LP2+: a robust symmetric-key AKE protocol with perfect forward secrecy, and an advocacy for thorough security proofs

Pierre-Alain Jacqmin[a] and Jean Liénardy[a]

[a] *Department of Mathematics, Royal Military Academy, Brussels, Belgium*

May 28, 2025

Abstract

Symmetric-key authenticated key establishment (AKE) protocols are particularly well suited in resource constraint environments such as internet of things (IoT) devices. Moreover, they often rely on better understood assumptions than asymmetric ones. In this paper, we review the security model for symmetric-key AKE protocols. We show why several existing models allow trivial attacks while they do not protect against some non-trivial ones. We fix these issues with our new security definitions.

We show that the protocols LP2 and LP3 of Boyd et al. do not satisfy the claimed security properties. We propose a new 2-message protocol based on them, called LP2+. This protocol is proved to satisfy correctness, weak synchronization robustness, entity authentication, key indistinguishability and, as a consequence, it admits perfect forward secrecy. An instantiation of LP2+ is presented, whose security only relies on that of a pseudo-random function (PRF). Its total execution time in normal cases is dominated by only 14 evaluations of the PRF, making it a lightweight protocol that is particularly well suited for resource-constrained environments such as IoT devices.

The flaws found in the security models as well as in the security arguments could have been avoided with precise and detailed proofs. We thus take this paper as an opportunity to advocate for thorough security proofs. Therefore, we have made the choice of rigor over concision.

Contents

1 Introduction

The most widely key establishment mechanisms used nowadays are probably protocols based on public-key primitives such as RSA, Diffie-Hellman or, more recently, post-quantum alternatives such as Kyber. Such asymmetric-key protocols have the major advantage that parties involved in a protocol run do not need to share a secret in advance. In particular, this allows, with the help of a public-key infrastructure, for very large networks. However, public-key primitives often require more computational, communication, power, time, and space resources than their private-key counterparts. In many scenarios, e.g., when a server or a laptop is available at each node, these requirements are easily met and do not cause any issue. However, in resource constraint environments such as the Internet of Things (IoT), Wireless Sensor Networks (WSNs) or medical implants, these limitations may have a greater impact. In some cases, resource limitations can lead to the reduction of the security parameters of the public-key primitives below an acceptable threshold. In those cases, symmetric-key protocols may be the only adequate solution. Indeed, symmetric-key primitives are particularly well suited when lightweight cryptography is needed.

In addition to their low resource requirements, symmetric-key protocols have another advantage over public-key ones. Indeed, the latter often rely on the unproven assumptions that certain complex mathematical problems are hard to solve, while the former rely on the security of well-understood primitives such as pseudo-random functions or hash functions. Moreover, many public-key primitives used today, such as RSA, are vulnerable to the advent of quantum computers. More recent post-quantum algorithms like Kyber are being introduced and standardized, but their underlying problems are less mature than that of RSA. In contrast, private-key primitives are, up to current knowledge, naturally immune to quantum computers (up to, e.g., double the secret-key length). Therefore, for situations where security is critical, symmetric-key solutions might be the most appropriate option.

The main drawback of private-key protocols is the need to exchange secret material in advance between each pair of parties. While for large networks this task may be infeasible, for small or medium-sized networks in a given area, this might be achievable and even a cost-effective solution. Examples of use-cases are industrial networks (e.g. fleet of vehicles of a company), governmental-grade product, cryptographic systems with short lifetime, domotics, etc.

In order to avoid man-in-the-middle attacks, a key establishment protocol must provide entity authentication. That is, once each party has accepted a new session key, they are both sure to have actually communicated with their intended partner. Moreover, to avoid replay attacks, i.e., that the adversary resends a previous message between the same parties, each protocol session needs to authenticate the corresponding protocol session on its partner side. This property is called *entity authentication* in the literature and gives the name to Authenticated Key Establishment (AKE) protocols. Of course, such an informal definition could not lead to descent security proofs, and the trust that it would provide in protocols would be almost null. The seminal work [5] of Bellare and Rogaway introduced the first precise model of authenticated protocols, and in particular of AKE protocols, giving rise to meaningful definitions. This model has then been modified and improved many times to encompass other contexts or to define other security properties, see e.g. [7, 4, 9, 16, 14, 10, 19, 8].

These security models give great power to the adversary. In a nutshell, the adversary has complete control over the communication channel. It can delay, delete, modify, forward, and swap messages sent by honest parties, as well as create new messages. Moreover, in order to give the strongest guarantees and to be as close as possible from real use cases, the security models often take place in a multi-party environment where concurrent sessions are allowed. The adversary can thus initiate new initiator and responder sessions at will. With these abilities, one of the goals of the adversary in the security definitions is to distinguish between a session key established by two parties from a random key. From [4], the adversary is even allowed to corrupt the parties after the establishment of the tested session key. The extra security guarantee captured by this additional ability of the adversary is that session keys established at a given time are immune to future corruption of the parties. This property is often referred to as *perfect forward secrecy* (PFS), also known as pre-compromise secrecy. Generally speaking, it is easier to achieve PFS in a public-key protocol than in a symmetric-key protocol since, in the former case, one can generate ephemeral secrets on one side such as a secret key for a key establishment mechanism (KEM) or an exponent for a generator of a group and transmit to the other party the corresponding public

information. In a symmetric-key setting, an ephemeral secret does not have a public counterpart to be sent to the partner, and so it is harder to use such a secret to generate the session key. In order to achieve PFS in this case, one can use a (linear) key evolution scheme where the master key is updated irreversibly after the session key establishment. This technique is used, for example, in [2, 8] and in the present paper.

One problem that may arise with symmetric-key AKE protocols achieving PFS with a key evolution scheme is when the master keys of the legitimate parties are desynchronized (due to, e.g., an adversarial behavior or to the accidental loss of messages in transit). Such desynchronizations can cause a break in communication between the two parties, which can lead to denial-of-service (DoS) attacks. In this context, it is therefore important to design protocols that are resistant to desynchronizations. This property has been captured in [8] where *(weak) synchronization robustness* has been introduced. Informally speaking, this notion requires that, whichever actions the adversary has made before, if an honest run of the protocol is executed between two parties, then both parties will accept and generate the same session key.

Our contributions. The contributions made in this paper are divided into four main parts and can be summarized as follows.

1. We first explain why several widely used security notions in AKE protocols are flawed.
2. In order to correct them, we propose a new security model for symmetric-key AKE protocols.
3. Regardless of the flaws in their security model, we show that the protocols LP2 and LP3 of [8] do not satisfy their claimed security properties.
4. Finally, we propose a new symmetric-key AKE protocol called LP2+ and comprehensively prove its security properties.

Let us describe informally our contributions. We refer the reader to the body of the text for the rigorous statements. The first differences in our security model concerns the notion of matching conversation and session partnership. In order to avoid trivial wins from the adversary, one needs to impose some restrictions on the corruption query it can make. In particular, the adversary is not allowed to reveal the session key of the partner of the session being tested. To properly define the notion of partner session, the concept of *matching conversations* [5] is often used (other possibilities consist of using *partner functions* [6] or *session identifiers* [9]). The definitions of matching conversations (and of partner functions) are based on the properties of the *transcript*, that is the ordered list of messages sent and received by a party session. In [14, 2, 1, 8], to treat the case where a party P_i sent the last message of the conversation (and therefore cannot be sure that its intended partner received it), this session of P_i is said to have a matching conversation with that of P_j if the transcript of the session of P_j is a prefix of that of P_i. Therefore, the conversation at P_j's can be much shorter than on P_i's side. As shown in Subsection 2.3, this both leads to trivial attacks on the entity authentication and this makes real threats considered as trivial attacks and ignored. To correct this flaw, we introduce the notion of *partially matching conversations* in this paper (see Definition 1), a notion much closer to the original definition of matching conversations of Bellare and Rogaway from [5]. The idea is to require that P_j has received all messages except, maybe, the last one which might have been deleted or modified. Moreover, as in the seminal work [5] and contrary to [14, 2, 1, 8], we require the existence of a *strong* partner session, that is, one in which the adversary did not manage to deliver a message to some party before its partner actually created it (see Definition 8).

A second distinction in our security model regards the definition of *key indistinguishability*. In the asymmetric setting of [14], the adversary is allowed to corrupt any party right after the tested session has accepted. In particular, the adversary may corrupt before the last message of the protocol is delivered. In this asymmetric setting, this does not seem to raise any issue. However, the same permission to the adversary was made in [2, 1], but in the symmetric setting. As noted and corrected in [8], one should not allow the adversary in that setting to corrupt the partner P_j of the tested session before it has received the last message. Indeed, at that moment, P_j has not yet updated its master key and its recovery would permit to recover the tested session key. However, [8] still allows the adversary to corrupt the tested party P_i before its partner P_j has received the last message. Since, in the symmetric setting, authentication is often done with a message authentication code (MAC) whose key lies in the master key of the party, the corruption of P_i would make it

possible for the adversary to start a protocol session with P_j spoofing P_i. Since P_j has not yet updated its master key, the adversary can often recover the tested session key with this illegitimate conversation. Therefore, in our model, we disallow any corruption between the partners before both parties have received all messages (see Definition 10).

The security model we propose in this paper is described in Subsections 2.1 and 2.2. The notions of *correctness*, *weak synchronization robustness*, *entity authentication* and *key indistinguishability* are considered. The adversary has full control over the network via the queries NewSessionI, NewSessionR and Send. It has also access to a lot of secret information (as long as it does not allow trivial attacks) via the queries RevealKey, RevealState and Corrupt which respectively steal the accepted session key, the internal state of an oracle (except the master key) and the master key itself. As a consequence of the presence of the Corrupt query, our model also considers perfect forward secrecy. Note that the RevealState query allows to dump the memory of a protocol session *in between* two protocol steps, and not *during* the execution of a protocol step. Therefore, in general, the adversary has no access to the internal randomness used during such a step, although one can always specify the protocol to include this randomness in the internal state if security is not impacted. Finally, the adversary can make a unique Test query in order to define key indistinguishability.

Let us make explicit that, since our model pertains to symmetric-key cryptography, it does not protect against key compromise impersonation (KCI) attacks. That is, if a party is corrupted, the attacker can use its master key to establish a session key with that compromised party. Moreover, it does not offer *post-compromise security*, also known as backward secrecy. We also note that the paper is written in the context of concrete security. This means that we define the advantages of a fixed arbitrary adversary in various security experiments and do not specify when these advantages are sufficiently small. This is of course more general than the asymptotic approach, as one can always particularize our definitions to the case of a protocol and an adversary depending on a security parameter n and require that, under some restrictions on the adversary, the advantage is negligible in n.

LP2 and LP3 are symmetric-key AKE protocols from [8]. LP2 is a 2-message protocol which can only be used in a one-way mode, that is, one party is always the initiator and the other is always the responder. This restriction can be overcome using the protocol in duplex mode, with two independent instances of the protocol where the roles are switched. This protocol does not use nonces and we show in Subsection 4.1 how one can attack both the entity authentication and the key indistinguishability of LP2 using this lack of nonces. LP3 was designed as a 3-message two-way protocol (that is, any party can be the initiator) that uses nonces. However, in Subsection 4.1, we also show an attack on the key indistinguishability of LP3. This attack is based specifically on the fact that the protocol was claimed to allow two-way use. Let us make explicit here that these attacks do not rely on the modifications of the security models and take place both in the model of [8] and in that of the present paper.

In order to correct these protocols, we introduce in this paper the protocol LP2+ (see Figure 4). This is a symmetric-key AKE protocol with 2 messages that uses nonces, built on the same ideas as LP2 and LP3. It is designed as a two-way protocol (to fit our security model) but is essentially the duplex mode of a one-way protocol, in the sense that part of the master key is only used when the party is the initiator while another part is only used when the party is the responder. The building blocks of LP2+ are a pseudo-random function (PRF) F, a message authentication code (MAC) Σ and a nonce generator GenN, which makes it suitable for resource-constrained environments. We prove in Subsection 4.3 the correctness, weak synchronization robustness, entity authentication and key indistinguishability of LP2+ (thus including perfect forward secrecy): see Theorems 19, 20, 23 and 24. These security properties only rely on the pseudo-randomness of F, the strong unforgeability of Σ and the absence of nonce collision on the initiator side. Moreover, all our security proofs sit in the standard model. Let us also notice that LP2+ is, by its symmetric-key nature, a quantum-safe protocol. The correctness and the weak synchronization robustness of LP2+ imply that, in a multi-party environment with concurrent sessions, whatever the behavior of the adversary was before, and however the parties are desynchronized, if an uninterrupted honest run of the protocol is executed between two parties, they will both accept with the same session key, and they will be resynchronized at the end of the protocol run. Let us notice however that, if an honest run of the protocol is interrupted by other queries, the honest run might abort and the parties be desynchronized. This desynchronization is not a big issue in view of the weak synchronization

robustness as explained above and the fact that this does not break the security of other session keys.

In Subsection 4.4, we instantiate the protocol LP2+ to obtain LP2+(n, F) which depends only on the security parameter n and the pseudo-random function $F\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$. The MAC Σ is instantiated using F in CBC-MAC mode and nonces are uniformly chosen. The resulting protocol consists of two messages of $\frac{5n}{2}$ bits each and its complexity is dominated, if the parties are synchronized, by a total of 14 evaluations of the PRF F, making it a lightweight protocol. Its security relies solely on the security of F (see Theorem 26).

Let us stress the fact that many of the flaws found in previous security models and protocols would not have existed if complete security proofs were proposed instead of just mere sketches of proofs. We take this paper as an opportunity to advocate for comprehensive, detailed, and precise security proofs, whence the title of the paper. Such proofs may seem long and boring, but the uninterested reader can skip them and, more importantly, they should be considered as the minimum requirement to trust a security item. Another direction to corroborate the correctness and rigor in proofs is the use of automated formal verification methods. Although these tools have been proven valuable in eliminating ambiguities and uncovering subtle flaws, they remain relatively uncommon. Although our work does not rely on such tools, we recognize their potential as a promising direction to strengthen the trustworthiness of security proofs in cryptography. However, we stress that one should keep in mind the inherent limitations of these tools: they can only verify the properties explicitly specified by the user, and their conclusions are valid only within the confines of the chosen model.

Related works. The formal treatment of AKE protocols and their security properties was initiated in [5]. In that paper, adversaries have full control over the communication channel and they can make RevealKey queries. However, they have no access to the internal states of the protocol sessions, neither they can corrupt the master keys of the parties. Therefore, PFS was not considered there. As in our model, authentication requires the existence of a (unique) partner session via the notion of matching conversations. The Corrupt queries were introduced in [7], but were not allowed to be queried on the parties involved in the session being tested. Perfect forward secrecy was formalized in [4] where Corrupt queries were allowed on these parties *after* the Test query. The RevealState queries were first considered in the model of [9]. There are countless different security models in the literature, making it infeasible to list them all here. Let us cite e.g. [14], where, on the contrary to [5, 7, 9], the adversary does not specify the intended partner of a session when it initializes one. That paper defines matching conversations using prefixes of transcripts, leading to trivial attacks as explained above and in Subsection 2.3.

The paper [10] introduces a symmetric-key AKE protocol with forward security named FORSAKES. This protocol and its security is only based on a random oracle. A important feature in [10] is that the protocol updates the master key at specific time intervals. As a consequence, one needs synchronized clocks at each party to properly run FORSAKES. This technique has the advantage that concurrent runs of the protocol are less likely to cause abortion than with LP2+ or the protocols of [2, 1, 8]. On the other hand, this has the disadvantage that forward secrecy is only provided once the time interval at which the session key was established is over. Thus, this provides only *delayed forward secrecy* and not perfect forward secrecy.

In [2], the authors introduce two symmetric-key AKE protocols called SAKE and SAKE-AM, running in 5 and 4 messages, respectively. They both admit perfect forward security and are built, similarly as LP2+, from a MAC, some PRFs and a (uniform) nonce generator. These protocols were slightly modified in [1] in order to better protect them against denial-of-service (DoS) attacks. The main difference between these protocols and LP2+ is the efficiency since our protocol only needs 2 messages. Both papers [2, 1] use the same definition of matching conversations as in [14], which causes trivial attacks of the security model of [14]. However, it is unclear how this notion is used in the security model of [2, 1].

Our inspiration for the design of the protocol LP2+ comes from LP2 and LP3 from [8]. These protocols are symmetric-key AKE protocols based on a MAC and a PRF and use, respectively, 2 and 3 messages. While LP2 is designed as a one-way protocol not using nonces, LP3 is a two-way protocol with nonces. We show in Subsection 4.1 that these two protocols do not meet their security claims. In addition, the security model of [8] is also affected by the flaws described in Subsection 2.3.

Besides, that paper introduces the security notion of (weak) synchronization robustness whose weak version is satisfied by LP2+.

Let us make explicit here that our protocol LP2+ does *not* require a trusted authority as in [11, 13, 20] and that we do *not* consider in this document privacy preservation as in [19, 12].

Outline. The rest of the paper is organized as follows. In Section 2, we present our security model: Subsection 2.1 is devoted to the execution environment and Subsection 2.2 to security definitions, while we present the flaws of previous models in Subsection 2.3. In Section 3, we recall the necessary background on the building blocks of LP2+, i.e., pseudo-random functions (Subsection 3.1), message authentication codes (Subsection 3.2) and nonce generators (Subsection 3.3). We start Section 4 by exposing the attacks on LP2 and LP3 (Subsection 4.1). Then, we introduce our new protocol LP2+ (Subsection 4.2) and prove its security properties (Subsection 4.3). We conclude the paper by mentioning the instantiation LP2+(n, F) in Subsection 4.4.

Acknowledgment. Both authors were partially funded by the European project BeQCI (No 101091625), as well as by the Defense Funded Research (DFR) study DAP/23-01.

2 The security model

In this section, we describe our security model for symmetric-key authenticated key establishment protocols with perfect forward secrecy. We define the security environment and the allowed queries of the adversary in Subsection 2.1 and provide security definitions in Subsection 2.2. Although, as far as we know, our model is new, it shares many similarities with existing ones, see [5, 9, 14, 2, 8]. In Subsection 2.3, we compare our model with previous ones from the literature and exhibit some weaknesses in some of them.

Let us remark beforehand that we are interested in concrete security. Our definitions thus do not depend on a security parameter. In particular, our (probabilistic) algorithms are not *polynomial-time*, as this notion depends on a security parameter. Moreover, in security notions, we only define the advantage of an adversary, but not when this advantage is sufficiently small. This is of course more general than the asymptotical approach. Indeed, one can always instantiate our definitions with protocols depending on a parameter n and require that the relevant probabilistic algorithms are polynomial-time with respect to n and that the advantages of certain adversaries are bounded by negligible functions in terms of n.

2.1 Execution environment

We consider a set $\{P_1, \ldots, P_M\}$ of $M \geqslant 2$ parties that are potential participants in an authenticated key establishment (AKE) protocol Π. We assume that initially each party P_i is in possession, for each $j \in \{1, \ldots, M\} \setminus \{i\}$, of a secret master key MK_{ij}. In order to achieve perfect forward secrecy, we allow these keys to be modified over time. Note that we do *not* require that $\mathrm{MK}_{ij} = \mathrm{MK}_{ji}$, although this is often the case. We model parallel executions of the protocol by equipping each party P_i with session oracles $\pi_i^1, \pi_i^2, \ldots$, where each of them represents a process that executes a single instance of the protocol. Each of these oracles π_i^s has access to P_i's master keys MK_{ij} and is able to modify them. Moreover, each oracle maintains a set of local variables that are described in Table 1. We indicate the value of a variable v for the oracle π_i^s by $\pi_i^s.v$.

Variable	Description
α	execution state $\in \{\perp, \text{negotiating}, \text{accepted}, \text{rejected}\}$
pid	identity of the intended partner $\in \{\perp, P_1, \ldots, P_M\}$
ρ	role $\in \{\perp, \text{initiator}, \text{responder}\}$
T	transcript
st	state to store ephemeral values
sk	session key $\in \{\perp\} \cup \mathcal{K}_\mathrm{s}$ for some session key space $\mathcal{K}_\mathrm{s}$

Table 1: Local variables stored by each oracle.

At the initialization of the parties, the local variables of each session oracle π_i^s are set to the default values: $\pi_i^s.\alpha = \pi_i^s.\text{pid} = \pi_i^s.\rho = \pi_i^s.\text{sk} = \perp$ and $\pi_i^s.\text{T}$ and $\pi_i^s.\text{st}$ are empty. The variables $\pi_i^s.\alpha$, $\pi_i^s.\text{pid}$ and $\pi_i^s.\rho$ are set to a value different from $\perp$ when the adversary first interacts with the oracle π_i^s. For the sake of clarity, we will often write $\pi_i^s.\text{T}$ as T_i^s. This variable represents the transcript of the oracle π_i^s, i.e., the sequence of all messages sent and received by π_i^s in chronological order. The local state $\pi_i^s.\text{st}$ serves as the oracle's memory to save variables from one round to another. The variable $\pi_i^s.\text{sk}$ is the established session key of the oracle and one has $\pi_i^s.\text{sk} \neq \perp$ if and only if $\pi_i^s.\alpha = \text{accepted}$. Once the variables $\pi_i^s.\text{pid}$, $\pi_i^s.\rho$ and $\pi_i^s.\text{sk}$ are set to a value different from $\perp$, they can no longer be modified. Similarly, once the variable $\pi_i^s.\alpha$ reaches the value accepted or rejected, it can no longer be modified and the oracle π_i^s accepts no more messages.

To begin any of the experiments in this section, the challenger initializes M parties $\{P_1, \ldots, P_M\}$ as described above. That is, it initializes the oracles π_i^s as above and, for all $i, j \in \{1, \ldots, M\}$ with $i \neq j$, it generates MK_{ij} as prescribed by the protocol and gives it to P_i, i.e., all oracles $\pi_i^1, \pi_i^2, \ldots$ have access to it.

We assume that the adversary has complete control over the communication network. It can forward, modify, drop, and delay any message exchanged between the parties $\{P_1, \ldots, P_M\}$, as well as create new messages. More precisely, the adversary $\mathcal{A}$ can interact with the session oracles π_i^s and the parties P_i by issuing the following queries, answered by the challenger.

- NewSessionI(π_i^s, P_j) (where $\pi_i^s.\alpha = \perp$ and $i \neq j$): Starts a new initiator session for party P_i with intended partner P_j. Specifically, this query sets $\pi_i^s.\alpha = \text{negotiating}$, $\pi_i^s.\text{pid} = P_j$ and $\pi_i^s.\rho = \text{initiator}$ and runs the actions prescribed by the protocol. If a message is output, it is returned to $\mathcal{A}$ and added to T_i^s.
- NewSessionR(π_i^s, P_j, m) (where $\pi_i^s.\alpha = \perp$ and $i \neq j$): Starts a new responder session for party P_i with intended partner P_j and input message m. Specifically, this query sets $\pi_i^s.\alpha = \text{negotiating}$, $\pi_i^s.\text{pid} = P_j$ and $\pi_i^s.\rho = \text{responder}$, adds m to T_i^s and runs the actions prescribed by the protocol. If a message is output, it is returned to $\mathcal{A}$ and added to T_i^s.
- Send(π_i^s, m) (where $\pi_i^s.\alpha = \text{negotiating}$): Delivers message m to oracle π_i^s which processes it. Specifically, this query adds m to T_i^s and runs the actions prescribed by the protocol. If a message is output, it is returned to $\mathcal{A}$ and added to T_i^s.
- RevealKey(π_i^s) (where $\pi_i^s.\alpha = \text{accepted}$): Reveals the session key. Specifically, this query returns $\pi_i^s.\text{sk}$ to $\mathcal{A}$.
- RevealState(π_i^s): Reveals the state. Specifically, this query returns $\pi_i^s.\text{st}$ to $\mathcal{A}$.
- Corrupt(P_i, P_j) (where $i \neq j$): Corrupts party P_i with respect to P_j. Specifically, this query returns MK_{ij} to $\mathcal{A}$.
- Test(π_i^s) (where $\pi_i^s.\alpha = \text{accepted}$): Selects π_i^s as the test oracle for key indistinguishability. Specifically, this query uniformly chooses sk_0 in $\mathcal{K}_\text{s}$, sets $sk_1 = \pi_i^s.\text{sk}$, uniformly chooses a *test bit* $b_\text{test} \in \{0, 1\}$ and returns sk_{b_test} to $\mathcal{A}$. This action can only be queried at most once.

It is implicit that a query made by the adversary which does not respect a constraint (e.g., querying Test(π_i^s) for π_i^s such that $\pi_i^s.\alpha \neq \text{accepted}$) is disregarded by the challenger and left unanswered.

In view of the above capabilities of the adversary, $\mathcal{A}$ always knows the value of the variables $\pi_i^s.\text{pid}$, $\pi_i^s.\rho$ and T_i^s. We also assume that $\mathcal{A}$ always has access to the value of $\pi_i^s.\alpha$. The value $\pi_i^s.\alpha$ can be modified by some actions prescribed by the protocol. If $\pi_i^s.\text{sk}$ is set to a value different from $\perp$, then $\pi_i^s.\alpha$ is set to accepted, while other actions of the protocol may set $\pi_i^s.\alpha = \text{rejected}$.

Since we consider two-party protocols, we assume that the communication is alternated between the initiator and the responder. That is, we assume that, at the end of each NewSessionI, NewSessionR and Send query, the targeted oracle π_i^s outputs at most one message, and if the oracle is still in execution state $\pi_i^s.\alpha = \text{negotiating}$, it actually outputs a follow-up message. In those cases, we say that the oracle has sent that message of its transcript. Similarly, we say that the oracle has received a message m if it was added to its transcript at the start of a query NewSessionR(π_i^s, P_j, m) or Send(π_i^s, m). With the above remark, note that in the transcript T_i^s, the sent and received messages alternate.

The adversary $\mathcal{A}$ can only make one query at a time. Therefore, the RevealState query gives it access to the state $\pi_i^s.\text{st}$ of an oracle only after the completion of an entire step of the protocol (or

before the protocol has started, which is useless since $\pi_i^s.\mathrm{st} = \emptyset$ in that case). In the case where algorithms were considered as deterministic with access to some internal predefined randomness, one could also give the possibility to the adversary to have access to this randomness. In this paper, we instead chose to work with probabilistic algorithms generating their randomness on the fly and for which the adversary has, in general, no access to it. Of course, if there is no impact on security, one can still force in the specification of a protocol that this randomness is included in $\pi_i^s.\mathrm{st}$ once the session has started in order to increase the security guarantees provided by the definitions.

2.2 Security definitions

Now that we have described the execution environment of our model, we can define the security notions. In order to define partnership between sessions, the usual idea is to use matching conversations. The requirement $\mathrm{T}_i^s = \mathrm{T}_j^t$ that the transcripts of two oracles be identical (*guaranteed delivery matching conversation* in the terminology of [8]) is too strong to define partnership (see [5, 14, 8]). This is due to the fact that the party who sends the last message cannot be sure that this message was indeed received by its partner. Thus, one needs the notion of partially matching conversation. Let us stress here that, as far as we know, the following Definition 1 is new. The standard notion of *partial-transcript matching conversation* (using the terminology of [8]) is not suitable here, as will be explained in Subsection 2.3.

For an non-empty transcript T, we denote by T^* the transcript T to which the last message has been removed (whereas if T is empty, T^* also denotes the empty transcript).

Definition 1. At a given time of the execution of an adversary $\mathcal{A}$, a session oracle π_i^s is said to have a *partially matching conversation* with oracle π_j^t if and only if, at that given time:

1. either π_i^s has sent the last message of its transcript and $(\mathrm{T}_i^s)^* \in \{\mathrm{T}_j^t, (\mathrm{T}_j^t)^*\}$,
2. or π_i^s has received the last message of its transcript and $\mathrm{T}_i^s = \mathrm{T}_j^t$.

Notice that, in the above definition, it is suitable to use the transcript T^* where we remove the last message from T since we consider that protocols are such that transcripts alternate between sent and received messages (otherwise, one would have to remove maybe more than one message). Partially matching conversations are the main ingredient to define partnership of oracles.

Definition 2. At a given time of the execution of an adversary $\mathcal{A}$, for two session oracles π_i^s and π_j^t, we say that π_i^s *has* π_j^t *as partner* if and only if the following conditions hold (at that given time):

1. $\pi_i^s.\mathrm{pid} = P_j$,
2. $\pi_j^t.\mathrm{pid} = P_i$,
3. $\pi_i^s.\rho \neq \pi_j^t.\rho$,
4. $\pi_i^s.\alpha = \mathrm{accepted}$,
5. π_i^s has a partially matching conversation with π_j^t.

Let us stress that the above definition is not symmetric in π_i^s and π_j^t. The largest symmetric relation contained in the partnership relation is made explicit in the next definition.

Definition 3. At a given time of the execution of an adversary $\mathcal{A}$, we say that two session oracles π_i^s and π_j^t are *mutual partners* if and only if π_i^s has π_j^t as partner and π_j^t has π_i^s as partner; that is, if and only if the following conditions hold (at that given time):

1. $\pi_i^s.\mathrm{pid} = P_j$,
2. $\pi_j^t.\mathrm{pid} = P_i$,
3. $\pi_i^s.\rho \neq \pi_j^t.\rho$,
4. $\pi_i^s.\alpha = \mathrm{accepted} = \pi_j^t.\alpha$,
5. $\mathrm{T}_i^s = \mathrm{T}_j^t$.

We can now define the correctness of a protocol in an adversarial context. To this end, we adopt from now on the standard cryptographic approach of defining security through games (also called experiments), wherein an adversary interacts with a challenger under specified rules. Recall from Subsection 2.1 that we have already required that, for any session oracle π_i^s, one has $\pi_i^s.\mathrm{sk} \neq \perp$ if and only if $\pi_i^s.\alpha = \mathrm{accepted}$.

Definition 4 (Correctness)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\mathrm{Corr}_{\mathcal{A},\Pi}$ be the following game:

1. The challenger initializes M parties $\{P_1, \ldots, P_M\}$.
2. $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined above.
3. Once $\mathcal{A}$ has concluded, the experiment outputs 1 if and only if there exist two oracles π_i^s and π_j^t such that the following conditions hold:
   (a) π_i^s and π_j^t are mutual partners,
   (b) $\pi_i^s.\mathrm{sk} \neq \pi_j^t.\mathrm{sk}$.

The advantage of $\mathcal{A}$ in the correctness experiment $\mathrm{Corr}_{\mathcal{A},\Pi}$ is defined as

$$\mathrm{Adv}_{\Pi}^{\mathrm{Corr}}(\mathcal{A}) = \Pr(\mathrm{Corr}_{\mathcal{A},\Pi} = 1).$$

We note that the adversary is allowed to corrupt any party in the above experiment. We also notice that the correctness takes place in a concurrent setting, but nothing is required when only one oracle accepts. In particular, it is not asked that π_j^t accepts if π_i^s does since, in a key evolving setting, concurrent runs of the protocols may make honest runs abort. In the case where no queries interrupt the honest run of a protocol, one needs the notion of *weak synchronization robustness* from [8] as described below.

Definition 5. For an adversary $\mathcal{A}$, we say that an *honest run* of the protocol was executed between two session oracles π_i^s and π_j^t if and only if $\mathcal{A}$ has made the following queries in that order, potentially after swapping the roles of π_i^s and π_j^t and potentially interleaved with other queries, excluding NewSessionI, NewSessionR or Send queries to π_i^s or π_j^t:

1. the query $\mathrm{NewSessionI}(\pi_i^s, P_j)$ was made and produced a message m_1,
2. the query $\mathrm{NewSessionR}(\pi_j^t, P_i, m_1)$ was made,
3. if the above query produced a message m_2, then the query $\mathrm{Send}(\pi_i^s, m_2)$ was made,
4. if the above query produced a message m_3, then the query $\mathrm{Send}(\pi_j^t, m_3)$ was made,
5. and so on, alternating between the oracles, until no more message is produced by those query or until $\mathcal{A}$ tries to deliver a message to an oracle which is no longer negotiating.

We say that an honest run of the protocol is *interrupted* if there is any other query in between the above queries.

Definition 6 (Weak Synchronization Robustness)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\mathrm{wSR}_{\mathcal{A},\Pi}$ be the following game:

1. The challenger initializes M parties $\{P_1, \ldots, P_M\}$.
2. $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined above.
3. Once $\mathcal{A}$ has concluded, the experiment outputs 1 if and only if there exist two oracles π_i^s and π_j^t such that the following conditions hold:
   (a) an honest run of the protocol was executed between π_i^s and π_j^t,
   (b) no queries were made by $\mathcal{A}$ to interrupt the honest run between π_i^s and π_j^t,
   (c) $\mathcal{A}$ did not use queries $\mathrm{Corrupt}(P_i, P_j)$ or $\mathrm{Corrupt}(P_j, P_i)$,
   (d) $\pi_i^s.\alpha \neq$ accepted or $\pi_j^t.\alpha \neq$ accepted or $\pi_i^s.\mathrm{sk} \neq \pi_j^t.\mathrm{sk}$.

The advantage of $\mathcal{A}$ in the weak synchronization robustness security experiment $\mathrm{wSR}_{\mathcal{A},\Pi}$ is defined as

$$\mathrm{Adv}_{\Pi}^{\mathrm{wSR}}(\mathcal{A}) = \Pr(\mathrm{wSR}_{\mathcal{A},\Pi} = 1).$$

The notion of *weak* in the above definition comes from [8] and is due condition 3(b) that must be satisfied. Indeed, a stronger version allows $\mathcal{A}$ to perform some queries during the run.

Note that, opposite to what is done in [8], we allow $\mathcal{A}$ to make RevealKey, RevealState and Corrupt queries in the above definition (except from those mentioned in point 3(c)). In particular,

$\mathcal{A}$ can query RevealKey(π_i^s), RevealKey(π_j^t), RevealState(π_i^s) and RevealState(π_j^t) before or after the honest run. Moreover, we do not require that $\mathcal{A}$ specify which are the oracles π_i^s and π_j^t.

In order to define the entity authentication experiment, one still needs the following two notions.

Definition 7. At a given time of the execution of an adversary $\mathcal{A}$, a session oracle π_i^s is said to be *two-sided uncorrupted* if and only if the following conditions hold (at that given time):

1. $\pi_i^s.\alpha \neq \perp$,
2. $\mathcal{A}$ did not make the query Corrupt(P_i, π_i^s.pid),
3. $\mathcal{A}$ did not make the query Corrupt(π_i^s.pid, P_i).

We will also need the notion of strong partnership. This idea goes back at least to [5] and addresses the undesirable scenario in which an oracle receives a message from $\mathcal{A}$ before its partner actually creates it. The idea is that π_i^s has π_j^t as a stronger partner if it has it as a partner and an honest run of the protocol was executed between π_i^s and π_j^t, up to maybe the last message if π_i^s has sent it.

Definition 8. At a given time of the execution of an adversary $\mathcal{A}$, for two session oracles π_i^s and π_j^t, we say that π_i^s *has* π_j^t *as strong partner* if and only if, at that given time, π_i^s has π_j^t as partner and moreover, for each $k \in \{1, \ldots, |\mathrm{T}_i^s| - 1\}$ in the case π_i^s has sent the last message of its transcript, or for each $k \in \{1, \ldots, |\mathrm{T}_i^s|\}$ in the case π_i^s has received the last message of its transcript, then

- if π_i^s received the k^{th} message m_k of its transcript during the τ_k^{th} query of $\mathcal{A}$, then π_j^t output its k^{th} message m_k of its transcript during the ${\tau'_k}^{\text{th}}$ query of $\mathcal{A}$ with $\tau'_k < \tau_k$;
- if π_i^s sent the k^{th} message m_k of its transcript during the τ_k^{th} query of $\mathcal{A}$, then π_j^t received its k^{th} message m_k of its transcript during the ${\tau'_k}^{\text{th}}$ query of $\mathcal{A}$ with $\tau'_k > \tau_k$.

As usually done, we say that the adversary breaks entity authentication if it forces a two-sided uncorrupted oracle to accept maliciously. Note that this definition does not impose any condition on the session key π_i^s.sk.

Definition 9 (Entity Authentication)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\mathrm{EntAuth}_{\mathcal{A},\Pi}$ be the following game:

1. The challenger initializes M parties $\{P_1, \ldots, P_M\}$.
2. $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined above.
3. Once $\mathcal{A}$ has concluded, the experiment outputs 1 if and only if there exists an oracle π_i^s such that the following conditions hold:
   (a) $\pi_i^s.\alpha =$ accepted,
   (b) π_i^s was two-sided uncorrupted at the time it accepted,
   (c) there are more than one oracles π_j^t such that π_i^s has π_j^t as partner, or there is no π_j^t such that π_i^s has π_j^t as strong partner.

The advantage of $\mathcal{A}$ in the entity authentication security experiment $\mathrm{EntAuth}_{\mathcal{A},\Pi}$ is defined as

$$\mathrm{Adv}_{\Pi}^{\mathrm{EntAuth}}(\mathcal{A}) = \Pr(\mathrm{EntAuth}_{\mathcal{A},\Pi} = 1).$$

An oracle π_i^s accepting in the above sense is said to *accept maliciously*.

Concerning the secrecy of the session keys established by the AKE protocol, one says that an adversary breaks the key indistinguishability of the protocol if it can distinguish a random key from a session key, provided that it cannot access the session key trivially. The definition below makes precise what is meant by a trivial attack.

Definition 10 (Key Indistinguishability)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\mathrm{KeyInd}_{\mathcal{A},\Pi}$ be the following game:

1. The challenger initializes M parties $\{P_1, \ldots, P_M\}$.
2. $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined above.

3. When $\mathcal{A}$ concludes, it outputs a bit $b' \in \{0,1\}$.
4. If any of the following conditions is not satisfied, the output of the experiments is uniformly chosen at random from $\{0,1\}$:
   (a) $\mathcal{A}$ issued a unique Test query on an oracle π_i^s with $\pi_i^s.\mathrm{pid} = P_j$,
   (b) π_i^s was two-sided uncorrupted at the time it accepted,
   (c) when $\mathcal{A}$ issued any of the queries $\mathrm{Corrupt}(P_i, P_j)$ or $\mathrm{Corrupt}(P_j, P_i)$, one has $\mathrm{T}_i^s = \mathrm{T}_j^t$ for any π_j^t such that π_i^s has π_j^t as partner,
   (d) $\mathcal{A}$ did not issue a RevealKey query to π_i^s,
   (e) $\mathcal{A}$ did not issue a RevealKey query to any π_j^t such that π_i^s and π_j^t are mutual partners;

   otherwise, if all these conditions are satisfied, the output of the experiment is 1 if and only if $b' = b_{\mathrm{test}}$ (and 0 otherwise) where b_{test} is the test bit from the Test query.

The advantage of $\mathcal{A}$ in the key indistinguishability security experiment $\mathrm{KeyInd}_{\mathcal{A},\Pi}$ is defined as

$$\begin{aligned}\mathrm{Adv}_{\Pi}^{\mathrm{KeyInd}}(\mathcal{A}) &= \left|\Pr(\mathrm{KeyInd}_{\mathcal{A},\Pi} = 1) - \Pr(\mathrm{KeyInd}_{\mathcal{A},\Pi} = 0)\right| \\ &= 2\left|\Pr(\mathrm{KeyInd}_{\mathcal{A},\Pi} = 1) - \frac{1}{2}\right|.\end{aligned}$$

Remark 11. Let us comment here on the definitions of correctness, weak synchronization robustness, entity authentication, and key indistinguishability. First, we compare the definition of correctness with what is done in the literature. To this end, and as it often the case, we consider that, once an oracle π_i^s accepts and generates a session key $\pi_i^s.\mathrm{sk}$, it also generates a (public) session identifier $\pi_i^s.\mathrm{sid}$ as

$$\pi_i^s.\mathrm{sid} = \begin{cases}(P_i, \pi_i^s.\mathrm{pid}, \mathrm{T}_i^s) & \text{if } \pi_i^s.\rho = \text{initiator} \\ (\pi_i^s.\mathrm{pid}, P_i, \mathrm{T}_i^s) & \text{if } \pi_i^s.\rho = \text{responder.}\end{cases}$$

The notation $\pi_i^s.\mathrm{sid} = \perp$ should be understood as $\pi_i^s.\alpha \neq$ accepted (or equivalently $\pi_i^s.\mathrm{sk} = \perp$). The correctness of a protocol can then be stated as: if the adversary did not win the game $\mathrm{Corr}_{\mathcal{A},\Pi}$, then the implication

$$\left(\pi_i^s.\rho \neq \pi_j^t.\rho\right) \wedge \left(\pi_i^s.\alpha = \text{accepted} = \pi_j^t.\alpha\right) \wedge \left(\pi_i^s.\mathrm{sid} = \pi_j^t.\mathrm{sid}\right) \implies \begin{cases}\pi_i^s.\mathrm{sk} = \pi_j^t.\mathrm{sk} \\ \pi_i^s.\mathrm{pid} = P_j \\ \pi_j^t.\mathrm{pid} = P_i\end{cases}$$

holds. Combined with the requirement

$$(\pi_i^s.\alpha = \text{accepted}) \implies (\pi_i^s.\mathrm{sk} \neq \perp) \wedge (\pi_i^s.\mathrm{sid} \neq \perp),$$

which holds from our definition of execution environment, this is almost the correctness requirement of [2, 1]. The main difference is that we added $\pi_i^s.\rho \neq \pi_j^t.\rho$ in the premise of the first implication in order not to consider as an adversarial win the case where there are two different oracles π_i^s and $\pi_i^{s'}$ at the same party P_i with $\pi_i^s.\alpha = \pi_i^{s'}.\alpha =$ accepted, $\pi_i^s.\mathrm{pid} = \pi_i^{s'}.\mathrm{pid}$, $\pi_i^s.\rho = \pi_i^{s'}.\rho$ and $\mathrm{T}_i^s = \mathrm{T}_i^{s'}$.

We stress that correctness does not ensure that, if an honest run of the protocol was executed between two session oracles, they will both accept. In full generality, this is impossible for many symmetric-key AKE protocols because concurrent runs of the protocol, even done honestly by faithfully transmitting all messages in the right order, may cause sessions to abort due to a lack of synchronization. If no concurrent runs are allowed during an honest run of the protocol, if the parties are not corrupted with respect to each other, and if the adversary does not win, the weak synchronization robustness requires the acceptance and generation of the same session key by the parties. This captures the ability of the parties to resynchronize after the adversary has had a full control over the communication channel.

For entity authentication, it is required that, if $\mathcal{A}$ does not win and does not corrupt P_i or $\pi_i^s.\mathrm{pid} = P_j$, an accepting oracle π_i^s have a unique π_j^t as partner, and moreover that this be a strong partner. This implies in particular that:

- Uniqueness: there is at most one accepting oracle π_k^u with $\pi_i^s.\text{sid} = \pi_k^u.\text{sid}$ and $\pi_i^s.\rho \neq \pi_k^u.\rho$, and in this case one has $k = j$ and $u = t$, in other words, there is at most one oracle π_j^t such that π_i^s and π_j^t are mutual partners.
- Existence: an honest run of the protocol was executed between π_i^s and an oracle π_j^t of its partner party P_j, up to maybe the last message if π_i^s sent it (in that case, π_i^s is not sure whether π_j^t received it and whether π_j^t has accepted / will accept it).

For key indistinguishability, we require that an adversary cannot (substantially) distinguish a random key from a session key generated by π_i^s provided that

- the adversary did not corrupt P_i or its intended partner P_j before π_i^s has accepted,
- the adversary did not corrupt P_i or its intended partner P_j before the last message that π_i^s sent was delivered to its partners,
- the session key generated by π_i^s was not revealed to the adversary,
- the session key generated by any other oracle with the same sid but opposite role was not revealed to the adversary.

These four conditions correspond, respectively, to the items 4(b), 4(c), 4(d), and 4(e) in Definition 10, and provide the assumptions under which the key indistinguishability property holds.

Therefore, if key indistinguishability and entity authentication are achieved, given two two-sided uncorrupted oracles π_i^s and π_j^t that are mutual partners, future corruption of one of the parties P_i or P_j does not compromise the secrecy of their session key. This property is called *perfect forward secrecy*.

2.3 Comparison with previous security models

Our model is largely inspired by what is classically done in the literature, but there are several important differences. In this subsection, we make them explicit and explain why these modifications must have been made. Since the small differences concerning the definitions of correctness and weak synchronization robustness have already been described in Remark 11 and after Definition 6 respectively, we focus here on the definitions of entity authentication and key indistinguishability.

One of the first major differences with what already exists in the literature is the definition of partially matching conversations (Definition 1), see [14, 2, 1, 8]. In these papers, π_i^s is said to have a *partial-transcript matching conversation* to π_j^t (using the terminology of [8]) if

- T_j^t is a prefix of T_i^s and π_i^s has sent the last message, or
- $\mathrm{T}_i^s = \mathrm{T}_j^t$ and π_j^t has sent the last message.

In the above lines, a transcript T_j^t is a *prefix* of T_i^s if T_j^t contains at least one message and the messages in T_j^t are identical to and in the same order as the first $|\mathrm{T}_j^t|$ messages of T_i^s. One can see that this definition is not implied by and does not imply our Definition 1. The use of matching conversations in the models of [2, 1] is unclear. Besides, the definition of entity authentication in [14, 8] is almost identical to our Definition 9 except that

- *partially matching conversations* as in Definition 1 are replaced with *partial-transcript matching conversations* as above in the definition of partner (Definition 2);
- point 3(c) is replaced by the equivalent of the following statement: "there is no unique π_j^t such that π_i^s has π_j^t as partner".

The other differences (if any) are irrelevant for this discussion. Let us explain here why, on the one hand, the definition of [14, 8] is almost impossible to satisfy as it allows for trivial attacks and, on the other hand, does not forbid adversarial behavior that should be considered as a winning attack in our opinion. In the following, we consider two legitimate parties, P_i and P_j that we call Alice and Bob, respectively, and an adversary, Eve.

Firstly, let us consider the case where Eve lets a protocol instance running honestly between Alice's oracle π_i^s and Bob's oracle π_j^t up to the point where Alice creates the last message m_ℓ of the protocol. Then, Eve transmits a modified message m'_ℓ to Bob (who will probably reject it). In that scenario, Alice's oracle π_i^s will accept (with high probability), while T_i^s and T_j^t have the same

number $|\mathrm{T}_i^s|$ of messages, have the same $|\mathrm{T}_i^s| - 1$ first messages, but $\mathrm{T}_i^s \neq \mathrm{T}_j^t$. Hence, T_j^t is not a prefix of T_i^s and there is no $\pi_j^{t'}$ such that π_i^s has a partial-transcript matching conversation to $\pi_j^{t'}$. Therefore, Eve trivially succeeded in her entity authentication attack according to the definition of [14, 8]. This problem disappears with our Definition 1 since $(\mathrm{T}_i^s)^* = (\mathrm{T}_j^t)^*$ and π_i^s has a partially matching conversation with π_j^t.

For the second problem, Eve faithfully delivers the first message of Alice's oracle π_i^s to Bob's oracle π_j^t, and then she delivers Bob's answer from π_j^t to π_i^s. Then, suppose that Eve manages to spoof Bob and continues the discussion for several rounds with Alice until π_i^s accepts. We suppose that Alice sent the last message of the protocol. In that case, T_j^t contains two messages (Alice's first message and Bob's response), while T_i^s may contain many more. Since T_j^t is a prefix of T_i^s, the accepting π_i^s has a partial-transcript matching conversation to π_j^t. Therefore, the above scenario is not considered a valid attack in the sense of entity authentication of [14, 8], although Eve has spoofed Bob for maybe several rounds. In our Definition 9, this is considered as an attack, since $(\mathrm{T}_i^s)^* \notin \{\mathrm{T}_j^t, (\mathrm{T}_j^t)^*\}$ and so π_i^s has no partners.

Thirdly, we consider the following scenario. Eve initializes Alice's oracle π_i^s and spoofs Bob making Alice's oracle π_i^s accept a certain number of messages, but keeping π_i^s negotiating. Then, Eve delivers Alice's messages to Bob and manages that Bob replies with exactly the same messages Eve delivered to Alice. Then, Eve faithfully delivers subsequent messages between Alice and Bob until both Alice and Bob accept. In that scenario, one has $\mathrm{T}_i^s = \mathrm{T}_j^t$ and therefore this is not considered a winning attack in the sense of entity authentication of [14, 2, 1, 8]. However, Alice did not have an honest conversation with Bob, since she received messages before Bob created them. This makes us believe that the above scenario should be considered as a valid attack. This is why we explicitly added that the unique partner π_j^t of π_i^s must be a strong partner. This idea already appears in the seminal work [5] of Bellare and Rogaway. Note that the uniqueness requirement in Definition 9 is on (simple) partners and not only on strong partners so that each accepting oracle can compute its sid as in Remark 11 (without knowledge of Eve's information) and this definition implies there is at most one other oracle with the same sid and opposite role (if $\mathcal{A}$'s attack was not successful and it did not corrupt the parties).

Let us also remark that, with both styles of definition (that of [14, 2, 8] or of the present paper), a protocol terminating by a constant unsigned message "Confirmation" will never satisfy entity authentication. Indeed, Eve can modify the last message Alice sends to Bob and reply with the correct answer "Confirmation". This means that, if a secure protocol is modified by adding this last "Confirmation" message, it becomes insecure in that sense. This has already been noticed in [17]. Although it might look counterintuitive, we think that it should indeed be considered as not providing entity authentication in a context where the adversary controls the entire communication network. The reason is that the reception of this last "Confirmation" message does not bring any information to its receiver (since anyone could have sent it), while it makes the sender of that message not sure whether its partner received it (whereas it was sure that all messages have been transmitted so far). A similar situation occurs with our Definition 9 if a constant unsigned message "Hello" is added at the beginning of the protocol. This time, the violation of the entity authentication comes from the requirement that the partner π_j^t must be a strong partner although Eve is able to first deliver the message "Hello" to Bob before Alice actually creates it.

Let us now look at the definition of key indistinguishability. According to points 4(b) and 4(c) of Definition 10, in order to hope for an advantage, the adversary $\mathcal{A}$ is only allowed to use the queries $\mathrm{Corrupt}(P_i, P_j)$ and $\mathrm{Corrupt}(P_j, P_i)$ after π_i^s has accepted and after the last message it sent has been correctly delivered to its partners. In contrast, in [14, 2, 1, 8], $\mathcal{A}$ is allowed to corrupt P_i with respect to P_j as soon as π_i^s accepts ($\mathcal{A}$ thus does not need to deliver this last message before corrupting P_i). In the asymmetric setting of [14], this does not seem to raise any problem, but in the symmetric-key setting of [2, 1, 8], where perfect forward secrecy is often achieved via key evolution, the following might occur.

We describe a generic attack for an n-message protocol for which the key is updated after the last message is received. This attack works, e.g., for LP2 and LP3 from [8] but does not as such to the protocols of [2, 1] due to the extra message they have after the last key evolution. We refer the reader to Subsection 4.1 for a description of the protocols LP2 and LP3. In this attack, the adversary $\mathcal{A}$ first launches an honest run of the protocol between Alice's oracle π_i^1 and Bob's

oracle π_j^1. Suppose, without loss of generality, that Bob creates the $(n-1)^{\text{th}}$ message. To launch the attack, the adversary $\mathcal{A}$ does not deliver this message to Alice's π_i^1. Instead, $\mathcal{A}$ starts a new honest run of the protocol between Alice's oracle π_i^2 and Bob's oracle π_j^2 but does not deliver the last message to Bob's π_j^2. The adversary then queries Test(π_i^2) and corrupts Alice with respect to Bob. That way, $\mathcal{A}$ can reply to Bob's first oracle π_j^1 since $\mathcal{A}$ can now spoof Alice's identity (having her master key). The problem that might occur at that moment is that $\mathcal{A}$ can query RevealKey(π_j^1) and get π_j^1.sk which might be close to (or equal to) π_i^2.sk since they both might have been generated with the same update of the master key. The problem comes from the fact that, since Bob's second oracle π_j^2 never received its last message, it might not have updated Bob's master key to achieve forward secrecy. A similar attack also works for LP1 from [8] if the MAC used is not deterministic. Here, $\mathcal{A}$ queries NewSessionI(π_i^s, P_j), which makes π_i^s already accepts and outputs a signed message m. Then $\mathcal{A}$ queries Test(π_i^s) and Corrupt(P_i, P_j). Having the MAC key, $\mathcal{A}$ can then modify the signature of m to another valid one and deliver it to Bob with NewSessionR(π_j^t, P_i, m'). The session keys π_i^s.sk and π_j^t.sk will be the same but $\mathcal{A}$ is allowed to query RevealKey(π_j^t). This attack on LP1 is a *no-match attack* in the terminology of [17].

Although these attacks break the claimed security of LP2, LP3 and (if the MAC is not deterministic) LP1, we think they should not. That is the reason why we made the conditions stronger on the ability of $\mathcal{A}$ to corrupt P_i and P_j. The drawback is that when a party notices it has been corrupted at a certain time, it does not know by itself which accepted session keys can still be trusted and which ones are unsafe. However, if entity authentication and key indistinguishability are achieved, it knows that every session key that has actually been used by its mutual partner before one of them got corrupted is safe (assuming these session keys were not directly revealed to the adversary). This is the reason why we chose Definition 10 as it stands.

As a final remark on Definition 10, let us notice that, in the case where $\mathcal{A}$ does not follow the rules of the game (point 4), the output of the experiment is uniformly chosen in $\{0,1\}$ instead of always being set to 0 as is often done in the literature. This prevents the trivial attack that $\mathcal{A}$ does not respect the rules on purpose to have a probability of success far from $\frac{1}{2}$, and thus a big advantage.

3 Building blocks and their security assumptions

This section serves as a reminder on pseudo-random functions (PRF), message authentication codes (MAC), and nonce generators, the three building blocks of our protocol LP2+. As in the previous section, we describe their security properties in the concrete setting, thus not depending on a security parameter.

3.1 Pseudo-random functions

A keyed function is a function $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ where $\mathcal{K}$ is considered as the *key space*, $\mathcal{D}$ is the *domain* or *input space* and $\mathcal{R}$ is the *range* or *codomain*. For a key $k \in \mathcal{K}$, we write as usual $F_k\colon \mathcal{D} \to \mathcal{R}$ for the function $F(k,\cdot)$. Roughly speaking, a keyed function is called a pseudo-random function if it is infeasible for some kind of adversary to distinguish a random function $\mathcal{D} \to \mathcal{R}$ from a function F_k for an unknown key k uniformly chosen. More precisely, we define the advantage of an adversary as follows.

Definition 12 (Pseudo-randomness)**.** Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function. For an adversary $\mathcal{A}$ attacking the pseudo-randomness of F, let $\text{PRF}_{\mathcal{A},F}$ be the following game:

1. The challenger uniformly chooses a key $k \in \mathcal{K}$.
2. The challenger uniformly and independently chooses a bit $b \in \{0,1\}$. If $b = 0$, the challenger uniformly chooses a function $g\colon \mathcal{D} \to \mathcal{R}$ in the set of all functions $\mathcal{D} \to \mathcal{R}$; if $b = 1$, the challenger lets $g = F_k$.
3. $\mathcal{A}$ has access to an oracle $\mathcal{O}_g$ which, upon query $x \in \mathcal{D}$, returns $g(x) \in \mathcal{R}$.
4. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.
5. The output of the experiment is 1 if and only if $b = b'$ (and 0 otherwise).

The advantage of $\mathcal{A}$ in the pseudo-randomness security experiment $\mathrm{PRF}_{\mathcal{A},F}$ is

$$\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{A}) = |\Pr(\mathrm{PRF}_{\mathcal{A},F} = 1) - \Pr(\mathrm{PRF}_{\mathcal{A},F} = 0)| = 2\left|\Pr(\mathrm{PRF}_{\mathcal{A},F} = 1) - \frac{1}{2}\right|.$$

As already mentioned, in our definitions, it is implicit that a query made by the adversary which does not respect a constraint on the domain of the oracle (e.g., querying $x \notin \mathcal{D}$ to $\mathcal{O}_g$) is disregarded by the oracle and unanswered.

As in the protocols LP1, LP2 and LP3 of [8], we use a linear key evolution scheme. That is, we use a pseudo-random function $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ with $\mathcal{K} = \mathcal{R}$, and we fix a constant $\mathrm{upd} \in \mathcal{D}$. Part of the initial master key will consist of a random key $k_0 \in \mathcal{K}$, and the subsequent ones are generated by induction as

$$k_{i+1} = F_{k_i}(\mathrm{upd}) =: \mathrm{update}(k_i).$$

The session key of a protocol execution is obtained by a key derivation function (KDF) as

$$\mathrm{sk}_{i,data_i} = F_{k_i}(data_i) =: \mathrm{derive}(k_i, data_i)$$

where $data_i$ is some element of $\mathcal{D} \setminus \{\mathrm{upd}\}$ derived from the data generated during the protocol execution (typically, nonces). While in [8], a constant element of $\mathcal{D} \setminus \{\mathrm{upd}\}$ was used, here we allow a nonce-dependent value in order to provide some key freshness and reduce key control. The key evolution scheme can be displayed as in Figure 1. Its security property is formalized in Definition 13 and follows from the pseudo-randomness of F as shown in Lemma 14.

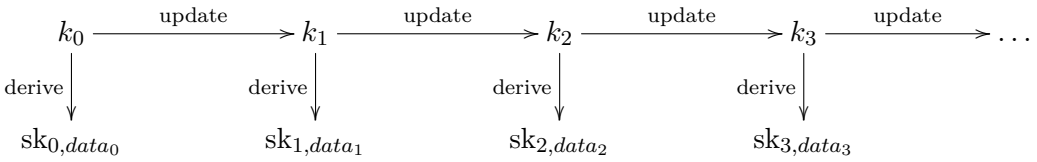


Figure 1: The linear key evolution scheme, similar as in [8]

Definition 13 (Key Evolution Security)**.** Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function with $\mathcal{K} = \mathcal{R}$, let $\mathrm{upd} \in \mathcal{D}$ be a fixed input and $q \in \mathbb{N}$. For an adversary $\mathcal{A}$ attacking the key evolution of F, let $\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}$ be the following game:

1. The challenger uniformly chooses a bit $b \in \{0,1\}$.
2. The challenger uniformly and independently chooses a key $k_0 \in \mathcal{K}$.
3. For each $i \in \{0,\dots,q-1\}$, the challenger computes $k_{i+1} = F_{k_i}(\mathrm{upd}) \in \mathcal{K}$.
4. $\mathcal{A}$ has access to the following three oracles:
   - an oracle $\mathcal{O}_{\mathrm{Reveal}}$ that on input (i,x) with $0 \leqslant i < q$ and $x \in \mathcal{D} \setminus \{\mathrm{upd}\}$, returns $\mathrm{sk}_{i,x} = F_{k_i}(x)$;
   - an oracle $\mathcal{O}_{\mathrm{Corrupt}}$ that on input i with $0 \leqslant i \leqslant q$, returns k_i;
   - an oracle $\mathcal{O}_{\mathrm{Test}}$ that on input (i,x) with $0 \leqslant i < q$ and $x \in \mathcal{D} \setminus \{\mathrm{upd}\}$, returns $\mathrm{sk}_{i,x}$, where, if $b = 0$, $\mathrm{sk}_{i,x} \in \mathcal{K}$ is chosen uniformly at random, while if $b = 1$, $\mathrm{sk}_{i,x} = F_{k_i}(x)$;

   restricted to the following constraints:
   (a) $\mathcal{A}$ can only use its $\mathcal{O}_{\mathrm{Test}}$ oracle once;
   (b) $\mathcal{A}$ cannot make both a query $\mathcal{O}_{\mathrm{Reveal}}(i,x)$ and a query $\mathcal{O}_{\mathrm{Test}}(i,y)$ (with the same first component i);
   (c) $\mathcal{A}$ cannot make three queries $\mathcal{O}_{\mathrm{Reveal}}(i,x)$, $\mathcal{O}_{\mathrm{Reveal}}(i,y)$ and $\mathcal{O}_{\mathrm{Reveal}}(i,z)$ (with the same first component i);
   (d) $\mathcal{A}$ cannot make both a query $\mathcal{O}_{\mathrm{Test}}(i,x)$ and a query $\mathcal{O}_{\mathrm{Corrupt}}(j)$ with $j \leqslant i$.
5. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.
6. The output of the experiment is 1 if and only if $b = b'$ (and 0 otherwise).

The advantage of $\mathcal{A}$ in the key evolution security experiment $\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}$ is

$$\begin{aligned}\mathrm{Adv}^{\mathrm{KEvol}}_{F,\mathrm{upd}}(\mathcal{A},q) &= |\Pr(\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}=1)-\Pr(\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}=0)|\\ &= 2\left|\Pr(\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}=1)-\frac{1}{2}\right|.\end{aligned}$$

In the above definition, we allow in 4(c) the adversary to ask for two session keys on the same level i derived with different $data_i$ (but no more) since, after corruption, an initiator oracle and a responder oracle may accept with the same k_i but different nonces. Constraint 4(d) forbids a trivial win but allows the adversary to corrupt on levels higher than that of the test query, hence ensuring perfect forward secrecy.

The following lemma derives the security of the key evolving scheme from the pseudo-randomness of F. A similar lemma, although less general, has been proved in [8].

Lemma 14. Let $F\colon \mathcal{K}\times\mathcal{D}\to\mathcal{R}$ be a function with $\mathcal{K}=\mathcal{R}$, let $\mathrm{upd}\in\mathcal{D}$ be a fixed input and $q\in\mathbb{N}$. Let also $\mathcal{A}$ be an adversary attacking F in the experiment $\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}$. There exists an adversary $\mathcal{B}$ attacking the pseudo-randomness of F such that

$$\mathrm{Adv}^{\mathrm{KEvol}}_{F,\mathrm{upd}}(\mathcal{A},q)\leqslant 2q\cdot\mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{B}).$$

Moreover, $\mathcal{B}$ queries at most three times its oracle and its running time is about the running time of $\mathcal{A}$ plus the time needed to generate $3q$ elements of $\mathcal{K}$ either by evaluation of F or by uniform selection.

Proof. The statement being trivial for $q=0$, let us suppose that $q>0$. For each $j\in\{0,\dots,q\}$, let G_j be the game $\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}$ except for the following differences:

- For each $i\in\{0,\dots,j\}$, the challenger uniformly chooses $k_i\in\mathcal{K}$ and for each $i\in\{j,\dots,q-1\}$, the challenger computes $k_{i+1}=F_{k_i}(\mathrm{upd})\in\mathcal{K}$.
- $\mathcal{O}_{\mathrm{Reveal}}$, on input (i,x), answers as follows. If $i<j$, the oracle returns $\mathrm{sk}_{i,x}\in\mathcal{K}$ uniformly chosen at random (although on twice the same $\mathcal{O}_{\mathrm{Reveal}}(i,x)$ query, it outputs the same $\mathrm{sk}_{i,x}$). Otherwise, if $j\leqslant i<q$, it returns $\mathrm{sk}_{i,x}=F_{k_i}(x)\in\mathcal{K}$.
- If $\mathcal{O}_{\mathrm{Corrupt}}$ is queried on input i with $i\leqslant j$, the game aborts and outputs a uniform value in $\{0,1\}$.
- If $\mathcal{O}_{\mathrm{Test}}$ is queried on input (i,x) with $i<j$, the game aborts and outputs a uniform value in $\{0,1\}$.

It is easily checked that the games G_0 and $\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}$ are similar, except for the query $\mathcal{O}_{\mathrm{Corrupt}}(0)$, which is allowed in the latter, but forbids a further query $\mathcal{O}_{\mathrm{Test}}$, and aborts the game in the former game. As this discrepancy does not impact the output of the game (if it queries $\mathcal{O}_{\mathrm{Corrupt}}(0)$, the output will be random), we have $\Pr(G_0=1)=\Pr(\mathrm{KEvol}_{\mathcal{A},F,\mathrm{upd},q}=1)$. It is furthermore obvious that $\Pr(G_q=1)=\frac{1}{2}$. For each $j\in\{0,\dots,q-1\}$, let us construct an adversary $\mathcal{B}_j$ attacking the pseudo-randomness of F such that

$$|\Pr(G_j=1)-\Pr(G_{j+1}=1)|=\mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{B}_j).$$

$\mathcal{B}_j$ simulates an execution of G_j except that, at initialization, k_j is left undefined and $k_{j+1}=\mathcal{O}_g(\mathrm{upd})$. Moreover, when queried $\mathcal{O}_{\mathrm{Reveal}}(j,x)$, $\mathcal{B}_j$ returns $\mathrm{sk}_{j,x}=\mathcal{O}_g(x)$ to $\mathcal{A}$. Finally, when queried $\mathcal{O}_{\mathrm{Test}}(j,x)$, $\mathcal{B}_j$ returns a uniformly chosen $\mathrm{sk}_{j,x}\in\mathcal{K}$ if $b=0$ or $\mathrm{sk}_{j,x}=\mathcal{O}_g(x)$ if $b=1$. At the end of the experiment, $\mathcal{B}_j$ will output as guess bit the output of the experiment.

Let $b''\in\{0,1\}$ be the bit chosen at the beginning of the experiment $\mathrm{PRF}_{\mathcal{B}_j,F}$. If $b''=1$ (i.e. $g=F_k$), one has that $\mathcal{B}_j$ perfectly simulates G_j, so that

$$\Pr(1\leftarrow\mathcal{B}_j\,|\,b''=1)=\Pr(G_j=1).$$

Moreover, if $b''=0$ (i.e. g is a random function), let E be the event that, in the run of $\mathcal{B}_j$, $\mathcal{A}$ queries $\mathcal{O}_{\mathrm{Test}}(j,x)$ or $\mathcal{O}_{\mathrm{Corrupt}}(j+1)$. In the former case, sk_j is uniformly chosen whatever b is and no further information on it can be obtained by $\mathcal{A}$ since it cannot make a query $\mathcal{O}_{\mathrm{Reveal}}(j,x)$ in that case. In the

latter case, either $\mathcal{A}$ does not query $\mathcal{O}_{\text{Test}}$, or $\mathcal{A}$ queries $\mathcal{O}_{\text{Test}}(j,x)$, or $\mathcal{A}$ queries $\mathcal{O}_{\text{Test}}(i,x)$ for some $i<j$. In all these cases, $\mathcal{B}_j$ outputs 1 with probability $\frac{1}{2}$. Therefore, $\Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0 \wedge E) = \frac{1}{2}$. We denote by E' the event that, in the run of game G_{j+1}, $\mathcal{A}$ queries $\mathcal{O}_{\text{Test}}(j,x)$ or $\mathcal{O}_{\text{Corrupt}}(j+1)$. If $b''=0$, $\mathcal{B}_j$ perfectly simulates G_{j+1} up to the point where E (or equivalently E') occurs. So one has $\Pr(E \,|\, b''=0) = \Pr(E')$ and $\Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0 \wedge \neg E) = \Pr(G_{j+1}=1 \,|\, \neg E')$. Thus,

$$\begin{aligned}
&\Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0)\\
&= \Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0 \wedge E)\Pr(E \,|\, b''=0) + \Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0 \wedge \neg E)\Pr(\neg E \,|\, b''=0)\\
&= \frac{1}{2}\Pr(E \,|\, b''=0) + \Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0 \wedge \neg E)\Pr(\neg E \,|\, b''=0)\\
&= \Pr(G_{j+1}=1 \,|\, E')\Pr(E') + \Pr(G_{j+1}=1 \,|\, \neg E')\Pr(\neg E')\\
&= \Pr(G_{j+1}=1).
\end{aligned}$$

Therefore,

$$\begin{aligned}
|\Pr(G_j=1) - \Pr(G_{j+1}=1)| &= |\Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=1) - \Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=0)|\\
&= |\Pr(1 \leftarrow \mathcal{B}_j \,|\, b''=1) + \Pr(0 \leftarrow \mathcal{B}_j \,|\, b''=0) - 1|\\
&= \left|2\Pr(\text{PRF}_{\mathcal{B}_j,F}=1) - 1\right|\\
&= \text{Adv}_F^{\text{PRF}}(\mathcal{B}_j)
\end{aligned}$$

as announced. Taking $\mathcal{B}$ as the $\mathcal{B}_j$ that maximizes $\text{Adv}_F^{\text{PRF}}(\mathcal{B}_j)$, one has

$$\begin{aligned}
\text{Adv}_{F,\text{upd}}^{\text{KEvol}}(\mathcal{A},q) &= 2\left|\Pr(G_0=1) - \frac{1}{2}\right|\\
&\leqslant 2\sum_{j=0}^{q-1}|\Pr(G_j=1) - \Pr(G_{j+1}=1)|\\
&\leqslant 2\sum_{j=0}^{q-1}\text{Adv}_F^{\text{PRF}}(\mathcal{B})\\
&= 2q\cdot\text{Adv}_F^{\text{PRF}}(\mathcal{B})
\end{aligned}$$

as desired. Notice that the complexity of $\mathcal{B}$ is as mentioned in the statement since, at initialization, $\mathcal{B}_j$ generates $q-1$ elements of $\mathcal{K}$ and uses once its oracle and, for each level i, at most two $\mathcal{O}_{\text{Reveal}}$ or $\mathcal{O}_{\text{Test}}$ queries can be made. □

3.2 Message authentication codes

In our protocol LP2+, a MAC is used to authenticate messages. A MAC is a triplet $\Sigma = (\text{KeyGen}, \text{Mac}, \text{Vrfy})$ of (probabilistic) algorithms where:

- KeyGen takes no input and outputs a key $k \in \mathcal{K}_{\text{MAC}}$ in a *key space* $\mathcal{K}_{\text{MAC}}$,
- Mac takes as input a key $k \in \mathcal{K}_{\text{MAC}}$ and a message $m \in \mathcal{M}$ in the *message space* $\mathcal{M}$, and outputs a tag $\sigma \in \mathcal{T}$ in the *tag space* $\mathcal{T}$,
- Vrfy is a deterministic algorithm that takes as input a key $k \in \mathcal{K}_{\text{MAC}}$, a message $m \in \mathcal{M}$ and a tag $\sigma \in \mathcal{T}$, and outputs a bit $\text{Vrfy}_k(m,\sigma) \in \{0,1\}$.

We require that MACs be *correct*, that is, for every key $k \in \mathcal{K}_{\text{MAC}}$, for every message $m \in \mathcal{M}$ and for every tag generated as $\sigma \leftarrow \text{Mac}_k(m)$, one has $\text{Vrfy}_k(m,\sigma) = 1$.

The security property that we will need on our MAC is the *strong existential unforgeability under chosen message attack* (SEUF-CMA). Roughly speaking, this means that it is hard for an adversary to forge a new pair message–valid tag, even when equipped with a tagging oracle. This is formalized by the following definition.

Definition 15 (Strong Unforgeability)**.** Let $\Sigma = (\text{KeyGen}, \text{Mac}, \text{Vrfy})$ be a MAC with key space $\mathcal{K}_{\text{MAC}}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$. For an adversary $\mathcal{A}$ attacking Σ, let $\text{SEUF-CMA}_{\mathcal{A},\Sigma}$ be the following game:

1. The challenger runs KeyGen to obtain a key $k \in \mathcal{K}_{\mathrm{MAC}}$.
2. The challenger initializes the set $\mathcal{Q}$ to the empty set $\emptyset$.
3. $\mathcal{A}$ has access to an oracle $\mathcal{O}_{\mathrm{Mac}}$ which, when queried on $m \in \mathcal{M}$, runs $\sigma \leftarrow \mathrm{Mac}_k(m)$, returns σ to $\mathcal{A}$ and updates $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$.
4. $\mathcal{A}$ outputs $(m, \sigma) \in \mathcal{M} \times \mathcal{T}$.
5. The output of the experiment is defined as 1 if and only if $(m, \sigma) \notin \mathcal{Q}$ and $\mathrm{Vrfy}_k(m, \sigma) = 1$.

The advantage of $\mathcal{A}$ in the strong unforgeability security experiment SEUF-CMA$_{\mathcal{A},\Sigma}$ is

$$\mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{A}) = \Pr(\text{SEUF-CMA}_{\mathcal{A},\Sigma} = 1).$$

In order to ease the readability of our security proofs, we consider a version of the above definition where the adversary has access to a verification oracle to which it can make q tries.

Definition 16. Let $\Sigma = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a MAC with key space $\mathcal{K}_{\mathrm{MAC}}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$, and let $q \in \mathbb{N}$. For an adversary $\mathcal{A}$ attacking Σ, let SEUF-CMA$_{\mathcal{A},\Sigma,q}$ be the following game:

1. The challenger runs KeyGen to obtain a key $k \in \mathcal{K}_{\mathrm{MAC}}$.
2. The challenger initializes the sets $\mathcal{Q}$ and $\mathcal{V}$ to the empty set $\emptyset$.
3. - $\mathcal{A}$ has access to an oracle $\mathcal{O}_{\mathrm{Mac}}$ which, when queried on $m \in \mathcal{M}$, runs $\sigma \leftarrow \mathrm{Mac}_k(m)$, returns σ to $\mathcal{A}$ and updates $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$.
   - $\mathcal{A}$ has access to an oracle $\mathcal{O}_{\mathrm{Vrfy}}$ that $\mathcal{A}$ can query at most q times and which, when queried on $(m, \sigma) \in \mathcal{M} \times \mathcal{T}$, runs $b \leftarrow \mathrm{Vrfy}_k(m, \sigma)$, returns b to $\mathcal{A}$ and, if $b = 1$, updates $\mathcal{V} \leftarrow \mathcal{V} \cup \{(m, \sigma)\}$.
4. Once $\mathcal{A}$ has concluded, the output of the experiment is defined as 1 if and only if $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$.

The advantage of $\mathcal{A}$ in the experiment SEUF-CMA$_{\mathcal{A},\Sigma,q}$ is

$$\mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{A}, q) = \Pr(\text{SEUF-CMA}_{\mathcal{A},\Sigma,q} = 1).$$

The above two security notions are linked via the following theorem from [3].

Theorem 17. [3] Let $\Sigma = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a MAC, let $q \in \mathbb{N}$ and let $\mathcal{A}$ be an adversary attacking Σ in the experiment SEUF-CMA$_{\mathcal{A},\Sigma,q}$. There exists an adversary $\mathcal{B}$ attacking Σ in the experiment SEUF-CMA$_{\mathcal{B},\Sigma}$ such that

$$\mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{A}, q) \leqslant q \cdot \mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}).$$

Moreover, the number of $\mathcal{O}_{\mathrm{Mac}}$ queries made by $\mathcal{B}$ is upper bounded by the number of $\mathcal{O}_{\mathrm{Mac}}$ queries made by $\mathcal{A}$, and the running time of $\mathcal{B}$ is about by that of $\mathcal{A}$ plus the time to save (m, σ) in $\mathcal{Q}$ for each $\mathcal{O}_{\mathrm{Mac}}$ query of $\mathcal{A}$ plus the time to check if $(m, \sigma) \in \mathcal{Q}$ for each $\mathcal{O}_{\mathrm{Vrfy}}$ query of $\mathcal{A}$.

3.3 Nonce generators

A *nonce generator* with *nonce space* $\mathcal{N}$ and *state space* $\mathcal{S}$ is a (potentially stateful) algorithm GenN that on input state $\mathrm{st} \in \mathcal{S}$, outputs an element $N \in \mathcal{N}$ and updates the state st. We denote these operations by $N \leftarrow \mathrm{GenN}(\mathrm{st})$. When $\mathcal{S}$ is a singleton, we say that the nonce generator is *stateless*.

For q consecutive calls to the algorithm GenN starting with initial state st_0, we denote by $\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0)$ the probability that a collision occurs between two of the q outputs of these calls.

As an example of GenN, one can choose a stateless uniform selection of an element of $\mathcal{N}$, in which case one has

$$\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0) \leqslant \frac{q^2}{2|\mathcal{N}|}.$$

Another example for GenN is a cyclic counter, where $\mathcal{N} = \mathcal{S} = \mathbb{Z}_n$ and $N \leftarrow \mathrm{GenN}(\mathrm{st})$ outputs $N = \mathrm{st}$ and updates $\mathrm{st} \leftarrow \mathrm{st} + 1$. In this case, one has

$$\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0) = \begin{cases} 0 & \text{if } q \leqslant |\mathcal{N}|, \\ 1 & \text{if } q > |\mathcal{N}|. \end{cases}$$

In LP2+, only the absence of collision on the initiator side is important, not on the responder side (see Theorems 23 and 24). We can thus also choose $\mathcal{S} = \{0, 1\}$ with the initial state on the initiator side always being $\mathrm{st}^{\mathrm{initN}} = 1$ and the initial state on the responder side always being $\mathrm{st}^{\mathrm{respN}} = 0$. Then $N \leftarrow \mathrm{GenN}(1)$ uniformly chooses $N \in \mathcal{N}$ (and does not update the state), while $N \leftarrow \mathrm{GenN}(0)$ always selects a constant nonce $\mathrm{cst} \in \mathcal{N}$ (and does not update the state). In this case, one has

$$\mathrm{Coll}(\mathrm{GenN}, q, 1) \leqslant \frac{q^2}{2|\mathcal{N}|}$$

while $\mathrm{Coll}(\mathrm{GenN}, q, 0) = 1$ as soon as $q > 1$.

4 The protocol LP2+ and its security proofs

In this section, we first expose the protocols LP2 and LP3 from [8] and explain why they fail to satisfy their claimed security properties. Then, in Subsection 4.2, we propose a new 2-message protocol LP2+, based on LP2 and LP3, and prove in Subsection 4.3 that LP2+ achieves correctness, weak synchronization robustness, entity authentication, and key indistinguishability (and thus also perfect forward secrecy). These security proofs rely on the assumptions that a function is a PRF and that a MAC is strongly unforgeable. Finally, in Subsection 4.4, we instantiate the protocol in order to have a lightweight AKE protocol with perfect forward secrecy that relies only on a PRF hypothesis.

Let us make explicit here that our security proofs are complete proofs and not just mere sketches of proofs. Those sketches occur frequently in the cryptographic literature and are the cause of many errors. We take the opportunity here to promote rigorous and complete proofs in a domain where mistakes, as small as they might appear, may have dramatic consequences.

4.1 LP2, LP3 and their security flaws

The ingredients that are common to both protocols LP2 and LP3 of [8] are the following:

- a PRF $F \colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ with $\mathcal{K} = \mathcal{R}$;
- two distinct constants $\mathrm{upd}, \mathrm{der} \in \mathcal{D}$;
- a strongly unforgeable MAC $\Sigma = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ with key space $\mathcal{K}_{\mathrm{MAC}}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$.

We start with the protocol LP2. It has been designed with fixed roles: Alice will always be the initiator and Bob always the responder (or vice versa). As mentioned in [8], we can use it in duplex mode to remove this constraint. That is, for each pair of parties, we can generate two master keys (one for each direction). In order to integrate it into our model from Section 2, we follow this approach. Note that this choice is made for the coherence of the paper and that the security flaw has nothing to do with it.

Therefore, in the initialization phase of the protocol LP2, for each (unordered) pair of parties $\{A, B\} \subseteq \{P_1, \ldots, P_M\}$, one uniformly and independently chooses $K_{AB}^{\mathrm{init,kdf}} = K_{BA}^{\mathrm{resp,kdf}} \in \mathcal{K}$ and $K_{BA}^{\mathrm{init,kdf}} = K_{AB}^{\mathrm{resp,kdf}} \in \mathcal{K}$. Moreover, KeyGen is run twice to get $K_{AB}^{\mathrm{init,mac}} = K_{BA}^{\mathrm{resp,mac}} \in \mathcal{K}_{\mathrm{MAC}}$ and $K_{BA}^{\mathrm{init,mac}} = K_{AB}^{\mathrm{resp,mac}} \in \mathcal{K}_{\mathrm{MAC}}$. Finally, one initializes four counters $\mathrm{CTR}_{AB}^{\mathrm{init}}$, $\mathrm{CTR}_{AB}^{\mathrm{resp}}$, $\mathrm{CTR}_{BA}^{\mathrm{init}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ to 0. The keys $K^{\mathrm{init,kdf}}$ and $K^{\mathrm{init,mac}}$ and the counters $\mathrm{CTR}^{\mathrm{init}}$ serve for the initiator, while the keys $K^{\mathrm{resp,kdf}}$ and $K^{\mathrm{resp,mac}}$ and the counters $\mathrm{CTR}^{\mathrm{resp}}$ serve for the responder. The keys $K^{\mathrm{init,kdf}}$ and $K^{\mathrm{resp,kdf}}$ are used for the KDF, while the keys $K^{\mathrm{init,mac}}$ and $K^{\mathrm{resp,mac}}$ are used for the MAC. The master keys on both sides are initially

$$\mathrm{MK}_{AB} = \left(K_{AB}^{\mathrm{init,kdf}}, K_{AB}^{\mathrm{init,mac}}, \mathrm{CTR}_{AB}^{\mathrm{init}}, K_{AB}^{\mathrm{resp,kdf}}, K_{AB}^{\mathrm{resp,mac}}, \mathrm{CTR}_{AB}^{\mathrm{resp}}\right)$$

and

$$\mathrm{MK}_{BA} = \left(K_{BA}^{\mathrm{init,kdf}}, K_{BA}^{\mathrm{init,mac}}, \mathrm{CTR}_{BA}^{\mathrm{init}}, K_{BA}^{\mathrm{resp,kdf}}, K_{BA}^{\mathrm{resp,mac}}, \mathrm{CTR}_{BA}^{\mathrm{resp}}\right).$$

Although they are public state values, we include the counters as part of the master keys since they should be accessible and modifiable by any oracle of the party with the correct intended partner.

In the first step of the protocol LP2, a function NextOdd: $\mathbb{N} \to \mathbb{N}$ is used, which associates to a natural number n the least odd natural number $m > n$. We also assume that the message space $\mathcal{M}$ of the MAC contains the product $\{P_1, \ldots, P_M\} \times \mathbb{N}$. The protocol LP2 is described in Figure 2.

Alice $\left(K_{AB}^{\text{init,kdf}}, K_{AB}^{\text{init,mac}}, \text{CTR}_{AB}^{\text{init}}, K_{AB}^{\text{resp,kdf}}, K_{AB}^{\text{resp,mac}}, \text{CTR}_{AB}^{\text{resp}}\right)$		Bob $\left(K_{BA}^{\text{init,kdf}}, K_{BA}^{\text{init,mac}}, \text{CTR}_{BA}^{\text{init}}, K_{BA}^{\text{resp,kdf}}, K_{BA}^{\text{resp,mac}}, \text{CTR}_{BA}^{\text{resp}}\right)$
$C \leftarrow \text{NextOdd}(\text{CTR}_{AB}^{\text{init}})$		
while $C > \text{CTR}_{AB}^{\text{init}}$		
$\quad K_{AB}^{\text{init,kdf}} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(\text{upd})$		
$\quad \text{CTR}_{AB}^{\text{init}} \leftarrow \text{CTR}_{AB}^{\text{init}} + 1$		
$\sigma_A \leftarrow \text{Mac}_{K_{AB}^{\text{init,mac}}}\left(A, \text{CTR}_{AB}^{\text{init}}\right)$		
	$\xrightarrow{m_A = \left(\text{CTR}_{AB}^{\text{init}}, \sigma_A\right)}$	
		if $\text{Vrfy}_{K_{BA}^{\text{resp,mac}}}\left(\left(A, \text{CTR}_{AB}^{\text{init}}\right), \sigma_A\right) = 0$
		$\quad$ **abort**
		if $\text{CTR}_{AB}^{\text{init}} < \text{CTR}_{BA}^{\text{resp}}$
		$\quad$ **abort**
		while $\text{CTR}_{AB}^{\text{init}} > \text{CTR}_{BA}^{\text{resp}}$
		$\quad K_{BA}^{\text{resp,kdf}} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(\text{upd})$
		$\quad \text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{BA}^{\text{resp}} + 1$
		$\sigma_B \leftarrow \text{Mac}_{K_{BA}^{\text{resp,mac}}}(B, \text{CTR}_{BA}^{\text{resp}})$
	$\xleftarrow{m_B = (\text{CTR}_{BA}^{\text{resp}}, \sigma_B)}$	
if $\text{Vrfy}_{K_{AB}^{\text{init,mac}}}((B, \text{CTR}_{BA}^{\text{resp}}), \sigma_B) = 0$		$\text{sk} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(\text{der})$
$\quad$ **abort**		$K_{BA}^{\text{resp,kdf}} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(\text{upd})$
if $\text{CTR}_{AB}^{\text{init}} \neq \text{CTR}_{BA}^{\text{resp}}$		$\text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{BA}^{\text{resp}} + 1$
$\quad$ **abort**		
$\text{sk} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(\text{der})$		
$K_{AB}^{\text{init,kdf}} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(\text{upd})$		
$\text{CTR}_{AB}^{\text{init}} \leftarrow \text{CTR}_{AB}^{\text{init}} + 1$		

Figure 2: The protocol LP2 from [8].

Let us show an attack against entity authentication of LP2 (both in the model of [8] and in our model). This attack exploits the absence of nonces in LP2. The actions of the adversary $\mathcal{A}$ are as follows (where Alice is designated by $A = P_i$ and Bob by $B = P_j$):

1. NewSessionI(π_i^1, P_j) //$\mathcal{A}$ initiates a new initiator session at Alice.
   //This outputs the message $m_A = (1, \sigma_A)$.
   //This step concludes with $\text{CTR}_{AB}^{\text{init}} = 1$.

2. NewSessionI(π_i^2, P_j) //$\mathcal{A}$ initiates a new initiator session at Alice.
   //This outputs the message $m'_A = (3, \sigma'_A)$.
   //This step concludes with $\text{CTR}_{AB}^{\text{init}} = 3$.

3. NewSessionR(π_j^1, P_i, m'_A) //$\mathcal{A}$ initiates a new responder session at Bob with message m'_A.
//This outputs the message $m_B = (3, \sigma_B)$.
//π_j^1 accepts with π_j^1.sk derived from the 3rd update of $K_{BA}^{\text{resp,kdf}}$.
//This step concludes with $\text{CTR}_{BA}^{\text{resp}} = 4$.

4. Send(π_i^1, m_B) //$\mathcal{A}$ delivers m_B to Alice's first oracle π_i^1.
//π_i^1 accepts with π_i^1.sk derived from the 3rd update of $K_{AB}^{\text{init,kdf}}$.
//This step concludes with $\text{CTR}_{AB}^{\text{init}} = 4$.

At the end of these four steps, one has $\text{T}_i^1 = (m_A, m_B)$, $\text{T}_i^2 = (m'_A)$ and $\text{T}_j^1 = (m'_A, m_B)$. Therefore, the accepting oracle π_i^1 does not have any partner, as it does not have a partially matching conversation with π_j^1 (the only other party's oracle), nor a partial-transcript matching conversation in the sense of [8]. Hence this contradicts the entity authentication of the protocol.

One can continue this attack to turn it into an attack against the key indistinguishability of LP2 as follows:

5. Test(π_j^1) //$\mathcal{A}$ gets a test key sk.

6. RevealKey(π_i^1) //$\mathcal{A}$ gets π_i^1.sk $= \pi_j^1$.sk.

7. $(sk = \pi_i^1.\text{sk}) \leftarrow \mathcal{A}$ //$\mathcal{A}$ outputs 1 if and only if $sk = \pi_i^1$.sk.

The advantage of $\mathcal{A}$ in this key distinguishing attack is $\text{Adv}_{\text{LP2}}^{\text{KeyInd}}(\mathcal{A}) = 1 - \frac{1}{|\mathcal{K}|}$.

Let us now describe the protocol LP3 from [8]. In contrast to LP2, LP3 is designed as a non-fixed-role 3-message protocol. In addition to the PRF $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{K}$, the constants $\text{upd}, \text{der} \in \mathcal{D}$, and the MAC $\Sigma = (\text{KeyGen}, \text{Mac}, \text{Vrfy})$ as mentioned above, one also needs a constant "conf" as well as a nonce generator GenN with nonce space $\mathcal{N}$.

The initialization phase of LP3 is done as follows. For each (unordered) pair of parties $\{A, B\} \subseteq \{P_1, \ldots, P_M\}$, one uniformly chooses $K_{AB}^{\text{kdf}} = K_{BA}^{\text{kdf}} \in \mathcal{K}$ and KeyGen is run to get $K_{AB}^{\text{mac}} = K_{BA}^{\text{mac}} \in \mathcal{K}_{\text{MAC}}$. Finally, one initializes the counters CTR_{AB} and CTR_{BA} to 0 and selects the initial states $\text{st}_{AB}^{\text{nonce}}$ and $\text{st}_{BA}^{\text{nonce}}$ of GenN. The master keys are initially

$$\text{MK}_{AB} = \left(K_{AB}^{\text{kdf}}, K_{AB}^{\text{mac}}, \text{CTR}_{AB}, \text{st}_{AB}^{\text{nonce}}\right)$$

and

$$\text{MK}_{BA} = \left(K_{BA}^{\text{kdf}}, K_{BA}^{\text{mac}}, \text{CTR}_{BA}, \text{st}_{BA}^{\text{nonce}}\right).$$

Note that, as for LP2, we incorporate the state values that are common to every oracle of A with intended partner B in the master key MK_{AB} (and vice versa). The protocol LP3 is described in Figure 3 where we assume that the message space $\mathcal{M}$ of the MAC contains $\{P_1, \ldots, P_M\} \times \mathcal{N} \times \mathbb{N}$ and $\{P_1, \ldots, P_M\} \times \mathcal{N} \times \mathcal{N} \times \mathbb{N}$ and $\{P_1, \ldots, P_M\} \times \mathcal{N} \times \mathcal{N} \times \mathbb{N} \times \{\text{conf}\}$.

Alice $\left(K_{AB}^{\mathrm{kdf}}, K_{AB}^{\mathrm{mac}}, \mathrm{CTR}_{AB}, \mathrm{st}_{AB}^{\mathrm{nonce}}\right)$		Bob $\left(K_{BA}^{\mathrm{kdf}}, K_{BA}^{\mathrm{mac}}, \mathrm{CTR}_{BA}, \mathrm{st}_{BA}^{\mathrm{nonce}}\right)$
$N_1 \leftarrow \mathrm{GenN}(\mathrm{st}_{AB}^{\mathrm{nonce}})$		
$\sigma_1 \leftarrow \mathrm{Mac}_{K_{AB}^{\mathrm{mac}}}(A, N_1, \mathrm{CTR}_{AB})$		
	$\xrightarrow{m_1 = (N_1, \mathrm{CTR}_{AB}, \sigma_1)}$	
		if $\mathrm{Vrfy}_{K_{BA}^{\mathrm{mac}}}((A, N_1, \mathrm{CTR}_{AB}), \sigma_1) = 0$
		abort
		while $\mathrm{CTR}_{AB} > \mathrm{CTR}_{BA}$
		$K_{BA}^{\mathrm{kdf}} \leftarrow F_{K_{BA}^{\mathrm{kdf}}}(\mathrm{upd})$
		$\mathrm{CTR}_{BA} \leftarrow \mathrm{CTR}_{BA} + 1$
		$N_2 \leftarrow \mathrm{GenN}(\mathrm{st}_{BA}^{\mathrm{nonce}})$
		$\sigma_2 \leftarrow \mathrm{Mac}_{K_{BA}^{\mathrm{mac}}}(B, N_1, N_2, \mathrm{CTR}_{BA})$
	$\xleftarrow{m_2 = (N_2, \mathrm{CTR}_{BA}, \sigma_2)}$	
if $\mathrm{Vrfy}_{K_{AB}^{\mathrm{mac}}}((B, N_1, N_2, \mathrm{CTR}_{BA}), \sigma_2) = 0$		
abort		
while $\mathrm{CTR}_{BA} > \mathrm{CTR}_{AB}$		
$K_{AB}^{\mathrm{kdf}} \leftarrow F_{K_{AB}^{\mathrm{kdf}}}(\mathrm{upd})$		
$\mathrm{CTR}_{AB} \leftarrow \mathrm{CTR}_{AB} + 1$		
$\sigma_3 \leftarrow \mathrm{Mac}_{K_{AB}^{\mathrm{mac}}}(A, N_1, N_2, \mathrm{CTR}_{AB}, \mathrm{conf})$		
	$\xrightarrow{m_3 = (\mathrm{CTR}_{AB}, \sigma_3)}$	
$\mathrm{sk} \leftarrow F_{K_{AB}^{\mathrm{kdf}}}(\mathrm{der})$		**if** $\mathrm{Vrfy}_{K_{BA}^{\mathrm{mac}}}((A, N_1, N_2, \mathrm{CTR}_{AB}, \mathrm{conf}), \sigma_3) = 0$
$K_{AB}^{\mathrm{kdf}} \leftarrow F_{K_{AB}^{\mathrm{kdf}}}(\mathrm{upd})$		**abort**
$\mathrm{CTR}_{AB} \leftarrow \mathrm{CTR}_{AB} + 1$		**if** $\mathrm{CTR}_{AB} \neq \mathrm{CTR}_{BA}$
		abort
		$\mathrm{sk} \leftarrow F_{K_{BA}^{\mathrm{kdf}}}(\mathrm{der})$
		$K_{BA}^{\mathrm{kdf}} \leftarrow F_{K_{BA}^{\mathrm{kdf}}}(\mathrm{upd})$
		$\mathrm{CTR}_{BA} \leftarrow \mathrm{CTR}_{BA} + 1$

Figure 3: The protocol LP3 from [8].

Let us show a key distinguishing attack against LP3 (both in the model of [8] and in our model). The attack is based on the fact that the roles are not fixed in LP3. The actions of the adversary $\mathcal{A}$ are as follows (where Alice is designated by $A = P_i$ and Bob by $B = P_j$):

1. NewSessionI(π_j^1, P_i) //$\mathcal{A}$ initiates a new initiator session at Bob.
   //This outputs the message $m_1 = (N_1, 0, \sigma_1)$.
   //This step concludes with $\mathrm{CTR}_{BA} = 0$.

2. NewSessionR(π_i^1, P_j, m_1) //$\mathcal{A}$ initiates a new responder session at Alice with message m_1.
   //This outputs the message $m_2 = (N_2, 0, \sigma_2)$.
   //This step concludes with $\mathrm{CTR}_{AB} = 0$.

3.	NewSessionI(π_i^2, P_j)	//$\mathcal{A}$ initiates a new initiator session at Alice.
		//This outputs the message $m_1' = (N_1', 0, \sigma_1')$.
		//This step concludes with $\mathrm{CTR}_{AB} = 0$.

4.	NewSessionR(π_j^2, P_i, m_1')	//$\mathcal{A}$ initiates a new responder session at Bob with message m_1'.
		//This outputs the message $m_2' = (N_2', 0, \sigma_2')$.
		//This step concludes with $\mathrm{CTR}_{BA} = 0$.

5.	Send(π_i^2, m_2')	//$\mathcal{A}$ delivers m_2' to Alice's second oracle π_i^2.
		//This outputs the message $m_3' = (0, \sigma_3')$.
		//π_i^2 accepts with π_i^2.sk derived from the initial K_{AB}^{kdf}.
		//This step concludes with $\mathrm{CTR}_{AB} = 1$.

6.	Send(π_j^1, m_2)	//$\mathcal{A}$ delivers m_2 to Bob's first oracle π_j^1.
		//This outputs the message $m_3 = (0, \sigma_3)$.
		//π_j^1 accepts with π_j^1.sk derived from the initial K_{BA}^{kdf}.
		//This step concludes with $\mathrm{CTR}_{BA} = 1$.

7.	Test(π_i^2)	//$\mathcal{A}$ gets a test key sk.
8.	RevealKey(π_j^1)	//$\mathcal{A}$ gets π_j^1.sk = π_i^2.sk.
9.	$(sk = \pi_j^1.\mathrm{sk}) \leftarrow \mathcal{A}$	//$\mathcal{A}$ outputs 1 if and only if $sk = \pi_j^1$.sk.

Since $\mathrm{T}_i^1 = (m_1, m_2)$, $\mathrm{T}_i^2 = (m_1', m_2', m_3')$, $\mathrm{T}_j^1 = (m_1, m_2, m_3)$ and $\mathrm{T}_j^2 = (m_1', m_2')$, nothing forbids $\mathcal{A}$ to use RevealKey(π_j^1) which gives it the key π_i^2.sk. Again, the advantage of this attack against key indistinguishability is $\mathrm{Adv}_{\mathrm{LP3}}^{\mathrm{KeyInd}}(\mathcal{A}) = 1 - \frac{1}{|\mathcal{K}|}$. However, notice that, if $\mathcal{A}$ delivers m_3' to π_j^2, it will abort because the value of its internal counter equals 1 and is thus different from 0 as specified in m_3' (and similarly if $\mathcal{A}$ delivers m_3 to π_i^1). For this reason, we do not know how to apply this attack in practice. But in theory as in the real world, security notions are often composed many times and a mistake, which may appear insignificant at first sight, might have dramatic consequences. Therefore, we promote rigorous statements and proofs of security notions.

4.2 The new protocol LP2+

Our new protocol LP2+ has been designed based on the following ideas. The protocol LP3, which uses three messages while LP2 uses only two, was designed to work with no fixed roles. But, as we have shown above, one can actually not use it bidirectionally. In order to design a new fully proved AKE protocol that fixes the security flaws of LP2 and LP3, it thus seems more promising to design a 2-message protocol in duplex mode. However, as it stands, LP2 does not ensure entity authentication due to the lack of nonces. Hence, similarly as for LP3, we use a nonce selection in both parties. With these nonces, the update of the KDF key in the first step of LP2 is no longer necessary. Even though nonce collisions must be avoided only at the initiator side, we chose to explicitly include a nonce selection also at the responder side to prevent key control for the initiator, which could be unwanted in some applications. If one is not interested by this feature, one can use our last example of nonce generator in Subsection 3.3.

The design of LP2+ is based on the following ingredients:

- a PRF $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ with $\mathcal{K} = \mathcal{R}$ (and where $\mathcal{K} = \mathcal{K}_{\mathrm{s}}$ will be the session key space);
- a strongly unforgeable MAC $\Sigma = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ with key space $\mathcal{K}_{\mathrm{MAC}}$, message space $\mathcal{M}$ and tag space $\mathcal{T}$;
- a nonce generator GenN with nonce space $\mathcal{N}$;
- a constant $\mathrm{upd} \in \mathcal{D}$;
- a function $\mathrm{der}\colon \mathcal{N} \times \mathcal{N} \to \mathcal{D} \setminus \{\mathrm{upd}\}$.

The initialization phase of the protocol LP2+ is as follows. For each unordered pair of distinct parties $\{A, B\} \subseteq \{P_1, \ldots, P_M\}$, one uniformly and independently chooses $K_{AB}^{\text{init,kdf}} = K_{BA}^{\text{resp,kdf}} \in \mathcal{K}$ and $K_{BA}^{\text{init,kdf}} = K_{AB}^{\text{resp,kdf}} \in \mathcal{K}$. Moreover, KeyGen is run to get $K_{AB}^{\text{mac}} = K_{BA}^{\text{mac}} \in \mathcal{K}_{\text{MAC}}$. Finally, one initializes four counters $\text{CTR}_{AB}^{\text{init}}$, $\text{CTR}_{AB}^{\text{resp}}$, $\text{CTR}_{BA}^{\text{init}}$ and $\text{CTR}_{BA}^{\text{resp}}$ to 0 and selects the initial states $\text{st}_{AB}^{\text{initN}}$, $\text{st}_{AB}^{\text{respN}}$, $\text{st}_{BA}^{\text{initN}}$ and $\text{st}_{BA}^{\text{respN}}$ of GenN. The master keys on both sides are initially

$$\text{MK}_{AB} = \left(K_{AB}^{\text{init,kdf}}, \text{CTR}_{AB}^{\text{init}}, \text{st}_{AB}^{\text{initN}}, K_{AB}^{\text{resp,kdf}}, \text{CTR}_{AB}^{\text{resp}}, \text{st}_{AB}^{\text{respN}}, K_{AB}^{\text{mac}}\right)$$

and

$$\text{MK}_{BA} = \left(K_{BA}^{\text{init,kdf}}, \text{CTR}_{BA}^{\text{init}}, \text{st}_{BA}^{\text{initN}}, K_{BA}^{\text{resp,kdf}}, \text{CTR}_{BA}^{\text{resp}}, \text{st}_{BA}^{\text{respN}}, K_{BA}^{\text{mac}}\right).$$

Again, we put the state values that are common to several oracles in the master keys to provide them access to those states. We assume that the message space $\mathcal{M}$ of the MAC contains as disjoint subsets $\{P_1, \ldots, P_M\} \times \mathbb{N} \times \mathcal{N}$ and $\{P_1, \ldots, P_M\} \times \mathbb{N} \times \mathcal{N} \times \mathcal{N}$. The protocol LP2+ is described in Figure 4. Let us make precise the following points:

- an **abort** instruction for an oracle π_i^s means $\pi_i^s.\alpha \leftarrow$ rejected and the execution of the query aborts, meaning that the next instructions are not performed;
- an instruction $\text{sk} \leftarrow X$ for an oracle π_i^s means $\pi_i^s.\text{sk} \leftarrow X$; $\pi_i^s.\alpha \leftarrow$ accepted;
- in a NewSessionI(π_i^s, P_j) query, the nonce N_A that is generated is stored in the state of the oracle $\pi_i^s.\text{st} \leftarrow N_A$ and reused during a Send(π_i^s, m) query.

In the security theorems below, we denote by $\text{Coll}(\text{GenN}, q, \text{st}_0^{\text{initN}})$ the maximum value of $\text{Coll}(\text{GenN}, q, \text{st}_0)$ as in Subsection 3.3 when st_0 runs through the initial value of $\text{st}_{AB}^{\text{initN}}$ for all distinct $A, B \in \{P_1, \ldots, P_M\}$.

4.3 The full security proofs

In this subsection, we prove the correctness (Theorem 19), weak synchronization robustness (Theorem 20), entity authentication (Theorem 23), and key indistinguishability (Theorem 24) of LP2+. These properties rely on the assumptions that the function F is a PRF, the MAC Σ is strongly unforgeable, and on the absence of nonce collision at the initiator side.

Let us first start with a lemma that we will use several times without always making an explicit reference to it. For point 3 below, for a function $f\colon \mathcal{K} \to \mathcal{K}$, we define as usual the following functions $\mathcal{K} \to \mathcal{K}$ by induction: $f^0 = \text{id}_{\mathcal{K}}$ and $f^{n+1} = f \circ f^n$ for each $n \in \mathbb{N}$. We apply in particular this definition to the function update$\colon \mathcal{K} \to \mathcal{K}$, which maps $k \mapsto F_k(\text{upd})$.

Lemma 18. Consider an adversary $\mathcal{A}$ attacking LP2+ as usual: a challenger initializes M parties $\{P_1, \ldots, P_M\}$ and $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined in Section 2. Let $A, B \in \{P_1, \ldots, P_M\}$ be two distinct parties. Then, the following properties hold.

1. At any time during the experiment, one has $K_{AB}^{\text{mac}} = K_{BA}^{\text{mac}}$ and this key does not evolve over time.
2. The counters $\text{CTR}_{AB}^{\text{init}}$, $\text{CTR}_{AB}^{\text{resp}}$, $\text{CTR}_{BA}^{\text{init}}$ and $\text{CTR}_{BA}^{\text{resp}}$ are non-decreasing over time.
3. At any time during the experiment (except between consecutive instructions $K_Y^{X,\text{kdf}} \leftarrow F_{K_Y^{X,\text{kdf}}}(\text{upd})$; $\text{CTR}_Y^X \leftarrow \text{CTR}_Y^X + 1$ with $(X, Y) = (\text{init}, AB)$ or $(X, Y) = (\text{resp}, BA)$), one has

$$K_{AB}^{\text{init,kdf}} = \text{update}^{\text{CTR}_{AB}^{\text{init}}}(K) \quad \text{and} \quad K_{BA}^{\text{resp,kdf}} = \text{update}^{\text{CTR}_{BA}^{\text{resp}}}(K)$$

where K is the initial $K_{AB}^{\text{init,kdf}} = K_{BA}^{\text{resp,kdf}}$.

Proof. 1. No query of $\mathcal{A}$ can ever change K_{AB}^{mac} or K_{BA}^{mac}. Since the equality $K_{AB}^{\text{mac}} = K_{BA}^{\text{mac}}$ holds at the initialization phase, it remains true at all time.

2. The only way these counters can evolve is via protocol actions. Since all instructions modifying these counters are of the form $\text{CTR} \leftarrow \text{CTR} + 1$, the claim is obvious.

Alice $\left(K_{AB}^{\text{init,kdf}}, \text{CTR}_{AB}^{\text{init}}, \text{st}_{AB}^{\text{initN}}, K_{AB}^{\text{resp,kdf}}, \text{CTR}_{AB}^{\text{resp}}, \text{st}_{AB}^{\text{respN}}, K_{AB}^{\text{mac}}\right)$		Bob $\left(K_{BA}^{\text{init,kdf}}, \text{CTR}_{BA}^{\text{init}}, \text{st}_{BA}^{\text{initN}}, K_{BA}^{\text{resp,kdf}}, \text{CTR}_{BA}^{\text{resp}}, \text{st}_{BA}^{\text{respN}}, K_{BA}^{\text{mac}}\right)$
$N_A \leftarrow \text{GenN}(\text{st}_{AB}^{\text{initN}})$		
$\sigma_A \leftarrow \text{Mac}_{K_{AB}^{\text{mac}}}\left(A, \text{CTR}_{AB}^{\text{init}}, N_A\right)$	$\xrightarrow{m_A = \left(\text{CTR}_{AB}^{\text{init}}, N_A, \sigma_A\right)}$	
		if $m_A \notin \mathbb{N} \times \mathcal{N} \times \mathcal{T}$
		abort
		if $\text{Vrfy}_{K_{BA}^{\text{mac}}}\left(\left(A, \text{CTR}_{AB}^{\text{init}}, N_A\right), \sigma_A\right) = 0$
		abort
		while $\text{CTR}_{AB}^{\text{init}} > \text{CTR}_{BA}^{\text{resp}}$
		$K_{BA}^{\text{resp,kdf}} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(\text{upd})$
		$\text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{BA}^{\text{resp}} + 1$
		$N_B \leftarrow \text{GenN}(\text{st}_{BA}^{\text{respN}})$
		$\sigma_B \leftarrow \text{Mac}_{K_{BA}^{\text{mac}}}\left(B, \text{CTR}_{BA}^{\text{resp}}, N_A, N_B\right)$
	$\xleftarrow{m_B = (\text{CTR}_{BA}^{\text{resp}}, N_B, \sigma_B)}$	
if $m_B \notin \mathbb{N} \times \mathcal{N} \times \mathcal{T}$		$\text{sk} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(\text{der}(N_A, N_B))$
abort		$K_{BA}^{\text{resp,kdf}} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(\text{upd})$
if $\text{Vrfy}_{K_{AB}^{\text{mac}}}\left((B, \text{CTR}_{BA}^{\text{resp}}, N_A, N_B), \sigma_B\right) = 0$		$\text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{BA}^{\text{resp}} + 1$
abort		
if $\text{CTR}_{BA}^{\text{resp}} < \text{CTR}_{AB}^{\text{init}}$		
abort		
while $\text{CTR}_{BA}^{\text{resp}} > \text{CTR}_{AB}^{\text{init}}$		
$K_{AB}^{\text{init,kdf}} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(\text{upd})$		
$\text{CTR}_{AB}^{\text{init}} \leftarrow \text{CTR}_{AB}^{\text{init}} + 1$		
$\text{sk} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(\text{der}(N_A, N_B))$		
$K_{AB}^{\text{init,kdf}} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(\text{upd})$		
$\text{CTR}_{AB}^{\text{init}} \leftarrow \text{CTR}_{AB}^{\text{init}} + 1$		

Figure 4: The AKE protocol LP2+ with perfect forward secrecy and weak synchronization robustness.

3. Since the counters $\mathrm{CTR}_{AB}^{\mathrm{init}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ are initialized to 0, the statement is clearly true at the beginning. The statement for $K_{AB}^{\mathrm{init,kdf}}$ then follows from the fact that $K_{AB}^{\mathrm{init,kdf}}$ evolves always just before $\mathrm{CTR}_{AB}^{\mathrm{init}}$ evolves, $\mathrm{CTR}_{AB}^{\mathrm{init}}$ evolves always just after $K_{AB}^{\mathrm{init,kdf}}$ evolves and they do it always as $K_{AB}^{\mathrm{init,kdf}} \leftarrow \mathrm{update}(K_{AB}^{\mathrm{init,kdf}})$ followed directly by $\mathrm{CTR}_{AB}^{\mathrm{init}} \leftarrow \mathrm{CTR}_{AB}^{\mathrm{init}} + 1$. One shows with similar arguments the statement for $K_{BA}^{\mathrm{resp,kdf}}$.

□

The correctness of LP2+ depends neither on the security of the PFR, the MAC or the nonce generator, nor on any assumptions on the power of the adversary.

Theorem 19 (Correctness of LP2+)**.** Let $\mathcal{A}$ be an adversary attacking the correctness of the protocol LP2+. One has

$$\mathrm{Adv}_{\mathrm{LP2+}}^{\mathrm{Corr}}(\mathcal{A}) = 0.$$

Proof. In view of Definition 4, one must show that if the oracles π_i^s and π_j^t are mutual partners, then $\pi_i^s.\mathrm{sk} = \pi_j^t.\mathrm{sk}$. From Definition 3, we know that both oracles have accepted with the same transcript and $\pi_i^s.\mathrm{pid} = P_j$, $\pi_j^t.\mathrm{pid} = P_i$ and $\pi_i^s.\rho \neq \pi_j^t.\rho$. Without loss of generality, we can thus assume that $\pi_i^s.\rho =$ initiator and $\pi_j^t.\rho =$ responder and we denote $P_i = A$ and $P_j = B$.

Since π_i^s has accepted, its transcript T_i^s must consist of two messages m_A, m_B of the form $m_A = (C, N_A, \sigma_A) \in \mathbb{N} \times \mathcal{N} \times \mathcal{T}$ and $m_B = (C', N_B, \sigma_B) \in \mathbb{N} \times \mathcal{N} \times \mathcal{T}$. Moreover, because of the third **if** test in the query $\mathrm{Send}(\pi_i^s, m_B)$ and the subsequent **while** loop, we know that $\mathrm{CTR}_{AB}^{\mathrm{init}} = C'$ when π_i^s accepted, that is, at the time of instruction $\mathrm{sk} \leftarrow F_{K_{AB}^{\mathrm{init,kdf}}}(\mathrm{der}(N_A, N_B))$. In view of Lemma 18.3, this means that $\pi_i^s.\mathrm{sk} = F_{\mathrm{update}^{C'}(K)}(\mathrm{der}(N_A, N_B))$ where K is the initial $K_{AB}^{\mathrm{init,kdf}} = K_{BA}^{\mathrm{resp,kdf}}$.

Since $\mathrm{T}_i^s = \mathrm{T}_j^t = (m_A, m_B)$, we know that the nonces received and generated by π_j^t are respectively N_A and N_B. Moreover, when π_j^t accepted, that is, at the time of instruction $\mathrm{sk} \leftarrow F_{K_{BA}^{\mathrm{resp,kdf}}}(\mathrm{der}(N_A, N_B))$, we know that $\mathrm{CTR}_{BA}^{\mathrm{resp}} = C'$. In view of Lemma 18.3, this means that $\pi_j^t.\mathrm{sk} = F_{\mathrm{update}^{C'}(K)}(\mathrm{der}(N_A, N_B))$ and so $\pi_i^s.\mathrm{sk} = \pi_j^t.\mathrm{sk}$. □

As for correctness, the weak synchronization robustness of LP2+ also holds for an adversary with unlimited resources, and independently of the security of the PRF, the MAC and the nonce generator.

Theorem 20 (Weak synchronization robustness of LP2+)**.** Let $\mathcal{A}$ be an adversary attacking the weak synchronization robustness of the protocol LP2+. One has

$$\mathrm{Adv}_{\mathrm{LP2+}}^{\mathrm{wSR}}(\mathcal{A}) = 0.$$

Proof. We prove here that if an honest run of the protocol was executed between oracles π_i^s and π_j^t and $\mathcal{A}$ made no queries to interrupt the protocol execution between π_i^s and π_j^t, then $\pi_i^s.\alpha = \pi_j^t.\alpha =$ accepted and $\pi_i^s.\mathrm{sk} = \pi_j^t.\mathrm{sk}$. Note that, contrary to Definition 6, we even allow the adversary to corrupt P_i with respect to P_j and vice versa. Without loss of generality, we assume that $\pi_i^s.\rho =$ initiator and $\pi_j^t.\rho =$ responder and we denote $P_i = A$ and $P_j = B$.

Let us denote by C and C' the respective values of $\mathrm{CTR}_{AB}^{\mathrm{init}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ at the start of the honest run of the protocol. We also denote by k the constant key $k = K_{AB}^{\mathrm{mac}} = K_{BA}^{\mathrm{mac}}$ (see Lemma 18.1), and by K the initial key $K_{AB}^{\mathrm{init,kdf}} = K_{BA}^{\mathrm{resp,kdf}}$.

The honest run starts with the query $\mathrm{NewSessionI}(\pi_i^s, P_j)$. Upon this query, a nonce $N_A \in \mathcal{N}$ is selected, a tag $\sigma_A \in \mathcal{T}$ is generated as $\sigma_A \leftarrow \mathrm{Mac}_k(A, C, N_A)$ and the message $m_A = (C, N_A, \sigma_A)$ is output.

The immediate next query of $\mathcal{A}$ is $\mathrm{NewSessionR}(\pi_j^t, P_i, m_A)$. By construction, one has $m_A \in \mathbb{N} \times \mathcal{N} \times \mathcal{T}$ and $\mathrm{Vrfy}_k((A, C, N_A), \sigma_A) = 1$ since our MACs are required to be always correct (see Subsection 3.2). After these checks, the **while** loop has the effect of turning $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ to $\max\{C, C'\}$ and updating $K_{BA}^{\mathrm{resp,kdf}}$ accordingly. A nonce $N_B \in \mathcal{N}$ is then selected and a tag $\sigma_B \in \mathcal{T}$ is generated as $\sigma_B \leftarrow \mathrm{Mac}_k(B, \max\{C, C'\}, N_A, N_B)$. The message $m_B = (\max\{C, C'\}, N_B, \sigma_B)$ is then output and π_j^t accepts with $\pi_j^t.\mathrm{sk} = F_{\mathrm{update}^{\max\{C,C'\}}(K)}(\mathrm{der}(N_A, N_B))$ (see Lemma 18.3). Moreover, another update of $K_{BA}^{\mathrm{resp,kdf}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ is performed.

The last query of the honest run by $\mathcal{A}$ is $\text{Send}(\pi_i^s, m_B)$. By construction, one has $m_B \in \mathbb{N}\times\mathcal{N}\times\mathcal{T}$,

$$\text{Vrfy}_k((B, \max\{C, C'\}, N_A, N_B), \sigma_B) = 1$$

and $\max\{C, C'\} \geqslant \text{CTR}_{AB}^{\text{init}} = C$. After these checks, the **while** loop has the effect of turning $\text{CTR}_{AB}^{\text{init}}$ to $\max\{C, C'\}$ and updating $K_{AB}^{\text{init,kdf}}$ accordingly. Then, π_i^s accepts with $\pi_i^s.\text{sk} = F_{\text{update}^{\max\{C,C'\}}(K)}(\text{der}(N_A, N_B))$ (see Lemma 18.3). Finally, another update of $K_{AB}^{\text{init,kdf}}$ and $\text{CTR}_{AB}^{\text{init}}$ is done.

We thus have proved that, after the execution of the honest run of the protocol between π_i^s and π_j^t, one has $\pi_i^s.\alpha = \pi_j^t.\alpha = \text{accepted}$ and $\pi_i^s.\text{sk} = F_{\text{update}^{\max\{C,C'\}}(K)}(\text{der}(N_A, N_B)) = \pi_j^t.\text{sk}$. □

Before proceeding with the proofs of entity authentication and key indistinguishability of LP2+, we first need the following lemma.

Lemma 21. Consider an adversary $\mathcal{A}$ attacking LP2+ as usual: a challenger initializes M parties $\{P_1, \ldots, P_M\}$ and $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined in Section 2. Let $\{A, B\}$ be an unordered pair of distinct parties. Let E_{AB} be the event that, at some point in the execution of $\mathcal{A}$,

(a) $\mathcal{A}$ made a NewSessionR query or a Send query to an oracle π_i^s which was, at the end of this query, two-sided uncorrupted with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$,

(b) that this query passed the Vrfy test of the oracle,

(c) and that the message-tag pair (m, σ) that passed this Vrfy test had not been previously obtained by an instruction $\sigma \leftarrow \text{Mac}_{(\cdot)}(m)$ of an oracle π_j^t with $P_j = \pi_i^s.\text{pid}$, $\pi_j^t.\text{pid} = P_i$ and $\pi_j^t.\rho \neq \pi_i^s.\rho$.

Then, there exists an adversary $\mathcal{B}_{AB}$ attacking the MAC Σ such that

$$\Pr(E_{AB}) = \text{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}_{AB}, q) \tag{1}$$

where q is an upper bound on the number of oracles π_i^s with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$. Moreover, $\mathcal{B}_{AB}$ makes at most q queries to each of its oracles $\mathcal{O}_{\text{Mac}}$ and $\mathcal{O}_{\text{Vrfy}}$, and its running time is about the same as that of $\mathcal{A}$ plus the time needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries plus the time needed to initialize the parties plus the time needed to check if $\mathcal{V} \setminus \mathcal{Q} = \emptyset$ after each successful query of $\mathcal{B}_{AB}$ to its oracle $\mathcal{O}_{\text{Vrfy}}$.

Proof. The algorithm $\mathcal{B}_{AB}$ starts by initializing the oracles of the parties $\{P_1, \ldots, P_M\}$ as usual and, for each unordered pair $\{A', B'\}$ of distinct parties, $\mathcal{B}_{AB}$ uniformly and independently chooses $K_{A'B'}^{\text{init,kdf}} = K_{B'A'}^{\text{resp,kdf}} \in \mathcal{K}$ and $K_{B'A'}^{\text{init,kdf}} = K_{A'B'}^{\text{resp,kdf}} \in \mathcal{K}$, it initializes the counters $\text{CTR}_{A'B'}^{\text{init}}$, $\text{CTR}_{A'B'}^{\text{resp}}$, $\text{CTR}_{B'A'}^{\text{init}}$ and $\text{CTR}_{B'A'}^{\text{resp}}$ to 0, and it selects the initial states $\text{st}_{A'B'}^{\text{initN}}$, $\text{st}_{A'B'}^{\text{respN}}$, $\text{st}_{B'A'}^{\text{initN}}$ and $\text{st}_{B'A'}^{\text{respN}}$ of GenN. Moreover, if $\{A', B'\} \neq \{A, B\}$, $\mathcal{B}_{AB}$ generates $K_{A'B'}^{\text{mac}} = K_{B'A'}^{\text{mac}}$ by running KeyGen. The key $K_{AB}^{\text{mac}} = K_{BA}^{\text{mac}} \in \mathcal{K}_{\text{MAC}}$ is not chosen by $\mathcal{B}_{AB}$ but it will use its oracles instead. Then $\mathcal{B}_{AB}$ runs $\mathcal{A}$ and answers its queries as specified in the protocol (see Figure 4). In more detail, for all queries NewSessionI, NewSessionR and Send issued to oracles π_i^s with intended partner P_j with $\{P_i, P_j\} \neq \{A, B\}$, $\mathcal{B}_{AB}$ can answer exactly as specified. If $\{P_i, P_j\} = \{A, B\}$, $\mathcal{B}_{AB}$ will simply use its oracles $\mathcal{O}_{\text{Mac}}$ and $\mathcal{O}_{\text{Vrfy}}$ to sign and verify the tags, respectively (see Definition 16). Upon queries RevealKey, RevealState and Test, $\mathcal{B}_{AB}$ simply answers as in their definitions. For queries $\text{Corrupt}(P_i, P_j)$ with $\{P_i, P_j\} \neq \{A, B\}$, $\mathcal{B}_{AB}$ also answers as in its definition. Upon query $\text{Corrupt}(P_i, P_j)$ with $\{P_i, P_j\} = \{A, B\}$, $\mathcal{B}_{AB}$ aborts and the output is considered as 0. As soon as $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$ (see Definition 16), $\mathcal{B}_{AB}$ stops and thus wins. Notice that the running time of $\mathcal{B}_{AB}$ is as given in the statement. For each oracle π_i^s, $\mathcal{A}$ can issue at most one NewSessionR or Send query to it, thus $\mathcal{B}_{AB}$ makes at most q queries to $\mathcal{O}_{\text{Vrfy}}$. Moreover, $\mathcal{A}$ can issue at most one NewSessionI or NewSessionR query to π_i^s, thus $\mathcal{B}_{AB}$ makes at most q queries to $\mathcal{O}_{\text{Mac}}$.

It remains to prove the equality (1). Let us notice that, while $\mathcal{B}_{AB}$ is running, it perfectly simulates $\mathcal{A}$. Moreover, there are only two possibilities for $\mathcal{B}_{AB}$ to stop $\mathcal{A}$ in advance. Either $\mathcal{A}$ issues a $\text{Corrupt}(A, B)$ query or a $\text{Corrupt}(B, A)$ query (and in both of these events, $\mathcal{B}_{AB}$ looses), or, at some point, $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$ (and in that case $\mathcal{B}_{AB}$ wins).

In the first case, it is already known at that point whether the event E_{AB} will occur: if it has not been realized before that corruption, it is known that it will not occur afterwards since all oracles π_i^s with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$ are no longer two-sided uncorrupted.

In the second case, $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$ means that $\mathcal{A}$ made a NewSessionR query or a Send query to an oracle π_i^s with intended partner P_j with $\{P_i, P_j\} = \{A, B\}$, as it is the only way for $\mathcal{V}$ to be updated. The oracle was two-sided uncorrupted at the end of this query, otherwise $\mathcal{B}_{AB}$ would have aborted. Hence, condition (a) is met. Furthermore, $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$ means that the query passed the Vrfy test of the oracle (hence condition (b) is met) and that the message-tag pair (m, σ) that passed this Vrfy test had not been previously obtained by an instruction $\sigma \leftarrow \text{Mac}_{(.)}(m)$ of an oracle π_j^t with $\{P_j, \pi_j^t.\text{pid}\} = \{A, B\}$ (which implies condition (c)). Therefore, $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$ implies that E_{AB} occurs. Conversely, if E_{AB} occurs and if the message-tag pair (m, σ) of condition (c) had been obtained by an oracle π_j^t with $\{P_j, \pi_j^t.\text{pid}\} = \{A, B\}$, since $\{P_1, \dots, P_M\} \times \mathbb{N} \times \mathcal{N}$ and $\{P_1, \dots, P_M\} \times \mathbb{N} \times \mathcal{N} \times \mathcal{N}$ are disjoint subsets of $\mathcal{M}$ (by assumption, see Subsection 4.2) and since the identity of the tagging party is included in the tagged message, one must have $\pi_j^t.\rho \neq \pi_i^s.\rho$, $P_j = \pi_i^s.\text{pid}$, and $\pi_j^t.\text{pid} = P_i$, contradicting condition (c). Therefore, $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$ occurs exactly when E_{AB} occurs.

Hence, the probability that E_{AB} occurs in a normal execution of $\mathcal{A}$ is equal to the probability that E_{AB} occurs in an execution of $\mathcal{A}$ as a subroutine of $\mathcal{B}_{AB}$, which is equal to the probability that $\mathcal{B}_{AB}$ wins. □

Corollary 22. Consider an adversary $\mathcal{A}$ attacking LP2+ as usual: a challenger initializes M parties $\{P_1, \dots, P_M\}$ and $\mathcal{A}$ may issue queries NewSessionI, NewSessionR, Send, RevealKey, RevealState, Corrupt and Test as defined in Section 2. Suppose that, for each ordered pair (A, B) of distinct parties, $\mathcal{A}$ makes at most q queries NewSessionI and q queries NewSessionR to A with intended partner B. For each unordered pair $\{A, B\}$ of distinct parties, let E_{AB} be the event that, at some point in the execution of $\mathcal{A}$,

(a) $\mathcal{A}$ made a NewSessionR query or a Send query to an oracle π_i^s which was, at the end of this query, two-sided uncorrupted with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$,

(b) that this query passed the Vrfy test of the oracle,

(c) and that the message-tag pair (m, σ) that passed this Vrfy test had not been previously obtained by an instruction $\sigma \leftarrow \text{Mac}_{(.)}(m)$ of an oracle π_j^t with $P_j = \pi_i^s.\text{pid}$, $\pi_j^t.\text{pid} = P_i$ and $\pi_j^t.\rho \neq \pi_i^s.\rho$.

Then, there exists an adversary $\mathcal{B}$ attacking the MAC Σ such that

$$\Pr(E_{AB}) \leqslant 4q \cdot \text{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B})$$

for all unordered pairs $\{A, B\}$ of distinct parties. Moreover, $\mathcal{B}$ queries at most $4q$ times its oracle $\mathcal{O}_{\text{Mac}}$. In terms of F, Mac and Vrfy evaluations, $\mathcal{B}$ makes at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries.

Proof. For each unordered pair $\{A, B\}$ of distinct parties, we know that there are at most

- q oracles π_i^s with $P_i = A$, $\pi_i^s.\text{pid} = B$ and $\pi_i^s.\rho = \text{initiator}$,
- q oracles π_i^s with $P_i = A$, $\pi_i^s.\text{pid} = B$ and $\pi_i^s.\rho = \text{responder}$,
- q oracles π_i^s with $P_i = B$, $\pi_i^s.\text{pid} = A$ and $\pi_i^s.\rho = \text{initiator}$,
- q oracles π_i^s with $P_i = B$, $\pi_i^s.\text{pid} = A$ and $\pi_i^s.\rho = \text{responder}$.

Therefore, there are at most $4q$ oracles π_i^s with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$. For each such $\{A, B\}$, let $\mathcal{B}_{AB}$ be the adversary given by Lemma 21. We thus know that $\Pr(E_{AB}) = \text{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}_{AB}, 4q)$. Taking $\mathcal{B}'$ to be the $\mathcal{B}_{AB}$ that maximizes $\text{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}_{AB}, 4q)$, we have

$$\Pr(E_{AB}) \leqslant \text{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}', 4q)$$

for all $\{A, B\}$. Then, taking $\mathcal{B}$ to be given by Theorem 17, we have $\Pr(E_{AB}) \leqslant 4q \cdot \text{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B})$ as desired. Since $\mathcal{B}'$ makes at most $4q$ queries to its oracle $\mathcal{O}_{\text{Mac}}$, so does $\mathcal{B}$. The number of computations made by $\mathcal{B}$ comes from that of $\mathcal{B}'$ (see Theorem 17) which comes from those of the $\mathcal{B}_{AB}$'s (see Lemma 21). □

The following theorem shows that entity authentication of LP2+ is independent of nonce collision at the responder side and of the security of the PRF F.

Theorem 23 (Entity authentication of LP2+)**.** Let $\mathcal{A}$ be an adversary attacking the entity authentication of the protocol LP2+ with M parties. Suppose that, for each ordered pair (A, B) of distinct parties, $\mathcal{A}$ makes at most q queries NewSessionI and q queries NewSessionR to A with intended partner B. Then, there exists an adversary $\mathcal{B}$ attacking the MAC Σ such that

$$\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{LP2+}}(\mathcal{A}) \leqslant M(M-1) \cdot \left(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q \cdot \mathrm{Adv}^{\mathrm{SEUF\text{-}CMA}}_{\Sigma}(\mathcal{B}) \right).$$

Moreover, $\mathcal{B}$ queries at most $4q$ times its oracle $\mathcal{O}_{\mathrm{Mac}}$. In terms of F, Mac and Vrfy evaluations, $\mathcal{B}$ makes at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries.

Proof. We proceed via a sequence of reductions using different games.

Game 0. This is the original game $G_0 = \mathrm{EntAuth}_{\mathcal{A},\mathrm{LP2+}}$ as defined in Definition 9. We thus have $\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{LP2+}}(\mathcal{A}) = \Pr(G_0 = 1)$.

Game 1. This game G_1 is the same as G_0 except that, as soon as some party A generates the same nonce N_A twice in two different queries NewSessionI with the same intended partner B, then the experiment aborts and outputs 0. We call this event D and we have $\Pr(G_0 = 1|\neg D) = \Pr(G_1 = 1|\neg D)$. For a fixed ordered pair (A, B) of distinct parties, the probability that A, as an initiator, generates twice the same nonce for B is upper bounded by $\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}})$ since A is queried at most q times a NewSessionI query with intended partner B. Therefore, the probability that G_1 aborts early (in comparison to G_0), that is $\Pr(D)$, is upper bounded by $M(M-1) \cdot \mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}})$. This means that

$$\begin{aligned}
\Pr(G_0 = 1) &= \Pr(G_0 = 1|\neg D)\Pr(\neg D) + \Pr(G_0 = 1|D)\Pr(D) \\
&\leqslant \Pr(G_1 = 1|\neg D)\Pr(\neg D) + \Pr(D) \\
&\leqslant \Pr(G_1 = 1) + M(M-1) \cdot \mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}).
\end{aligned}$$

Game 2. This game G_2 is the same as G_1 except that, at the beginning, the challenger guesses an unordered pair $\{A, B\}$ of distinct parties uniformly and, once $\mathcal{A}$ has concluded, the experiment outputs 1 if and only if there exists an oracle π_i^s that accepts maliciously with $\{P_i, \pi_i^s.\mathrm{pid}\} = \{A, B\}$ (see Definition 9). Then, denoting the guess of the challenger by "guess", we have

$$\begin{aligned}
\Pr(G_1 = 1) &= \Pr\left(\bigvee_{\{A,B\}} (\text{some } \pi_i^s \text{ accepts maliciously with } \{P_i, \pi_i^s.\mathrm{pid}\} = \{A, B\}) \right) \\
&\leqslant \sum_{\{A,B\}} \Pr\left(\text{some } \pi_i^s \text{ accepts maliciously with } \{P_i, \pi_i^s.\mathrm{pid}\} = \{A, B\}\right) \\
&= \sum_{\{A,B\}} \Pr(G_2 = 1 \,|\, \text{guess} = \{A, B\}) \\
&= \frac{M(M-1)}{2} \cdot \sum_{\{A,B\}} \Pr(G_2 = 1 \,|\, \text{guess} = \{A, B\}) \Pr(\text{guess} = \{A, B\}) \\
&= \frac{M(M-1)}{2} \cdot \Pr(G_2 = 1).
\end{aligned}$$

Now, for each unordered pair $\{A, B\}$ of distinct parties, let E_{AB} be the event defined in Corollary 22 and let $\mathcal{B}$ be the adversary attacking Σ given by this corollary. We will prove that for each such $\{A, B\}$, one has

$$\Pr(G_2 = 1 \,|\, \text{guess} = \{A, B\} \wedge \neg E_{AB}) = 0. \tag{2}$$

This would mean that

$$\begin{aligned}\Pr(G_2 = 1 \,|\, \text{guess} = \{A,B\}) &= \Pr(G_2 = 1 \,|\, \text{guess} = \{A,B\} \wedge E_{AB}) \Pr(E_{AB} \,|\, \text{guess} = \{A,B\}) \\ &\quad + \Pr(G_2 = 1 \,|\, \text{guess} = \{A,B\} \wedge \neg E_{AB}) \Pr(\neg E_{AB} \,|\, \text{guess} = \{A,B\}) \\ &= \Pr(G_2 = 1 \,|\, \text{guess} = \{A,B\} \wedge E_{AB}) \Pr(E_{AB}) \\ &\leqslant 4q \cdot \mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}).\end{aligned}$$

Therefore,

$$\begin{aligned}\Pr(G_2 = 1) &= \sum_{\{A,B\}} \Pr\left(G_2 = 1 \,|\, \text{guess} = \{A,B\}\right) \Pr\left(\text{guess} = \{A,B\}\right) \\ &\leqslant \sum_{\{A,B\}} 4q \cdot \mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}) \Pr\left(\text{guess} = \{A,B\}\right) \\ &= 4q \cdot \mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B})\end{aligned}$$

which, putting everything together, will conclude the proof.

It thus remains to prove (2). We fix an unordered pair $\{A,B\}$ of distinct parties, we assume that guess $= \{A,B\}$ and that E_{AB} does not occur, and we must prove that $G_2 = 1$ is impossible, that is, there is no oracle π_i^s that accepts maliciously with $\{P_i, \pi_i^s.\text{pid}\} = \{A,B\}$. By contradiction, let us suppose that such a π_i^s exists. In particular, π_i^s accepted and it was two-sided uncorrupted when it did. In view of the definition of G_1, we also suppose that all initiator oracles of a party that have as intended partner the same fixed party produce different nonces.

Let us first treat the case where $\pi_i^s.\rho = \text{initiator}$. By symmetry, we can suppose $P_i = A$ and $\pi_i^s.\text{pid} = P_j = B$. Let us notice that it is not possible that there are two distinct π_j^t and $\pi_j^{t'}$ such that π_i^s has both π_j^t and $\pi_j^{t'}$ as partners. Indeed, if it were the case, as π_i^s had received the last message of its transcript, one would have $\mathrm{T}_i^s = \mathrm{T}_j^t = \mathrm{T}_j^{t'}$ (see Definitions 1 and 2). This would mean that π_j^t and $\pi_j^{t'}$ both output the same message m_B, and in particular they both accepted with the same value of $\mathrm{CTR}_{BA}^{\text{resp}}$. This is impossible since $\mathrm{CTR}_{BA}^{\text{resp}}$ was increased when these oracles accepted and this counter is non-decreasing (see Lemma 18.2). As we have assumed that π_i^s has accepted maliciously, this implies, in view of Definition 9, that there is no π_j^t such that π_i^s has π_j^t as strong partner. Let us show that this leads to a contradiction. Since $\pi_i^s.\alpha = \text{accepted}$, the second message of its transcript must be of the form $m_B = (C, N_B, \sigma_B)$ with $C \in \mathbb{N}$, $N_B \in \mathcal{N}$, $\sigma_B \in \mathcal{T}$ and $\mathrm{Vrfy}_{K_{AB}^{\text{mac}}}((B,C,N_A,N_B),\sigma_B) = 1$ where N_A, stored in $\pi_i^s.\text{st}$, is the nonce from the first message of π_i^s. Since E_{AB} does not occur, there exists an oracle π_j^t that previously produced the tag $\sigma_B \leftarrow \mathrm{Mac}_{K_{BA}^{\text{mac}}}(B,C,N_A,N_B)$ and such that $\pi_j^t.\text{pid} = A$ and $\pi_j^t.\rho = \text{responder}$. Moreover, it must have been initialized with a NewSessionR query with a message m_A of the form $m_A = (C', N_A, \sigma_A)$ with $C' \in \mathbb{N}$, $N_A \in \mathcal{N}$, $\sigma_A \in \mathcal{T}$ and $\mathrm{Vrfy}_{K_{BA}^{\text{mac}}}((A,C',N_A),\sigma_A) = 1$. Since E_{AB} does not occur, there exists an oracle $\pi_i^{s'}$ that previously produced the tag $\sigma_A \leftarrow \mathrm{Mac}_{K_{AB}^{\text{mac}}}(A,C',N_A)$ and such that $\pi_i^{s'}.\text{pid} = B$ and $\pi_i^{s'}.\rho = \text{initiator}$. Moreover, at initialization, it has produced the nonce N_A, which means that $\pi_i^{s'} = \pi_i^s$ (i.e. $s' = s$) by definition of G_1. Therefore, one has $\mathrm{T}_i^s = (m_A, m_B) = \mathrm{T}_j^t$ and π_i^s first produced m_A, then m_A was delivered to π_j^t which produced m_B and accepted and finally, m_B was delivered to π_i^s which accepted. This means that π_i^s has π_j^t as strong partner, which is a contradiction.

Let us conclude with the case $\pi_i^s.\rho = \text{responder}$. By symmetry, we can suppose $P_i = B$ and $\pi_i^s.\text{pid} = P_j = A$. We first remark that it is not possible that there are two distinct π_j^t and $\pi_j^{t'}$ such that π_i^s has both π_j^t and $\pi_j^{t'}$ as partners. Indeed, if this were the case, the first message of π_i^s, π_j^t and $\pi_j^{t'}$ would be the same. In particular, A would have generated as initiator twice the same nonce for B, which would contradicts the definition of G_1. Therefore and as before, in view of Definition 9, we can suppose that there is no π_j^t such that π_i^s has π_j^t as strong partner. Since $\pi_i^s.\alpha = \text{accepted}$, the first message of its transcript must be of the form $m_A = (C, N_A, \sigma_A)$ with $C \in \mathbb{N}$, $N_A \in \mathcal{N}$, $\sigma_A \in \mathcal{T}$ and $\mathrm{Vrfy}_{K_{BA}^{\text{mac}}}((A,C,N_A),\sigma_A) = 1$. Since E_{AB} does not occur, there exists an oracle π_j^t that previously produced the tag $\sigma_A \leftarrow \mathrm{Mac}_{K_{AB}^{\text{mac}}}(A,C,N_A)$ and such that $\pi_j^t.\text{pid} = B$ and $\pi_j^t.\rho = \text{initiator}$. Moreover, at initialization, this oracle π_j^t must have output the message m_A.

Therefore, π_j^t first produced the message m_A and then it was delivered to π_i^s which accepted. This means that π_i^s has π_j^t as strong partner, which is a contradiction. □

Finally, we show the key indistinguishability of LP2+. Note that it does not depend on the collision resistance of nonces in the responder side.

Theorem 24 (Key indistinguishability of LP2+)**.** Let $\mathcal{A}$ be an adversary attacking the key indistinguishability of the protocol LP2+ with M parties. Suppose that, for each ordered pair (A, B) of distinct parties, $\mathcal{A}$ makes at most q queries NewSessionI and q queries NewSessionR to A with intended partner B. Then, there exist adversaries $\mathcal{B}$ and $\mathcal{C}$ attacking the MAC Σ and the pseudo-random function F respectively, such that

$$\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{LP2+}}(\mathcal{A}) \leqslant M(M-1)\cdot\left(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q\cdot\mathrm{Adv}^{\mathrm{SEUF\text{-}CMA}}_{\Sigma}(\mathcal{B}) + 2q\cdot\mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C})\right).$$

Moreover, $\mathcal{B}$ queries at most $4q$ times its oracle $\mathcal{O}_{\mathrm{Mac}}$ and $\mathcal{C}$ queries at most 3 times its oracle $\mathcal{O}_g$. In terms of F, Mac and Vrfy evaluations, $\mathcal{B}$ makes at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries; while $\mathcal{C}$ makes at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries plus $3q$ evaluations of F.

Proof. We proceed via a sequence of reductions using different games.

Game 0. This is the original game $G_0 = \mathrm{KeyInd}_{\mathcal{A},\mathrm{LP2+}}$ as defined in Definition 10. We thus have

$$\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{LP2+}}(\mathcal{A}) = 2\left|\Pr(G_0 = 1) - \frac{1}{2}\right|. \tag{3}$$

Game 1. This game G_1 is the same as G_0 except that, as soon as some party A generates the same nonce N_A twice in two different queries NewSessionI with the same intended partner B, then the experiment aborts and outputs a uniformly chosen value in $\{0, 1\}$. We denote by D the event that such an abortion occurs. For a fixed ordered pair (A, B) of distinct parties, the probability that A, as an initiator, generates twice the same nonce for B is upper bounded by $\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}})$ since A is queried at most q times a NewSessionI query with intended partner B. Therefore,

$$\Pr(D) \leqslant M(M-1)\cdot\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}).$$

We have

$$\begin{aligned}\Pr(G_1 = 1) &= \Pr(G_1 = 1 \mid D)\Pr(D) + \Pr(G_1 = 1 \mid \neg D)\Pr(\neg D)\\ &= \frac{1}{2}\Pr(D) + \Pr(G_0 = 1 \mid \neg D)\Pr(\neg D)\\ &= \frac{1}{2}\Pr(D) + \Pr(G_0 = 1) - \Pr(G_0 = 1 \mid D)\Pr(D)\end{aligned}$$

and thus,

$$\begin{aligned}\left|\Pr(G_0 = 1) - \frac{1}{2}\right| &= \left|\left(\Pr(G_0 = 1 \mid D) - \frac{1}{2}\right)\Pr(D) + \Pr(G_1 = 1) - \frac{1}{2}\right|\\ &\leqslant \frac{1}{2}\Pr(D) + \left|\Pr(G_1 = 1) - \frac{1}{2}\right|.\end{aligned}$$

This yields

$$2\left|\Pr(G_0 = 1) - \frac{1}{2}\right| \leqslant M(M-1)\cdot\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2\left|\Pr(G_1 = 1) - \frac{1}{2}\right|. \tag{4}$$

Game 2. Let E be the event that, at some point in the execution of $\mathcal{A}$,

(a) $\mathcal{A}$ makes a NewSessionR query or a Send query to an oracle π_i^s which is, at the end of this query, two-sided uncorrupted,

(b) this query passes the Vrfy test of the oracle,

(c) and the message-tag pair (m, σ) that passes this Vrfy test has not been previously obtained by an instruction $\sigma \leftarrow \mathrm{Mac}_{(.)}(m)$ of an oracle π_j^t with $P_j = \pi_i^s.\mathrm{pid}$, $\pi_j^t.\mathrm{pid} = P_i$ and $\pi_j^t.\rho \neq \pi_i^s.\rho$.

The game G_2 is the same as G_1 except that, if E occurs, then the experiment aborts and outputs a uniform value in $\{0,1\}$. One has that

$$\begin{aligned}\Pr(G_2 = 1) &= \Pr(G_2 = 1 \,|\, E)\Pr(E) + \Pr(G_2 = 1 \,|\, \neg E)\Pr(\neg E)\\ &= \frac{1}{2}\Pr(E) + \Pr(G_1 = 1 \,|\, \neg E)\Pr(\neg E)\\ &= \frac{1}{2}\Pr(E) + \Pr(G_1 = 1) - \Pr(G_1 = 1 \,|\, E)\Pr(E)\end{aligned}$$

and therefore,

$$\begin{aligned}\left|\Pr(G_1 = 1) - \frac{1}{2}\right| &= \left|\left(\Pr(G_1 = 1 \,|\, E) - \frac{1}{2}\right)\Pr(E) + \Pr(G_2 = 1) - \frac{1}{2}\right|\\ &\leqslant \frac{1}{2}\Pr(E) + \left|\Pr(G_2 = 1) - \frac{1}{2}\right|.\end{aligned}$$

By Corollary 22, we know that there exists an adversary $\mathcal{B}$ of Σ with the complexity as in the statement and such that $\Pr(E_{AB}) \leqslant 4q \cdot \mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B})$ for all unordered pair $\{A, B\}$ of distinct parties where E_{AB} is as in Corollary 22. Since $E = \bigvee_{\{A,B\}} E_{AB}$, we know that

$$\Pr(E) \leqslant 2M(M-1)q \cdot \mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B})$$

and so

$$2\left|\Pr(G_1 = 1) - \frac{1}{2}\right| \leqslant 2M(M-1)q \cdot \mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}) + 2\left|\Pr(G_2 = 1) - \frac{1}{2}\right|. \tag{5}$$

Game 3. This game G_3 is the same as G_2 except that, at the beginning, the challenger guesses an ordered pair (A, B) of distinct parties uniformly and, once $\mathcal{A}$ has concluded, if $\mathcal{A}$ has made its Test query on π_i^s with $(P_i, \pi_i^s.\mathrm{pid}, \pi_i^s.\rho) \notin \{(A, B, \text{initiator}), (B, A, \text{responder})\} = S_{AB}$, then the experiment aborts and outputs a uniform value in $\{0,1\}$ (otherwise, the output of the experiment is as in G_2). Let T be the random variable that takes the value $(P_i, \pi_i^s.\mathrm{pid}, \pi_i^s.\rho)$ if $\mathcal{A}$ makes its Test query on π_i^s (and $\perp$ if $\mathcal{A}$ does not make any Test query). From the description of G_3 and of T, we have

$$\Pr(G_3 = 1 \,|\, T \in S_{AB}) = \Pr(G_2 = 1 \,|\, T \in S_{AB}) = \Pr(G_2 = 1 \,|\, T \neq \perp).$$

By condition 4(a) in Definition 10, one has $\Pr(G_2 = 1 \,|\, T = \perp) = \frac{1}{2}$. One can then compute

$$\begin{aligned}\Pr(G_2 = 1) &= \Pr(G_2 = 1 \,|\, T = \perp)\Pr(T = \perp) + \Pr(G_2 = 1 \,|\, T \neq \perp)\Pr(T \neq \perp)\\ &= \frac{1}{2}(1 - \Pr(T \neq \perp)) + \Pr(G_2 = 1 \,|\, T \neq \perp)\Pr(T \neq \perp)\\ &= \frac{1}{2} + \Pr(T \neq \perp)\left(\Pr(G_2 = 1 \,|\, T \neq \perp) - \frac{1}{2}\right)\end{aligned}$$

and so

$$\begin{aligned}\Pr(G_3 = 1) &= \Pr(G_3 = 1 \,|\, T \notin S_{AB})\Pr(T \notin S_{AB}) + \Pr(G_3 = 1 \,|\, T \in S_{AB})\Pr(T \in S_{AB})\\ &= \frac{1}{2}\Pr(T \notin S_{AB}) + \Pr(G_2 = 1 \,|\, T \in S_{AB})\Pr(T \in S_{AB})\\ &= \frac{1}{2}\left(1 - \Pr(T \in S_{AB})\right) + \Pr(G_2 = 1 \,|\, T \neq \perp)\Pr(T \in S_{AB})\\ &= \frac{1}{2} + \Pr(T \in S_{AB})\left(\Pr(G_2 = 1 \,|\, T \neq \perp) - \frac{1}{2}\right)\\ &= \frac{1}{2} + \Pr(T \in S_{AB} \,|\, T \neq \perp)\Pr(T \neq \perp)\left(\Pr(G_2 = 1 \,|\, T \neq \perp) - \frac{1}{2}\right).\end{aligned}$$

Using $\Pr(T \in S_{AB} \,|\, T \neq \perp) = \frac{1}{M(M-1)}$ and putting the last two results together, we obtain

$$\begin{aligned}\Pr(G_3 = 1) &= \frac{1}{2} + \frac{1}{M(M-1)} \Pr(T \neq \perp) \left(\Pr(G_2 = 1 \,|\, T \neq \perp) - \frac{1}{2} \right) \\ &= \frac{1}{2} + \frac{1}{M(M-1)} \left(\Pr(G_2 = 1) - \frac{1}{2} \right).\end{aligned}$$

This means that

$$2 \left| \Pr(G_2 = 1) - \frac{1}{2} \right| = 2M(M-1) \left| \Pr(G_3 = 1) - \frac{1}{2} \right|. \tag{6}$$

Combining (3), (4), (5) and (6), it remains to prove that $2 \left| \Pr(G_3 = 1) - \frac{1}{2} \right| \leqslant 2q \cdot \mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C})$ for some adversary $\mathcal{C}$ satisfying the conditions in the statement.

Claim 1. At any time in the course of the execution of G_3 and as far as A is not corrupted with respect to B and vice versa, one has $\mathrm{CTR}_{AB}^{\mathrm{init}} \leqslant q$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}} \leqslant q$.

To prove this claim, let us denote by x_{AB} the number (depending on the stage of execution of G_3) of oracles π_j^t with $P_j = B$, $\pi_j^t.\mathrm{pid} = A$, $\pi_j^t.\rho =$ responder and $\pi_j^t.\alpha =$ accepted. Let us prove that, in the course of the execution of G_3, as far as A is not corrupted with respect to B and vice versa, we have at any time

$$\max\{\mathrm{CTR}_{AB}^{\mathrm{init}}, \mathrm{CTR}_{BA}^{\mathrm{resp}}\} \leqslant x_{AB}. \tag{7}$$

At initialization, these numbers are all equal to 0 and thus the inequality holds. A NewSessionI, RevealKey, RevealState, Corrupt or Test query does not modify these numbers, so we only have to consider the effect of NewSessionR and Send queries.

For NewSessionR queries, only those of the form $\mathrm{NewSessionR}(\pi_j^t, A, m_A)$ with $P_j = B$ may have an impact. Moreover, for such a query to have an impact, one must also have $m_A = (C, N_A, \sigma_A)$ for some $C \in \mathbb{N}$, $N_A \in \mathcal{N}$ and $\sigma_A \in \mathcal{T}$ with $\mathrm{Vrfy}_{K_{BA}^{\mathrm{mac}}}((A, C, N_A), \sigma_A) = 1$. During such a query, a new responder oracle of B with intended partner A accepts, therefore x_{AB} is increased by 1. Furthermore, $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ is increased to $\max\{\mathrm{CTR}_{BA}^{\mathrm{resp}}, C\} + 1$ (due to the **while** loop and the last instruction) while $\mathrm{CTR}_{AB}^{\mathrm{init}}$ does not change. Since π_j^t is supposed to be two-sided uncorrupted at the end of this query, and by definition of G_2, we know that there exists an oracle π_i^s that previously produced the tag $\sigma_A \leftarrow \mathrm{Mac}_{K_{AB}^{\mathrm{mac}}}(A, C, N_A)$ and such that $P_i = A$, $\pi_i^s.\mathrm{pid} = B$ and $\pi_i^s.\rho =$ initiator. By Lemma 18.2, this implies that C is at most the value of $\mathrm{CTR}_{AB}^{\mathrm{init}}$ at the time of the considered NewSessionR query. Therefore, the left-hand side of (7) increases by at most 1 with this NewSessionR query, and thus this cannot break the validity of the inequality.

As for Send queries, only those of the form $\mathrm{Send}(\pi_i^s, m_B)$ with $P_i = A$ and $\pi_i^s.\mathrm{pid} = B$ may have an impact. Moreover, for such a query to have an impact, one must also have $m_B = (C', N_B, \sigma_B)$ for some $C' \in \mathbb{N}$, $N_B \in \mathcal{N}$ and $\sigma_B \in \mathcal{T}$ with $\mathrm{Vrfy}_{K_{AB}^{\mathrm{mac}}}((B, C', N_A, N_B), \sigma_B) = 1$ and $C' \geqslant \mathrm{CTR}_{AB}^{\mathrm{init}}$ where N_A, stored in $\pi_i^s.\mathrm{st}$, is the nonce from the first message of π_i^s. During such a query, due to the **while** loop and the last instruction, $\mathrm{CTR}_{AB}^{\mathrm{init}}$ is increased to $C' + 1$ while $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ and x_{AB} do not change. Since π_i^s is supposed to be two-sided uncorrupted at that time and by definition of G_2, we know that there exists an oracle π_j^t that previously produced the tag $\sigma_B \leftarrow \mathrm{Mac}_{K_{BA}^{\mathrm{mac}}}(B, C', N_A, N_B)$ and such that $P_j = B$, $\pi_j^t.\mathrm{pid} = A$ and $\pi_j^t.\rho =$ responder. After this tag is produced, both $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ and x_{AB} increased by 1, so that, by Lemma 18.2, $\mathrm{CTR}_{BA}^{\mathrm{resp}} \geqslant C' + 1$ at the time of the considered Send query. Therefore, the left-hand side of (7) remains unchanged with this Send query, which concludes the proof of (7).

Since one has at any time $x_{AB} \leqslant q$, this concludes the proof of Claim 1.

Let us now construct an adversary $\mathcal{C}'$ attacking the key evolution security of F (see Definition 13) such that

$$2 \left| \Pr(G_3 = 1) - \frac{1}{2} \right| = \mathrm{Adv}_{F,\mathrm{upd}}^{\mathrm{KEvol}}(\mathcal{C}', q). \tag{8}$$

The algorithm $\mathcal{C}'$ starts by uniformly choosing a pair (A, B) of distinct parties. It then initializes the oracles of the parties $\{P_1, \ldots, P_M\}$ as usual and, for each ordered pair (A', B') of distinct parties with $(A', B') \neq (A, B)$, $\mathcal{C}'$ uniformly and independently chooses $K_{A'B'}^{\mathrm{init,kdf}} = K_{B'A'}^{\mathrm{resp,kdf}} \in \mathcal{K}$. Moreover, for each unordered pair $\{A', B'\}$ of distinct parties, $\mathcal{C}'$ runs KeyGen to generate $K_{A'B'}^{\mathrm{mac}} = K_{B'A'}^{\mathrm{mac}} \in \mathcal{K}_{\mathrm{MAC}}$,

initializes the counters $\mathrm{CTR}^{\mathrm{init}}_{A'B'}$, $\mathrm{CTR}^{\mathrm{resp}}_{A'B'}$, $\mathrm{CTR}^{\mathrm{init}}_{B'A'}$ and $\mathrm{CTR}^{\mathrm{resp}}_{B'A'}$ to 0 and selects the initial states $\mathrm{st}^{\mathrm{initN}}_{A'B'}$, $\mathrm{st}^{\mathrm{respN}}_{A'B'}$, $\mathrm{st}^{\mathrm{initN}}_{B'A'}$ and $\mathrm{st}^{\mathrm{respN}}_{B'A'}$ of GenN. The key $K^{\mathrm{init,kdf}}_{AB} = K^{\mathrm{resp,kdf}}_{BA}$ is not generated by $\mathcal{C}'$ but initially corresponds to the key k_0 generated by its challenger. $\mathcal{C}'$ sets $K^{\mathrm{init,kdf}}_{AB} \leftarrow \perp$, $K^{\mathrm{resp,kdf}}_{BA} \leftarrow \perp$, and $k_0 \leftarrow \perp$ to indicate that it does not know their value yet, and initializes $k_1, k_2, \cdots \leftarrow \perp$, which correspond to the updates of k_0 (yet unknown to $\mathcal{C}'$). $\mathcal{C}'$ keeps track of a register SK for keys sk that are accepted but unknown to it. This register is initialized to the empty set. Then $\mathcal{C}'$ runs $\mathcal{A}$ and answers its queries as follows.

- NewSessionI(π^s_i, P_j): $\mathcal{C}'$ answers exactly as prescribed by the protocol. If the nonce generated was already generated during a previous query NewSessionI$(\pi^{s'}_i, P_j)$, then $\mathcal{C}'$ aborts the experiment and outputs a uniform value in $\{0,1\}$.
- NewSessionR(π^s_i, P_j, m): $\mathcal{C}'$ answers exactly as prescribed by the protocol if $(P_i, P_j) \neq (B, A)$. If $(P_i, P_j) = (B, A)$, $\mathcal{C}'$ answers as prescribed by the protocol except that
  - it replaces each occurrence of the pair of instructions
$$K^{\mathrm{resp,kdf}}_{BA} \leftarrow F_{K^{\mathrm{resp,kdf}}_{BA}}(\mathrm{upd})$$
$$\mathrm{CTR}^{\mathrm{resp}}_{BA} \leftarrow \mathrm{CTR}^{\mathrm{resp}}_{BA} + 1$$
    that are present in two places by
$$\mathrm{CTR}^{\mathrm{resp}}_{BA} \leftarrow \mathrm{CTR}^{\mathrm{resp}}_{BA} + 1$$
$$\textbf{if } K^{\mathrm{resp,kdf}}_{BA} \neq \perp$$
$$\quad K^{\mathrm{resp,kdf}}_{BA} \leftarrow F_{K^{\mathrm{resp,kdf}}_{BA}}(\mathrm{upd})$$
$$\textbf{else if } k_{\mathrm{CTR}^{\mathrm{resp}}_{BA}} \neq \perp$$
$$\quad K^{\mathrm{resp,kdf}}_{BA} \leftarrow k_{\mathrm{CTR}^{\mathrm{resp}}_{BA}}$$
  - it replaces the instruction $\mathrm{sk} \leftarrow F_{K^{\mathrm{resp,kdf}}_{BA}}(\mathrm{der}(N_A, N_B))$ by
$$\textbf{if } K^{\mathrm{resp,kdf}}_{BA} \neq \perp$$
$$\quad \mathrm{sk} \leftarrow F_{K^{\mathrm{resp,kdf}}_{BA}}(\mathrm{der}(N_A, N_B))$$
$$\textbf{else}$$
$$\quad \text{save } (i, s, \mathrm{CTR}^{\mathrm{resp}}_{BA}, \mathrm{der}(N_A, N_B)) \text{ in SK}$$
$$\quad \pi^s_i.\alpha \leftarrow \text{accepted.}$$

  In both cases, if π^s_i is two-sided uncorrupted at the end of this query, if the Vrfy test of π^s_i has been passed and if the message-tag pair (m, σ) that passes this Vrfy test has not been previously obtained by an instruction $\sigma \leftarrow \mathrm{Mac}_{(.)}(m)$ of an oracle π^t_j with $\pi^t_j.\mathrm{pid} = P_i$ and $\pi^t_j.\rho = \text{initiator}$, then $\mathcal{C}'$ aborts the experiment and outputs a uniform value in $\{0,1\}$.
- Send(π^s_i, m): $\mathcal{C}'$ answers exactly as prescribed by the protocol if $(P_i, \pi^s_i.\mathrm{pid}) \neq (A, B)$. If $(P_i, \pi^s_i.\mathrm{pid}) = (A, B)$, then $\mathcal{C}'$ answers as prescribed by the protocol except that
  - it replaces each occurrence of the pair of instructions
$$K^{\mathrm{init,kdf}}_{AB} \leftarrow F_{K^{\mathrm{init,kdf}}_{AB}}(\mathrm{upd})$$
$$\mathrm{CTR}^{\mathrm{init}}_{AB} \leftarrow \mathrm{CTR}^{\mathrm{init}}_{AB} + 1$$
    that are present in two places by
$$\mathrm{CTR}^{\mathrm{init}}_{AB} \leftarrow \mathrm{CTR}^{\mathrm{init}}_{AB} + 1$$
$$\textbf{if } K^{\mathrm{init,kdf}}_{AB} \neq \perp$$
$$\quad K^{\mathrm{init,kdf}}_{AB} \leftarrow F_{K^{\mathrm{init,kdf}}_{AB}}(\mathrm{upd})$$
$$\textbf{else if } k_{\mathrm{CTR}^{\mathrm{init}}_{AB}} \neq \perp$$
$$\quad K^{\mathrm{init,kdf}}_{AB} \leftarrow k_{\mathrm{CTR}^{\mathrm{init}}_{AB}}$$

– it replaces the instruction $\mathrm{sk} \leftarrow F_{K_{AB}^{\mathrm{init,kdf}}}(\mathrm{der}(N_A, N_B))$ by

if $K_{AB}^{\mathrm{init,kdf}} \neq \perp$
  $\mathrm{sk} \leftarrow F_{K_{AB}^{\mathrm{init,kdf}}}(\mathrm{der}(N_A, N_B))$
else
  save $(i, s, \mathrm{CTR}_{AB}^{\mathrm{init}}, \mathrm{der}(N_A, N_B))$ in SK
  $\pi_i^s.\alpha \leftarrow$ accepted.

In both cases, if π_i^s is two-sided uncorrupted at the end of this query, if the Vrfy test of π_i^s has been passed and if the message-tag pair (m, σ) that passes this Vrfy test has not been previously obtained by an instruction $\sigma \leftarrow \mathrm{Mac}_{(\cdot)}(m)$ of an oracle π_j^t with $P_j = \pi_i^s.\mathrm{pid}$, $\pi_j^t.\mathrm{pid} = P_i$ and $\pi_j^t.\rho =$ responder, then $\mathcal{C}'$ aborts the experiment and outputs a uniform value in $\{0,1\}$.

- RevealKey(π_i^s): If $\mathcal{C}'$ has $\pi_i^s.\mathrm{sk} \neq \perp$, it simply returns it to $\mathcal{A}$ and stops its answer there. Otherwise, $\mathcal{C}'$ looks for the unique entry of the form (i, s, C, x) in SK. If $\mathcal{C}'$ has already made a $\mathcal{O}_{\mathrm{Test}}(C, y)$ query for some y, it aborts the experiment and outputs a uniform value in $\{0,1\}$. Otherwise, it queries its $\mathcal{O}_{\mathrm{Reveal}}$ oracle on input (C, x) to get $\mathrm{sk}_{C,x}$, sets $\pi_i^s.\mathrm{sk} \leftarrow \mathrm{sk}_{C,x}$, returns it to $\mathcal{A}$ and deletes the entry (i, s, C, x) of SK.
- RevealState(π_i^s): $\mathcal{C}'$ always knows $\pi_i^s.\mathrm{st}$ so it simply returns it to $\mathcal{A}$.
- Corrupt(P_i, P_j): If $\{P_i, P_j\} \neq \{A, B\}$ or if $(P_i, P_j) = (A, B)$ and $\mathcal{C}'$ has $K_{AB}^{\mathrm{init,kdf}} \neq \perp$ or if $(P_i, P_j) = (B, A)$ and $\mathcal{C}'$ has $K_{BA}^{\mathrm{resp,kdf}} \neq \perp$, then $\mathcal{C}'$ knows MK_{ij}, it returns it to $\mathcal{A}$ and stops its answer there. Otherwise, if $(P_i, P_j) = (A, B)$ and $\mathcal{C}'$ still has $K_{AB}^{\mathrm{init,kdf}} = \perp$, let $C = \mathrm{CTR}_{AB}^{\mathrm{init}}$; while if $(P_i, P_j) = (B, A)$ and $\mathcal{C}'$ still has $K_{BA}^{\mathrm{resp,kdf}} = \perp$, let $C = \mathrm{CTR}_{BA}^{\mathrm{resp}}$. If $\mathcal{C}'$ has already made a query $\mathcal{O}_{\mathrm{Test}}(C', x)$ with $C' \geqslant C$, it aborts the experiment and outputs a uniform value in $\{0,1\}$. Otherwise, $\mathcal{C}'$ queries $\mathcal{O}_{\mathrm{Corrupt}}(C)$ to get k_C and it inductively sets $k_{u+1} \leftarrow F_{k_u}(\mathrm{upd})$ for each $C \leqslant u < \max\{\mathrm{CTR}_{AB}^{\mathrm{init}}, \mathrm{CTR}_{BA}^{\mathrm{resp}}\}$. If $\mathrm{CTR}_{AB}^{\mathrm{init}} \geqslant C$ and $K_{AB}^{\mathrm{init,kdf}} = \perp$, it sets $K_{AB}^{\mathrm{init,kdf}} \leftarrow k_{\mathrm{CTR}_{AB}^{\mathrm{init}}}$; and if $\mathrm{CTR}_{BA}^{\mathrm{resp}} \geqslant C$ and $K_{BA}^{\mathrm{resp,kdf}} = \perp$, it sets $K_{BA}^{\mathrm{resp,kdf}} \leftarrow k_{\mathrm{CTR}_{BA}^{\mathrm{resp}}}$. Thus, $\mathcal{C}'$ now knows MK_{ij} and returns it to $\mathcal{A}$.
- Test(π_i^s): If $\mathcal{C}'$ has $\pi_i^s.\mathrm{sk} \neq \perp$, it aborts the experiment and outputs a uniform value in $\{0,1\}$. Otherwise, $\mathcal{C}'$ looks for the unique entry of the form (i, s, C, x) in its register SK. If $\mathcal{C}'$ has already made a $\mathcal{O}_{\mathrm{Reveal}}(C, y)$ query for some y or a $\mathcal{O}_{\mathrm{Corrupt}}(C')$ query for some $C' \leqslant C$, it aborts the experiment and outputs a uniform value in $\{0,1\}$. Otherwise, $\mathcal{C}'$ queries its $\mathcal{O}_{\mathrm{Test}}$ oracle on input (C, x) to obtain $\mathrm{sk}_{C,x}$ and returns it to $\mathcal{A}$. (Notice that $\mathcal{C}'$ does not choose the test bit b_{test} but considers it as the bit hidden by its challenger).

Once $\mathcal{A}$ has concluded, if any of the conditions of point 4 of Definition 10 is not satisfied, $\mathcal{C}'$ aborts the experiment and outputs a uniform value in $\{0,1\}$. Otherwise, $\mathcal{C}'$ outputs the same bit b' as $\mathcal{A}$.

Let us make four remarks showing that $\mathcal{C}'$ is well-defined and that, while running, it perfectly simulates an execution of $\mathcal{A}$ in G_3.

First, in view of Lemma 18.3 and the way the k_C's are computed, one has for each entry (i, s, C, x) of the register SK that $\pi_i^s.\mathrm{sk} = F_{k_C}(x)$ (although $\mathcal{C}'$ cannot compute it immediately).

Second, from the construction of the register SK, for each π_i^s, there can be at most one entry of the form (i, s, C, x). Furthermore, such an entry exists if and only if π_i^s has accepted but $\mathcal{C}'$ still has $\pi_i^s.\mathrm{sk} = \perp$. Therefore, the condition that π_i^s has accepted before $\mathcal{A}$ can make a Test or RevealKey ensures that the entry (i, s, C, x) exists in SK when $\mathcal{C}'$ looks for it.

Third, let us observe that the inputs of the oracles of $\mathcal{C}'$ satisfy their respective domain restrictions, as given in Definition 13. As long as A is not corrupted with respect to B and vice versa, one has $k_i = \perp$ for all i, $K_{AB}^{\mathrm{init,kdf}} = \perp$ and $K_{BA}^{\mathrm{resp,kdf}} = \perp$. Moreover, in that case, in view of Claim 1, one has $\mathrm{CTR}_{AB}^{\mathrm{init}} \leqslant q$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}} \leqslant q$. Therefore, when A gets corrupted with respect to B or vice versa for the first time, if $\mathcal{C}'$ does not abort, it gets k_C for $C \leqslant q$. Hence, after this corruption, since $\mathrm{CTR}_{AB}^{\mathrm{init}}$ increases one by one, $\mathcal{C}'$ has $K_{AB}^{\mathrm{init,kdf}} \neq \perp$ at the latest when $\mathrm{CTR}_{AB}^{\mathrm{init}} = C \leqslant q$ (the assignment to $K_{AB}^{\mathrm{init,kdf}}$ of a value different from $\perp$ may occur in a Send or Corrupt query). Therefore, in any case,

if $K_{AB}^{\text{init,kdf}} = \perp$, one has $\text{CTR}_{AB}^{\text{init}} \leqslant q$ and analogously, if $K_{BA}^{\text{resp,kdf}} = \perp$, one has $\text{CTR}_{BA}^{\text{resp}} \leqslant q$. Since the corresponding counter is increased one more time after saving an entry in the register SK, this means that, for each entry (i, s, C, x) in SK, one has $0 \leqslant C < q$ (and $x \in \mathcal{D} \setminus \{\text{upd}\}$, by definition of the function der). Moreover, this also means that, if $\mathcal{C}'$ queries $\mathcal{O}_{\text{Corrupt}}(C)$ in a Corrupt query, then $0 \leqslant C \leqslant q$.

Fourth, we show that the constraints on the number and dependencies of calls given in Definition 13 to the oracles are fulfilled:

- $\mathcal{C}'$ queries at most once its oracle $\mathcal{O}_{\text{Test}}$ since $\mathcal{A}$ makes at most one Test query.
- By the way RevealKey and Test queries are answered, $\mathcal{C}'$ does not make both a query $\mathcal{O}_{\text{Reveal}}(C, x)$ and a query $\mathcal{O}_{\text{Test}}(C, y)$.
- By the way Corrupt and Test queries are answered, $\mathcal{C}'$ does not make both a query $\mathcal{O}_{\text{Corrupt}}(C')$ and a query $\mathcal{O}_{\text{Test}}(C, x)$ with $C' \leqslant C$.
- $\mathcal{C}'$ does not make three queries $\mathcal{O}_{\text{Reveal}}(C, x)$, $\mathcal{O}_{\text{Reveal}}(C, y)$ and $\mathcal{O}_{\text{Reveal}}(C, z)$. Indeed, if it were the case, these queries would correspond to $\mathcal{A}$'s queries RevealKey(π_i^s), RevealKey(π_j^t) and RevealKey(π_ℓ^u) and respectively to entries (i, s, C, x), (j, t, C, y) and (ℓ, u, C, z) of the register SK. Two of these three oracles π_i^s, π_j^t and π_ℓ^u would have the same role, say $\pi_i^s.\rho = \pi_j^t.\rho$ without loss of generality. In view of how the register SK is filled in, we know that $(P_i, \pi_i^s.\text{pid}, \pi_i^s.\rho) = (P_j, \pi_j^t.\text{pid}, \pi_j^t.\rho) \in \{(A, B, \text{initiator}), (B, A, \text{responder})\}$. If $\pi_i^s = \pi_j^t$, $\mathcal{C}'$ would not have queried its $\mathcal{O}_{\text{Reveal}}$ oracle for the second RevealKey query to the same oracle. If $\pi_i^s \neq \pi_j^t$, these oracles would not have accepted with the same counter value C (due to the non-decreasing nature of the counter, see Lemma 18.2), reaching a contradiction.

Let Z be the event that, in the course of an execution of $\mathcal{A}$ in G_3, the experiment aborts (in which case, it outputs a uniform value in $\{0, 1\}$). This event can be caused by any of the reasons mentioned in G_1, G_2 or G_3 or because one of the conditions mentioned in point 4 of Definition 10 is not satisfied. Let Z' be the event that, in the course of an experiment $\text{KEvol}_{\mathcal{C}', F, \text{upd}, q}$, the experiment aborts (and outputs a uniform value in $\{0, 1\}$). We are going to prove that Z' occurs if and only if Z occurs during the simulation of G_3 by $\mathcal{C}'$. This would imply that $\Pr(Z) = \Pr(Z')$ and $\Pr(G_3 = 1 \,|\, \neg Z) = \Pr(\text{KEvol}_{\mathcal{C}', F, \text{upd}, q} = 1 \,|\, \neg Z')$. It is obvious that $\Pr(G_3 = 1 \,|\, Z) = \frac{1}{2}$ and $\Pr(\text{KEvol}_{\mathcal{C}', F, \text{upd}, q} = 1 \,|\, Z') = \frac{1}{2}$. Therefore, one would have

$$\begin{aligned}
\Pr(G_3 = 1) &= \Pr(G_3 = 1 \,|\, Z) \Pr(Z) + \Pr(G_3 = 1 \,|\, \neg Z) \Pr(\neg Z) \\
&= \Pr(\text{KEvol}_{\mathcal{C}', F, \text{upd}, q} = 1 \,|\, Z') \Pr(Z') + \Pr(\text{KEvol}_{\mathcal{C}', F, \text{upd}, q} = 1 \,|\, \neg Z') \Pr(\neg Z') \\
&= \Pr(\text{KEvol}_{\mathcal{C}', F, \text{upd}, q} = 1)
\end{aligned}$$

and (8) would follow.

If Z occurs because of the new rule of G_1, then Z' occurs in view of the way NewSessionI queries are answered. If Z occurs because of the new rule of G_2, then Z' occurs based on the way NewSessionR and Send queries are answered. If Z occurs because of the new rule of G_3, i.e., if $\mathcal{A}$ has made its Test query on π_i^s with $(P_i, \pi_i^s.\text{pid}, \pi_i^s.\rho) \notin \{(A, B, \text{initiator}), (B, A, \text{responder})\}$, this means that the corresponding keys $K_{A'B'}^{\text{init,kdf}}$ and $K_{B'A'}^{\text{resp,kdf}}$ are set from the beginning. Thus $\pi_i^s.\text{sk} \neq \perp$ and Z' occurs because of the way Test queries are answered. Finally, if Z occurs because any of the conditions mentioned in point 4 of Definition 10 is not satisfied, then Z' occurs by the way $\mathcal{C}'$ acts once $\mathcal{A}$ has concluded.

Let us now prove that, if Z' occurs, then Z occurs in the simulation of G_3 by $\mathcal{C}'$. We suppose by contradiction that Z does not occur. If Z' occurs by the way $\mathcal{C}'$ acts once $\mathcal{A}$ has concluded, then one has a contradiction because one of the conditions mentioned in point 4 of Definition 10 is not satisfied. Let us now treat the case where Z' occurs by the way $\mathcal{C}'$ answers $\mathcal{A}$'s queries. By the rule of G_1, Z' cannot occur in a NewSessionI query. By the rule of G_2, Z' cannot occur in a NewSessionR or Send query. Before proving the other cases, let us prove the following two claims (still supposing that Z does not occur).

Claim 2. If π_i^s was a two-sided uncorrupted oracle when it accepted, then there is a unique π_j^t such that π_i^s has π_j^t as strong partner.

The proof of this claim is the same as the final part of the proof of Theorem 23 and we omit it here for conciseness. It relies on the rules of G_1 and G_2.

Claim 3. If (i, s, C, x) and (j, t, C, y) are in the register SK (with the same counter value C), if $\pi_i^s \neq \pi_j^t$, and if a Test query has been issued to π_i^s or π_j^t, then π_i^s and π_j^t are mutual partners.

In view of how SK is filled in, one has $(P_i, \pi_i^s.\text{pid}, \pi_i^s.\rho) \in \{(A, B, \text{initiator}), (B, A, \text{responder})\}$ and similarly for π_j^t. By Lemma 18.2, since both π_i^s and π_j^t accepted with counter value C, one has $\pi_i^s.\rho \neq \pi_j^t.\rho$. Without loss of generality, we can suppose that $(P_i, \pi_i^s.\text{pid}, \pi_i^s.\rho) = (A, B, \text{initiator})$ and $(P_j, \pi_j^t.\text{pid}, \pi_j^t.\rho) = (B, A, \text{responder})$. Let us first assume that π_i^s was not two-sided uncorrupted when it accepted. By condition 4(b) of Definition 10, this implies $\text{Test}(\pi_i^s)$ was not queried, and thus $\text{Test}(\pi_j^t)$ was queried. Thus, by the same condition, π_j^t was two-sided uncorrupted when it accepted. By Claim 2, there exists an oracle $\pi_i^{s'}$ such that π_j^t has $\pi_i^{s'}$ as strong partner. By condition 4(c) of Definition 10, one has $\text{T}_i^{s'} = \text{T}_j^t$ before the corruption occurred between A and B (which occurred before π_i^s accepted by assumption). If $C \geqslant \text{CTR}_{AB}^{\text{init}}$ when $\pi_i^{s'}$ received its second message (faithfully delivered from π_j^t), it accepted with counter value C, which is impossible since π_i^s accepted later with the same counter value. If $C < \text{CTR}_{AB}^{\text{init}}$ when $\pi_i^{s'}$ received its second message, π_i^s could not have accepted with counter value C, reaching a contradiction. Therefore, π_i^s was two-sided uncorrupted when it accepted. By Claim 2, there is a unique accepting oracle $\pi_j^{t'}$ such that π_i^s has $\pi_j^{t'}$ as strong partner. Since they both share the same second message in their transcripts, $\pi_j^{t'}$ also accepted when $\text{CTR}_{BA}^{\text{resp}} = C$. Since this counter is non-decreasing, this implies that $\pi_j^{t'} = \pi_j^t$ and so π_i^s and π_j^t are mutual partners, concluding the proof of Claim 3.

Let us now consider the case where Z' occurs during a RevealKey, Corrupt, or Test query (still supposing that Z does not occur).

First, we analyze the case where Z' occurs because $\mathcal{C}'$ tries to make queries $\mathcal{O}_{\text{Test}}(C, x)$ and $\mathcal{O}_{\text{Reveal}}(C, y)$. This may happen during a RevealKey or a Test query. $\mathcal{C}'$ queries correspond to entries (i, s, C, x) and (j, t, C, y) of the register SK and $\mathcal{A}$ made, in some order, a $\text{Test}(\pi_i^s)$ query and a $\text{RevealKey}(\pi_j^t)$ query. By condition 4(d) of Definition 10, we know that $\pi_i^s \neq \pi_j^t$. By Claim 3, π_i^s and π_j^t are mutual partners, which contradicts condition 4(e) of Definition 10.

We now treat the case where Z' occurs for the first reason stated in the description of a $\text{Test}(\pi_i^s)$ query, that is, $\mathcal{C}'$ has $\pi_i^s.\text{sk} \neq \perp$ at the time of the Test query. By the rule of G_3, we know that $(P_i, \pi_i^s.\text{pid}, \pi_i^s.\rho) \in \{(A, B, \text{initiator}), (B, A, \text{responder})\}$. By condition 4(b) in Definition 10, we also know that π_i^s was two-sided uncorrupted when it accepted. Therefore, $K_{AB}^{\text{init,kdf}} = \perp$ and $K_{BA}^{\text{resp,kdf}} = \perp$ when π_i^s accepted. Thus, the only way one could have $\pi_i^s.\text{sk} \neq \perp$ when the Test query is made is if a $\text{RevealKey}(\pi_i^s)$ query was made before. But this is impossible by condition 4(d) of Definition 10.

It remains to treat the case where Z' occurs because $\mathcal{C}'$ tries to make queries $\mathcal{O}_{\text{Test}}(C, x)$ and $\mathcal{O}_{\text{Corrupt}}(C')$ with $C' \leqslant C$. This $\mathcal{O}_{\text{Corrupt}}$ query corresponds to a $\text{Corrupt}(A, B)$ (in which case $C' = \text{CTR}_{AB}^{\text{init}}$ at that time) or to a $\text{Corrupt}(B, A)$ (in which case $C' = \text{CTR}_{BA}^{\text{resp}}$ at that time). The $\mathcal{O}_{\text{Test}}$ query corresponds to a $\text{Test}(\pi_i^s)$ query and to an entry (i, s, C, x) of the register SK. By condition 4(b) of Definition 10, π_i^s was two-sided uncorrupted when it accepted. By Claim 2, there exists π_j^t such that π_i^s has π_j^t as strong partner. If $\pi_i^s.\rho = \text{initiator}$, then $\text{T}_i^s = \text{T}_j^t$ when π_i^s accepted and both oracles accepted before the Corrupt query, with the same counter value C. After π_i^s accepted, one has thus $\text{CTR}_{AB}^{\text{init}} > C \geqslant C'$ and $\text{CTR}_{BA}^{\text{resp}} > C \geqslant C'$, which is in contradiction with the definition of C'. If $\pi_i^s.\rho = \text{responder}$, by condition 4(c) of Definition 10, one has $\text{T}_i^s = \text{T}_j^t$ before the corruption. Moreover, once π_i^s has accepted, thus before the corruption, one has $\text{CTR}_{BA}^{\text{resp}} > C \geqslant C'$. If π_j^t accepted, it is thus with counter value C and one also has $\text{CTR}_{AB}^{\text{init}} > C \geqslant C'$ at the time of corruption, which is a contradiction. Since the deliverance of the second message of $\text{T}_i^s = \text{T}_j^t$ was faithful and both oracles used the same nonces, the only reason π_j^t could have rejected, is because $\text{CTR}_{AB}^{\text{init}} > C \geqslant C'$ at the time π_j^t rejected. This is again a contradiction, which concludes the proof of (8).

Let us notice that, up to ask $\mathcal{C}'$ to save its F computations in order to make at most once an evaluation of F for a fixed input, $\mathcal{C}'$ makes at most the F, Mac and Vrfy evaluations made by $\mathcal{A}$ plus those needed to answer its queries. Indeed, the F evaluations made to answer a Corrupt query would have been made earlier in a normal execution of $\mathcal{A}$ when the challenger knows the keys $K_{AB}^{\text{init,kdf}}$ and $K_{BA}^{\text{resp,kdf}}$.

By Lemma 14, there exists an adversary $\mathcal{C}$ attacking the pseudo-randomness of F, with the

complexity as in the statement, such that

$$\mathrm{Adv}^{\mathrm{KEvol}}_{F,\mathrm{upd}}(\mathcal{C}', q) \leqslant 2q \cdot \mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C}). \tag{9}$$

Putting everything together, we get

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{LP2+}}(\mathcal{A}) \\
&= 2\left|\Pr(G_0 = 1) - \frac{1}{2}\right| && (3)\\
&\leqslant M(M-1) \cdot \mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2\left|\Pr(G_1 = 1) - \frac{1}{2}\right| && (4)\\
&\leqslant M(M-1) \cdot \left(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q \cdot \mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B})\right) + 2\left|\Pr(G_2 = 1) - \frac{1}{2}\right| && (5)\\
&= M(M-1) \cdot \left(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q \cdot \mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}) + 2\left|\Pr(G_3 = 1) - \frac{1}{2}\right|\right) && (6)\\
&= M(M-1) \cdot \left(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q \cdot \mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}) + \mathrm{Adv}^{\mathrm{KEvol}}_{F,\mathrm{upd}}(\mathcal{C}', q)\right) && (8)\\
&\leqslant M(M-1) \cdot \left(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q \cdot \mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}) + 2q \cdot \mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C})\right) && (9)
\end{aligned}$$

as desired. □

4.4 Instantiation only based on a PRF

We conclude this paper with an instantiation of LP2+ whose security is only based on a PRF. Let n be an even positive integer and let $\mathbb{F}_2^n$ be the n-dimensional vector space over the two-element field $\mathbb{F}_2$ (elements of $\mathbb{F}_2^n$ are thus represented by n-bit strings). Let also $F\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a pseudo-random function. Our instantiation LP2+(n, F) of LP2+ is described as follows.

We require that the number M of parties is such that $2 \leqslant M \leqslant 2^{\frac{n}{2}-1}$. Each party P_i will be represented by an element of $\mathbb{F}_2^{\frac{n}{2}-1}$. Moreover, we represent the counters $\mathrm{CTR}^{\mathrm{init}}_{AB}$ and $\mathrm{CTR}^{\mathrm{resp}}_{BA}$ by elements of $\mathbb{F}_2^{\frac{n}{2}}$. If, at the end of an accepting protocol step, one of theses counters reaches $1^{\frac{n}{2}}$, one must renew the master key generation and clear all sessions between these parties. We also renew the master key generation (and clear the sessions) if one of these counters cycles and reaches $0^{\frac{n}{2}}$, to avoid the the case where a message with counter value $1^{\frac{n}{2}}$ is sent (e.g. when parties are corrupted).

To give an order of magnitude, in the case no attack is successful, if n is 128 and if an adversary makes one billion NewSessionR(π_i^s, P_j, m) queries per second, more than 500 years are necessary to make the counter reach $1^{\frac{n}{2}}$.

The nonce generator GenN is a (stateless) uniform selection of an element in $\mathcal{N} = \mathbb{F}_2^n$. We denote such a selection by $N \xleftarrow{\mathrm{U}} \mathbb{F}_2^n$. For the constant $\mathrm{upd} \in \mathbb{F}_2^n$, we simply use $\mathrm{upd} = 0^n$. In order to reduce key control by the parties and to improve key freshness, we choose the function $\mathrm{der}\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \setminus \{0^n\}$ as

$$\mathrm{der}(N_A, N_B) = \begin{cases} [N_A]_{\frac{n}{2}} \| [N_B]_{\frac{n}{2}} & \text{if } [N_A]_{\frac{n}{2}} \| [N_B]_{\frac{n}{2}} \neq 0^n \\ 1^n & \text{if } [N_A]_{\frac{n}{2}} \| [N_B]_{\frac{n}{2}} = 0^n \end{cases}$$

where $N \| N'$ denotes the concatenation of the strings N and N', and $[N]_{\frac{n}{2}} \in \mathbb{F}_2^{\frac{n}{2}}$ denotes the first $\frac{n}{2}$ bits of a nonce $N \in \mathbb{F}_2^n$.

It remains to describe the MAC Σ. Since one needs to sign messages of the form $(A, \mathrm{CTR}^{\mathrm{init}}_{AB}, N_A)$ and $(B, \mathrm{CTR}^{\mathrm{resp}}_{BA}, N_A, N_B)$ which are respectively represented by strings of $2n-1$ and $3n-1$ bits, we take the message space as $\mathcal{M} = \mathbb{F}_2^{2n-1} \cup \mathbb{F}_2^{3n-1}$. We then instantiate the scheme Σ as the deterministic CBC-MAC using F and taking care not to use this authentication mode with a message and one of its prefixes. More precisely, we take KeyGen as the uniform selection of a key in $\mathcal{K}_{\mathrm{MAC}} = \mathbb{F}_2^n$, and, for $k, m_2, m_3 \in \mathbb{F}_2^n$ and $m_1 \in \mathbb{F}_2^{n-1}$, we set

$$\mathrm{Mac}_k(m_1 \| m_2) = F_k(m_2 \oplus F_k(0 \| m_1))$$

and

$$\mathrm{Mac}_k(m_1\|m_2\|m_3) = F_k(m_3 \oplus F_k(m_2 \oplus F_k(1\|m_1)))$$

where $\oplus$ is the component-wise XOR (i.e., the addition in $\mathbb{F}_2^n$) and the tag space is $\mathcal{T} = \mathbb{F}_2^n$. The verification is done in the canonical way, i.e., for $k, m_2, m_3, t \in \mathbb{F}_2^n$ and $m_1 \in \mathbb{F}_2^{n-1}$:

$$\mathrm{Vrfy}_k(m_1\|m_2, t) = 1 \quad \Longleftrightarrow \quad t = F_k(m_2 \oplus F_k(0\|m_1))$$

and

$$\mathrm{Vrfy}_k(m_1\|m_2\|m_3, t) = 1 \quad \Longleftrightarrow \quad t = F_k(m_3 \oplus F_k(m_2 \oplus F_k(1\|m_1))).$$

The security of CBC-MAC for prefix-free sets of arbitrary-length messages has been, to our knowledge, first mentioned in [18]. We borrow the following theorem from the presentation of [15].

Theorem 25. Let n and q be positive integers, $F\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a function and Σ be the CBC-MAC on $\mathcal{M} = \mathbb{F}_2^{2n-1} \cup \mathbb{F}_2^{3n-1}$ as described above. Let also $\mathcal{B}$ be an adversary attacking Σ in the experiment $\text{SEUF-CMA}_{\mathcal{B},\Sigma}$ making at most q queries to its $\mathcal{O}_{\mathrm{Mac}}$ oracle. There exists an adversary $\mathcal{C}$ attacking the pseudo-randomness of F such that

$$\mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}) \leqslant \mathrm{Adv}_{F}^{\mathrm{PRF}}(\mathcal{C}) + \frac{7q^2 + 7q + 1}{2^n}.$$

Moreover, $\mathcal{C}$ queries at most $3(q+1)$ times its oracle $\mathcal{O}_g$ and makes as many evaluations of F as $\mathcal{B}$ does.

Proof. By prepending a 0 before the $2n-1$ bit messages and a 1 before the $3n-1$ bit messages in $\mathcal{M}$, the CBC-MAC is used here on a prefix-free set of messages. The result then follows from (the proof of) Theorems 4.6 and 4.11 in [15]. Note that, analyzing the proofs in details and borrowing the notation from them, one first constructs a distinguisher $\mathcal{D}$ for the CBC function that queries at most $q+1$ times its oracle and such that

$$\mathrm{Adv}_{\Sigma}^{\text{SEUF-CMA}}(\mathcal{B}) \leqslant \left|\Pr\left(1 \leftarrow \mathcal{D}^{\mathrm{CBC}_{F_k}(\cdot)}\right) - \Pr\left(1 \leftarrow \mathcal{D}^{f(\cdot)}\right)\right| + \frac{1}{2^n}$$

where k is uniformly chosen in $\mathbb{F}_2^n$ and f is a uniformly chosen function $\mathcal{M} \to \mathbb{F}_2^n$. Then, one can prove that the CBC function is $(q+1, 3, \delta)$-smooth where $\delta = \frac{7q(q+1)}{2^n}$ (using $q > 0$). From this, there exists $\mathcal{C}$ as in the statement such that

$$\left|\Pr\left(1 \leftarrow \mathcal{D}^{\mathrm{CBC}_{F_k}(\cdot)}\right) - \Pr\left(1 \leftarrow \mathcal{D}^{f(\cdot)}\right)\right| \leqslant \mathrm{Adv}_{F}^{\mathrm{PRF}}(\mathcal{C}) + \delta$$

concluding the proof. □

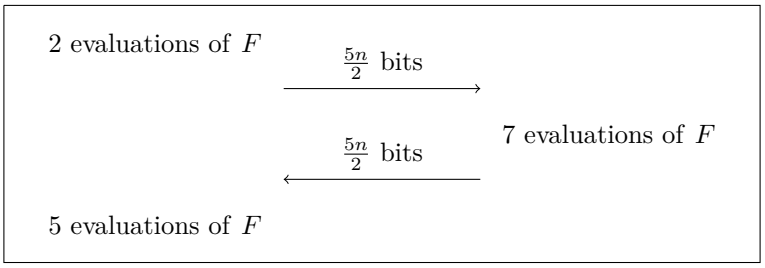
The protocol LP2+(n, F) is initialized as follows. For each unordered pair of distinct parties $\{A, B\} \subseteq \{P_1, \ldots, P_M\}$, one uniformly and independently chooses $K_{AB}^{\mathrm{init,kdf}} = K_{BA}^{\mathrm{resp,kdf}} \xleftarrow{\mathrm{U}} \mathbb{F}_2^n$, $K_{BA}^{\mathrm{init,kdf}} = K_{AB}^{\mathrm{resp,kdf}} \xleftarrow{\mathrm{U}} \mathbb{F}_2^n$ and $K_{AB}^{\mathrm{mac}} = K_{BA}^{\mathrm{mac}} \xleftarrow{\mathrm{U}} \mathbb{F}_2^n$. One also initializes to $0^{\frac{n}{2}}$ four counters $\mathrm{CTR}_{AB}^{\mathrm{init}}$, $\mathrm{CTR}_{AB}^{\mathrm{resp}}$, $\mathrm{CTR}_{BA}^{\mathrm{init}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ in $\mathbb{F}_2^{\frac{n}{2}}$. The master keys on both sides are initially

$$\mathrm{MK}_{AB} = \left(K_{AB}^{\mathrm{init,kdf}}, \mathrm{CTR}_{AB}^{\mathrm{init}}, K_{AB}^{\mathrm{resp,kdf}}, \mathrm{CTR}_{AB}^{\mathrm{resp}}, K_{AB}^{\mathrm{mac}}\right) \in \mathbb{F}_2^{4n}$$

and

$$\mathrm{MK}_{BA} = \left(K_{BA}^{\mathrm{init,kdf}}, \mathrm{CTR}_{BA}^{\mathrm{init}}, K_{BA}^{\mathrm{resp,kdf}}, \mathrm{CTR}_{BA}^{\mathrm{resp}}, K_{BA}^{\mathrm{mac}}\right) \in \mathbb{F}_2^{4n}.$$

The protocol LP2+(n, F) is described in Figure 5. It thus consists of two messages of $\frac{5n}{2}$ bits each. If the parties start synchronized (i.e., $\mathrm{CTR}_{AB}^{\mathrm{init}} = \mathrm{CTR}_{BA}^{\mathrm{resp}}$) and if an uninterrupted honest run of the protocol is executed between Alice and Bob, Alice's first step requires 2 evaluations of F, Bob's step requires 7 evaluations of F and Alice's last step requires 5 evaluations of F, for a total of 14 evaluations of F.



We conclude with a summary of the security theorems of LP2+(n, F).

Theorem 26. Let $n, M, q \in \mathbb{N}$ be integers such that $n > 0$ is even and $2 \leqslant M \leqslant 2^{\frac{n}{2}-1}$. Let also $F\colon \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a function. Let $\mathcal{A}$ be an adversary attacking the protocol LP2+(n, F) with M parties. Suppose that, for each ordered pair (A, B) of distinct parties, $\mathcal{A}$ makes at most q queries NewSessionI and q queries NewSessionR to A with intended partner B. Then, there exists an adversary $\mathcal{C}$ attacking the pseudo-randomness of F such that

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{Corr}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) = 0\\
&\mathrm{Adv}^{\mathrm{wSR}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) = 0\\
&\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) \leqslant M(M-1)\cdot\left(2q\cdot\mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C}) + \frac{283q^3}{2^n}\right)\\
&\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) \leqslant M(M-1)\cdot\left(4q\cdot\mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C}) + \frac{283q^3}{2^n}\right).
\end{aligned}$$

Moreover, $\mathcal{C}$ queries at most $12q + 3$ times its oracle $\mathcal{O}_g$. In terms of F evaluations, $\mathcal{C}$ makes at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries plus $3q$.

Proof. The first two equalities follow respectively from Theorems 19 and 20. For the last two inequalities, if $q = 0$ or $q \geqslant 2^{\frac{n}{3}-2}$, the theorem is vacuous since no oracle π_i^s can accept in the first case and the upper bounds are larger than 1 in the second case. So we can suppose that $0 < q < 2^{\frac{n}{3}-2} < 2^{\frac{n}{2}} - 1$ and so the addition of the re-initialization of the master keys when a counter reaches $0^{\frac{n}{2}}$ or $1^{\frac{n}{2}}$ has no influence here, provided no corruption or tag forgery occurred. By Theorems 23 and 24, there exists adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ attacking the MAC Σ and an adversary $\mathcal{C}_3$ attacking the pseudo-randomness of F such that

$$\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) \leqslant M(M-1)\cdot\Big(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q\cdot\mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}_1)\Big)$$

and

$$\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) \leqslant M(M-1)\cdot\Big(\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) + 2q\cdot\mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}_2) + 2q\cdot\mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C}_3)\Big).$$

Moreover, $\mathcal{B}_1$ and $\mathcal{B}_2$ query at most $4q$ times their oracle $\mathcal{O}_{\mathrm{Mac}}$ and $\mathcal{C}_3$ queries at most 3 times its oracle $\mathcal{O}_g$. In terms of F evaluations, $\mathcal{B}_1$ and $\mathcal{B}_2$ make at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries; while $\mathcal{C}_3$ makes at most those made by $\mathcal{A}$ plus those needed to simulate the protocol instructions to answer $\mathcal{A}$'s queries plus $3q$. By Theorem 25, there exist adversaries $\mathcal{C}_1$ and $\mathcal{C}_2$ attacking the pseudo-randomness of F such that

$$\mathrm{Adv}^{\text{SEUF-CMA}}_{\Sigma}(\mathcal{B}_i) \leqslant \mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C}_i) + \frac{7\cdot(4q)^2 + 7\cdot(4q) + 1}{2^n} \leqslant \mathrm{Adv}^{\mathrm{PRF}}_{F}(\mathcal{C}_i) + \frac{141q^2}{2^n}$$

for each $i \in \{1, 2\}$. Moreover, $\mathcal{C}_1$ and $\mathcal{C}_2$ query at most $3\cdot(4q+1)$ times their oracle $\mathcal{O}_g$ and make as many evaluations of F as $\mathcal{B}_1$ and $\mathcal{B}_2$ do, respectively. Finally, as mentioned in Subsection 3.3, one has

$$\mathrm{Coll}(\mathrm{GenN}, q, \mathrm{st}_0^{\mathrm{initN}}) \leqslant \frac{q^2}{2^{n+1}}$$

Alice	Bob
$\left(K_{AB}^{\text{init,kdf}}, \text{CTR}_{AB}^{\text{init}}, K_{AB}^{\text{resp,kdf}}, \text{CTR}_{AB}^{\text{resp}}, K_{AB}^{\text{mac}}\right) \in \mathbb{F}_2^{4n}$	$\left(K_{BA}^{\text{init,kdf}}, \text{CTR}_{BA}^{\text{init}}, K_{BA}^{\text{resp,kdf}}, \text{CTR}_{BA}^{\text{resp}}, K_{BA}^{\text{mac}}\right) \in \mathbb{F}_2^{4n}$
$N_A \xleftarrow{\text{U}} \mathbb{F}_2^n$	
$\sigma_A = F_{K_{AB}^{\text{mac}}}\left(N_A \oplus F_{K_{AB}^{\text{mac}}}\left(0\|A\|\text{CTR}_{AB}^{\text{init}}\right)\right)$	
$m_A = \text{CTR}_{AB}^{\text{init}}\|N_A\|\sigma_A$	
$\xrightarrow{m_A \in \mathbb{F}_2^{\frac{5n}{2}}}$	
	if $m_A \notin \mathbb{F}_2^{\frac{5n}{2}}$
	abort
	decompose m_A as $m_A = C\|N_A\|\sigma_A$
	with $\|C\| = \frac{n}{2},\ \|N_A\| = \|\sigma_A\| = n$
	if $\sigma_A \neq F_{K_{BA}^{\text{mac}}}\left(N_A \oplus F_{K_{BA}^{\text{mac}}}(0\|A\|C)\right)$
	abort
	while $C > \text{CTR}_{BA}^{\text{resp}}$
	$K_{BA}^{\text{resp,kdf}} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(0^n)$
	$\text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{BA}^{\text{resp}} + 1$
	$N_B \xleftarrow{\text{U}} \mathbb{F}_2^n$
	$\sigma_B = F_{K_{BA}^{\text{mac}}}\left(N_B \oplus F_{K_{BA}^{\text{mac}}}\left(N_A \oplus F_{K_{BA}^{\text{mac}}}(1\|B\|\text{CTR}_{BA}^{\text{resp}})\right)\right)$
	$m_B = \text{CTR}_{BA}^{\text{resp}}\|N_B\|\sigma_B$
$\xleftarrow{m_B \in \mathbb{F}_2^{\frac{5n}{2}}}$	
if $m_B \notin \mathbb{F}_2^{\frac{5n}{2}}$	**if** $[N_A]_{\frac{n}{2}}\|[N_B]_{\frac{n}{2}} \neq 0^n$
abort	$\text{sk} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}\left([N_A]_{\frac{n}{2}}\|[N_B]_{\frac{n}{2}}\right)$
decompose m_B as $m_B = C'\|N_B\|\sigma_B$	**else**
with $\|C'\| = \frac{n}{2},\ \|N_B\| = \|\sigma_B\| = n$	$\text{sk} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(1^n)$
if $\sigma_B \neq F_{K_{AB}^{\text{mac}}}\left(N_B \oplus F_{K_{AB}^{\text{mac}}}\left(N_A \oplus F_{K_{AB}^{\text{mac}}}(1\|B\|C')\right)\right)$	$K_{BA}^{\text{resp,kdf}} \leftarrow F_{K_{BA}^{\text{resp,kdf}}}(0^n)$
abort	$\text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{BA}^{\text{resp}} + 1$
if $C' < \text{CTR}_{AB}^{\text{init}}$	**if** $\text{CTR}_{BA}^{\text{resp}} \in \{0^{\frac{n}{2}}, 1^{\frac{n}{2}}\}$
abort	re-initialize master keys
while $C' > \text{CTR}_{AB}^{\text{init}}$	
$K_{AB}^{\text{init,kdf}} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(0^n)$	
$\text{CTR}_{AB}^{\text{init}} \leftarrow \text{CTR}_{AB}^{\text{init}} + 1$	
if $[N_A]_{\frac{n}{2}}\|[N_B]_{\frac{n}{2}} \neq 0^n$	
$\text{sk} \leftarrow F_{K_{AB}^{\text{init,kdf}}}\left([N_A]_{\frac{n}{2}}\|[N_B]_{\frac{n}{2}}\right)$	
else	
$\text{sk} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(1^n)$	
$K_{AB}^{\text{init,kdf}} \leftarrow F_{K_{AB}^{\text{init,kdf}}}(0^n)$	
$\text{CTR}_{AB}^{\text{init}} \leftarrow \text{CTR}_{AB}^{\text{init}} + 1$	
if $\text{CTR}_{AB}^{\text{init}} \in \{0^{\frac{n}{2}}, 1^{\frac{n}{2}}\}$	
re-initialize master keys	

Figure 5: The AKE protocol LP2+(n, F), an instantiation of LP2+.

for the chosen nonce generator. Taking $\mathcal{C}$ as the $\mathcal{C}_i$ maximizing $\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C}_i)$, we get

$$\begin{aligned}\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) &\leqslant M(M-1)\cdot\left(\mathrm{Coll}(\mathrm{GenN},q,\mathrm{st}_0^{\mathrm{initN}})+2q\cdot\mathrm{Adv}_\Sigma^{\text{SEUF-CMA}}(\mathcal{B}_1)\right)\\ &\leqslant M(M-1)\cdot\left(\frac{q^2}{2^{n+1}}+2q\cdot\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C}_1)+\frac{282q^3}{2^n}\right)\\ &\leqslant M(M-1)\cdot\left(2q\cdot\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C})+\frac{283q^3}{2^n}\right)\end{aligned}$$

and

$$\begin{aligned}\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{LP2+}(n,F)}(\mathcal{A}) &\leqslant M(M-1)\cdot\left(\mathrm{Coll}(\mathrm{GenN},q,\mathrm{st}_0^{\mathrm{initN}})+2q\cdot\mathrm{Adv}_\Sigma^{\text{SEUF-CMA}}(\mathcal{B}_2)+2q\cdot\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C}_3)\right)\\ &\leqslant M(M-1)\cdot\left(\frac{q^2}{2^{n+1}}+2q\cdot\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C}_2)+\frac{282q^3}{2^n}+2q\cdot\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C}_3)\right)\\ &\leqslant M(M-1)\cdot\left(4q\cdot\mathrm{Adv}_F^{\mathrm{PRF}}(\mathcal{C})+\frac{283q^3}{2^n}\right)\end{aligned}$$

as desired. For the number of oracle queries and F evaluations made by $\mathcal{C}$, we take in each case the worst among those of the $\mathcal{C}_i$'s. □

References

[1] S.F. Aghili, A.A. Jolfaei and A. Abidin, SAKE+: Strengthened symmetric-key authenticated key exchange with perfect forward secrecy for IoT, *Cryptology ePrint Archive* **2020/778**, (2020).

[2] G. Avoine, S. Canard and L. Ferreira, Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy, *Topics in Cryptology – CT-RSA 2020, Springer Lecture Notes in Computer Science* **12006**, 199–224 (2020).

[3] M. Bellare, O. Goldreich and A. Mityagin, The power of verification queries in message authentication and authenticated encryption, *Cryptology ePrint Archive* **2004/309**, (2004).

[4] M. Bellare, D. Pointcheval and P. Rogaway, Authenticated key exchange secure against dictionary attacks, *Advances in Cryptology – EUROCRYPT 2000, Springer Lecture Notes in Computer Science* **1807**, 139–155 (2000).

[5] M. Bellare and P. Rogaway, Entity authentication and key distribution, *Advances in Cryptology – CRYPTO 1993, Springer Lecture Notes in Computer Science* **773**, 232–249 (1994).

[6] M. Bellare and P. Rogaway, Provably secure session key distribution: the three party case, *STOC 1995: Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing*, 57–66 (1995).

[7] S. Blake-Wilson, D. Johnson and A. Menezes, Key agreement protocols and their security analysis, *Cryptography and Coding 1997, Springer Lecture Notes in Computer Science* **1355**, 30–45 (1997).

[8] C. Boyd, G.T. Davies, B. de Kock, K. Gellert, T. Jager and L. Millerjord, Symmetric key exchange with full forward security and robust synchronization, *Advances in Cryptology – ASIACRYPT 2021, Springer Lecture Notes in Computer Science* **13093**, 681–710 (2021).

[9] R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, *Advances in Cryptology – EUROCRYPT 2001, Springer Lecture Notes in Computer Science* **2045**, 453–474 (2001).

[10] M.S. Dousti and R. Jalili, FORSAKES: A forward-secure authenticated key exchange protocol based on symmetric key-evolving schemes, *Advances in Mathematics of Communications* **9**, 471–514 (2015).

[11] Q. Fan, J. Chen, M. Shojafar, S. Kumari and D. He, SAKE*: A symmetric authenticated key exchange protocol with perfect forward secrecy for industrial internet of things, *IEEE Transactions on Industrial Informatics* **18**, 6424–6434 (2022).

[12] L. Ferreira, Privacy-preserving authenticated key exchange for constrained devices, *ACNS 2022, Springer Lecture Notes in Computer Science* **13269**, 293–312 (2022).

[13] Y. Guo and Y. Guo, CS-LAKA: A lightweight authenticated key agreement protocol with critical security properties for IoT environments, *IEEE Transactions on Services Computing* **16**, 4102–4114 (2023).

[14] T. Jager, F. Kohlar, S. Schäge and J. Schwenk, On the security of TLS-DHE in the standard model, *Advances in Cryptology – CRYPTO 2012, Springer Lecture Notes in Computer Science* **7417**, 273–293 (2012).

[15] J. Katz and Y. Lindell, Introduction to modern cryptography (3rd edition), *Chapman & Hall/CRC Cryptography and Network Security Series* (2021).

[16] B. LaMacchia, K. Lauter and A. Mityagin, Stronger security of authenticated key exchange, *ProvSec 2007, Springer Lecture Notes in Computer Science* **4784**, 1–16 (2007).

[17] Y. Li and S. Schäge, No-match attacks and robust partnering definitions: Defining trivial attacks for security protocols is not trivial, *CCS 2017: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1343–1360 (2017).

[18] E. Petrank and C. Rackoff, CBC MAC for real-time data sources, *Journal of Cryptology* **13**, 315–338 (2000).

[19] S. Schäge, J. Schwenk and S. Lauer, Privacy-preserving authenticated key exchange and the case of IKEv2, *PKC 2020, Springer Lecture Notes in Computer Science* **12111**, 567–596 (2020).

[20] Y. Zhang, D. He, P. Vijayakumar, M. Luo and X. Huang, SAPFS: An efficient symmetric-key authentication key agreement scheme with perfect forward secrecy for industrial internet of things, *IEEE Internet of Things Journal* **10**, 9716–9726 (2023).