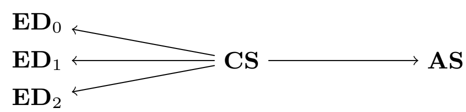


Fig. 1: Connection between end-devices (ED) and an application server (AS) through a communication server (CS)

Cryptographic Separation of the Layers. The (Industrial) IoT involves low-resources end-devices which are not able to apply heavy computations implied by asymmetric schemes. Consequently, security protocols used on currently deployed IoT networks usually implement symmetric-key functions only, and are based on a unique (per end-device) symmetric root key. Using the same root key implies that the communication layer and the application layer are entangled. The communication provider must guarantee that only legitimate parties can send data through its network, but does not need to get the application data. The application provider must keep full control over its connected objects, but must not be able to interfere with the management of the communication network. Therefore, the communication and the application layers must be cryptographically distinct.

Forward Secrecy. The (Industrial) IoT protocols based on symmetric-key functions do not provide strong security properties usually ensured by asymmetric schemes, in particular *forward secrecy.* The disclosure of the root key compromises all the past sessions established with that key, not to mention the consequences of an intrusion into the back-end server that centralises all root keys. The current symmetric-key based IoT protocols lack in providing this fundamental security property.

Session Resumption. A session resumption scheme allows establishing a new session at a reduced cost: once two parties have performed a first key exchange, they can use some shared key material to execute subsequent runs faster. This means less data exchanged during the key agreement, and reduced time and energy, which is particularly convenient and advantageous for low-resource end-devices. Yet, the symmetric-key based IoT protocols always execute the same full key exchange.

1.2 Related Work

Several protocols for the (Industrial) IoT have been proposed. Among these, the following are widely deployed or strongly promoted. They all build their security on symmetric-key functions, and make use of a static and unique (per end-device) root key shared between the end-device and the back-end network.

Sigfox [30,31] corresponds to a centric model: one entity (the Sigfox company or one of its partners) manages a proprietary network. The application data (sent by the end-devices) is managed by the central entity (Sigfox) and then delivered to the different Sigfox's customers. Hence, the latter are compelled to have confidence in Sigfox, and, in a way, to let the company disintermediate them. Sigfox *owns* all the application end-devices in the sense that the (fixed) root key used to protect the data is known to Sigfox.

LoRaWAN 1.0 [33] provides more flexibility: any company can deploy a LoRa network. Two session keys are derived from the end-device's root key to protect the communication and the application layers. Hence, this root key gives access to both (cryptographic) layers, and knowledge of this key gives virtually ownership of the end-devices. Moreover, several weaknesses in LoRaWAN 1.0 have been identified which lead to likely practical attacks [4].

In LoRaWAN 1.1 [32], a "Join Server" is added to the architecture (compared to version 1.0). It is in charge of doing the key exchange with the end-device [34]. Two distinct static symmetric root keys are used. Each yields a session key. This allows to cryptographically separate the communication and the application layers. The specification [32] does not make clear if the application and the communication providers can own their respective root key.[4] Yet, a companion document [34] states that these keys must be stored at the Join Server which is in charge of doing the key exchange with the end-device. When the Join Server computes the session keys, each one is respectively sent to the communication server, and to the application server. In such a context, the Join Server is always solicited during the key exchange, including when the end-device makes a new run with the *same* server. Furthermore, only the end-device can initiate a key exchange (as in version 1.0) even though the Network Server can, in some specific cases, request the end-device to initiate a new key exchange.

Contrary to the previous technologies, Narrowband IoT (NB-IoT), enhanced Machine-Type Communication (eMTC), Extended Coverage GSM IoT (EC-GSM-IoT) are cellular technologies. eMTC provides enhancements to the Long Term Evolution (LTE/4G) technology for machine type communications. NB-IoT is also based on LTE, whereas EC-GSM-IoT is based on GSM/EDGE technologies and dedicated to low-cost end-devices. These technologies aims at decreasing the end-device complexity (hence its cost), power consumption, extending autonomy, and increasing coverage [16]. The security of all these systems relies on the underlying technology (GSM, EDGE, LTE), hence on a static symmetric root key known to a central authority, likely the telecom operator. They inherit the intrinsic security limitations of the symmetric-key schemes they are

[4] This would imply that the end-device be tied for its whole life to unique application and communication providers.

built on.

Furthermore, *none* of the aforementioned protocols and technologies provide forward secrecy.

To the best of our knowledge, no IoT protocol proposes a session resumption scheme. Such schemes exist in other contexts. In TLS 1.2 [13], the server can encrypt the “master secret” and store that “Session Ticket” [25] at the client. In TLS 1.3 [21], the server encrypts a “resumption master secret” (RMS) output by the previous key exchange, and stores it at the client. In IKEv2 [19], a similar approach is used [27].

From the *same* secret value, used as symmetric master key, successive runs can be executed with these (TLS, IKE) procedures. Hence disclosure of the reused secret may compromise several past sessions: this breaks forward secrecy. In TLS 1.3, a fresh secret can be added to the key derivation computation, but this implies applying the Diffie-Hellman scheme [14]. Moreover, in TLS, the same Session Ticket Encryption Key (STEK) is used by the server to encrypt several RMS values (corresponding to different clients). Hence a STEK may be persistent in the server’s memory and its disclosure compromises past sessions. Therefore these solutions are not satisfactory with respect to forward secrecy.

Aviram, Gellert, and Jager [2] propose a resumption scheme aiming at guaranteeing forward secrecy and non-replayability when 0-RTT is used in TLS 1.3. They describe two concrete instantiations. One is based on RSA [22], the other is a tree-based scheme. As all the aforementioned session resumption schemes, Aviram et al.’s proposal implies to store a ticket at the client. Therefore the number of tickets to store grows with the number of servers the client can resume a session with. Hence, low-resource end-devices with constrained memory cannot apply these schemes.

Reversing the roles taken by the client and the server (i.e., the client computes and the server stores the ticket) is not sufficient. First, the Aviram et al.’s RSA based scheme is excluded, despite its elegance, for low-cost IoT end-devices that can only implement symmetric-key functions. Moreover, their tree-based scheme implies that the decryption key grows (up to some point) each time a ticket is used, which is prohibitive for the end-device (client).[5] They also propose an alternative that trades decryption key size for ticket size. However sending (and retrieving) big tickets is an issue for a low-resource end-device. As noticed by Aviram et al., each transmitted bit costs energy, which limits the battery lifetime of self-powered end-devices.

The resumption scheme we describe reverses the roles of client (end-device) and server. At the same time it mitigates the issues related to memory space,

[5] The two schemes (RSA- and tree-based) described by Aviram et al. allow computing a fixed number of tickets (say n). One key (asymmetric or symmetric depending on the scheme) is used to yield the n tickets. In order to compute a new batch of n tickets, a new key must be generated and stored by the server. We observe that, if a new batch of n tickets is computed whereas it remains even one ticket not used yet from the previous batch, two keys (the current and the new one) must be stored concurrently in the server’s memory.

computation cost, and amount of transmitted data. Yet, solving this problem *without* an asymmetric scheme is not trivial.

1.3 Contribution

In this paper, we present a 3-party authenticated key exchange (3-AKE) protocol executed between an end-device, a server, and a trusted third party, which matches *at the same time* the following properties:

- The protocol is solely based on symmetric-key functions (regarding the computations done by the end-device).
- Application and communication security layers are separated.
- The protocol enables session resumption.
- The protocol provides forward secrecy.

In addition, we describe a security model in order to formally prove the security of our protocol, and give a concrete instantiation of the latter.

Finally, we describe how to use our 3-party key exchange protocol in a realistic IoT deployment that involves numerous end-devices and servers, and such that it inherits the security properties of the 3-AKE protocol (in particular forward secrecy).

1.4 Outline of the Paper

In Section 2, we describe our generic 3-party authenticated key exchange protocol, and, based on it, a more general construction for the IoT context. The session resumption procedure is explained in Section 3. In Section 4, we introduce the security model that we use to prove the security of our protocol. Section 5 presents a concrete instantiation of our protocol. Finally, we conclude in Section 6.

2 Description of the 3-party AKE Protocol

In this section, we describe our generic 3-party authenticated key exchange (3-AKE) protocol. The main purpose of our protocol is to output session keys. This subsequently enables to establish two distinct secure channels, with a communication server on the one hand, and an application server on the other hand. We do not detail these channels, and let it be defined depending on their specific context.

2.1 The Different Roles

The real-case IoT deployment we consider involves four *roles*: the trusted third party that we name *Authentication and Key Server* (KS), the *Application End-device* (ED), the *Communication Server* (CS), and the *Application Server* (AS). The purpose of AS is to provide some service (e.g., telemetry, asset tracking, equipment automation, etc.). The AS exploits ED (e.g., a sensor, an actuator, etc.) to ensure that service. In order to exchange data, ED and AS use a communication network. The entry point is CS, which grants ED access to that network. Typically CS is managed by a telecom operator.

One KS can manage several ED. An ED can be either static or mobile, hence may have to connect one or several CS. An AS can use several ED in order to provide its service.[6] The kind of ED we consider is a (low-resource) wireless end-device whereas we assume that KS, CS, and AS use high-speed (wired) connections with each other, and have heavier capabilities, in particular computational.

The data exchanged between ED and AS must be accessible to these two parties only. Moreover, CS needs also to privately communicate with ED, e.g., in order to regulate the radio interface. The KS is in charge of the overall security of the system: its main purpose is to authenticate ED, and to allow AS and CS to share distinct session keys with ED. These keys aim at establishing two separate secure channels.

As said, the architecture we consider involves four types of entities: KS, ED, CS, and AS. However, from a cryptographic perspective, CS and AS behave the same way with respect to KS and ED. The main goal to reach is to allow ED to share a session key with a server XS $\in \{\mathrm{CS}, \mathrm{AS}\}$ which ensures some functionality (communication or application in our case). This is achieved with our 3-AKE protocol: executed between KS, ED, and XS, the protocol outputs key material that allows ED and XS to establish a secure channel. In the remainder of the paper, we will mention for simplicity only the two *types* of CS and AS servers. Nonetheless, recall that they represent in fact the several servers which are actually involved in the IoT architecture we consider.

2.2 Key Computation and Distribution

Our 3-AKE protocol is based on a pre-shared symmetric key mk known only to two parties: ED, and KS which ED is affiliated to. Each ED owns a distinct master key mk. A 3-AKE run is split in two main phases. Each phase appeals to a 2-party authenticated key exchange (2-AKE) protocol, whose security properties will be made explicit in Section 2.3. During the first phase, ED and KS perform a 2-AKE run with the shared master key mk. During the second phase, ED and XS $\in \{\mathrm{CS}, \mathrm{AS}\}$ use the output of the first key exchange to perform an additional 2-AKE run. This yields a session key used to establish a secure channel between ED and XS. In practice, since our architecture involves two types of XS servers, a 3-AKE run is done first between KS, ED, and CS, and then between KS, ED, and AS. This yields two distinct session keys. With each session key, a secure channel can be established between ED and CS on the one hand, and ED and AS on the other hand.

[6] For the sake of genericness, it may also be technically (i.e., cryptographically) possible with our protocol that the same ED be used by several AS (each one providing a different service).

More precisely, the following steps are executed between KS, ED, CS, and AS (see Figures 2 and 3).

1. Based on the shared master key mk, KS and ED perform an AKE, relayed by CS (Figure 2a). This first AKE outputs a *communication intermediary key* ik_c.
2. The previous step (2-AKE) is repeated between KS and ED. It outputs an *application intermediary key* ik_a.
3. KS sends ik_c to CS, and ik_a to AS through two distinct pre-existing secure channels (Figure 2b). Then, upon reception of the keys by CS and AS, KS deletes its own copies in order to enhance the security of the subsequent phases of the protocol (we elaborate more on this in Section 2.4).
4. Using ik_c, ED and CS perform an AKE which outputs a *communication session key* sk_c (Figure 3a).
5. Using ik_a, ED and AS perform an AKE which outputs an *application session key* sk_a (Figure 3b).
6. Using the application session key sk_a, ED and AS can now establish an *application secure channel*. Likewise, with the communication session key sk_c, ED and CS can establish a distinct *communication secure channel* (Figure 3c).

We call P the protocol that involves ED, and is used to perform the 2-AKE runs between ED and KS (steps 1-2), ED and CS (step 4), and ED and AS (step 5). We call P' the 2-AKE protocol used on the back-end side between KS and AS (resp. CS). Let Enc be the function used to set up the secure channel between KS and AS (resp. CS) with the session key output by P' (step 3).

For the sake of clarity, we have depicted (Figure 2a) the case where the two intermediary keys ik_c and ik_a are successively computed. But the computation of either key can be completely *dissociated*.[7]

2.3 The Building Blocks P, P', and Enc

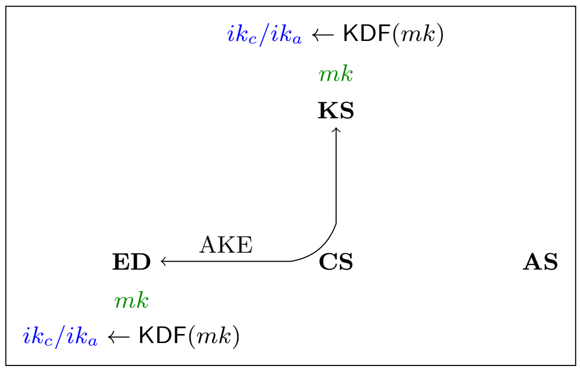
Our 3-AKE protocol depends crucially on the 2-party protocols P and P', and function Enc. Before making clear the properties of our 3-party protocol, we list below the main features we require these three building blocks to have.

Protocol P. We require protocol P to fulfill the following properties.

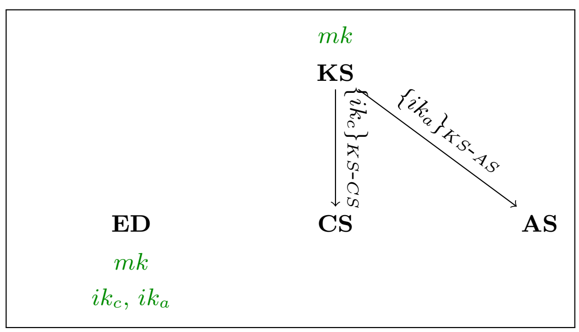
- The scheme is a 2-party AKE protocol that provides mutual authentication.
- The scheme is based on symmetric-key functions solely.
- The scheme guarantees forward secrecy.

Although it is not related to the main goals we tackle, we add the following requirement in order to improve the flexibility of the 3-AKE protocol:

[7] Conversely, it may also be possible that both keys be computed *at once* during the same run. The same key exchange protocol can be used in either case, the difference lying in an additional derivation step that yields two keys from the unique output of the original 2-party AKE.

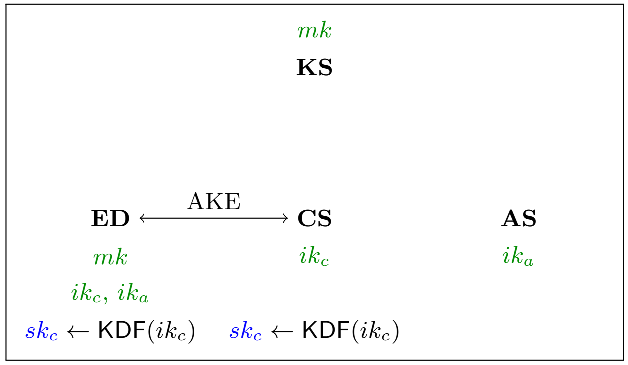


(a) 2-AKE executed between ED and KS (relayed by CS) with mk

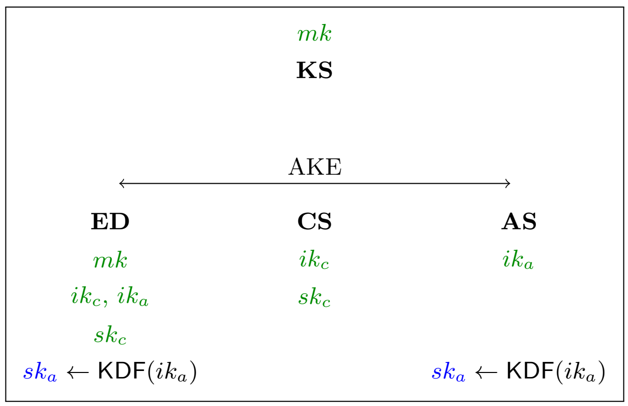


(b) Transmission by KS of intermediary keys ik_c (to CS) and ik_a (to AS) respectively through the secure channels $\{\cdot\}_{KS\text{-}CS}$ established between KS and CS, and $\{\cdot\}_{KS\text{-}AS}$ established between KS and AS

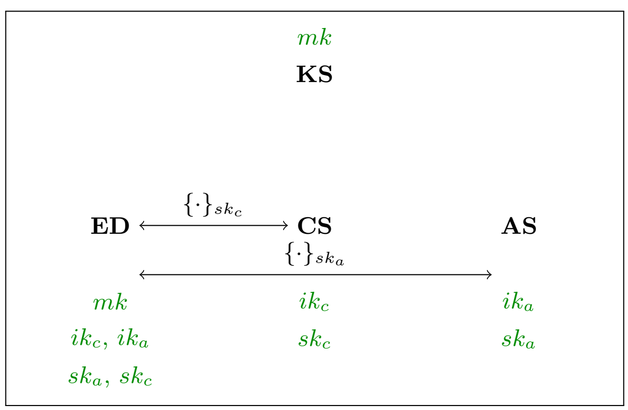
Fig. 2: 2-AKE executed between ED and KS with mk, and distribution of ik_c, ik_a



(a) 2-AKE executed between ED and CS with ik_c



(b) 2-AKE executed between ED and AS (relayed by CS) with ik_a



(c) Secure channels established: communication channel $\{\cdot\}_{sk_c}$ between ED and CS, and application channel $\{\cdot\}_{sk_a}$ between ED and AS

Fig. 3: 2-AKE executed between ED and AS (resp. CS) with ik_a (resp. ik_c), and subsequent secure channels

– Any of the two parties can initiate a run of protocol P.

Combining symmetric-key cryptography and forward secrecy may appear counterintuitive. Therefore, we informally recall what such a property means in that context. Once a 2-AKE run of P is complete, *past* output secrets must remain private even if the current symmetric root key (used to authenticate the parties and compute the shared secret) is revealed.

More precisely, in a 2-AKE run done between ED and KS, the disclosure of the current master key mk (used as root key) must not compromise past intermediary keys ik computed by these two parties. Likewise, in a 2-AKE run done between ED and some XS $\in \{\text{CS}, \text{AS}\}$, the disclosure of the current intermediary key ik (used as root key in that case) must not compromise past session keys sk computed by ED and XS.[8]

Protocol P'. We demand P' to be a secure 2-AKE protocol that provides mutual authentication, and forward secrecy. Since P' is applied between KS and XS, asymmetric functions may be used.

Function Enc. We demand Enc to provide data confidentiality and data authenticity. In the latter we include non-replayability of messages.

2.4 Main Features of the 3-AKE Protocol

In Section 4, we formally define the properties we demand for a 3-AKE protocol, and prove that P, P', and Enc yield a secure 3-AKE protocol. Before, we detail in this section the main features provided by our 3-AKE protocol and informally justify these properties.

Management of the security. The key hierarchy (between mk, ik, and sk), allows ED and KS to manage the overall security of the system. The key exchange done between KS and ED (steps 1-2, Section 2.2) can be initiated by any but *only* these two entities. Each 2-AKE done between ED and KS creates a new intermediary key ik. This obsoletes the current intermediary key shared by ED and XS $\in \{\text{CS}, \text{AS}\}$, and "disconnects" ED from XS by resetting ik at ED. Hence, KS and ED can defend against a dishonest or corrupted XS.

Cryptographic separation of the layers. The use of two distinct intermediary keys ik_c and ik_a allows separating the communication layer (between ED and CS) and the application layer (between ED and AS). The mutual authentication done between KS and, respectively, CS and AS, guarantees that the intermediary keys are sent to and received from legitimate parties only.ly.

[8] A concrete instantiation of P is given in Section 5.

Secure connection to any server. The 3-AKE protocol allows ED to share an intermediary key ik with *any* (communication or application) server. Moreover, ED can connect any such server without impairing the security with another server. First, each 2-AKE run done between ED and KS yields a different intermediary key ik. Hence each partnered ED and XS use a distinct key ik. Next, KS deletes its copy of ik as soon as it has been received by XS. Finally, P provides forward secrecy. The disclosure of the current master key mk (stored at ED and KS) does not compromise a past output key ik. The forward secrecy ensured by P' and the security of the channel established with Enc participate also in the privacy of ik. Likewise, due to the forward secrecy of P, past session keys sk (computed between ED and XS) remain private, even if the current key ik (stored at ED and XS) is exposed

Quick session establishment. Once a first intermediary key ik is shared between ED and XS, these two parties can perform as many 2-AKE runs (hence set up as many successive secure channels) as wished *without* soliciting KS anymore (i.e., ED and XS repeat several times step 4 or 5, Section 2.2). This avoids overloading KS (which has to manage many ED and XS). At the same time this hides to KS the number and the frequency of the connections established between ED and XS.

3 Session Resumption Procedure

3.1 Rationale for a Session Resumption Procedure

As explained in Section 2.4, after a first 2-AKE run with KS, ED shares an intermediary key ik with $\mathrm{XS} \in \{\mathrm{CS}, \mathrm{AS}\}$. Then, ED and XS can execute, from ik, subsequent 2-AKE runs without soliciting KS anymore. Consequently, as soon as ED shares (distinct) intermediary keys with several servers, it can quickly switch from one server to another back and forth without the help of KS. This is particularly convenient for a mobile ED which must connect different communication providers (hence different CS servers). Likewise, this allows ED to connect several AS servers, hence to be securely used by different application providers. Moreover, since P guarantees forward secrecy, the disclosure of (the current value of) ik does not compromise past session keys sk. We call this faster mode (without KS) a *session resumption* procedure.

Due to the intrinsic properties of the 2-AKE scheme P (see Section 2.3), any peer (ED or XS) can initiate the key exchange. This implies that *both* peers can initiate the session resumption procedure.

The main benefit of this procedure is to give the ability to switch between servers without soliciting KS. Avoiding the involvement of KS (that is, avoiding a whole 2-AKE run between ED and KS), allows to save time, computation cost and communication cost for KS but mainly for ED. Indeed, the ED we consider are low-resource, self-powered devices. The energy cost to transmit and to receive data usually exceeds the cost of cryptographic processing [26]. Hence it is worth saving as much as possible the amount of data exchanged to compute a new session key.

Another limitation of a low-resource ED is its memory space. Being able to resume a session with several servers implies to store simultaneously as many intermediary keys. This is likely possible for a server but becomes prohibitive for such kind of ED. In Section 3.2 we present a session resumption scheme that solves this issue.

3.2 Session Resumption Procedure for Low-resource ED

Overview of the Procedure. The session resumption procedure for a low-resource ED with $\mathrm{XS} \in \{\mathrm{CS}, \mathrm{AS}\}$ is made of two phases:

(a) The *storage phase*. ED and XS have an ongoing secure channel set up with a session key sk (output by P). Both share an intermediary key ik. First, ED encrypts ik under a key known only to itself (we elaborate on this in Section 3.2). Next, ED sends this "ticket" to XS through the ongoing secure channel. Upon reception of the ticket by XS, ED deletes ik. Then ED can close the channel any time.

(b) The *retrieval phase*. ED starts a new 2-AKE run with a known XS. First, ED gets, in the continuity of the run, the ticket it has sent previously. Next, ED decrypts the ticket and gets the corresponding key ik. Then, ED and XS complete the run with ik, and compute a new session key sk.

This procedure is reminiscent of existing schemes (e.g., [21, 25, 27]). However none of the latter succeeds in combining session resumption and forward secrecy without asymmetric cryptography or prohibitive requirements (for a constrained ED) regarding memory, or the amount of transmitted data [2, 21]. In contrast, our 3-AKE protocol provides a nifty solution to this issue, as explained below.

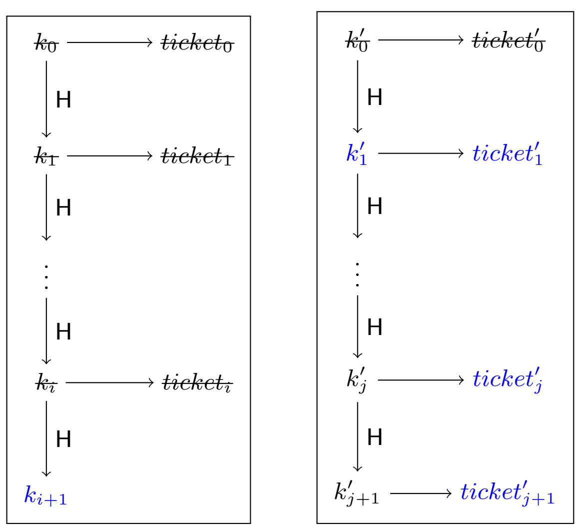
Computing the Ticket. The intermediary key ik that is stored at the server and later retrieved by ED is encrypted. Only ED needs to decrypt ik since the server stores its own copy of the key. Using the same encryption key k to protect different intermediary keys (sent to different servers) obviously breaks forward secrecy: revealing k allows decrypting *past* intermediary keys, hence compromising the session key sk computed with the latter. Therefore each intermediary key must be encrypted with a different key k. However, replacing in ED's memory each intermediary key ik with another (encryption) key k yields the same memory issue and is pointless. Therefore, we compute the keys k used to encrypt the intermediary keys as elements of a *one-way key chain*.

From an initial random key k_0, each ticket is computed as $ticket_{i+1} = \mathsf{KW}(k_{i+1}, ik)$ with $k_{i+1} = \mathsf{H}(k_i)$, $i \geq 0$. KW is a key-wrap function [24], and H a one-way function. ED keeps in memory only one key k_j. This key is the child of the key that has decrypted the last used ticket. When ED wants to consume $ticket_i$, it first computes the decryption key k_i from the current key k_j, $i \geq j$: $k_i = \mathsf{H}^{i-j}(k_j)$. Then k_j is replaced with $k_{i+1} = \mathsf{H}(k_i)$, and $ticket_i$ *cannot* be decrypted anymore.

This *unique* encryption key gives ED the ability to compute multiple tickets, therefore to resume as many sessions.

Two Chains of Keys. When $ticket_i$ is used, the current decryption key is replaced with $k_{i+1} = \mathsf{H}(k_i)$. Hence any previous $ticket_j$, $j \leq i$, is obsoleted. Let us consider the following scenario. A mobile low-resource ED is managed by one AS, and switches back and forth between two other servers CS_a and CS_b. ED stores fresh $ticket_i$, $ticket_j$, and $ticket_k$, $i < j < k$, respectively at AS, CS_a, and CS_b. ED keeps the decryption key k_i. When ED makes a new key exchange with CS_a, it retrieves $ticket_j$ and decrypts it with $k_j = \mathsf{H}^{j-i}(k_i)$. Then, ED replaces the current key k_i with $k_{j+1} = \mathsf{H}(k_j)$. Whenever ED alternates between CS_a and CS_b, the ticket decryption key is updated. Consequently, ED cannot use $ticket_i$. Even though $ticket_i$ was the most recent ticket, it would be obsoleted at some point. This makes the session resumption procedure unusable with AS. Therefore, we advocate the use of two chains of decryption keys corresponding to the *two types* of CS and AS servers, and the two possibly different behaviours of ED (see Figure 4). Nonetheless, if a different context requires so, a unique chain of decryption keys can also be maintained. Note that, if the tickets are used in the same order they are computed, all can be (legitimately) decrypted.

Figure 4a depicts the case where a CS ticket ($ticket_i$) is used. The corresponding decryption key k_i is deleted, and ED keeps only k_{i+1}. This obsoletes all previous CS tickets. Figure 4b depicts the case where an AS ticket ($ticket'_0$) is used. The decryption key k'_0 is deleted, and ED keeps only k'_1. All AS $ticket'_j$, $j \geq 1$, are still usable.



(a) CS tickets and keys (b) AS tickets and keys

Fig. 4: Chains of keys used to compute a ticket

Maintaining Forward Secrecy. When $ticket_i$ is used, the current encryption key k_j, $j \leq i$, stored at ED, is replaced with the next encryption key $k_{i+1} = \mathsf{H}(k_i)$. This forbids any old ticket from being decrypted. All the remaining tickets that can be decrypted (from the now current key k_{i+1}) have not been used yet. Moreover, the protocol P provides forward secrecy. Hence, the disclosure of the intermediary key ik protected into a (not used yet) ticket does *not* compromise past session keys sk. In a way, the session resumption procedure inherits the forward secrecy from P (and also from the one-wayness of H).

The use of the (forward secret) intermediary key ik highlights also why encrypting the session key sk in the ticket is not a good choice. The more data the *same* session key protects, the worse its disclosure.[9]

4 3-AKE Security Model

Before describing in Section 5 a concrete instantiation of our generic 3-AKE protocol, we present the essential building blocks of the security model that we employ to formally prove the security of the 3-AKE protocol and its instantiation.

In a nutshell, we use the security experiments of a 2-AKE model (entity authentication, key indistinguishability), as described by Brzuska, Jacobsen, and Stebila [12]. Taking inspiration from the 3(S)ACCE model of Bhargavan, Boureanu, Fouque, Onete, and Richard [10], we extend the 2-AKE model to incorporate the three parties of our 3-AKE protocol, and their interleaved operations.

In our 3-AKE security model, the adversary has full control over the communication network. It can forward, alter, drop any message exchanged by honest parties, or insert new messages. Our 3-AKE model captures also *forward secrecy.*

4.1 Preliminaries

In this section, we recall the definitions of the basic security notions we use in our results. The security definitions of a secure pseudo-random function (PRF), can be found in Bellare, Desai, Jokipii, and Rogaway [6]. We take the definition of a (stateful) authenticated encryption scheme (sAE) from Shrimpton [29] that we rephrase below. The security definition of a MAC strongly unforgeable under chosen-message attacks ($\mathsf{SUF\text{-}CMA}$) can be found in Bellare and Namprempre [7].

Secure PRF. A *pseudo-random function* (PRF) F is a deterministic algorithm which given a key $K \in \{0,1\}^\lambda$ and a bit string $x \in \{0,1\}^*$ outputs a string

[9] Another reason to opt for ik is *efficiency.* In fact, sk may be quite large (e.g., two pairs of keys, encryption and MAC, for each direction, and the last value of the uplink and downlink frame counters). The ticket is transmitted twice between ED and XS. As explained above, the amount of data exchanged with the server is a burden for a wireless low-resource ED. From a single intermediary key ik, any kind of security parameters can be computed. Hence the choice of ik.

$y = F(K, x) \in \{0,1\}^\gamma$ (with γ being polynomial in λ). Let $Func$ be the set of all functions of domain $\{0,1\}^*$ and range $\{0,1\}^\gamma$. The security of a PRF is defined with the following experiment between a challenger and an adversary $\mathcal{A}$:

1. The challenger samples $K \xleftarrow{\$} \{0,1\}^\lambda$, $G \xleftarrow{\$} Func$, and $b \xleftarrow{\$} \{0,1\}$ uniformly at random.
2. The adversary may adaptively query values x to the challenger. The challenger replies to each query with either $y = F(K, x)$ if $b = 1$, or $y = G(x)$ if $b = 0$.
3. Finally, the adversary outputs its guess $b' \in \{0,1\}$ of b.

The adversary's advantage is defined as

$$\mathsf{adv}_F^{\mathsf{PRF}}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right|.$$

Definition 1 (Secure PRF). *A function $F: \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\gamma$ is said to be a* secure pseudo-random function (PRF) *if, for all probabilistic polynomial time adversary $\mathcal{A}$, $\mathsf{adv}_F^{\mathsf{PRF}}(\mathcal{A})$ is a negligible function in λ.*

Secure MAC. A message authentication code (MAC) consists of two algorithms $(\mathsf{Mac}, \mathsf{Vrf})$. The tagging algorithm Mac takes as input a key $K \in \{0,1\}^k$ and a message $m \in \{0,1\}^*$ and returns a tag $\tau \in \{0,1\}^\gamma$ (with γ being polynomial in k). The verification algorithm Vrf takes as input the key K, a message m, and a candidate tag τ for m. It outputs 1 if τ is a valid tag on message m with respect to K. Otherwise, it returns 0. The notion of *strong unforgeability under chosen-message attacks* ($\mathsf{SUF\text{-}CMA}$) for a MAC $G = (\mathsf{Mac}, \mathsf{Vrf})$ is defined with the following experiment between a challenger and an adversary $\mathcal{A}$:

1. The challenger samples $K \xleftarrow{\$} \{0,1\}^k$, and sets $S \leftarrow \emptyset$.
2. The adversary may adaptively query values m to the challenger. The challenger replies to each query with $\tau = \mathsf{Mac}(K, m)$ and records (m, τ): $S \leftarrow S \cup \{(m, \tau)\}$.
3. Finally, the adversary sends (m^*, τ^*) to the challenger.

The adversary's advantage is defined as

$$\mathsf{adv}_G^{\mathsf{SUF\text{-}CMA}}(\mathcal{A}) = \Pr[\mathsf{Vrf}(K, m^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin S].$$

Definition 2 ($\mathsf{SUF\text{-}CMA}$). *A message authentication code $G = (\mathsf{Mac}, \mathsf{Vrf})$ with $\mathsf{Mac}: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^\gamma$ is said to be* strongly unforgeable under chosen-message attacks ($\mathsf{SUF\text{-}CMA}$) *if, for all probabilistic polynomial time adversary $\mathcal{A}$, $\mathsf{adv}_G^{\mathsf{SUF\text{-}CMA}}(\mathcal{A})$ is a negligible function in k.*

Stateful Authenticated Encryption. A *stateful authenticated encryption scheme* (sAE) consists of two algorithms $\mathsf{StAE} = (\mathsf{StAE.Enc}, \mathsf{StAE.Dec})$. The encryption algorithm, given as $(C, st'_e) \leftarrow \mathsf{StAE.Enc}(K, H, M, st_e)$, takes as input a secret key $K \in \{0,1\}^\lambda$, a header data $H \in \{0,1\}^*$, a plaintext M, and the current encryption state $st_e \in \{0,1\}^*$. It outputs an updated state st'_e, and either a ciphertext $C \in \{0,1\}^*$ or an error symbol $\perp$. The decryption algorithm, given as $(M, st'_d) \leftarrow \mathsf{StAE.Dec}(K, H, C, st_d)$, takes as input a key K, a header data H, a ciphertext C, and the current decryption state st_d. It outputs an updated state st'_d, and either a value M, which is the message encrypted in C, or an error symbol $\perp$. The states st_e and st_d are initialised to the empty string $\emptyset$. The security of a sAE scheme is defined with the following experiment between a challenger and an adversary $\mathcal{A}$:

1. The challenger samples uniformly at random $K \xleftarrow{\$} \{0,1\}^\lambda$, and $b \xleftarrow{\$} \{0,1\}$.
2. The adversary may adaptively query the encryption oracle $\mathsf{Encrypt}$ and the decryption oracle $\mathsf{Decrypt}$, as described by Figure 5.
3. Finally, the adversary outputs her guess $b' \in \{0,1\}$ of b.

This game captures both the confidentiality and integrity properties of a stateful AEAD scheme. The adversary's advantage is defined as

$$\mathsf{adv}^{\mathsf{sAE}}_{\mathsf{StAE}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}} \Rightarrow 1 | b = 1] - \Pr[\mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}} \Rightarrow 1 | b = 0]\right|.$$

Definition 3 (Secure sAE). *The encryption scheme* StAE *is said to be a* secure stateful authenticated encryption scheme (sAE) *if, for all probabilistic polynomial time adversary $\mathcal{A}$, $\mathsf{adv}^{\mathsf{sAE}}_{\mathsf{StAE}}(\mathcal{A})$ is a negligible function in λ.*

$\underline{\mathsf{Encrypt}(M, H)}$	$\underline{\mathsf{Decrypt}(C, H)}$
$u \leftarrow u + 1$	if $b = 0$ then return $\perp$
$M^0 \xleftarrow{\$} \{0,1\}^{\lvert M\rvert}$	$v \leftarrow v + 1$
$M^1 \leftarrow M$	$(M, st_d) \leftarrow \mathsf{StAE.Dec}(K, H, C, st_d)$
$(C^b, st^b_e) \xleftarrow{\$} \mathsf{StAE.Enc}(K, H, M^b, st_e)$	if $v > u$ or $C \neq C_v$ or $H \neq H_v$
if $C^b = \perp$ then return $\perp$	then $sync \leftarrow \texttt{false}$
$(C_u, H_u, st_e) \leftarrow (C^b, H, st^b_e)$	if $sync = \texttt{false}$ then return M
return C_u	return $\perp$

Fig. 5: The $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ oracles in the sAE security experiment. The counters u and v are initialised to 0, and *sync* to `true` at the beginning of the experiment.

4.2 Execution Environment

Protocol Entities. Our model considers three sets of parties: a set $\mathcal{K}$ of KS servers, a set $\mathcal{E}$ of ED parties, and a set $\mathcal{X}$ of XS $\in$ {CS, AS}. Each party is given a long term key ltk.

Session Instances. Each party P_i maintains a set of instances $P_i.\mathsf{Instances} = \{\pi_i^0, \pi_i^1, \ldots\}$ modeling several (sequential or parallel) executions of the 3-party protocol Π. Each instance π_i^n has access to the long term key $P_i.\mathsf{ltk}$ of its party parent P_i. Moreover, each instance π_i^n maintains the following internal state:

- The *instance parent* $\pi_i^n.\mathsf{parent} \in \mathcal{K} \cup \mathcal{E} \cup \mathcal{X}$ indicating the party owning that instance.
- The *partner-party* $\pi_i^n.\mathsf{pid} \in \mathcal{K} \cup \mathcal{E} \cup \mathcal{X}$ indicating the intended party partner. A party in one of the three sets can be partnered with a party belonging to any of the two other sets.
- The *role* $\pi_i^n.\rho \in \{\mathsf{ed}, \mathsf{ks}, \mathsf{cs}, \mathsf{as}\}$ of $P_i = \pi_i^n.\mathsf{parent}$. If $P_i \in \mathcal{E}$, then $\pi_i^n.\rho = \mathsf{ed}$. If $P_i \in \mathcal{K}$, then $\pi_i^n.\rho = \mathsf{ks}$. If $P_i \in \mathcal{X}$, then $\pi_i^n.\rho \in \{\mathsf{as}, \mathsf{cs}\}$.
- The *session identifier* $\pi_i^n.\mathsf{sid}$ of an instance.
- The *acceptance flag* $\pi_i^n.\alpha \in \{\perp, \mathtt{running}, \mathtt{accepted}, \mathtt{rejected}\}$ originally set to `running` when the session is ongoing, and set to `accepted`/`rejected` when the party accepts/rejects the partner's authentication.
- The *session key* $\pi_i^n.\mathsf{ck}$ set to $\perp$ at the beginning of the session, and set to a non-null bitstring once π_i^n computes the session key.
- The *key material* $\pi_i^n.\mathsf{km}$. If $\pi_i^n.\mathsf{parent} \in \mathcal{K}$ (resp. $\pi_i^n.\mathsf{parent} \in \mathcal{X}$) and $\pi_i^n.\mathsf{pid} \in \mathcal{X}$ (resp. $\pi_i^n.\mathsf{pid} \in \mathcal{K}$), then $\pi_i^n.\mathsf{km}$ is set to $\perp$ at the beginning of the session, and set to a non-null bitstring once π_i^n ends in accepting state. Otherwise $\pi_i^n.\mathsf{km}$ is always set to $\perp$.
- The *status* $\pi_i^n.\kappa \in \{\perp, \mathtt{revealed}\}$ of the session key $\pi_i^n.\mathsf{ck}$.
- The *transcript* $\pi_i^n.\mathsf{trscrpt}$ of the messages sent and received by π_i^n.
- The *security bit* $\pi_i^n.\mathsf{b} \in \{0, 1\}$ sampled at random at the beginning of the security experiments.
- The *partner-instances set* $\pi_i^n.\mathsf{ISet}$ stores the *instances* that are involved in the same protocol run as π_i^n (including π_i^n).
- The *partner-parties set* $\pi_i^n.\mathsf{PSet}$ stores the *parties* parent of the instances in $\pi_i^n.\mathsf{ISet}$ (including $\pi_i^n.\mathsf{parent}$).

Adversarial queries. The adversary $\mathcal{A}$ is assumed to control the network, and interacts with the instances by issuing to them the queries described below.

In our 3-AKE model, we use familiar queries. Nonetheless we require some restrictions regarding the Test-query. This query aims at "evaluating" the quality of a key output by any of the 2-AKE runs done during a 3-AKE session. We use the vanilla real-or-random experiment. Nonetheless, some session keys output during a 3-AKE session are used in the same session, which allows the adversary to trivially distinguish between a "real" session key and a random key. Consequently, we forbid the adversary from issuing a Test-query with respect to a key as soon as this key is used (i.e., as input to a function).

Moreover, we require the adversary to be stateless with respect to the Test-query. That is, the key k_b sent in response to a Test-query cannot be used to interact with instances, nor contribute to answering other Test-challenges.

Instead of proving that the key is *good*, one could consider proving that the key is *good to be used for* some purpose [11,20]. But we chose not to use a weaker notion than the more established ones despite the necessity of these restrictions.

- $\mathsf{NewSession}(P_i, \rho, \mathsf{pid})$: this query creates a new instance π_i^n at party P_i, having role ρ, and intended partner pid.
- $\mathsf{Send}(\pi_i^n, M)$: this query allows the adversary to send any message M to π_i^n. If $\pi_i^n.\alpha \neq \texttt{running}$, it returns $\perp$. Otherwise π_i^n responds according to the protocol specification.
- $\mathsf{Corrupt}(P_i)$: this query returns the long-term key $P_i.\mathsf{ltk}$ of P_i. If $\mathsf{Corrupt}(P_i)$ is the ν-th query issued by the adversary, then we say that P_i is *ν-corrupted*. For a party that has not been corrupted, we define $\nu = +\infty$.
- $\mathsf{Reveal}(\pi_i^n)$: this query returns the session key $\pi_i^n.\mathsf{ck}$, and $\pi_i^n.\kappa$ is set to $\texttt{revealed}$. If $\mathsf{Reveal}(\pi_i^n)$ is the ν-th query issued by the adversary, then we say that π_i^n is *ν-revealed*. For a party that has not been revealed, we define $\nu = +\infty$.
- $\mathsf{Test}(\pi_i^n)$: this query may be asked only once per pairwise partnered instances throughout the game. If $\pi_i^n.\alpha \neq \texttt{accepted}$, then it returns $\perp$. Otherwise it samples an independent key $k_0 \xleftarrow{\$} \mathcal{KEY}$, and returns k_b, where $k_1 = \pi_i^n.\mathsf{ck}$. The key k_b is called the Test-*challenge*. We forbid the adversary from issuing a Test-query, and answering a Test-challenge as soon as the corresponding key $\pi_i^n.\mathsf{ck}$ is used during the session. Moreover, the adversary is stateless with respect to this query (it does not keep track of k_b).

4.3 Security Definitions

Partnership. In order to define the partnership between two instances involved in a 2-AKE run, we use the definition of matching conversations initially proposed by Bellare and Rogaway [8], and modified by Jager, Kohlar, Schäge and Schwenk [18].

Let $T_{i,n}$ be the sequence of all (valid) messages sent and received by an instance π_i^n in chronological order. For two transcripts $T_{i,n}$ and $T_{j,u}$, we say that $T_{i,n}$ is a prefix of $T_{j,u}$ if $T_{i,n}$ contains at least one message, and the messages in $T_{i,n}$ are identical to the first $|T_{i,n}|$ messages of $T_{j,u}$.

Definition 4 (Matching Conversations). *We say that π_i^n has a* matching conversation *to π_j^u, if*

- *π_i^n has sent all protocol messages and $T_{j,u}$ is a prefix of $T_{i,n}$, or*
- *π_j^u has sent all protocol messages and $T_{i,n} = T_{j,u}$.*

Consequently, we define sid to be the transcript, in chronological order, of all the (valid) messages sent and received by an instance during a 2-AKE run, but, possibly, the last message. We say that two instances π_i^n and π_j^u are pairwise partnered if $\pi_i^n.\mathsf{sid} = \pi_j^u.\mathsf{sid}$. Then, we define the 3-AKE partnering with the sets ISet and PSet. $\pi_i^n.\mathsf{ISet}$ stores instances partnered with π_i^n, and $\pi_i^n.\mathsf{PSet}$ stores parties partnered with π_i^n.

Definition 5 (Correctness). *We define the* correctness *of a 3-AKE protocol as follows. We demand that, for any instance π ending in an accepting state, the following conditions hold:*

- $|\pi.\mathsf{ISet}| = 6$. *Let* $\pi.\mathsf{ISet}$ *be* $\{\pi_i^n, \pi_i^m, \pi_k^\ell, \pi_k^s, \pi_j^u, \pi_j^v\}$.
- $\pi_i^n.\mathsf{parent} = \pi_i^m.\mathsf{parent} = P_i \in \mathcal{E}$
- $\pi_k^\ell.\mathsf{parent} = \pi_k^s.\mathsf{parent} = P_k \in \mathcal{K}$
- $\pi_j^u.\mathsf{parent} = \pi_j^v.\mathsf{parent} = P_j \in \mathcal{X}$
- $\pi.\mathsf{PSet} = \{P_i, P_j, P_k\}$
- $\pi_i^m.\mathsf{sid} = \pi_k^\ell.\mathsf{sid} \neq \perp$ *and* $\pi_i^m.\mathsf{ck} = \pi_k^\ell.\mathsf{ck} \neq \perp$
- $\pi_i^n.\mathsf{sid} = \pi_j^u.\mathsf{sid} \neq \perp$ *and* $\pi_i^n.\mathsf{ck} = \pi_j^u.\mathsf{ck} \neq \perp$
- $\pi_k^s.\mathsf{sid} = \pi_j^v.\mathsf{sid} \neq \perp$ *and* $\pi_k^s.\mathsf{ck} = \pi_j^v.\mathsf{ck} \neq \perp$
- $\pi_j^v.\mathsf{km} = \pi_k^s.\mathsf{km} = \pi_i^m.\mathsf{ck} = \pi_k^\ell.\mathsf{ck}$
- $\exists\ f \mid f(\pi_i^m.\mathsf{ck}, \pi_i^n.\mathsf{trscrpt}) = \pi_i^n.\mathsf{ck} = \pi_j^u.\mathsf{ck} = f(\pi_j^v.\mathsf{km}, \pi_j^u.\mathsf{trscrpt})$

The last two conditions aim at "binding" the six instances involved in a 3-AKE run. Function f corresponds typically to the session key derivation function used by P_i (ED) and P_j (XS $\in \{\text{CS}, \text{AS}\}$) together.

More concretely, ck corresponds either to ik output by the 2-AKE run done between ED and KS, or to sk output by the 2-AKE run done between ED and XS (both with protocol P), or to the session key output by the 2-AKE run done between KS and XS (with protocol P'). km denotes the intermediary key ik sent by KS to XS (see Figure 6).

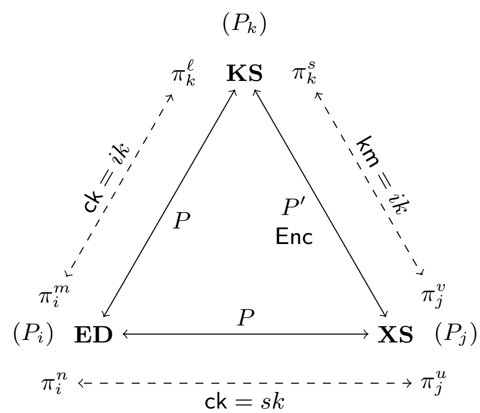


Fig. 6: The six instances involved in a correct 3-AKE run

Security of a 3-AKE protocol is defined in terms of an experiment played between a challenger and an adversary. This experiment uses the execution environment described in Section 4.2. The adversary can win the 3-AKE experiment in one of two ways: (i) by making an instance accept maliciously, or (ii) by guessing the secret bit of the Test-instance. In both, the adversary can query all oracles $\mathsf{NewSession}$, Send, Reveal, $\mathsf{Corrupt}$, and Test.

Entity Authentication (EA). This security property must guarantee that (i) any instance $\pi \in \{\pi_i^n, \pi_i^m, \pi_k^\ell, \pi_k^s, \pi_j^u, \pi_j^v\}$ ending in accepting state is pairwise partnered with a *unique* instance, and (ii) the output of a 2-AKE run done between P_k and P_i is *used as root key* in a 2-AKE run done between P_i and P_j.

Definition 6 (Entity Authentication (EA)). *An instance π of a protocol Π is said to* maliciously accept *in the 3-AKE security experiment with intended partner $\tilde{P}$, if*

(a) $\pi.\alpha = \mathtt{accepted}$ *and* $\pi.\mathsf{pid} = \tilde{P}$ *when $\mathcal{A}$ issues its ν_0-th query.*
(b) Any party in $\pi.\mathsf{PSet}$ *is ν-corrupted with $\nu > \nu_0$.*
(c) Any instance in $\pi.\mathsf{ISet}$ *is ν'-revealed with $\nu' > \nu_0$.*
(d) There is no unique instance $\tilde{\pi}$ such that $\pi.\mathsf{sid} = \tilde{\pi}.\mathsf{sid}$, *or there is no instances* $\pi_i^m, \pi_i^n, \pi_j^u, \pi_j^v \in \pi.\mathsf{ISet}$ *such that*
  – $\pi_i^m.\mathsf{pid} = \pi_j^v.\mathsf{pid} \in \mathcal{K}$,
  – $\pi_i^n.\mathsf{parent} = \pi_i^m.\mathsf{parent} \in \mathcal{E}$,
  – $\pi_j^u.\mathsf{parent} = \pi_j^v.\mathsf{parent} \in \mathcal{X}$, *and*
  – $f(\pi_i^m.\mathsf{ck}, \pi_i^n.\mathsf{trscrpt}) = \pi_i^n.\mathsf{ck} = \pi_j^u.\mathsf{ck} = f(\pi_j^v.\mathsf{km}, \pi_j^u.\mathsf{trscrpt})$.

The adversary's advantage is defined as its winning probability:

$$\mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) = \Pr[\mathcal{A} \textit{ wins the EA game}].$$

Key Indistinguishability. This security property must guarantee that the adversary can do no more than *guessing* in order to distinguish from random the session key output by any of the 2-AKE runs performed during a 3-AKE protocol session.

Definition 7 (Key Indistinguishability). *An adversary $\mathcal{A}$ against a protocol Π, that issues its* Test*-query to instance π during the 3-AKE security experiment,* answers the Test-challenge correctly *if it terminates with output b', such that*

(a) $\pi.\alpha = \mathtt{accepted}$
(b) Let $\tilde{\pi}$ be the last instance in $\pi.\mathsf{ISet}$ *to end in accepting state:* $\tilde{\pi}.\alpha = \mathtt{accepted}$ *when $\mathcal{A}$ issues its ν_0-th query.*
(c) Any party in $\pi.\mathsf{PSet}$ *is ν-corrupted with $\nu > \nu_0$.*
(d) No instance in $\pi.\mathsf{ISet}$ *has been queried in* Reveal *queries.*
(e) $\pi.\mathsf{b} = b'$

The adversary's advantage is defined as

$$\mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A}) = \left| \Pr[\pi.\mathsf{b} = b'] - \frac{1}{2} \right|.$$

The definitions of entity authentication and key indistinguishability allow an adversary to corrupt a party involved in the 3-AKE security experiment (up to some point, in order to preclude trivial attacks). Therefore, protocols secure with respect to Definition 8 below provide *forward secrecy.*

Definition 8 (3-AKE Security). *A protocol Π is 3-AKE-secure if Π satisfies correctness, and for all probabilistic polynomial time adversary $\mathcal{A}$,* $\mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A})$ *and* $\mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A})$ *are a negligible function of the security parameter.*

4.4 Security Proof of our Generic 3-AKE Protocol

In this section we use the security model described in Sections 4.2 and 4.3 to prove the security of the generic 3-AKE protocol Π depicted in Section 2. The protocol Π is based on: (i) the 2-AKE protocol P executed between ED and KS, ED and XS, (ii) the 2-AKE protocol P' executed between KS and XS, and (iii) the function Enc used to set up a secure channel between KS and XS with a session key output by P'. Informally, the security of the 3-AKE protocol Π relies on the 2-AKE-security [12] of P and P', and on the sAE-security [29] of the function Enc. Based on the security of P, P', and Enc, we show that Π is a secure 3-AKE protocol according to Definition 8.

Theorem 1. *The protocol Π is a secure 3-AKE protocol under the assumption that P is a secure 2-AKE protocol, P' is a secure 2-AKE protocol, and* Enc *is a secure* sAE *function, and for any probabilistic polynomial time adversary $\mathcal{A}$ in the 3-AKE security experiment against Π*

$$\begin{aligned}\mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) &= n_K \cdot n_E \cdot n_X \Big[2\mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1) + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0) + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) \\ &\qquad + 2\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2)\Big] \\ \mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A}) &= \mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \Big[2\mathsf{adv}_{P}^{\mathsf{key\text{-}ind}}(\mathcal{B}_1) + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) \\ &\qquad + \mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2)\Big]\end{aligned}$$

where n_K, n_E, n_X are respectively the number of KS, ED, and XS parties, and $\mathcal{B}_0$ is an adversary against the 2-AKE-security of P', $\mathcal{B}_1$ an adversary against the 2-AKE-security of P, and $\mathcal{B}_2$ an adversary against the sAE*-security of* Enc.

Below, we give a sketch proof of Theorem 1. The full proof is given in Appendix A.

Entity authentication. First we consider the entity authentication experiment described in Section 4.3. We use the following hops.

Game 0. This game corresponds to the entity authentication security experiment.

Game 1. The adversary interacts with three unique parties, respectively in the set $\mathcal{K}$, $\mathcal{X}$ and $\mathcal{E}$. This is equivalent to guessing the targeted parties, hence implies a security loss equal to $\frac{1}{n_K \cdot n_E \cdot n_X}$.

Game 2. Let $P_k \in \mathcal{K}$, $P_j \in \mathcal{X}$ and $P_i \in \mathcal{E}$ be the three parties. The challenger aborts the game if the adversary succeeds in impersonating P_j to P_k. We reduce this event to the 2-AKE-security of the protocol P' applied between P_k and P_j. Hence a loss $\mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0)$.

Game 3. The challenger aborts if the adversary succeeds in impersonating P_i to P_k. We reduce this event to the 2-AKE-security of the protocol P applied between P_k and P_i. Hence a loss $\mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1)$.

Game 4. The challenger aborts if the adversary succeeds in getting the intermediary key ik sent by P_k to P_j through the secure channel established with the session key output by P' and the function Enc, or in forging a valid message that carries a key of the adversary's choice. We reduce both events to the sAE-security of Enc. In turn, we must assume that the Enc key is random. The latter is reduced to the 2-AKE-security of P'. Hence a loss $\mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) + 2\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2)$.

Game 5. The challenger aborts if the adversary succeeds in impersonating P_i to P_j (or conversely). We reduce this event to the 2-AKE-security of P. Hence a loss $\mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1)$. To that point, the adversary has no chance to win.

Collecting all the probabilities from Game 0 to Game 5 yields the given bound.

Key indistinguishability. Now we consider the key indistinguishability security experiment described in Section 4.3.

Game 0. This game corresponds to the key indistinguishability security experiment.

Game 1. The challenger aborts and chooses $b' \in \{0, 1\}$ uniformly at random if there exists an instance that maliciously accepts. In other words, we make the same modifications as in the games performed during the entity authentication proof. Hence a loss $\mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A})$.

Game 2. The adversary interacts with three unique parties, respectively in the set $\mathcal{K}$, $\mathcal{X}$ and $\mathcal{E}$. This is equivalent to guessing the targeted parties, hence a loss $\frac{1}{n_K \cdot n_E \cdot n_X}$.

Game 3. Let $P_k \in \mathcal{K}$, $P_j \in \mathcal{X}$ and $P_i \in \mathcal{E}$ be the three parties. In this game, we rely upon the $\mathsf{key\text{-}ind}$-security of the protocol P between P_i and P_k to replace the session key ik with a truly random value. Hence a loss $\mathsf{adv}_{P}^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)$.

Game 4. In this game, we rely upon the $\mathsf{key\text{-}ind}$-security of the protocol P' between P_k and P_j to replace the session key with a truly random value. Hence a loss $\mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0)$.

Game 5. Now the adversary can try to get the key (allegedly ik) sent by P_k to P_j, and to compute the session key sk (shared between P_i and P_j). The key ik is sent by P_k to P_j protected with Enc. Hence we reduce this event to the sAE-security of Enc. In turn, this relies implicitly on the fact that the encryption key (output by P') used to key Enc be indistinguishable from random. This is the case due to Game 4. Therefore the challenger aborts the game if the adversary succeeds in getting ik. Hence a loss $\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2)$.

Game 6. The challenger aborts if the adversary succeeds in breaking the $\mathsf{key\text{-}ind}$-security of P between P_i and P_j. Hence a loss $\mathsf{adv}_{P}^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)$. To that point, the adversary can do no better than guessing.

Collecting all the probabilities from Game 0 to Game 6 yields the given bound.

5 Instantiation of the 3-AKE Protocol

In this section we present a concrete instantiation of our 3-AKE protocol described in Section 2. We have to choose a 2-AKE-secure protocol P, a 2-AKE-secure protocol P', and a sAE-secure function Enc. Recall that the protocol P must fulfill the properties listed in Section 2.3, which includes the essential *forward secrecy*.

P is a symmetric-key based protocol, whereas P' can implement asymmetric schemes. Therefore we define the long term key ltk of each party to be $\mathsf{ltk} = (\mathsf{pubk}, \mathsf{prvk}, \mathsf{rootk})$ where (i) pubk is a certified public key, (ii) prvk is the corresponding private key, and (iii) rootk is a symmetric root key. For any party in $\mathcal{K}$, the three components of ltk are defined. For any party in $\mathcal{X}$, ltk is fully defined *after* the first 3-AKE run (before, $\mathsf{rootk} = \perp$). For any party in $\mathcal{E}$, $\mathsf{ltk} = (\perp, \perp, \mathsf{rootk})$.

We describe an instantiation of P with (Section 5.3) and without (Section 5.2) the session resumption procedure for low-resource ED.

5.1 Protocol P' and Function Enc

We instantiate the 2-AKE protocol P' with TLS 1.3. In order not to impair the security, we enforce (EC)DHE and forbid 0-RTT mode. We define the Enc function to be the record layer of TLS 1.3.

5.2 Forward Secret 2-AKE Protocol P

We instantiate the 2-AKE protocol P with the SAKE protocol [3]. SAKE fulfills all the properties listed in Section 2.3.[10]

SAKE uses a key-evolving scheme, based on a one-way function, to update the symmetric root key shared by the two peers. The root key is made of two main components: a derivation key K, and an independent authentication key K'. K is used in the session key derivation. K' allows authenticating the messages, tracking the root key evolution (i.e., its successive updates), and, if necessary, resynchronising in the *continuity* of the protocol run. After a complete and correct run of SAKE, both peers have the guarantee that their root key is updated and synchronised. That is, SAKE is *self-synchronising*. When *any* of the two peers deems the session key can be safely used, it has the guarantee that its partner is synchronised (in particular, it is not late). Hence, SAKE guarantees forward secrecy.

Thus, applying SAKE, ED and KS compute an intermediary ik with their shared master key mk (used as SAKE root key). The current master key is then updated with the one-way function update: $mk^{i+1} \leftarrow \mathsf{update}(mk^i)$. Likewise, applying SAKE, ED and XS $\in \{\mathrm{CS}, \mathrm{AS}\}$ compute a session key sk with the key ik they share (used as SAKE root key in that case). Eventually, the SAKE root

[10] Any other 2-AKE protocol can be used, as long as it provides the same properties as SAKE, but we are not aware of other such protocols.

key used in that case is updated: $ik^{t+1} \leftarrow \mathsf{update}(ik^t)$.

Figure 7 depicts the evolution of the three types of keys over time: the master key mk, the intermediary key ik, and the session key sk. The computation of ik^0 and mk^1 from mk^0 corresponds to the 2-AKE run executed between ED and KS as depicted by Figure 2a. The computation of sk^2 and ik^3 from ik^2 corresponds to the 2-AKE run done between ED and CS (resp. AS) with $ik = ik_c$ (resp. $ik = ik_a$) as depicted by Figure 3a (resp. Figure 3b). Note that the keys ik_c and ik_a are computed from two different values mk (i.e., yielded by two different 2-AKE runs between ED and KS). In Figure 7, the branch $mk^0 \to mk^1 \to \cdots$ corresponds to the evolution of mk throughout successive key exchange runs executed between ED and KS. Each of these runs yields a new intermediary key $ik = ik^0$. The branch $ik^0 \to ik^1 \to \cdots$ corresponds to the evolution of ik throughout successive key exchange runs (each one outputting a session key sk^i) executed between ED and XS *without* the involvement of KS.

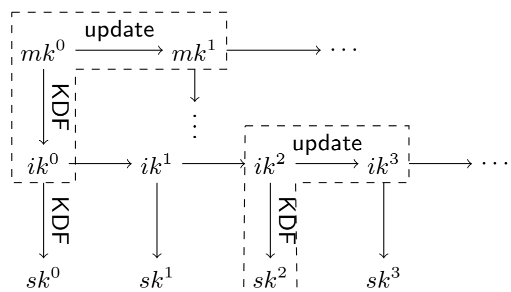


Fig. 7: Key chains

5.3 Protocol P with Session Resumption Scheme for Low-resource ED

In this section, we describe a session resumption scheme dedicated to low-resource ED. This scheme (i) fulfills the features of the procedure described in Section 3.2, and (ii) is a 2-AKE-secure protocol (which include, in particular, forward secrecy). Recall that the session resumption procedure is made of two phases (see Section 3.2): the *storage phase* (phase (a)) and the *retrieval phase* (phase (b)). The *retrieval phase* of the scheme for low-resource ED is a variant of the SAKE protocol adapted to include the use of the ticket. We call this variant *SAKE-R*.

Session Resumption Procedure with SAKE-R. Figure 8 depicts the two phases of the procedure regarding the evolution of the keys.

Figure 9 depicts the *storage phase* of the session resumption procedure. The computation $\mathsf{H}^n(k_j) = \mathsf{H}(\cdots\mathsf{H}(\mathsf{H}(k_j))\cdots)$ corresponds to n times the application of function H, where n is the "distance" between the current key k_j stored

$$
\begin{tikzcd}
mk^0 \arrow[r] \arrow[d] & mk^1 \arrow[r] & \cdots & \\
 & & & \textit{pause (phase (a))} \\
ik^0 \arrow[r] \arrow[d] & ik^1 \arrow[r] \arrow[d] & \boxed{ik^2} \arrow[ur, dashed, bend left] & \\
 & & & \textit{resume (phase (b))} \arrow[dl, dashed] \\
sk^0 & sk^1 & ik^2 \arrow[r] \arrow[d] & ik^3 \arrow[r] \arrow[d] & \cdots \\
 & & sk^2 & sk^3 &
\end{tikzcd}
$$

Fig. 8: Resuming a chain of intermediary keys (from ik^2)

by ED and the (new) key k_i needed to encrypt ik (i.e., $n = i - j$).[11] SAKE (hence SAKE-R) uses two main keys: an authentication key K' and a derivation key K. Therefore, ik corresponds to $K \| K'$.

ED		$\mathbf{XS} \in \{\mathbf{CS}, \mathbf{AS}\}$
$k_i \leftarrow \mathsf{H}^n(k_j)$ $//\ ik = K \| K'$ $ticket \leftarrow \mathsf{KW}(k_i, ik)$ store $(id_{ticket}, id_{server})$		
	$\xrightarrow{id_{ticket} \| ticket}$	
		store $(id_{ticket}, ticket, ik)$
	$\xleftarrow{\text{ack}}$	
delete ik		

Fig. 9: Storage of a ticket

During the *storage phase*, ED merely sends the ticket through the ongoing secure channel established with XS (see Section 3.2). When ED initiates the *retrieval phase* with SAKE-R (see Figure 10), the first message sent to XS carries an identifier of the ticket to retrieve. XS responds with the corresponding ticket. The parameter id_{ticket} indicates ED which key k_i (i.e., essentially its index i) must be used to decrypt the corresponding ticket.

The subsequent messages are essentially the same as the original SAKE protocol. They embed pseudo-random values that participate in the mutual authentication, and the session key computation. When the server is the initiator, the ticket is sent in the first message (see Figure 11).

For the sake of clarity we use the following notations:

[11] See Section 3.2.

– kdf corresponds to: $sk \leftarrow \mathsf{KDF}(K, g(r_A, r_B))$[12]
– upd corresponds to
  1. $K \leftarrow \mathsf{update}(K)$
  2. $K' \leftarrow \mathsf{update}(K')$

KDF is the session key derivation function used in SAKE (and SAKE-R). update is the one-way function used to update the root key (i.e., the intermediary key ik in that case). $\mathsf{verif}(k, m, \tau)$ denotes the MAC verification function. It takes as input a secret key k, a message m, and a tag τ. It outputs `true` if τ is a valid tag on message m with respect to k. Otherwise, it returns `false`.

We chose to model H and KDF as PRFs, and KW as an AE function.

Once ED gets ik, the ticket decryption key k_j currently kept by ED is replaced with $k_{i+1} = \mathsf{H}(k_i)$, where k_i is the key used to decrypt the retrieved ticket (see Section 3.2). Therefore, ED rejects any replay of an already consumed ticket.

Observe that ED updates its root key $ik = K \| K'$ upon reception of message m_B. If XS does not receive message τ_A, it does not update its own root key ik. Hence ED and the server are "desynchronised" (i.e., they do not share the same value of ik). When ED initiates anew the key exchange, it executes the SAKE protocol (and not SAKE-R) since it has already retrieved ik. SAKE enables ED and XS to resynchronise in the *continuity* of the protocol run (i.e., SAKE is *self-synchronising*). Therefore, ED and XS can perform a correct key exchange, and eventually compute a shared session key sk.

Security Proof for SAKE-R. We consider the case where ED initiates SAKE-R (see Figure 10).[13] With the following theorem we claim that SAKE-R is a secure 2-AKE protocol with respect to the security model of Brzuska et al. [12].

Theorem 2. *The protocol SAKE-R is a secure 2-AKE protocol, and for any probabilistic polynomial time adversary $\mathcal{A}$ in the 2-AKE security experiment against SAKE-R*

$$\begin{aligned}
\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A}) \leq nq \Big[&(nq-1)2^{-(\lambda-1)} + 2(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0) + 2\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) \\
&+ 3\mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2) + q \cdot \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)\Big] \\
\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R}(\mathcal{A}) \leq \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A}) + nq \Big[&2\left(\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{KDF}}(\mathcal{B}_4)\right) \\
&+ (q-1)\left(\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + 2\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right)\Big]
\end{aligned}$$

where n is the number of parties (ED and XS), q the maximum number of instances (sessions) per party, λ the size of the pseudo-random values (r_A, r_B), $\mathcal{B}_0$ an adversary against the PRF*-security of* H*, $\mathcal{B}_1$ an adversary against the* AE*-security of* KW*, $\mathcal{B}_2$ an adversary against the* $\mathsf{SUF\text{-}CMA}$*-security of* MAC*, $\mathcal{B}_3$ an*

[12] Function g is deliberately left undefined. For instance, $g(r_A, r_B)$ can be equal to the concatenation or the bitwise addition of r_A and r_B.

[13] The converse case, where the XS is the initiator, follows the same reasoning.

ED [A]		$\mathbf{XS} \in \{\mathbf{CS}, \mathbf{AS}\}$ **[B]**
$(-)$		(K, K')
(k_j, id_{ticket})		$(id_{ticket}, ticket)$
$r_A \xleftarrow{\$} \{0,1\}^\lambda$ $m_A \leftarrow A \Vert r_A \Vert id_{ticket}$		
	$\xrightarrow{m_A}$	
		`if` $(id_{ticket}$ not found$)$ `then` abort
		$r_B \xleftarrow{\$} \{0,1\}^\lambda$ $\tau_B \leftarrow \mathsf{MAC}(K', B \Vert A \Vert r_B \Vert r_A \Vert id_{ticket} \Vert ticket)$ $m_B \leftarrow r_B \Vert ticket \Vert \tau_B$
	$\xleftarrow{m_B}$	
$k_i \leftarrow \mathsf{H}^n(k_j)$ $ik \leftarrow \mathsf{KW}^{-1}(k_i, ticket)$ `if` $(ik = \bot)$ `then` abort		
$//\ ik = K \Vert K'$ `if` $(\mathsf{verif}(K', B \Vert A \Vert r_B \Vert r_A \Vert id_{ticket} \Vert ticket, \tau_B) = \texttt{false})$ `then` abort		
$k_{i+1} \leftarrow \mathsf{H}(k_i)$ delete k_j, k_i		
$\tau_A \leftarrow \mathsf{MAC}(K', A \Vert B \Vert r_A \Vert r_B)$		
$sk \leftarrow \mathsf{kdf}$ $K, K' \leftarrow \mathsf{upd}$		
	$\xrightarrow{\tau_A}$	
		`if` $(\mathsf{verif}(K', A \Vert B \Vert r_A \Vert r_B, \tau_A) = \texttt{false})$ `then` abort
		$sk \leftarrow \mathsf{kdf}$ $K, K' \leftarrow \mathsf{upd}$
		$\tau'_B \leftarrow \mathsf{MAC}(K', r_B \Vert r_A)$ delete $ticket$
	$\xleftarrow{\tau'_B}$	
`if` $(\mathsf{verif}(K', r_B \Vert r_A, \tau'_B) = \texttt{false})$ `then` abort		

Fig. 10: Session Resumption with SAKE-R initiated by ED

ED [**B**]		**XS** $\in$ {**CS**, **AS**} [**A**]
$(-)$		(K, K')
(k_j, id_{ticket})		$(id_{ticket}, ticket)$
		$r_A \xleftarrow{\$} \{0,1\}^\lambda$ $m_A \leftarrow A \Vert r_A \Vert id_{ticket} \Vert ticket$
	$\xleftarrow{m_A}$	

if (id_{ticket} not found) **then**
 abort
 	
 $k_i \leftarrow \mathsf{H}^n(k_j)$
 $ik \leftarrow \mathsf{KW}^{-1}(k_i, ticket)$
 if ($ik = \perp$) **then**
 abort
 	
 $k_{i+1} \leftarrow \mathsf{H}(k_i)$
 delete k_j, k_i
 	
 $//\ ik = K \Vert K'$
 $r_B \xleftarrow{\$} \{0,1\}^\lambda$
 $\tau_B \leftarrow \mathsf{MAC}(K', B \Vert A \Vert r_B \Vert r_A)$
 $m_B \leftarrow r_B \Vert \tau_B$
 	
 $sk \leftarrow \mathsf{kdf}$
 $K, K' \leftarrow \mathsf{upd}$		

	$\xrightarrow{m_B}$	
		if ($\mathsf{verif}(K', B \Vert A \Vert r_B \Vert r_A, \tau_B) =$ **false**) **then** abort $sk \leftarrow \mathsf{kdf}$ $K, K' \leftarrow \mathsf{upd}$ $\tau_A \leftarrow \mathsf{MAC}(K', A \Vert B \Vert r_A \Vert r_B)$ delete $ticket$
	$\xleftarrow{\tau_A}$	
if ($\mathsf{verif}(K', A \Vert B \Vert r_A \Vert r_B, \tau_A) =$ **false**) **then** abort		

Fig. 11: Session Resumption with SAKE-R initiated by XS $\in$ {CS, AS}

adversary against the PRF*-security of* update*, and* $\mathcal{B}_4$ *an adversary against the* PRF*-security of* KDF*.*

Below we give a sketch proof of Theorem 2. The full proof is given in Appendix B.

We consider the 2-AKE security model of Brzuska et al. [12], and define the entity authentication and the key indistinguishability security experiments accordingly.

Entity authentication. We start with the 2-AKE entity authentication experiment. Let $\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A})$ be the probability that the adversary $\mathcal{A}$) wins the entity authentication game. Let $\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A})$ bounds the probability that the adversary succeeds against a client (ED), and $\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$ bounds the probability that the adversary succeeds against a server ($\mathrm{XS} \in \{\mathrm{CS}, \mathrm{AS}\}$). We have that $\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A}) \leq \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) + \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$. We use the following hops. We first consider an adversary that targets a client.

Game 0. This game corresponds to the 2-AKE entity authentication security experiment when the adversary targets a client instance.

Game 1. The challenger aborts if there exists an instance that chooses a random value r_A or r_B that is not unique. There is at most $n \times q$ random values, each uniformly drawn at random in $\{0,1\}^\lambda$. Hence the two games are equivalent up to a collision term $\frac{nq(nq-1)}{2^\lambda}$.

Game 2. The adversary interacts with a single client instance. This is equivalent to guessing the targeted instance, hence a loss $\frac{1}{nq}$.

Game 3. The first key k_0 used to compute a cookie is uniformly drawn at random. Each subsequent key is computed with the function H. If k_0 is random, one can rely on the PRF-security of $\mathsf{H} = \mathsf{PRF}_\mathsf{H}(\cdot,\cdot)$. In turn, since $\mathsf{PRF}_\mathsf{H}(k_0,\cdot)$ can be replaced with a truly random function, its output k_1 is random. Therefore, one can rely on the pseudo-randomness of the function H keyed with this new value k_1, and so forth. Each transition (i.e., each computation of $k_{t+1} = \mathsf{H}(k_t)$, $t \geq 0$) implies a loss equal to $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$. Hence an overall loss at most $(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$.

Game 4. In this game, one relies on the AE-security of function KW to guarantee that $ik = K \| K'$ (hence K') is indistinguishable from random. This is possible because k_i is indistinguishable from random due to Game 3. Hence a loss $\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1)$.

Game 5. The challenger aborts if the targeted instance π ever receives a valid message m_B but no instance partnered with π has output that message. We reduce this event to the SUF-CMA-security of the MAC function used to compute m_B. Hence a loss $\mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)$.

Game 6. The challenger aborts if the targeted instance π ever receives a valid message τ'_B but no instance partnered with π has output that message. We reduce this event to the SUF-CMA-security of the MAC function used to compute τ'_B. In turn, we must rely on the PRF-security of the function update used to output the current MAC key. Hence a loss $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)$. To that point, the adversary has no chance to win.

Now we consider an adversary that targets a server.

Game 0. This game corresponds to the 2-AKE entity authentication security experiment when the adversary targets a server instance.

Game 1. The challenger aborts if there exists an instance that chooses a random value r_A or r_B that is not unique. There is at most $n \times q$ random values, each uniformly drawn at random in $\{0,1\}^\lambda$. Hence the two games are equivalent up to a collision term $\frac{nq(nq-1)}{2^\lambda}$.

Game 2. The adversary interacts with a single server instance. This is equivalent to guessing the targeted instance, hence a loss $\frac{1}{nq}$.

Game 3. The first key k_0 used to compute a cookie is uniformly drawn at random. Each subsequent key is computed with the function H. Since k_0 is random, one relies on the PRF-security of $\mathsf{H} = \mathsf{PRF}_\mathsf{H}(\cdot,\cdot)$. In turn, since $\mathsf{PRF}_\mathsf{H}(k_0,\cdot)$ can be replaced with a truly random function, its output k_1 is random. Therefore, one relies on the pseudo-randomness of the function H keyed with this new value k_1, and so forth. Each transition (i.e., each computation of $k_{t+1} = \mathsf{H}(k_t)$, $t \geq 0$) implies a loss $\mathsf{adv}_\mathsf{H}^\mathsf{PRF}(\mathcal{B}_0)$. Hence an overall loss at most $(q-1)\mathsf{adv}_\mathsf{H}^\mathsf{PRF}(\mathcal{B}_0)$.

Game 4. In this game, one relies on the AE-security of function KW to guarantee that $ik = K \| K'$ (hence K') is indistinguishable from random. This is possible because k_i is indistinguishable from random due to Game 3. Hence a loss $\mathsf{adv}_\mathsf{KW}^\mathsf{AE}(\mathcal{B}_1)$.

Game 5. The first value of K' is uniformly chosen at random. During each run of SAKE-R, K' is updated with the function update. Since the first K' is random, one relies on the PRF-security of $\mathsf{update} = \mathsf{PRF}_\mathsf{update}(\cdot,\cdot)$. In turn, since $\mathsf{PRF}_\mathsf{update}(K',\cdot)$ can be replaced with a truly random function, its output (updated K') is random. Therefore, one relies upon the pseudo-randomness of the function update keyed with this new value K', and so forth. Each transition (i.e., each update of K') implies a loss $\mathsf{adv}_\mathsf{update}^\mathsf{PRF}(\mathcal{B}_3)$. Hence an overall loss at most $(q-1)\mathsf{adv}_\mathsf{update}^\mathsf{PRF}(\mathcal{B}_3)$.

Game 6. The challenger aborts if the targeted instance π ever receives a valid message τ_A but no instance partnered with π has output that message. Such a forgery can be achieved in one of two ways: either the adversary guesses the MAC key used to compute τ_A, or it gets the key carried in *cookie*. The first possibility is reduced to the $\mathsf{SUF\text{-}CMA}$-security of the MAC function. The second possibility is reduced to the AE-security of function KW, which is already assumed due to Game 4. Hence a loss $\mathsf{adv}_\mathsf{MAC}^\mathsf{SUF\text{-}CMA}(\mathcal{B}_2)$. To that point, the adversary has no chance to win.

Collecting all the probabilities yields the indicated bound.

Key indistinguishability. Now we consider the 2-AKE key indistinguishability security experiment.

Game 0. This game corresponds to the 2-AKE key indistinguishability security experiment.

Game 1. The challenger aborts and chooses $b \in \{0,1\}$ uniformly at random if there exists an instance that accepts maliciously. Hence a loss $\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A})$.

Game 2. The adversary interacts with a single instance. This is equivalent to guessing the targeted instance, hence a loss $\frac{1}{nq}$.

Game 3. We distinguish two cases: the adversary targets either a client instance or a server instance, corresponding respectively to an advantage $\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A})$ and $\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$. Let adv_3 be the advantage of the adversary of winning in Game 3. We have that $\mathsf{adv}_3 \leq \mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) + \mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$. We begin with the first case.

Game$^{\mathsf{client}}$ *3.* We rely on the AE-security of KW. In turn, we recursively rely on the PRF-security of the function H used to output the cookie decryption key k_i. Hence a loss at most $\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$.

Game$^{\mathsf{client}}$ *4.* Due to Game$^{\mathsf{client}}$ 3, $ik = K \| K'$ (hence K) is indistinguishable from random. Hence we rely on the PRF-security of the function KDF used to compute the session key sk. Hence a loss $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{KDF}}(\mathcal{B}_4)$. To that point the adversary can do no better than guessing.

Now we consider the case where the adversary targets a server instance.

Game$^{\mathsf{server}}$ *3.* The key K used by the server instance to compute the session key is updated throughout the successive runs. We rely on the PRF-security of the function update. Hence a loss at most $(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$.

Game$^{\mathsf{server}}$ *4.* We rely on the AE-security of KW. In turn, we recursively rely on the PRF-security of the function H used to output the cookie decryption key k_i. Hence a loss at most $\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$.

Game$^{\mathsf{server}}$ *5.* Due to Game$^{\mathsf{server}}$ 3 and Game$^{\mathsf{server}}$ 4, K is indistinguishable from random. Hence we rely on the PRF-security of function KDF used to compute the session key sk. Hence a loss $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{KDF}}(\mathcal{B}_4)$. To that point the adversary can do no better than guessing.

Collecting all the probabilities yields the indicated bound.

5.4 Achieving 3-AKE Security

P = SAKE is proved to be a secure 2-AKE protocol [3] in the Brzuska et al. security model (which captures forward secrecy) [12]. P = SAKE-R is a 2-AKE-secure protocol from Theorem 2. With respect to the 3-AKE security model, we define the long-term key component rootk of any ED to be $\mathsf{rootk} = (K, K')$ if P = SAKE, and $\mathsf{rootk} = (K, K', k_j)$ if P = SAKE-R. That is, in the latter case, we allow the 3-AKE adversary to get the ticket encryption key k_j through a $\mathsf{Corrupt}$-query, in addition to the derivation master key K and the authentication master key K'.

P' = TLS 1.3 is proved to be a secure 2-AKE protocol [15]. Although this result applies to an earlier draft of the protocol, we may reasonably assume that the final version also guarantees 2-AKE-security.

Enc defined as the record layer of TLS 1.3 is proved to be AE-secure [5] in the sense of Rogaway [23] (indistinguishability from random bits) which implies AE-security in the sense of Shrimpton [29] (real-from-random indistinguishability). In addition, in TLS 1.3, a per-record nonce derived from a sequence number aims at guaranteeing non-replayability of the records (the sequence number being maintained independently at both sides). Hence we assume the sAE-security of Enc.

Hence, from Theorem 1, our instantiation (with and without the session resumption scheme for low-resource ED) is a 3-AKE-secure protocol according to Definition 8.

6 Conclusion

We have described a generic 3-party authenticated key exchange (3-AKE) protocol dedicated to IoT. Solely based on symmetric-key functions (regarding the computations done between the end-device and the back-end network), our 3-AKE protocol guarantees forward secrecy, in contrast to widely deployed symmetric-key based IoT protocols. It also enables session resumption without impairing security (in particular, forward secrecy is maintained). This allows saving communication and computation cost, and is advantageous for low-resource end-devices.

In addition, we have described a concrete instantiation of our 3-AKE protocol, and presented a security model used to formally prove the security of our 3-AKE protocol and its concrete instantiation.

Our 3-AKE protocol can be applied in a real-case IoT deployment (i.e., involving numerous end-devices and servers) such that the latter inherits from the security properties of the protocol. This results in the ability for the (mobile) end-device to securely switch from one server to another back and forth at a reduced (communication and computation) cost, without compromising the sessions established with other servers.

Acknowledgment. The authors would like to thank an anonymous reviewer from ESORICS 2019 for pointing out a mistake in the security model (Test-query).

References

1. Accenture: Winning with the Industrial Internet of Things – How to accelerate the journey to productivity and growth (2015)
2. Aviram, N., Gellert, K., Jager, T.: Session Resumption Protocols and Efficient Forward Security for TLS 1.3 0-RTT. Cryptology ePrint Archive, Report 2019/228 (2019), to appear at Eurocrypt 2019.
3. Avoine, G., Canard, S., Ferreira, L.: Symmetric-key Authenticated Key Exchange (SAKE) with Perfect Forward Secrecy. Cryptology ePrint Archive, Report 2019/444 (2019)

4. Avoine, G., Ferreira, L.: Rescuing LoRaWAN 1.0. In: Financial Cryptography and Data Security (FC 2018) (2018), https://fc18.ifca.ai/preproceedings/13.pdf
5. 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: 9th International Conference on Provable Security. Lecture Notes in Computer Science, vol. 9451, pp. 85–104. Springer, Heidelberg, Germany, Kanazawa, Japan (Nov 24–26, 2015). https://doi.org/10.1007/978-3-319-26059-4_5
6. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th Annual Symposium on Foundations of Computer Science. pp. 394–403. IEEE Computer Society Press, Miami Beach, Florida (Oct 19–22, 1997). https://doi.org/10.1109/SFCS.1997.646128
7. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. Journal of Cryptology **21**(4), 469–491 (Oct 2008). https://doi.org/10.1007/s00145-008-9026-x
8. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) Advances in Cryptology – CRYPTO'93. Lecture Notes in Computer Science, vol. 773, pp. 232–249. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 22–26, 1994). https://doi.org/10.1007/3-540-48329-2_21
9. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004), http://eprint.iacr.org/2004/331
10. Bhargavan, K., Boureanu, I., Fouque, P.A., Onete, C., Richard, B.: Content delivery over TLS: a cryptographic analysis of keyless SSL. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 1–16. IEEE (April 2017). https://doi.org/10.1109/EuroSP.2017.52
11. Brzuska, C., Fischlin, M., Smart, N.P., Warinschi, B., Williams, S.C.: Less is more: relaxed yet composable security notions for key exchange. International Journal of Information Security **12**(4), 267–297 (August 2013), https://doi.org/10.1007/s10207-013-0192-y
12. Brzuska, C., Jacobsen, H., Stebila, D.: Safely exporting keys from secure channels: On the security of EAP-TLS and TLS key exporters. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 670–698. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49890-3_26
13. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol – Version 1.2. https://tools.ietf.org/html/rfc5246 (August 2008)
14. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976)
15. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. Cryptology ePrint Archive, Report 2016/081 (2016), http://eprint.iacr.org/2016/081
16. GSMA: 3GPP Low Power Wide Area Technologies – GSMA White Paper (October 2016)
17. i-scoop: The Industrial Internet of Things (IIoT): the business guide to Industrial IoT. https://www.i-scoop.eu/internet-of-things-guide/industrial-internet-things-iiot-saving-costs-innovation/ (2018)
18. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. Cryptology ePrint Archive, Report 2011/219 (2011), http://eprint.iacr.org/2011/219
19. Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., Kiviner, T.: Internet Key Exchange Protocol Version 2 (IKEv2). https://tools.ietf.org/html/rfc7296 (October 2014)

20. Krawczyk, H.: A unilateral-to-mutual authentication compiler for key exchange (with applications to client authentication in TLS 1.3). In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16: 23rd Conference on Computer and Communications Security. pp. 1438–1450. ACM Press, Vienna, Austria (Oct 24–28, 2016). https://doi.org/10.1145/2976749.2978325
21. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. https://tools.ietf.org/html/rfc8446 (August 2018)
22. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signature and public-key cryptosystems. Communications of the Association for Computing Machinery **21**(2), 120–126 (1978)
23. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 02: 9th Conference on Computer and Communications Security. pp. 98–107. ACM Press, Washington D.C., USA (Nov 18–22, 2002). https://doi.org/10.1145/586110.586125
24. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 373–390. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006). https://doi.org/10.1007/11761679_23
25. Salowey, J., Zhou, H., Eronen, P., Tschofenig, H.: Transport Layer Security (TLS) Session Resumption without Server-Side State. https://tools.ietf.org/html/rfc5077 (January 2008)
26. Seys, S., Preneel, B.: Power Consumption Evaluation of Efficient Digital Signature Schemes for Low Power Devices. In: IEEE International Conference on Wireless And Mobile Computing, Networking And Communications. WiMob 2005, vol. 1, pp. 79–86. IEEE (August 2005). https://doi.org/10.1109/WIMOB.2005.1512820
27. Sheffer, Y., Tschofenig, H.: Internet Key Exchange Protocol Version 2 (IKEv2) – Session Resumption. https://tools.ietf.org/html/rfc5723 (January 2010)
28. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), http://eprint.iacr.org/2004/332
29. Shrimpton, T.: A characterization of authenticated-encryption as a form of chosen-ciphertext security. Cryptology ePrint Archive, Report 2004/272 (2004), http://eprint.iacr.org/2004/272
30. Sigfox: Secure SigFox Ready devices – Recommendation guide (2017)
31. Sigfox: SigFox Technical Overview (May 2017)
32. Sornin, N.: LoRaWAN 1.1 Specification (October 2017), LoRa Alliance, version 1.1, 11/10/2017
33. Sornin, N., Luis, M., Eirich, T., Kramp, T.: LoRaWAN Specification (July 2016), LoRa Alliance, version 1.0
34. Yegin, A.: LoRaWAN Backend Interfaces 1.0 Specification (October 2017), LoRa Alliance, version 1.0, 11/10/2017

A Security Proof for the Generic 3-AKE Protocol

In this section, we give a proof of Theorem 1. We proceed through a sequence of games [9, 28] between a challenger and an adversary $\mathcal{A}$.

A.1 Entity Authentication for the Generic 3-AKE Protocol

First we consider the entity authentication experiment described in Section 4.3. Let E_i be the event that the adversary succeeds in making an instance accept maliciously in Game i. We use the following hops.

Game 0. This game corresponds to the entity authentication security experiment described in Section 4.3. Therefore we have that

$$\Pr[E_0] = \mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A})$$

Game 1. In this game, the challenger aborts the experiment if it does not guess the party the adversary targets, and its party partners. There are respectively n_K, n_E, n_X parties in the sets $\mathcal{K}$, $\mathcal{E}$, and $\mathcal{X}$. Therefore we have that

$$\Pr[E_1] = \Pr[E_0] \times \frac{1}{n_K \cdot n_E \cdot n_X}$$

Game 2. Now the parties $P_k \in \mathcal{K}$, $P_j \in \mathcal{X}$ and $P_i \in \mathcal{E}$ are fixed. In this game, the challenger aborts the experiment if the adversary succeeds in impersonating P_j to P_k or conversely. We reduce this event to the 2-AKE-security of the protocol P' applied between P_k and P_j. Therefore we have that

$$\Pr[E_1] \leq \Pr[E_2] + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0)$$

where $\mathcal{B}_0$ is an adversary against the 2-AKE-security of P'.

This guarantees that to each instance $\pi_k^s \in P_k.\mathsf{Instances}$ there exists a unique instance $\pi_j^v \in P_j.\mathsf{Instances}$ such that $\pi_k^s.\mathsf{sid} = \pi_j^v.\mathsf{sid}$ (and conversely).

Game 3. In this game, the challenger aborts the experiment if the adversary succeeds in impersonating P_i to P_k or conversely. We reduce this event to the 2-AKE-security of the protocol P applied between P_k and P_i. Therefore we have that

$$\Pr[E_2] \leq \Pr[E_3] + \mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1)$$

where $\mathcal{B}_1$ is an adversary against the 2-AKE-security of P.

This guarantees that to each instance $\pi_i^m \in P_i.\mathsf{Instances}$ there exists a unique instance $\pi_k^\ell \in P_k.\mathsf{Instances}$ such that $\pi_i^m.\mathsf{sid} = \pi_k^\ell.\mathsf{sid}$ (and conversely).

Game 4. Another way for the adversary to win the entity authentication security experiment is to get the intermediary key ik used by P_i and P_j to mutually authenticate, or to forge a valid message intended to P_j that carries an intermediary key chosen by the adversary. In this game, the challenger aborts the experiment if the adversary succeeds in either case. The secure channel between P_k and P_j is established with the function Enc keyed with the session key output by P'. We reduce each of the two events to the sAE-security of Enc. In turn, we must assume that the Enc key is random. The latter is reduced to the 2-AKE-security of P'. Therefore we have that

$$\Pr[E_3] \leq \Pr[E_4] + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) + 2\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2)$$

where $\mathcal{B}_2$ is an adversary against the sAE-security of Enc.

Game 5. In this game, the challenger aborts the experiment if the adversary succeeds in impersonating P_i to P_j or conversely. We reduce this event to the 2-AKE-security of P. Therefore we have that

$$\Pr[E_4] \leq \Pr[E_5] + \mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1)$$

Due to Game 4 and Game 5, we have that, to each instance $\pi_i^n \in P_i.\mathsf{Instances}$, there exists a unique instance $\pi_j^u \in P_j.\mathsf{Instances}$ such that $\pi_i^n.\mathsf{sid} = \pi_j^u.\mathsf{sid}$ (and conversely). Due to Game 4, the intermediary key $\mathsf{ck} = ik$ shared by π_i^m and π_k^ℓ is also known to π_j^v (i.e., $ik = \pi_i^m.\mathsf{ck} = \pi_j^v.\mathsf{km}$). Due to Game 5, $\pi_i^n.\mathsf{sid} = \pi_j^u.\mathsf{sid}$. Hence $\pi_i^n.\mathsf{trscrpt} = \pi_j^u.\mathsf{trscrpt}$. We define function f to be the key derivation function that outputs the session key $\pi_i^n.\mathsf{ck} = \pi_j^u.\mathsf{ck} = sk$ from the root key ik and the messages exchanged between π_i^n and π_j^u. Therefore, we have that $f(\pi_i^m.\mathsf{ck}, \pi_i^n.\mathsf{trscrpt}) = \pi_i^n.\mathsf{ck} = \pi_j^u.\mathsf{ck} = f(\pi_j^v.\mathsf{km}, \pi_j^u.\mathsf{trscrpt})$.

To that point, the adversary has no chance to win. Therefore

$$\Pr[E_5] = 0$$

Collecting all the probabilities from Game 0 to Game 5, we have that

$$\begin{aligned}
\mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) &= \Pr[E_0] \\
&= n_K \cdot n_E \cdot n_X \cdot \Pr[E_1] \\
&\leq n_K \cdot n_E \cdot n_X \left[\Pr[E_2] + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0)\right] \\
&\leq n_K \cdot n_E \cdot n_X \left[\Pr[E_3] + \mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1) + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0)\right] \\
&\leq n_K \cdot n_E \cdot n_X \Big[\Pr[E_4] + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) + 2\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2) \\
&\qquad + \mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1) + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0)\Big] \\
&\leq n_K \cdot n_E \cdot n_X \Big[\Pr[E_5] + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) + 2\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2) \\
&\qquad + 2\mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1) + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0)\Big] \\
&= n_K \cdot n_E \cdot n_X \Big[\mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) + \mathsf{adv}_{P'}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_0) \\
&\qquad + 2\mathsf{adv}_{P}^{\mathsf{ent\text{-}auth}}(\mathcal{B}_1) + 2\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2)\Big]
\end{aligned}$$

A.2 Key Indistinguishability for the Generic 3-AKE Protocol

Now we consider the key indistinguishability security experiment described in Section 4.3. Let E_i be the event that the adversary wins in Game i, and $\mathsf{adv}_i = \Pr[E_i] - \frac{1}{2}$. We use the following hops.

Game 0. This game corresponds to the key indistinguishability security experiment described in Section 4.3. Therefore we have that

$$\Pr[E_0] = \mathsf{adv}_0 + \frac{1}{2} = \mathsf{adv}_{\Pi}^{\mathsf{key\text{-}ind}}(\mathcal{A}) + \frac{1}{2}$$

Game 1. In this game, the challenger aborts the experiment and chooses $b' \in \{0,1\}$ uniformly at random if there exists an instance that maliciously accepts. In other words, we make the same modifications as in the games performed during the entity authentication proof. Therefore

$$\mathsf{adv}_0 \leq \mathsf{adv}_1 + \mathsf{adv}_{\Pi}^{\mathsf{ent\text{-}auth}}(\mathcal{A})$$

Game 2. In this game, the adversary aborts the experiment if it does not guess the party the adversary targets, and its party partners. There are respectively n_K, n_E, n_X parties in the sets $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{X}$. Therefore

$$\mathsf{adv}_2 = \mathsf{adv}_1 \times \frac{1}{n_K \cdot n_E \cdot n_X}$$

Game 3. Now the parties $P_k \in \mathcal{K}$, $P_j \in \mathcal{X}$ and $P_i \in \mathcal{E}$ are fixed. Moreover, the conditions of Definition 6 are satisfied. This means in particular that each instance ending in accepting state is pairwise partnered with a unique instance.

In this game, we replace the session key ik output by the 2-AKE run of protocol P between P_i and P_k with a truly random value $\tilde{ik}$. The challenger aborts the game if there is an algorithm able to distinguish between both values. We reduce this event to the 2-AKE-security of P. Therefore, we have that

$$\mathsf{adv}_2 \leq \mathsf{adv}_3 + \mathsf{adv}_{P}^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)$$

where $\mathcal{B}_1$ is an adversary against the 2-AKE-security of P.

Game 4. In this game, we replace the session key output by the 2-AKE run of protocol P' between P_k and P_j with a truly random value. The challenger aborts the game if there is an algorithm able to distinguish between both values. We reduce this event to the 2-AKE-security of P'. Therefore, we have that

$$\mathsf{adv}_3 \leq \mathsf{adv}_4 + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0)$$

where $\mathcal{B}_0$ is an adversary against the 2-AKE-security of P'.

Game 5. The key (allegedly ik) sent by P_k to P_j is protected with Enc, which is keyed with the session key output by P'. The adversary can try to get the key $\tilde{ik}$, and then compute the session key sk shared between P_i and P_j. We reduce this event to the sAE-security of Enc. In turn, this relies implicitly on the fact that the encryption key (output by P') used to key Enc be indistinguishable from random. This is the case due to Game 4. Therefore, in this game, the challenger aborts the experiment if the adversary succeeds in getting ik. We have that

$$\mathsf{adv}_4 \leq \mathsf{adv}_5 + \mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2).$$

Game 6. In this game, the challenger aborts the experiment if the adversary succeeds in breaking the 2-AKE-security of P executed between P_i and P_j. Therefore we have that

$$\mathsf{adv}_5 \leq \mathsf{adv}_6 + \mathsf{adv}_P^{\mathsf{key\text{-}ind}}(\mathcal{B}_1).$$

To that point, the adversary can do no better than guessing. Therefore

$$\mathsf{adv}_6 = 0.$$

Collecting all the probabilities from Game 0 to Game 6, we have that

$$\begin{aligned}
\mathsf{adv}_\Pi^{\mathsf{key\text{-}ind}}(\mathcal{A}) &= \mathsf{adv}_0 \\
&\leq \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + \mathsf{adv}_1 \\
&= \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \cdot \mathsf{adv}_2 \\
&\leq \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \left[\mathsf{adv}_3 + \mathsf{adv}_P^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)\right] \\
&\leq \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \Big[\mathsf{adv}_4 + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) \\
&\qquad\qquad + \mathsf{adv}_P^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)\Big] \\
&\leq \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \Big[\mathsf{adv}_5 + \mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2) + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) \\
&\qquad\qquad + \mathsf{adv}_P^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)\Big] \\
&\leq \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \Big[\mathsf{adv}_6 + \mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2) + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) \\
&\qquad\qquad + 2\mathsf{adv}_P^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)\Big] \\
&= \mathsf{adv}_\Pi^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + n_K \cdot n_E \cdot n_X \Big[\mathsf{adv}_{\mathsf{Enc}}^{\mathsf{sAE}}(\mathcal{B}_2) + \mathsf{adv}_{P'}^{\mathsf{key\text{-}ind}}(\mathcal{B}_0) \\
&\qquad\qquad + 2\mathsf{adv}_P^{\mathsf{key\text{-}ind}}(\mathcal{B}_1)\Big]
\end{aligned}$$

B Security Proof for SAKE-R

In the section we give a proof of Theorem 2. That is, we consider only the case where ED initiates the protocol run (Figure 10). The converse case where $\mathrm{XS} \in \{\mathrm{CS}, \mathrm{AS}\}$ initiates the run (Figure 11) is similar.

We consider the 2-AKE security model of Brzuska et al. [12], and define the entity authentication and the key indistinguishability security experiments accordingly.

We define functions H and update to be two PRFs. That is $\mathsf{H} : y \mapsto \mathsf{PRF}_{\mathsf{H}}(y, x)$ and $\mathsf{update} : y \mapsto \mathsf{PRF}_{\mathsf{update}}(y, x')$ for some (constant) values x and x'.

B.1 Entity Authentication for SAKE-R

We start with the 2-AKE entity authentication experiment. Let $\mathsf{adv}_{SAKE\text{-}R}^{\mathsf{ent\text{-}auth}}(\mathcal{A})$ be the probability that the adversary wins the entity authentication game. Let

$\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A})$ be the probability that the adversary succeeds against a client (ED), and $\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$ the probability that the adversary succeeds against a server (XS). We have that

$$\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A}) \leq \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) + \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$$

Client Adversary. We first consider an adversary that targets a client. Let E_i be the event that the adversary succeeds in making a client instance accept maliciously in Game$^{\mathsf{client}}$ i.

Game$^{\mathsf{client}}$ *0.* This game corresponds to the 2-AKE entity authentication security experiment when the adversary targets a client instance. Therefore

$$\Pr[E_0] = \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A})$$

Game$^{\mathsf{client}}$ *1.* In this game, the challenger aborts the experiment if there exists an instance that chooses a random value r_A or r_B that is not unique. There is at most $n \times q$ random values, each uniformly drawn at random in $\{0,1\}^\lambda$. Hence the two games are equivalent up to a collision term $\frac{nq(nq-1)}{2^\lambda}$. Therefore

$$\Pr[E_0] \leq \Pr[E_1] + \frac{nq(nq-1)}{2^\lambda}$$

Game$^{\mathsf{client}}$ *2.* In this game, the challenger aborts the experiment if it does not guess which client instance will be the first to maliciously accept. There is n parties and q instances per party. Therefore we have that

$$\Pr[E_2] = \Pr[E_1] \times \frac{1}{nq}$$

Game$^{\mathsf{client}}$ *3.* The first key k_0 used to compute a ticket is uniformly drawn at random. The next encryption key k_1 is computed as $k_1 = \mathsf{H}(k_0) = \mathsf{PRF}_{\mathsf{H}}(k_0, x)$. Since k_0 is random, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_0, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_0}$. We do the same for any server instance that uses function H with the same key k_0 to compute k_1. Distinguishing the change implies an algorithm able to distinguish the H function from a random function. This corresponds to an advantage $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$ where $\mathcal{B}_0$ is an adversary against the PRF-security of H. Since $\mathsf{PRF}_{\mathsf{H}}(k_0, \cdot)$ is replaced with a random function $\mathsf{F}^{\mathsf{H}}_{k_0}$, $k_1 = \mathsf{F}^{\mathsf{H}}_{k_0}(x)$ is random. In turn, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_1, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_1}$. Recursively, we replace each function $\mathsf{PRF}_{\mathsf{H}}(k_i, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_i}$. There is at most q instances per party, hence at most $q-1$ updates of the original key k_0 before computing a ticket (that is, $0 \leq i < q$). Therefore, distinguishing the successive changes corresponds to an advantage at most $(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$. Consequently, in this game, the challenger aborts the experiment if the adversary is able to distinguish any of these changes. Therefore, we have that

$$\Pr[E_2] \leq \Pr[E_3] + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$$

Game$^{\mathsf{client}}$ *4.* In this game, the challenger aborts the experiment if the adversary is able to get the key ik from $ticket = \mathsf{KW}(k_i, ik)$, $0 \leq i < q$. We reduce this event to the AE-security of function KW (this is possible because k_i is indistinguishable from random due to Game$^{\mathsf{client}}$ 3). Therefore we have that

$$\Pr[E_3] \leq \Pr[E_4] + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1)$$

where $\mathcal{B}_1$ is an adversary against the AE-security of KW.

Game$^{\mathsf{client}}$ *5.* In this game, the challenger aborts the experiment if the targeted instance π ever receives a valid message m_B but no instance partnered with π has output that message. Due to Game$^{\mathsf{client}}$ 4, $ik = K \| K'$ (hence K') can be safely replaced with a truly random value. Therefore, we reduce this event to the $\mathsf{SUF\text{-}CMA}$-security of the MAC function (keyed with K') used to compute m_B. Therefore, we have that

$$\Pr[E_4] \leq \Pr[E_5] + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)$$

where $\mathcal{B}_2$ is an adversary against the $\mathsf{SUF\text{-}CMA}$-security of MAC.

Game$^{\mathsf{client}}$ *6.* The key used to compute the MAC tag τ'_B is $\mathsf{update}(K') = \mathsf{PRF}_{\mathsf{update}}(K', x')$. In this game, we replace $\mathsf{PRF}_{\mathsf{update}}(K', \cdot)$ with a random function $\mathsf{F}^{\mathsf{update}}_{K'}$. We do the same for any server instance that uses the update function with the same key K'. Distinguishing the change implies an algorithm able to distinguish the function update from a random function. Therefore, in this game, the challenger aborts the experiment if the adversary is able to distinguish such a change. Hence

$$\Pr[E_5] \leq \Pr[E_6] + \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$$

where $\mathcal{B}_3$ is an adversary against the PRF-security of update.

Game$^{\mathsf{client}}$ *7.* In this game, the challenger aborts the experiment if the targeted instance π ever receives a valid message τ'_B but no instance partnered with π has output that message. Due to Game$^{\mathsf{client}}$ 6, the key used to compute the MAC tag τ'_B is truly random. Hence, we reduce this event to the $\mathsf{SUF\text{-}CMA}$-security of the MAC function used to compute τ'_B. Therefore, we have that

$$\Pr[E_6] \leq \Pr[E_7] + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)$$

To that point, the adversary has no chance to win. Therefore

$$\Pr[E_7] = 0$$

Collecting all the probabilities from Game$^{\mathsf{client}}$ 0 to Game$^{\mathsf{client}}$ 7, we have that

$$\begin{aligned}
\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) &= \Pr[E_0] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + \Pr[E_1] \\
&= \frac{nq(nq-1)}{2^\lambda} + nq \times \Pr[E_2] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_3] + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_4] + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_5] + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2) + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1)\right. \\
&\qquad\qquad \left. + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_6] + \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)\right. \\
&\qquad\qquad \left. + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_7] + \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + 2\mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)\right. \\
&\qquad\qquad \left. + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&= \frac{nq(nq-1)}{2^\lambda} + nq\left[\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + 2\mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)\right. \\
&\qquad\qquad \left. + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&= nq\left[(nq-1)2^{-\lambda} + \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + 2\mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)\right. \\
&\qquad \left. + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right]
\end{aligned}$$

Server Adversary. Now we consider an adversary that targets a server. Let E_i be the event that the adversary succeeds in making a server instance accept maliciously in Game$^{\mathsf{server}}$ i.

Game$^{\mathsf{server}}$ *0.* This game corresponds to the 2-AKE entity authentication security experiment when the adversary targets a server instance. Therefore we have that

$$\Pr[E_0] = \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$$

Game$^{\mathsf{server}}$ *1.* In this game, the challenger aborts the experiment if there exists an instance that chooses a random value r_A or r_B that is not unique. There is at most $n \times q$ random values, each uniformly drawn at random in $\{0,1\}^\lambda$. Hence the two games are equivalent up to a collision term $\frac{nq(nq-1)}{2^\lambda}$. Therefore

$$\Pr[E_0] \leq \Pr[E_1] + \frac{nq(nq-1)}{2^\lambda}$$

Game$^{\mathsf{server}}$ *2.* In this game, the challenger aborts the experiment if it does not guess which server instance will be the first to maliciously accept. There is n parties and q instances per party. Therefore we have that

$$\Pr[E_2] = \Pr[E_1] \times \frac{1}{nq}$$

Game$^{\mathsf{server}}$ *3.* The first key k_0 used to compute a ticket is uniformly drawn at random. The next encryption key k_1 is computed as $k_1 = \mathsf{H}(k_0) = \mathsf{PRF}_{\mathsf{H}}(k_0, x)$. Since k_0 is random, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_0, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_0}$. We do the same for any client instance that uses the H function with the same key k_0 to compute k_1. Distinguishing the change implies an algorithm able to distinguish the H function from a random function. This corresponds to an advantage $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$ where $\mathcal{B}_0$ is an adversary against the PRF-security of H. Since $\mathsf{PRF}_{\mathsf{H}}(k_0, \cdot)$ is replaced with a random function $\mathsf{F}^{\mathsf{H}}_{k_0}$, $k_1 = \mathsf{F}^{\mathsf{H}}_{k_0}(x)$ is random. In turn, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_1, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_1}$. Recursively, we replace each function $\mathsf{PRF}_{\mathsf{H}}(k_i, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_i}$. There is at most q instances per party, hence at most $q - 1$ updates of the original key k_0 before computing a ticket (that is, $0 \leq i < q$). Therefore, distinguishing the successive changes corresponds to an advantage at most $(q - 1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$. Consequently, in this game, the challenger aborts the experiment if the adversary is able to distinguish any of these changes. Therefore, we have that

$$\Pr[E_2] \leq \Pr[E_3] + (q - 1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$$

Game$^{\mathsf{server}}$ *4.* In this game, the challenger aborts the experiment if the adversary is able to get the key ik from $ticket = \mathsf{KW}(k_i, ik)$, $0 \leq i < q$. We reduce this event to the AE-security of function KW (this is possible because k_i is indistinguishable from random due to Game$^{\mathsf{server}}$ 3). Therefore we have that

$$\Pr[E_3] \leq \Pr[E_4] + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1)$$

where $\mathcal{B}_1$ is an adversary against the AE-security of KW.

Game$^{\mathsf{server}}$ *5.* The first value of K' (K'_0) used to compute a MAC tag τ_A is uniformly chosen at random. During the next protocol run, the key is replaced with $\mathsf{update}(K'_0) = \mathsf{PRF}_{\mathsf{update}}(K'_0, x')$. Since K'_0 is random, we can replace $\mathsf{PRF}_{\mathsf{update}}(K'_0, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{update}}_{K'_0}$. We do the same for any client instance that uses update function with the same key K'_0. Distinguishing the change implies an algorithm able to distinguish the function update from a random function. This corresponds to an advantage $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$. Since $\mathsf{PRF}_{\mathsf{update}}(K'_0, \cdot)$ is replaced with a random function $\mathsf{F}^{\mathsf{update}}_{K'_0}$, $K'_1 = \mathsf{F}^{\mathsf{update}}_{K'_0}(x')$ is random. In turn, we can replace $\mathsf{PRF}_{\mathsf{update}}(K'_1, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{update}}_{K'_1}$. Recursively, we replace each function $\mathsf{PRF}_{\mathsf{update}}(K'_i, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{update}}_{K'_i}$. There is at most q instances per party, hence at most

$q-1$ updates of the original key K'_0 before computing a MAC tag τ_A (that is, $0 \leq i < q$). Therefore, distinguishing the successive changes corresponds to an advantage at most $(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$. Consequently, in this game, the challenger aborts the experiment if the adversary succeeds in distinguishing any of these changes. Therefore, we have that

$$\Pr[E_4] \leq \Pr[E_5] + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$$

Game$^{\mathsf{server}}$ *6.* In this game, the challenger aborts the experiment if the targeted instance π ever receives a valid message τ_A but no instance partnered with π has output that message. Such a forgery can be achieved in one of two ways: either the adversary succeeds in forging a valid MAC tag τ_A, or it gets the key K' carried in *ticket*. We reduce the first possibility to the SUF-CMA-security of the MAC function used to compute τ_A. We reduce the second possibility to the AE-security of function KW, which is already assumed due to Game$^{\mathsf{server}}$ 4. Therefore, we have that

$$\Pr[E_5] \leq \Pr[E_6] + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)$$

To that point, the adversary has no chance to win. Hence

$$\Pr[E_6] = 0$$

Collecting all the probabilities from Game$^{\mathsf{server}}$ 0 to Game$^{\mathsf{server}}$ 6, we have that

$$\begin{aligned}
\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A}) &= \Pr[E_0] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + \Pr[E_1] \\
&= \frac{nq(nq-1)}{2^\lambda} + nq \times \Pr[E_2] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_3] + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_4] + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_5] + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1)\right. \\
&\qquad\qquad\left. +(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&\leq \frac{nq(nq-1)}{2^\lambda} + nq\left[\Pr[E_6] + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2)\right. \\
&\qquad\qquad +(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3) + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) \\
&\qquad\qquad\left. +(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right] \\
&= nq\left[(nq-1)2^{-\lambda} + \mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)\right. \\
&\qquad\left. +\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)\right]
\end{aligned}$$

Finally, we have that

$$\begin{aligned}\mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A}) &\le \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) + \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A}) \\ &\le nq\Big[(nq-1)2^{-(\lambda-1)} + 2(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0) + 2\mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1) \\ &\qquad +3\mathsf{adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{B}_2) + q\cdot\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)\Big]\end{aligned}$$

B.2 Key Indistinguishability for SAKE-R

Now we consider the 2-AKE key indistinguishability security experiment. Let E_i be the event that the adversary succeeds in making an instance accept maliciously in Game i, and $\mathsf{adv}_i = \Pr[E_i] - \frac{1}{2}$.

Game 0. This game corresponds to the 2-AKE key indistinguishability security experiment. Therefore we have

$$\Pr[E_0] = \frac{1}{2} + \mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R}(\mathcal{A}) = \frac{1}{2} + \mathsf{adv}_0$$

Game 1. In this game, the challenger aborts the experiment and chooses $b \in \{0,1\}$ uniformly at random if there exists an instance that accepts maliciously. Therefore we have

$$\mathsf{adv}_0 \le \mathsf{adv}_1 + \mathsf{adv}^{\mathsf{ent\text{-}auth}}_{SAKE\text{-}R}(\mathcal{A})$$

Game 2. In this game, the challenger aborts the experiment if it does not guess which instance the adversary targets. Therefore, we have that

$$\mathsf{adv}_2 = \mathsf{adv}_1 \times \frac{1}{nq}$$

Game 3. We distinguish two cases: the adversary targets either a client instance or a server instance, corresponding respectively to an advantage $\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A})$ and $\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$. Therefore we have that

$$\mathsf{adv}_2 \le \mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) + \mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A})$$

We begin with the first case.

Game$^{\mathsf{client}}$ *3.* The first key k_0 used to compute a ticket is uniformly drawn at random. The next encryption key k_1 is computed as $k_1 = \mathsf{H}(k_0) = \mathsf{PRF}_{\mathsf{H}}(k_0, x)$. Since k_0 is random, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_0,\cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_0}$. We do the same for any server instance that uses the H function with the same key k_0 to compute k_1. Distinguishing the change implies an algorithm able to distinguish the H function from a random function. This corresponds to an advantage $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$ where $\mathcal{B}_0$ is an adversary against the PRF-security of H. Since $\mathsf{PRF}_{\mathsf{H}}(k_0,\cdot)$ is replaced with a random function $\mathsf{F}^{\mathsf{H}}_{k_0}$, $k_1 = \mathsf{F}^{\mathsf{H}}_{k_0}(x)$ is random. In turn, we can replace $\mathsf{PRF}_\mathsf{H}(k_1, \cdot)$ with a truly random function $\mathsf{F}^\mathsf{H}_{k_1}$. Recursively, we replace each function $\mathsf{PRF}_\mathsf{H}(k_i, \cdot)$ with a truly random function $\mathsf{F}^\mathsf{H}_{k_i}$. There is at most q instances per party, hence at most $q-1$ updates of the original key k_0 before computing a ticket (that is, $0 \leq i < q$). Therefore, distinguishing the successive changes corresponds to an advantage at most $(q-1)\mathsf{adv}^\mathsf{PRF}_\mathsf{H}(\mathcal{B}_0)$. Consequently, in this game, the challenger aborts the experiment if the adversary is able to distinguish any of these changes. Therefore, we have that

$$\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) \leq \mathsf{adv}^\mathsf{client}_3 + (q-1)\mathsf{adv}^\mathsf{PRF}_\mathsf{H}(\mathcal{B}_0)$$

Game$^\mathsf{client}$ *4.* In this game, the challenger aborts the experiment if the adversary succeeds in getting K from $ticket = \mathsf{KW}(k_i, ik)$. We reduce this event to the AE-security of the KW function (we use the fact that k_i be indistinguishable from random due to Game$^\mathsf{client}$ 3). Therefore we have that

$$\mathsf{adv}^\mathsf{client}_3 \leq \mathsf{adv}^\mathsf{client}_4 + \mathsf{adv}^\mathsf{AE}_\mathsf{KW}(\mathcal{B}_1)$$

Game$^\mathsf{client}$ *5.* In this game, we replace the KDF function used to compute the session key sk when keyed with K, with a random function $\mathsf{F}^\mathsf{KDF}_K$. We use the fact that K be indistinguishable from random due to Game$^\mathsf{client}$ 4. Consequently, the challenger aborts the experiment if the adversary succeeds in distinguishing the change. Therefore, we have that

$$\mathsf{adv}^\mathsf{client}_4 \leq \mathsf{adv}^\mathsf{client}_5 + \mathsf{adv}^\mathsf{PRF}_\mathsf{KDF}(\mathcal{B}_4)$$

where $\mathcal{B}_4$ is an adversary against the PRF-security of KDF.

To that point, $sk = \mathsf{F}^\mathsf{KDF}_K(f(r_A, r_B))$ is a random value. Therefore the adversary can do no better than guessing. Hence

$$\mathsf{adv}^\mathsf{client}_5 = 0$$

Collecting the probabilities from Game$^\mathsf{client}$ 3 to Game$^\mathsf{client}$ 5, we have that

$$\begin{aligned}
\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{client}}(\mathcal{A}) &\leq \mathsf{adv}^\mathsf{client}_3 + (q-1)\mathsf{adv}^\mathsf{PRF}_\mathsf{H}(\mathcal{B}_0)\\
&\leq \mathsf{adv}^\mathsf{client}_4 + \mathsf{adv}^\mathsf{AE}_\mathsf{KW}(\mathcal{B}_1) + (q-1)\mathsf{adv}^\mathsf{PRF}_\mathsf{H}(\mathcal{B}_0)\\
&\leq \mathsf{adv}^\mathsf{client}_5 + \mathsf{adv}^\mathsf{PRF}_\mathsf{KDF}(\mathcal{B}_4) + \mathsf{adv}^\mathsf{AE}_\mathsf{KW}(\mathcal{B}_1) + (q-1)\mathsf{adv}^\mathsf{PRF}_\mathsf{H}(\mathcal{B}_0)\\
&= \mathsf{adv}^\mathsf{PRF}_\mathsf{KDF}(\mathcal{B}_4) + \mathsf{adv}^\mathsf{AE}_\mathsf{KW}(\mathcal{B}_1) + (q-1)\mathsf{adv}^\mathsf{PRF}_\mathsf{H}(\mathcal{B}_0)
\end{aligned}$$

Now we consider the case where the adversary targets a server instance.

Game$^\mathsf{server}$ *3.* The first value of K (K_0) used to compute the session key is uniformly chosen at random. During the next protocol run, the key is replaced with $\mathsf{update}(K_0) = \mathsf{PRF}_\mathsf{update}(K_0, x')$. Since K_0 is random, we can replace $\mathsf{PRF}_\mathsf{update}(K_0, \cdot)$ with a truly random function $\mathsf{F}^\mathsf{update}_{K_0}$. We do the same for any client instance that uses update function with the same key K_0. Distinguishing the change implies an algorithm able to distinguish the function update from a random function. This corresponds to an advantage $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$. Since $\mathsf{PRF}_{\mathsf{update}}(K_0, \cdot)$ is replaced with a random function $\mathsf{F}^{\mathsf{update}}_{K_0}$, $K_1 = \mathsf{F}^{\mathsf{update}}_{K_0}(x')$ is random. In turn, we can replace $\mathsf{PRF}_{\mathsf{update}}(K_1, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{update}}_{K_1}$. Recursively, we replace each function $\mathsf{PRF}_{\mathsf{update}}(K_i, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{update}}_{K_i}$. There is at most q instances per party, hence at most $q-1$ updates of the original key K_0 before computing a session key (that is, $0 \le i < q$). Therefore, distinguishing the successive changes corresponds to an advantage at most $(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$. Consequently, in this game, the challenger aborts the experiment if the adversary succeeds in distinguishing any of these changes. Therefore, we have that

$$\mathsf{adv}^{\mathsf{key\text{-}ind}}_{SAKE\text{-}R,\mathsf{server}}(\mathcal{A}) \le \mathsf{adv}^{\mathsf{server}}_3 + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{update}}(\mathcal{B}_3)$$

Game$^{\mathsf{server}}$ *4.* The first key k_0 used to compute a ticket is uniformly drawn at random. The next encryption key k_1 is computed as $k_1 = \mathsf{H}(k_0) = \mathsf{PRF}_{\mathsf{H}}(k_0, x)$. Since k_0 is random, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_0, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_0}$. We do the same for any client instance that uses the H function with the same key k_0 to compute k_1. Distinguishing the change implies an algorithm able to distinguish the H function from a random function. This corresponds to an advantage $\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$ where $\mathcal{B}_0$ is an adversary against the PRF-security of H. Since $\mathsf{PRF}_{\mathsf{H}}(k_0, \cdot)$ is replaced with a random function $\mathsf{F}^{\mathsf{H}}_{k_0}$, $k_1 = \mathsf{F}^{\mathsf{H}}_{k_0}(x)$ is random. In turn, we can replace $\mathsf{PRF}_{\mathsf{H}}(k_1, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_1}$. Recursively, we replace each function $\mathsf{PRF}_{\mathsf{H}}(k_i, \cdot)$ with a truly random function $\mathsf{F}^{\mathsf{H}}_{k_i}$. There is at most q instances per party, hence at most $q-1$ updates of the original key k_0 before computing a ticket (that is, $0 \le i < q$). Therefore, distinguishing the successive changes corresponds to an advantage at most $(q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$. Consequently, in this game, the challenger aborts the experiment if the adversary is able to distinguish any of these changes. Therefore, we have that

$$\mathsf{adv}^{\mathsf{server}}_3 \le \mathsf{adv}^{\mathsf{server}}_4 + (q-1)\mathsf{adv}^{\mathsf{PRF}}_{\mathsf{H}}(\mathcal{B}_0)$$

Game$^{\mathsf{server}}$ *5.* In this game, the challenger aborts the experiment if the adversary succeeds in getting K from $ticket = \mathsf{KW}(k_i, ik)$. We reduce this event to the AE-security of the KW function (we use the fact that k_i be indistinguishable from random due to Game$^{\mathsf{server}}$ 4). Therefore we have that

$$\mathsf{adv}^{\mathsf{server}}_4 \le \mathsf{adv}^{\mathsf{server}}_5 + \mathsf{adv}^{\mathsf{AE}}_{\mathsf{KW}}(\mathcal{B}_1)$$

Game$^{\mathsf{client}}$ *6.* In this game, we replace the KDF function used to compute the session key sk when keyed with K, with a random function $\mathsf{F}^{\mathsf{KDF}}_K$. We use the fact that K be indistinguishable from random due to Game$^{\mathsf{server}}$ 3 and Game$^{\mathsf{server}}$ 5. Consequently, the challenger aborts the experiment if the adversary succeeds in distinguishing the change. Therefore, we have that

$$\mathsf{adv}^{\mathsf{server}}_5 \le \mathsf{adv}^{\mathsf{server}}_6 + \mathsf{adv}^{\mathsf{PRF}}_{\mathsf{KDF}}(\mathcal{B}_4)$$

To that point, $sk = \mathsf{F}_K^{\mathsf{KDF}}(f(r_A, r_B))$ is a random value. Therefore the adversary can do no better than guessing. Hence

$$\mathsf{adv}_6^{\mathsf{server}} = 0$$

Collecting the probabilities from Game$^{\mathsf{server}}$ 3 to Game$^{\mathsf{server}}$ 6, we have that

$$\begin{aligned}
\mathsf{adv}_{SAKE\text{-}R,\mathsf{server}}^{\mathsf{key\text{-}ind}}(\mathcal{A}) &\leq \mathsf{adv}_3^{\mathsf{server}} + (q-1)\mathsf{adv}_{\mathsf{update}}^{\mathsf{PRF}}(\mathcal{B}_3) \\
&\leq \mathsf{adv}_4^{\mathsf{server}} + (q-1)\mathsf{adv}_{\mathsf{H}}^{\mathsf{PRF}}(\mathcal{B}_0) + (q-1)\mathsf{adv}_{\mathsf{update}}^{\mathsf{PRF}}(\mathcal{B}_3) \\
&\leq \mathsf{adv}_5^{\mathsf{server}} + \mathsf{adv}_{\mathsf{KW}}^{\mathsf{AE}}(\mathcal{B}_1) + (q-1)\mathsf{adv}_{\mathsf{H}}^{\mathsf{PRF}}(\mathcal{B}_0) \\
&\quad + (q-1)\mathsf{adv}_{\mathsf{update}}^{\mathsf{PRF}}(\mathcal{B}_3) \\
&\leq \mathsf{adv}_6^{\mathsf{server}} + \mathsf{adv}_{\mathsf{KDF}}^{\mathsf{PRF}}(\mathcal{B}_4) + \mathsf{adv}_{\mathsf{KW}}^{\mathsf{AE}}(\mathcal{B}_1) + (q-1)\mathsf{adv}_{\mathsf{H}}^{\mathsf{PRF}}(\mathcal{B}_0) \\
&\quad + (q-1)\mathsf{adv}_{\mathsf{update}}^{\mathsf{PRF}}(\mathcal{B}_3) \\
&= \mathsf{adv}_{\mathsf{KDF}}^{\mathsf{PRF}}(\mathcal{B}_4) + \mathsf{adv}_{\mathsf{KW}}^{\mathsf{AE}}(\mathcal{B}_1) + (q-1)\mathsf{adv}_{\mathsf{H}}^{\mathsf{PRF}}(\mathcal{B}_0) \\
&\quad + (q-1)\mathsf{adv}_{\mathsf{update}}^{\mathsf{PRF}}(\mathcal{B}_3)
\end{aligned}$$

Finally, collecting all the probabilities, we have that

$$\begin{aligned}
\mathsf{adv}_{SAKE\text{-}R}^{\mathsf{key\text{-}ind}}(\mathcal{A}) &= \mathsf{adv}_0 \\
&\leq \mathsf{adv}_{SAKE\text{-}R}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + \mathsf{adv}_1 \\
&= \mathsf{adv}_{SAKE\text{-}R}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + nq \cdot \mathsf{adv}_2 \\
&\leq \mathsf{adv}_{SAKE\text{-}R}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + nq \left[\mathsf{adv}_{SAKE\text{-}R,\mathsf{client}}^{\mathsf{key\text{-}ind}}(\mathcal{A}) + \mathsf{adv}_{SAKE\text{-}R,\mathsf{server}}^{\mathsf{key\text{-}ind}}(\mathcal{A})\right] \\
&\leq \mathsf{adv}_{SAKE\text{-}R}^{\mathsf{ent\text{-}auth}}(\mathcal{A}) + nq \left[(q-1)\left(\mathsf{adv}_{\mathsf{update}}^{\mathsf{PRF}}(\mathcal{B}_3) + 2\mathsf{adv}_{\mathsf{H}}^{\mathsf{PRF}}(\mathcal{B}_0)\right)\right. \\
&\qquad\qquad \left. + 2\left(\mathsf{adv}_{\mathsf{KW}}^{\mathsf{AE}}(\mathcal{B}_1) + \mathsf{adv}_{\mathsf{KDF}}^{\mathsf{PRF}}(\mathcal{B}_4)\right)\right]
\end{aligned}$$