ITSAKE: an unconditionally secure authenticated key establishment protocol

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

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

February 10, 2026

Abstract

We introduce ITSAKE, a 2-message Information-Theoretic Secure Authenticated Key Establishment protocol. That is, ITSAKE is a protocol to establish keys between two parties from a pre-shared secret. Beside correctness and key indistinguishability, it offers entity authentication and perfect forward secrecy. Moreover, synchronization problems are avoided because ITSAKE satisfies the weak synchronization robustness property. The main advantage of ITSAKE is that all these security properties are unconditionally proved, i.e., they do not rely on unproven mathematical assumptions or on limitations on the power of the adversary. We provide complete and detailed security proofs in a well-defined multi-party security model for symmetric-key authenticated key establishment protocols admitting concurrent runs in which the adversary has a complete control on the communication channel.

The main drawback of ITSAKE is the length of the pre-shared keys that the parties need to hold. To give an example, we present an instantiation of the protocol for which the master-key is just slightly longer than the total keying material that can be established. At this cost, one obtains fresher keys, authenticity of the partner, information-theoretic security, and avoids synchronization problems.

ITSAKE is meant to be used to protect highly confidential information, for which trust in unproven assumptions or in limitations on the adversary is undesired. Moreover, as memory devices are typically much less expensive than dedicated quantum apparatus, and as physical key exchanges performed occasionally may be more convenient than maintaining those quantum apparatus, ITSAKE is a practical and cost-effective alternative to Quantum Key Distribution (QKD) in many of its potential use-cases, while offering the same security guarantees. Thus, ITSAKE challenges the necessity of QKD in most settings where these guarantees are the main objective.

Keywords: cryptographic protocol, information-theoretical security, authenticated key exchange, perfect forward secrecy, synchronization robustness, security proof.

Contents

1 Introduction **2**

2 Preliminaries **6**
2.1 Notation 6
2.2 The security model 6
2.3 τ-time MACCs 11
2.4 Nonce generators 15

*Email: pierre-alain.jacqmin@mil.be; ORCID: 0000-0001-9834-1433
†Email: jean.lienardy@mil.be; ORCID: 0009-0002-6377-0734

3 The protocol ITSAKE **15**
3.1 The general case 15
3.2 An instantiation 18

4 Security theorems **19**
4.1 Technical lemmas 19
4.2 Correctness 28
4.3 Weak synchronization robustness 29
4.4 Entity authentication 30
4.5 Key indistinguishability 31

References **36**

1 Introduction

One of the most important and most studied problems in cryptography is the establishment of a secret key between two parties. Such a secret key is then generally used by the parties to run a symmetric-key algorithm. Since the security of the overall solution is bounded by that of the key establishment protocol, one cannot neglect it.

The most commonly used solution for this key establishment problem is to use public-key cryptography schemes. This solution is very practical and is the only one known today that can efficiently be used in large networks. However, its security relies on unproven mathematical assumptions such as the difficulty to solve certain problems. The confidence one has in such a solution evolves according to the effort invested by the community to break these schemes and how the security margins are affected by it. Widely studied old schemes such as RSA are, in that respect, trusted by most people and organizations. However, due to Shor's algorithm, they are not standalone secure in a world where quantum computers are emerging. They should thus be replaced or combined with more recent post-quantum cryptographic schemes. Having received less attention from the community, these schemes have not yet passed the test of time. In view of this, a conservative way of enhancing the security of a public-key cryptographic scheme is to combine two schemes (or many more) in order to reduce the risk of future attacks breaking the combined scheme. Although being very conservative, this approach does not guarantee absolute security.

When one requires an even more conservative approach, a purely symmetric-key solution is definitely worth considering. In this case, every pair of parties exchanges a master key which serves to establish future session keys. Although this solution is not efficient for large networks due to the necessity of having this pre-shared keying material, it is a good alternative to public-key cryptosystems for smaller networks. In its most common variant, symmetric-key primitives such as block ciphers or hash functions are used. In that case, a small pre-shared master key serves for the establishment of many future session keys. The security of such a solution relies on that of its building blocks. The security of these symmetric-key primitives being extensively studied and remaining stable over time, the security of symmetric-key key establishment protocols is considered very conservative. Moreover, as far as we know, it is naturally immune to the advent of quantum computers. However, as for public-key solutions, the security is on one hand not mathematically fully proved, and on the other hand, only *computational*, i.e., it requires the adversary to have bounded resources (although the bound may be very large). This limitation concerns computational power, memory, and time, among others.

If one does not want to rely on computational assumptions about the adversary, which is usually referred to as 'unconditional security', then the notion of Information-Theoretic Security (ITS) comes into play. A cryptosystem is ITS if its security is mathematically fully proved and does not rely on an unproven assumption or on limitations on the power of the adversary. Such a level of security might be desired, for instance, to protect classified national information, military communications, or information exchanged between banking data servers. This theoretically perfect security usually comes at a cost: the pre-shared master keys need to be at least as large as the total length of the established keys.

One of the first methods one can think of to achieve key establishment in an ITS way is simply to use the One-Time Pad (OTP). This technique has two main drawbacks. On one hand, as mentioned

above, the pre-shared master key must be as long as the total length of all established keys. On the other hand, this technique does not ensure authenticity, i.e., identification of the parties. More evolved techniques using deals of cards as pre-shared secret information have been introduced [13, 14], but the same drawbacks persist. As a one-line summary, in this paper, we propose an ITS protocol that solves the authentication issue (and provides other security properties), but keeping long master keys. Other ITS key establishment protocols have been proposed in the literature in order to solve the restriction on the master key length. For instance, using noisy channels [27, 23, 24], the legitimate parties (Alice and Bob) can extract secret information based on the assumption that the adversary Eve does not have access to the exact same information that transits on the noisy channel between Alice and Bob. This assumption is quite restrictive and usually not considered safe to be made in modern cryptography. Moreover, these schemes are vulnerable to active attacks on the main channel. Besides, in theory, ITS key establishment schemes can also be obtained using laws of Physics. Quantum Key Distribution (QKD), introduced in [6], bases its security on the validity of Quantum Mechanics, while the security of Kirchhoff-law-Johnson-noise (KLJN) key establishment schemes, introduced in [19], rely on the Second Law of Thermodynamics [20]. However, this security is only theoretical as the difficulty of exchanging long master keys has been transferred to the difficulty of manufacturing devices that correctly execute these physical protocols. This task is far from being trivial as shown by several attacks (sometimes repaired by new protocol designs), see for instance [8, 22, 18] for QKD and [10, 25, 12] for KLJN.

Our contributions. In this paper, we propose an ITS key establishment protocol ITSAKE. As opposed to the plain OTP and protocols from [13, 14] based on deals of cards, we consider authenticity as one of the main security objectives and incorporate it directly in ITSAKE, whence the name Information-Theoretic Secure Authenticated Key Establishment. Authenticity is achieved via τ-time Message Authentication Codes with Counters [26] (τ-time MACCs). Our protocol only needs 2 messages, much less than those based on deals of cards. Using proper zeroization, we achieve *perfect forward secrecy* (PFS), i.e., if the master key of a party is leaked at some point in time, session keys previously established remain perfectly secret. Less trivially, ITSAKE also achieves *weak synchronization robustness*, a notion introduced in [7] and improved in [16]. This property ensures that, even if they start desynchronized, an uninterrupted honest run of the protocol between two uncorrupted parties makes them accept with the same session key. Moreover, in comparison with storing a key to be directly used for OTP, ITSAKE improves the freshness of the established session key. As expected in view of the information-theoretic security, the main drawback of our protocol is the length of the master key. As an example, we study a particular instantiation of the protocol, called ITSAKE(f), where f is a difference universal function used to build the MACCs. In that instantiation, the master key is about 3 times the total length of the session keys if they are short to be used with computationally secure symmetric-key algorithms, while the master key is about the total amount of keying material that can be established if session keys are long to be used with OTP (or if many short keys are required in one run). For instance, to establish a billion keys of 1 kB each, hence a total of 1 TB of secret, the parties are required to store a 1.1 TB master key, while if they want to establish a billion keys of 128 bits each, they need a master key of less than 52 GB.

The security properties of ITSAKE are stated and proved within the model of [16] for symmetric-key Authenticated Key Establishment (AKE) protocols. This model takes its roots in the sequence [5, 9, 17, 2, 7] of security models for AKE protocols. It is expressed in a multi-party context with concurrent runs. In this model, the adversary has a complete control over the communication channel, i.e., it can forward, modify, drop and delay any messages exchanged by the parties, as well as create new messages. We of course prove the correctness of ITSAKE (i.e., that mutual partner sessions output the same session key with very high probability, see Definition 4) as well as the key indistinguishability of ITSAKE (i.e., that the adversary cannot distinguish a random key from a session key provided it cannot trivially access it, see Definition 10). Key indistinguishability in the model of [16] includes in particular perfect forward secrecy. As mentioned above, we also prove the entity authentication (Definition 9) and weak synchronization robustness (Definition 6) of ITSAKE. With the same philosophy as in [16], we prove these properties in full details, not just with mere sketches of proofs. We think that these complete proofs are very important and cannot be neglected, especially in cryptography where small mistakes can have dramatic consequences.

A first scenario where ITSAKE would be useful is the following. Two parties meet once every few years, but regularly exchange highly confidential information for which they want unconditional security. They could exchange a big master key when they meet and use part of it as an OTP key each time they exchange messages (and delete this part of the key afterwards). However, this may create synchronization problems and authenticity is not taken into account. Instead, they can use this long-term keying material to establish one or several session keys with ITSAKE each time they want to exchange data, and then use them with OTP to transmit data with perfect security. Since ITSAKE is designed in a bidirectional way, both parties can initiate such a communication. In such a scenario, the size n of the session keys should be set to at least (but close to) the size of the information being regularly exchanged. This scenario is the main use-case of ITSAKE. Indeed, since the security of a solution is as weak as its weakest point, we recommend to use the keys established via ITSAKE within an ITS scheme.

Despite the above recommendation to use ITSAKE with ITS schemes, there are situations where it may be relevant to use keys established via ITSAKE within computationally secure schemes. Imagine a situation where people working in two distant antennas of an organization have many projects in common between them. Moreover, these projects evolve over time and are not yet known by the managers of the antennas. It then makes sense to have a crypto custodian in each antenna that communicate with each other to produce session keys for their colleagues using ITSAKE. Unconditional security might be desired for the backbone as an attack against the cryptographic scheme used between the crypto custodians implies an attack to all the projects of the organization. However, a computational security for each individual project may still be enough as a break of the cryptography used in one project may not impact the other ones. In such a scenario, typical session key lengths would be $n = 128$ or $n = 256$ bits. Notice that if the security of all the projects of this organization only relies on some pseudo-random functions (PRFs) within a small *fixed known* set of PRFs, then one would reach a similar level of security using LP2+ from [16] with a hybrid PRF combining all these PRFs in such a way that if at least one of them is secure, the hybrid PRF is secure too. In this case, the master key would be much smaller than with ITSAKE.

There is another theoretical motivation for ITSAKE. Let us compare QKD and ITSAKE. Quantum Key Distribution claims to extend a small pre-shared keying material (necessary for authentication) to longer session keys with perfect forward secrecy in an ITS way. Such a scheme requires the parties to have very expensive quantum devices in order to operate the protocol. On the other hand, ITSAKE requires the parties to exchange from time to time a large amount of keying material in order to be able to establish session keys with perfect forward secrecy in an ITS way. Both techniques thus aim to establish session keys in an ITS way with PFS from a pre-shared key. The main (or only) drawback of ITSAKE is that the master key should be refill once in a while, whereas QKD is expensive, slower, subject to availability issues, requires the parties to be close to each other, both its theoretical security and its implementation are difficult to evaluate, and most probably the quantum devices require maintenance that counter-balances the need of master key refilling for ITSAKE. Therefore, it is hard to find a use-case where QKD would be more appropriate than ITSAKE. The main arguments in favor of QKD being the unconditional security and PFS, ITSAKE can thus be viewed as an argument by contradiction against QKD, which was our original motivation.

Related works. A sequence of ITS key establishment protocols using noisy channels was initiated in [27]. An ITS protocol was presented based on the assumption that the adversary Eve listens the conversation between the legitimate parties Alice and Bob via a channel that adds noise on the way (or rather reduces the capacity of the channel). The assumption that Eve has access to a noisier channel than the main channel between Alice and Bob is quite dangerous and not usually made in classical cryptography. Moreover, authenticity was not treated. An improved protocol was presented in [23] where the noise on Eve's channel is supposed to be known, positive, but not necessarily higher than on the main channel. In addition to their noisy channel, Alice and Bob also need an authenticated error-free communication channel. In [24], the assumption that the error-free channel is authentic is dropped in some cases (but authentication of the noisy channel is still implicitly used). Many papers were then written on ITS protocols based on noisy channels, but since they rely on physical assumptions about the communication channels, one should be cautious when using them.

Some other types of ITS protocols depend on laws of Physics. As already mentioned above, such examples are KLJN protocols introduced in [19] relying on the Second Law of Thermodynamics and QKD protocols introduced in [6] relying on Quantum Mechanics. A long list of papers on these two topics appears in the literature. The overall idea with these schemes is to transfer the task of designing a secure classical cryptographic protocol appropriate for a given use-case to the task of manufacturing a device correctly implementing a physical protocol. This latter task is far from easy as shown by concrete attacks (see e.g. [8, 22, 18, 10, 25, 12]). Moreover, the task of evaluating the overall solution is certainly not made easier.

ITS key establishment protocols not based on physical assumptions have also been studied. The standard One-Time Pad is a classical example, but offers as such no authentication, and may suffer of synchronization problems. Another family of ITS protocols have been introduced in [13] using deals of cards. There, a deck of cards is shuffled, a portion p of it is given to Alice, another portion p is given to Bob and the rest is supposed to be in Eve's hands. Via public discussion, Alice and Bob can establish a secret key in a ITS way using this deal. However, this solution is not practical because the size of the deck needs to be super-polynomial in $\frac{1}{p}$, requires many messages to be exchanged and because such deals of cards are not easy to realize, especially without a trusted authority. The size of the deck has been reduced to a polynomial in $\frac{1}{p}$ in [14] but the other drawbacks remain.

In order to achieve unconditional security for our protocol, we need an ITS Message Authentication Code (MAC). Such MACs were introduced in [15]. Since one is allowed to use a key to authenticate only one message, these were called 1-time MACs. In order to increase the number of messages one can authenticate, τ-time MACs with counters were introduced in [26]. With such schemes, the size of the key grows linearly with the maximum number τ of messages that can be authenticated. To avoid counters, one can instead use τ-time MAC (without counters) as introduced in [1], but this comes at the cost that the size of the key grows quadratically with τ. In ITSAKE, we choose to use the former option, i.e., MACs with counters (MACCs). In [26], MACCs were constructed from strongly universal$_2$ hash functions. In order to have shorter keys, we generalize this construction to ε-difference universal functions, introduced in [21] under the name 'ε-otp-secure family of hash functions'. This generalized construction is then used in our instantiation ITSAKE(f) of ITSAKE.

The security properties of ITSAKE are stated and proved in the exact same security model as that of [16]. This model concerns the security of symmetric-key authenticated key establishment protocols. In particular, it formalizes the notions of correctness, weak synchronization robustness, entity authentication and key indistinguishability. This model takes its roots in a long sequence of security models. The first security model for AKE protocols as been introduced in the seminal work [5]. In that first model, the adversary already has a total control on the communication channel, but does not have the ability to corrupt parties. This ability has been considered in the model of [4] in order to guarantee perfect forward secrecy. The model has then been improved many times in the literature, e.g. in [9, 17, 2, 7, 16] to cite only a few. For instance, the property of weak synchronization robustness held by ITSAKE was introduced in [7].

Many AKE protocols based on symmetric-key cryptography have been proposed. In [5], AKEP1 and AKEP2 are designed as 3-message protocols, but they do not offer perfect forward secrecy. FORSAKES, introduced in [11], is a 3-message protocol that provides forward secrecy but for which parties are required to have synchronized clocks. SAKE and SAKE-AM were designed in [2] as symmetric-key AKE protocols providing PFS; they run in 5 and 4 messages respectively. The paper [7] introduces several symmetric-key AKE protocols such as LP2 and LP3 but, as noticed in [16], they do not meet their security claims. A corrected improved version of them, called LP2+ and using 2 messages, was designed and fully proved in [16]. All these mentioned symmetric-key AKE protocols are (at most) computationally secure, not ITS. To our knowledge, ITSAKE is the first ITS AKE protocol with complete security proofs in such a security model for AKE protocols.

Outline. The rest of the paper is divided as follows. Section 2 is devoted to the preliminaries. After setting the notation in Subsection 2.1, the security model of [16] is recalled in Subsection 2.2 and τ-time Message Authentication Codes with Counters (MACCs) are discussed in Subsection 2.3. Nonce generators are used in ITSAKE to increase freshness of the established keys and so they are briefly mentioned in Subsection 2.4. In Section 3, we describe the protocol ITSAKE. The general version is treated in Subsection 3.1 while the instantiation ITSAKE(f) is described in Subsection 3.2.

We then conclude the paper with the complete proofs of the security properties of ITSAKE in the model of [16]: correctness (Theorem 26), weak synchronization robustness (Theorem 27), entity authentication (Theorem 28) and key indistinguishability (Theorem 30).

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 Preliminaries

2.1 Notation

Here is a list of the notation used in this article.

- $\mathbb{N}$ denotes the set of natural numbers (including 0).
- $\mathbb{Z}$ denotes the set of integers.
- For $a \in \mathbb{N}$, we denote by $\mathbb{N}_{\geqslant a}$ (respectively by $\mathbb{N}_{<a}$) the set of natural numbers n such that $n \geqslant a$ (respectively such that $n < a$).
- For $n \in \mathbb{N}_{\geqslant 2}$, the ring of integers modulo n is denoted by $\mathbb{Z}_n$.
- If $\mathcal{X}$ is a finite non-empty set, $x \xleftarrow{\text{U}} \mathcal{X}$ denotes the uniform selection of an element $x \in \mathcal{X}$.
- For a function $f\colon \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ and for $k \in \mathcal{K}$ and $m \in \mathcal{M}$, we often denote $f(k,m)$ by $f_k(m)$.
- For a prime power q, we denote by $\mathbb{F}_q$ the q-element field. In particular, $\mathbb{F}_2$ is $\{0,1\}$ equipped with addition and multiplication modulo 2.
- For $n \in \mathbb{N}$, $\mathbb{F}_2^n$ denotes the n-dimensional vector space over $\mathbb{F}_2$, consisting of n-bit strings and with bit-wise XOR $\oplus$ as addition. In particular, for $n = 0$, $\mathbb{F}_2^0$ consists of the single empty string denoted ω.
- For $n, n' \in \mathbb{N}$, for $K \in \mathbb{F}_2^n$ and for $K' \in \mathbb{F}_2^{n'}$, the concatenation of K and K' is denoted by $K\|K'$ and lives in $\mathbb{F}_2^{n+n'}$. In particular, one has $K\|\omega = K = \omega\|K$.
- For $n, c \in \mathbb{N}$ with $n > c$ and for $K \in \mathbb{F}_2^n$, the cth bit of K (starting counting from 0) is denoted $K[c] \in \mathbb{F}_2$.
- For $n, c, \ell \in \mathbb{N}$ such that $n \geqslant c + \ell$, and for $K \in \mathbb{F}_2^n$, we denote by $K[c,\ell] \in \mathbb{F}_2^\ell$ the ℓ-bit string $K[c]\|K[c+1]\|\cdots\|K[c+\ell-1]$. In particular, one has $K[c,0] = \omega \in \mathbb{F}_2^0$ for any $c \leqslant n$.
- For events X and Y, $\Pr(X \,|\, Y)$ denotes the probability that X occurs given that Y occurs. If $\Pr(Y) > 0$, one has $\Pr(X \,|\, Y) = \frac{\Pr(X \wedge Y)}{\Pr(Y)}$; while if $\Pr(Y) = 0$, we use the convention $\Pr(X \,|\, Y) = 0$. In any case, $\Pr(X \,|\, Y)\Pr(Y) = \Pr(X \wedge Y)$.

2.2 The security model

Before describing our protocol ITSAKE, we describe the security model for symmetric-key authenticated key establishment protocols with perfect forward secrecy from [16]. We use exactly the same model to prove the security properties of ITSAKE. This model takes its roots in the sequence [5, 9, 17, 2, 7] of such models. We refer to [16] for a discussion and explanations on this security model. However, let us make explicit that we are concerned with concrete security. In particular, security definitions do not depend on a security parameter. We only consider the advantage of an adversary in a given security experiment, but do not precise when this advantage is small enough. This is more general than the asymptotical approach, since protocols can depend on a security parameter and the adversary's advantages required to be negligible with respect to that parameter.

We consider an authenticated key establishment (AKE) protocol Π that aims to establish session keys between two parties among a set $\{P_1, \ldots, P_M\}$ of $M \geqslant 2$ potential participants. Initially, each party P_i is assumed to be in possession, for each $j \in \{1, \ldots, M\} \setminus \{i\}$, of a secret master key $\text{MK}_{P_iP_j}$, which we often write MK_{ij} for the sake of simplicity. In order to achieve perfect forward secrecy, we allow these keys to be modified over time. Note that we do not necessarily require that $\text{MK}_{ij} = \text{MK}_{ji}$. To model parallel executions of the protocol, each party P_i is equipped 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 the master keys MK_{ij} of its party P_i and is able to modify them. Moreover, each oracle maintains a set of local variables as described in Table 1. We denote by $\pi_i^s.v$ the value of a variable v for the oracle π_i^s.

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 session oracle π_i^s.

At the initialization of the parties, the local variables of each session oracle π_i^s are set as follows: $\pi_i^s.\alpha = \pi_i^s.\mathrm{pid} = \pi_i^s.\rho = \pi_i^s.\mathrm{sk} = \perp$ and $\pi_i^s.\mathrm{T}$ and $\pi_i^s.\mathrm{st}$ are empty. The variables $\pi_i^s.\alpha$, $\pi_i^s.\mathrm{pid}$ and $\pi_i^s.\rho$ are set to a value different from $\perp$ when the adversary interacts for the first time with π_i^s. The transcript $\pi_i^s.\mathrm{T}$ of π_i^s, often denoted as T_i^s, is the sequence, in chronological order, of all messages sent and received by π_i^s. The state $\pi_i^s.\mathrm{st}$ serves as the oracle's memory to save information from one round to another. The variable $\pi_i^s.\mathrm{sk}$ is the established session key of the oracle and we require that $\pi_i^s.\mathrm{sk} \neq \perp$ if and only if $\pi_i^s.\alpha = \text{accepted}$. The session key space $\mathcal{K}_\mathrm{s}$ is supposed to be a finite non-empty set. Once the variables $\pi_i^s.\mathrm{pid}$, $\pi_i^s.\rho$ and $\pi_i^s.\mathrm{sk}$ are set to a value different from $\perp$, they can no longer be modified. Similarly, once the variable $\pi_i^s.\alpha$ reaches a value different from $\perp$, it can no longer be set back to $\perp$. Moreover, once $\pi_i^s.\alpha$ reaches the value accepted or rejected, it can no longer be modified and the oracle π_i^s accepts no more messages.

The security properties of a protocol are demonstrated through so-called experiments which involve a challenger and an adversary and that are represented by 1-bit outputs. The role of the challenger is to give the adversary access to oracles, and the role of the adversary is to interact with these oracles so that certain conditions are fulfilled. The ability of the adversary to meet these conditions is reflected in the probability that the experiment outputs 1 instead of 0. In the case where this success probability is expected to be small, it is also called the advantage of the adversary in breaking the given security property. When the success probability is expected to be at least $\frac{1}{2}$ for trivial reasons, the advantage of the adversary is defined as twice the gap between the success probability and $\frac{1}{2}$.

At the start of any of the experiments in this subsection, the challenger initializes M parties $\{P_1, \ldots, P_M\}$. That is, it initializes oracles π_i^s as described 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., for each s, oracle π_i^s has 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 $\pi_i^s.\alpha = \text{negotiating}$, $\pi_i^s.\mathrm{pid} = P_j$ 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 $\pi_i^s.\alpha = \text{negotiating}$, $\pi_i^s.\mathrm{pid} = P_j$ 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 $\pi_i^s.\alpha$ = accepted): Reveals the session key. Specifically, this query returns π_i^s.sk to $\mathcal{A}$.
- RevealState(π_i^s): Reveals the state. Specifically, this query returns π_i^s.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 $\pi_i^s.\alpha$ = accepted): Selects π_i^s as the test oracle for key indistinguishability. Specifically, this query uniformly chooses sk_0 in $\mathcal{K}_\mathrm{s}$, sets $sk_1 = \pi_i^s$.sk, uniformly chooses a *test bit* $b_\mathrm{test} \in \{0,1\}$ and returns sk_{b_test} to $\mathcal{A}$. This action can be queried at most once.

If the adversary makes a query that does not respect the constraints (e.g., a RevealKey(π_i^s) query for an oracle π_i^s such that $\pi_i^s.\alpha \neq$ accepted), the challenger disregards this query and leaves it unanswered.

In view of the queries available to $\mathcal{A}$, one can notice that $\mathcal{A}$ has always access to the value of the variables π_i^s.pid, $\pi_i^s.\rho$ and T_i^s. We also assume that $\mathcal{A}$ always knows the value of $\pi_i^s.\alpha$, which can be modified by some actions prescribed by the protocol. If π_i^s.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 $\pi_i^s.\alpha =$ rejected.

Since we consider protocols where each run is executed between two parties, 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 $\pi_i^s.\alpha =$ 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, one can see that the sent and the received messages alternate in the transcript T_i^s.

Remark that the adversary $\mathcal{A}$ can only make one query at a time. Therefore, the RevealState query gives it access to the state π_i^s.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 order to define the notion of partner, we first need that of partially matching conversation. 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 $\mathrm{T}_i^s = \mathrm{T}_j^t$.

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. $\pi_i^s.\alpha =$ 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. $\pi_i^s.\alpha = \text{accepted} = \pi_j^t.\alpha$,

5. $\mathrm{T}_i^s = \mathrm{T}_j^t$.

We can now recall the definition of correctness of an AKE protocol as stated in [16]. Recall 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 $\pi_i^s.\alpha = \text{accepted}$.

Definition 4 (Correctness)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\mathrm{Correct}_{\mathcal{A},\Pi}$ 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 $\mathrm{Correct}_{\mathcal{A},\Pi}$ is defined as

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

We note that in the above definition, it is not required that an honest uninterrupted run of the protocol executed between π_i^s and π_j^t leads the two oracles to accept. If the parties are uncorrupted, this requirement is captured by the notion of weak synchronization robustness introduced in [7]. This notion ensures that, even if two uncorrupted parties are desynchronized, they can recover simply by executing an honest uninterrupted run of the protocol. We recall here the version of weak synchronization robustness as in [16].

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,
2. if the above query produced a message m_1, then 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 queries or until $\mathcal{A}$ tries to deliver a message to an oracle that has already accepted or rejected.

We say that an honest run of the protocol is *uninterrupted* if there are no other queries 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 uninterrupted honest run of the protocol was executed between π_i^s and π_j^t,
   (b) $\mathcal{A}$ did not use queries $\mathrm{Corrupt}(P_i, P_j)$ or $\mathrm{Corrupt}(P_j, P_i)$,
   (c) $\pi_i^s.\alpha \neq \text{accepted}$ or $\pi_j^t.\alpha \neq \text{accepted}$ 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

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

A stronger version of this notion is also introduced in [7] and allows $\mathcal{A}$ to perform some other queries during the honest run.

We continue with the definition of entity authentication. For that purpose, we need some more terminology.

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 $\text{Corrupt}(P_i, \pi_i^s.\text{pid})$,
3. $\mathcal{A}$ did not make the query $\text{Corrupt}(\pi_i^s.\text{pid}, P_i)$.

If π_i^s has π_j^t as partner, nothing in Definition 2 prevents $\mathcal{A}$ from having delivered a message to π_j^t before it was output by π_i^s (or vice versa). The following definition adds this requirement.

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 $k \in \{1, \ldots, |\text{T}_i^s| - 1\}$ in the case π_i^s has sent the last message of its transcript, or for each $k \in \{1, \ldots, |\text{T}_i^s|\}$ in the case π_i^s has received the last message of its transcript, then

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

As usual, we say that the adversary breaks entity authentication if it forces a two-sided uncorrupted oracle to accept maliciously. Below is the precise definition as stated in [16].

Definition 9 (Entity Authentication)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\text{EntAuth}_{\mathcal{A},\Pi}$ 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) $\pi_i^s.\alpha = \text{accepted}$,
   (b) π_i^s was two-sided uncorrupted at the time it accepted,
   (c) there is more than one oracle 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 $\text{EntAuth}_{\mathcal{A},\Pi}$ is defined as

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

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

We finally recall the definition of key indistinguishability as stated in [16]. Roughly speaking, 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 following definition makes precise what is meant by a trivial attack.

Definition 10 (Key Indistinguishability)**.** For an adversary $\mathcal{A}$ attacking an AKE protocol Π, let $\text{KeyInd}_{\mathcal{A},\Pi}$ 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,
   (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 $\mathrm{Corrupt}(P_j, P_i)$ with $P_j = \pi_i^s.\mathrm{pid}$, 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

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

2.3 τ-time MACCs

The messages in our protocol ITSAKE need to be authenticated. However, when one desires unconditional security for a message authentication code (MAC), it often comes at the cost of being allowed to use a key to authenticate only 1 message (1-time MACs were introduced in [15]). If one wants to authenticate more messages with unconditional security, two options are available. Either one uses τ-time MACs as constructed in [1] where the size of the key grows quadratically with τ, or one uses τ-time MACs with counters, as introduced in [26], so that the size of the key only grows linearly with τ. In both cases, the maximum number τ of messages that one can authenticate has to be specified in advance. In order to keep the size of the key under control, we opt for counters in ITSAKE.

Definition 11. Given an integer $\tau \geqslant 1$, a *τ-time message authentication code with counters* (τ-time MACC) is a triplet $(\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ of (probabilistic) algorithms, where:

- KeyGen takes no input and outputs a key $k \in \mathcal{K}_{\mathrm{MACC}}$ in the key space $\mathcal{K}_{\mathrm{MACC}}$;
- Mac takes as input a key $k \in \mathcal{K}_{\mathrm{MACC}}$, a counter $i \in \mathbb{N}_{<\tau}$, 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 $k \in \mathcal{K}_{\mathrm{MACC}}$, a counter $i \in \mathbb{N}_{<\tau}$, a message $m \in \mathcal{M}$, and a tag $\sigma \in \mathcal{T}$, and outputs a bit $\mathrm{Vrfy}_{k,i}(m,\sigma) \in \{0,1\}$.

We require that τ-time MACCs be *correct*, that is, for every key $k \in \mathcal{K}_{\mathrm{MACC}}$, for every counter value $i \in \mathbb{N}_{<\tau}$, for every message $m \in \mathcal{M}$, and for every tag generated as $\sigma \leftarrow \mathrm{Mac}_{k,i}(m)$, one has $\mathrm{Vrfy}_{k,i}(m,\sigma) = 1$.

As with classical MACs, one distinguishes between 'unforgeability' and 'strong unforgeability' for a τ-time MACC. In ITSAKE, we need the stronger notion, as stated below.

Definition 12 (ε-strongly secure τ-time MACC)**.** Let $\varepsilon > 0$ be a real and $\tau \geqslant 1$ be an integer. A τ-time MACC $\Sigma = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ with key space $\mathcal{K}_{\mathrm{MACC}}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$ is *ε-strongly secure* if, for all (even unbounded) adversaries $\mathcal{A}$, one has

$$\Pr\left(\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma}^{\tau\text{-time}} = 1\right) \leqslant \varepsilon$$

where $\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma}^{\tau\text{-time}}$ is the following game:

1. The challenger runs KeyGen to obtain a key $k \in \mathcal{K}_{\mathrm{MACC}}$.
2. The challenger initializes a set $\mathcal{Q}$ to the empty set $\emptyset$.
3. $\mathcal{A}$ has access to an oracle $\mathcal{O}_{\mathrm{Mac}}$ that can be queried at most τ times. If the ith query of $\mathcal{A}$ to its oracle (starting counting from 0) is made on input $m \in \mathcal{M}$, the oracle runs $\sigma \leftarrow \mathrm{Mac}_{k,i}(m)$, returns σ to $\mathcal{A}$ and updates $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i,m,\sigma)\}$.

4. When $\mathcal{A}$ concludes, it outputs $(i, m, \sigma) \in \mathbb{N}_{<\tau} \times \mathcal{M} \times \mathcal{T}$.
5. The output of the experiment is defined as 1 if and only if $(i, m, \sigma) \notin \mathcal{Q}$ and $\mathrm{Vrfy}_{k,i}(m, \sigma) = 1$.

The weaker notion of 'ε-secure τ-time MACC' is obtained by replacing, in the above definition, '$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i, m, \sigma)\}$' by '$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i, m)\}$' in step 3 and the condition '$(i, m, \sigma) \notin \mathcal{Q}$' by '$(i, m) \notin \mathcal{Q}$' in step 5.

The following proposition is an adaptation of a result from [3] stating that a strongly unforgeable MAC remains secure if we allow the adversary to try q forgery attempts.

Proposition 13. Let $\varepsilon > 0$ be a real, $\tau \geqslant 1$ and $q \geqslant 0$ be integers, and $\Sigma = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ be an ε-strongly secure τ-time MACC with key space $\mathcal{K}_{\mathrm{MACC}}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$. For all adversaries $\mathcal{A}$ (even unbounded), one has

$$\Pr\left(\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma}^{\tau\text{-time},\, q\text{-vrfy}} = 1\right) \leqslant q\varepsilon$$

where $\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma}^{\tau\text{-time},\, q\text{-vrfy}}$ is the following game:

1. The challenger runs KeyGen to obtain a key $k \in \mathcal{K}_{\mathrm{MACC}}$.
2. The challenger initializes the sets $\mathcal{Q}$ and $\mathcal{V}$ to the empty set $\emptyset$.
3. - $\mathcal{A}$ has access to an oracle $\mathcal{O}_{\mathrm{Mac}}$ that can be queried at most τ times. If the ith query of $\mathcal{A}$ to its oracle (starting counting from 0) is made on input $m \in \mathcal{M}$, the oracle runs $\sigma \leftarrow \mathrm{Mac}_{k,i}(m)$, returns σ to $\mathcal{A}$ and updates $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i, m, \sigma)\}$.
   - $\mathcal{A}$ has access to an oracle $\mathcal{O}_{\mathrm{Vrfy}}$ that can be queried at most q times and which, when queried on $(i, m, \sigma) \in \mathbb{N}_{<\tau} \times \mathcal{M} \times \mathcal{T}$, runs $b \leftarrow \mathrm{Vrfy}_{k,i}(m, \sigma)$, returns b to $\mathcal{A}$ and, if $b = 1$, updates $\mathcal{V} \leftarrow \mathcal{V} \cup \{(i, m, \sigma)\}$.
4. The output of the experiment is defined as 1 if and only if, at any time during the experiment, one has $\mathcal{V} \setminus \mathcal{Q} \neq \emptyset$.

Proof. The proof is an adaptation of the proof of the analogous result from [3]. Since $\mathcal{V}$ is only updated at the end of an $\mathcal{O}_{\mathrm{Vrfy}}$ query, $\mathcal{A}$ wins if and only if it makes a $\mathcal{O}_{\mathrm{Vrfy}}(i, m, \sigma)$ query for which $\mathrm{Vrfy}_{k,i}(m, \sigma) = 1$ and $(i, m, \sigma) \notin \mathcal{Q}$. For an integer $z \in [0, q-1]$, let E_z be the event that $\mathcal{A}$ wins and the first winning $\mathcal{O}_{\mathrm{Vrfy}}$ query is the zth one (starting counting from 0). One has

$$\left(\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma}^{\tau\text{-time},\, q\text{-vrfy}} = 1\right) = \bigvee_{z=0}^{q-1} E_z$$

so that, by the union bound,

$$\Pr\left(\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma}^{\tau\text{-time},\, q\text{-vrfy}} = 1\right) \leqslant \sum_{z=0}^{q-1} \Pr\left(E_z\right).$$

It thus remains to prove that $\Pr(E_z) \leqslant \varepsilon$ for each $0 \leqslant z \leqslant q-1$. Given such a z, we construct an adversary $\mathcal{B}_z$ against the ε-strong security of Σ as in Definition 12 such that $\Pr\left(E_z\right) \leqslant \Pr\left(\mathrm{MaccStrongForge}_{\mathcal{B}_z,\Sigma}^{\tau\text{-time}} = 1\right) \leqslant \varepsilon$. The adversary $\mathcal{B}_z$ initiates its own set $\mathcal{Q}$ as the empty set and a counter CTR as 0. It then runs $\mathcal{A}$ and answers its $\mathcal{O}_{\mathrm{Mac}}$ queries simply using its own $\mathcal{O}_{\mathrm{Mac}}$ oracle and updating its own copy of $\mathcal{Q}$ as $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(i, m, \sigma)\}$ as in Step 3 of Definition 12. It answers queries $\mathcal{O}_{\mathrm{Vrfy}}(i, m, \sigma)$ of $\mathcal{A}$ (where $i \in \mathbb{N}_{<\tau}$, $m \in \mathcal{M}$ and $\sigma \in \mathcal{T}$) as follows:

if CTR $< z$ **and** $(i, m, \sigma) \in \mathcal{Q}$
  return 1 to $\mathcal{A}$
if CTR $< z$ **and** $(i, m, \sigma) \notin \mathcal{Q}$
  return 0 to $\mathcal{A}$
if CTR $= z$
  output (i, m, σ) **and stop**
CTR++

As long as E_z is still possible (i.e., CTR $\leqslant z$ and $\mathcal{A}$ has not won yet), $\mathcal{B}_z$ perfectly simulates $\mathcal{A}$. Indeed, if CTR $< z$ and $(i, m, \sigma) \in \mathcal{Q}$, then $\mathrm{Vrfy}_{k,i}(m, \sigma) = 1$ since our MACCs are considered to be correct. If CTR $< z$ and $(i, m, \sigma) \notin \mathcal{Q}$ and if $\mathcal{A}$ does not win at this query, one has $\mathrm{Vrfy}_{k,i}(m, \sigma) = 0$. Moreover, if E_z occurs, then $\mathcal{B}_z$ wins, proving that $\Pr(E_z) \leqslant \Pr\left(\mathrm{MaccStrongForge}_{\mathcal{B}_z,\Sigma}^{\tau\text{-time}} = 1\right) \leqslant \varepsilon$ as desired. □

In [26], a $\frac{1}{|\mathcal{T}|}$-strongly secure τ-time MACC was constructed from a strongly universal$_2$ hash function $f\colon \mathcal{K} \times \mathcal{M} \to \mathcal{T}$. We generalize here this construction to the case of an ε-difference universal function, as introduced in [21] (under the name 'ε-otp-secure family of hash functions').

Definition 14 (ε-difference universal function)**.** Let $\varepsilon > 0$ and let $f\colon \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a function where $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{T}$ are finite sets with $|\mathcal{K}| \geqslant 1$, $|\mathcal{M}| \geqslant 2$ and $\mathcal{T}$ admits a group structure, written additively. We say that f is an *ε-difference universal function* if for all distinct $m, m' \in \mathcal{M}$ and all $\Delta \in \mathcal{T}$,

$$\Pr_{k \xleftarrow{U} \mathcal{K}} (f_k(m) - f_k(m') = \Delta) \leqslant \varepsilon.$$

From the above definition, we deduce that, given any two distinct $m, m' \in \mathcal{M}$, one has

$$1 = \Pr_{k \xleftarrow{U} \mathcal{K}} \left(\bigvee_{\Delta \in \mathcal{T}} (f_k(m) - f_k(m') = \Delta) \right) \leqslant \sum_{\Delta \in \mathcal{T}} \Pr_{k \xleftarrow{U} \mathcal{K}} (f_k(m) - f_k(m') = \Delta) \leqslant |\mathcal{T}| \cdot \varepsilon,$$

so that $\frac{1}{|\mathcal{T}|} \leqslant \varepsilon$.

Example 15. Let $a \geqslant b > 0$ be integers. It is well-known that the additive group of $\mathbb{F}_2^a$ is isomorphic to the additive group of $\mathbb{F}_{2^a}$. Let $i\colon \mathbb{F}_2^a \to \mathbb{F}_{2^a}$ be such an isomorphism. Let also $\mathrm{p}_b\colon \mathbb{F}_2^a \to \mathbb{F}_2^b$ be the projection to the first b components. The function

$$\begin{aligned} f\colon \mathbb{F}_2^a \times \mathbb{F}_2^a &\longrightarrow \mathbb{F}_2^b \\ (k, m) &\longmapsto f_k(m) = \mathrm{p}_b\left(i^{-1}\left(i(k) \cdot i(m)\right)\right) \end{aligned}$$

is $\frac{1}{2^b}$-difference universal.

Example 16. Let $b \geqslant a > 0$ be integers. Let $i\colon \mathbb{F}_2^b \to \mathbb{F}_{2^b}$ be an isomorphism of additive groups. Let also $j\colon \mathbb{F}_2^a \to \mathbb{F}_2^b$ be an injective (linear) map. The function

$$\begin{aligned} f\colon \mathbb{F}_2^b \times \mathbb{F}_2^a &\longrightarrow \mathbb{F}_2^b \\ (k, m) &\longmapsto f_k(m) = i^{-1}\left(i(k) \cdot i(j(m))\right) \end{aligned}$$

is $\frac{1}{2^b}$-difference universal.

The following is a direct generalization of the construction of a $\frac{1}{|\mathcal{T}|}$-strongly secure τ-time MACC from a strongly universal$_2$ function from [26], to the case of an ε-difference universal function.

Proposition 17. Let $\varepsilon > 0$ be a real and $\tau \geqslant 1$ be an integer. Let also $f\colon \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be an ε-difference universal function. Then, one can construct an ε-strongly secure τ-time MACC $\Sigma_f^\tau = (\mathrm{KeyGen}, \mathrm{Mac}, \mathrm{Vrfy})$ as follows:

- The key space is $\mathcal{K}_{\mathrm{MACC}} = \mathcal{K} \times \mathcal{T}^\tau$, the message space is $\mathcal{M}$, and the tag space is $\mathcal{T}$.
- KeyGen uniformly selects a key $(\kappa, \kappa'_0, \kappa'_1, \ldots, \kappa'_{\tau-1})$ in $\mathcal{K}_{\mathrm{MACC}}$.
- Given a key $k = (\kappa, \kappa'_0, \kappa'_1, \ldots, \kappa'_{\tau-1}) \in \mathcal{K}_{\mathrm{MACC}}$, a counter $i \in \mathbb{N}_{<\tau}$ and a message $m \in \mathcal{M}$, Mac deterministically outputs the tag $\sigma = \mathrm{Mac}_{k,i}(m) = f_\kappa(m) + \kappa'_i \in \mathcal{T}$.
- Given a key $k \in \mathcal{K}_{\mathrm{MACC}}$, a counter $i \in \mathbb{N}_{<\tau}$, a message $m \in \mathcal{M}$, and a tag $\sigma \in \mathcal{T}$, $\mathrm{Vrfy}_{k,i}(m, \sigma) = 1$ if and only if $\sigma = \mathrm{Mac}_{k,i}(m)$.

Proof. The correctness of the MACC is trivially verified. To prove that Σ_f^τ satisfies Definition 12, let $\mathcal{A}$ be an adversary attacking Σ_f^τ in the experiment $\mathrm{MaccStrongForge}_{\mathcal{A},\Sigma_f^\tau}^{\tau\text{-time}}$. Suppose that the key obtained by the challenger in the first step is $k = (\kappa, \kappa'_0, \kappa'_1, \ldots, \kappa'_{\tau-1})$. Suppose also that $\mathcal{A}$ calls

its oracle $\tau' \leqslant \tau$ times on inputs $m_0, m_1, \ldots, m_{\tau'-1}$, and respectively gets the tags $\sigma_0, \sigma_1, \ldots, \sigma_{\tau'-1}$ where $\sigma_i = f_\kappa(m_i) + \kappa'_i$. Let us denote by (j, m, σ) the output of $\mathcal{A}$.

If $j \geqslant \tau'$, κ'_j is a uniform element of $\mathcal{T}$, completely independent of $\mathcal{A}$'s view. Therefore, in that case, the probability that $\mathcal{A}$ succeeds, i.e., that $\sigma = f_\kappa(m) + \kappa'_j$, is exactly $\frac{1}{|\mathcal{T}|} \leqslant \varepsilon$.

If $j < \tau'$, $\mathcal{A}$ succeeds if and only if $(j, m, \sigma) \notin \mathcal{Q}$ and $\sigma = f_\kappa(m) + \kappa'_j$, that is, if and only if $(m, \sigma) \neq (m_j, \sigma_j)$ and $\sigma = f_\kappa(m) - f_\kappa(m_j) + \sigma_j$. If $m = m_j$, one must have $\sigma \neq \sigma_j$ for $\mathcal{A}$ to win, but in that case $\sigma = f_\kappa(m) - f_\kappa(m_j) + \sigma_j = \sigma_j$, so $\mathcal{A}$ cannot win. Therefore, $\mathcal{A}$ wins if and only if $m \neq m_j$ and $\sigma = f_\kappa(m) - f_\kappa(m_j) + \sigma_j$. Since each κ'_i is a uniformly selected element of $\mathcal{T}$, each tag σ_i received by $\mathcal{A}$ is independent of the information $\mathcal{A}$ has at that time. In particular, at the end of the experiment, κ is still uniform in $\mathcal{A}$'s view. Therefore

$$\begin{aligned}
\Pr\left(\mathrm{MaccStrongForge}^{\tau\text{-time}}_{\mathcal{A},\Sigma^\tau_f} = 1 \,|\, j < \tau'\right) &= \Pr\left(\sigma - \sigma_j = f_\kappa(m) - f_\kappa(m_j) \,\wedge\, m \neq m_j \,|\, j < \tau'\right) \\
&\leqslant \Pr\left(\Delta_j = f_\kappa(m) - f_\kappa(m_j) \,|\, j < \tau' \,\wedge\, m \neq m_j\right) \\
&\leqslant \varepsilon,
\end{aligned}$$

where we introduced $\Delta_j = \sigma - \sigma_j$. The last inequality above can be formally proved as follows, using the fact that f is an ε-difference universal function:

$$\begin{aligned}
&\Pr\left(\Delta_j = f_\kappa(m) - f_\kappa(m_j) \,|\, j < \tau' \,\wedge\, m \neq m_j\right) \\
&= \sum_{\substack{\overline{j},\overline{\tau'} \\ \overline{m},\overline{m_j} \\ \overline{\Delta_j}}} \Pr\left(\Delta_j = f_\kappa(m) - f_\kappa(m_j) \,|\, j < \tau' \,\wedge\, m \neq m_j \,\wedge\, (j, \tau', m, m_j, \Delta_j) = (\overline{j}, \overline{\tau'}, \overline{m}, \overline{m_j}, \overline{\Delta_j})\right) \\
&\qquad\qquad \cdot \Pr\left((j, \tau', m, m_j, \Delta_j) = (\overline{j}, \overline{\tau'}, \overline{m}, \overline{m_j}, \overline{\Delta_j}) \,|\, j < \tau' \,\wedge\, m \neq m_j\right) \\
&= \sum_{\substack{\overline{j} < \overline{\tau'} \\ \overline{m} \neq \overline{m_j} \\ \overline{\Delta_j}}} \Pr\left(\overline{\Delta_j} = f_\kappa(\overline{m}) - f_\kappa(\overline{m_j}) \,|\, (j, \tau', m, m_j, \Delta_j) = (\overline{j}, \overline{\tau'}, \overline{m}, \overline{m_j}, \overline{\Delta_j})\right) \\
&\qquad\qquad \cdot \Pr\left((j, \tau', m, m_j, \Delta_j) = (\overline{j}, \overline{\tau'}, \overline{m}, \overline{m_j}, \overline{\Delta_j}) \,|\, j < \tau' \,\wedge\, m \neq m_j\right) \\
&= \sum_{\substack{\overline{j} < \overline{\tau'} \\ \overline{m} \neq \overline{m_j} \\ \overline{\Delta_j}}} \Pr_{\kappa \xleftarrow{\mathrm{U}} \mathcal{K}}\left(\overline{\Delta_j} = f_\kappa(\overline{m}) - f_\kappa(\overline{m_j})\right) \\
&\qquad\qquad \cdot \Pr\left((j, \tau', m, m_j, \Delta_j) = (\overline{j}, \overline{\tau'}, \overline{m}, \overline{m_j}, \overline{\Delta_j}) \,|\, j < \tau' \,\wedge\, m \neq m_j\right) \\
&\leqslant \sum_{\substack{\overline{j} < \overline{\tau'} \\ \overline{m} \neq \overline{m_j} \\ \overline{\Delta_j}}} \varepsilon \cdot \Pr\left((j, \tau', m, m_j, \Delta_j) = (\overline{j}, \overline{\tau'}, \overline{m}, \overline{m_j}, \overline{\Delta_j}) \,|\, j < \tau' \,\wedge\, m \neq m_j\right) \\
&\leqslant \varepsilon
\end{aligned}$$

where $\overline{j}$, $\overline{\tau'}$, $\overline{m}$, $\overline{m_j}$, and $\overline{\Delta_j}$ are the realizations of the random variables j, τ', m, m_j, and Δ_j respectively.

Combining the above results, we get

$$\begin{aligned}
\Pr\left(\mathrm{MaccStrongForge}^{\tau\text{-time}}_{\mathcal{A},\Sigma^\tau_f} = 1\right) &= \Pr\left(\mathrm{MaccStrongForge}^{\tau\text{-time}}_{\mathcal{A},\Sigma^\tau_f} = 1 \,|\, j \geqslant \tau'\right) \Pr\left(j \geqslant \tau'\right) \\
&\quad + \Pr\left(\mathrm{MaccStrongForge}^{\tau\text{-time}}_{\mathcal{A},\Sigma^\tau_f} = 1 \,|\, j < \tau'\right) \Pr\left(j < \tau'\right) \\
&\leqslant \varepsilon \cdot \Pr\left(j \geqslant \tau'\right) + \varepsilon \cdot \Pr\left(j < \tau'\right) \\
&= \varepsilon
\end{aligned}$$

as desired. □

Example 18. Given integers $a \geqslant b > 0$ and $\tau \geqslant 1$, instantiating Proposition 17 with the $\frac{1}{2^b}$-difference universal function $f\colon \mathbb{F}_2^a \times \mathbb{F}_2^a \to \mathbb{F}_2^b$ from Example 15 gives a $\frac{1}{2^b}$-strongly secure τ-time MACC with key space $\mathcal{K}_{\mathrm{MACC}} = \mathbb{F}_2^{a+\tau b}$, message space $\mathcal{M} = \mathbb{F}_2^a$ and tag space $\mathcal{T} = \mathbb{F}_2^b$.

2.4 Nonce generators

In opposite to LP2+ from [16], the protocol ITSAKE does not require nonces. However, we choose to give the possibility to use one if fresher session keys are desired. We thus recall here the notion of nonce generator.

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*.

As examples of nonce generator GenN, one can cite a stateless uniform selection of an element in a finite non-empty set $\mathcal{N}$; or a cyclic counter, where $\mathcal{N} = \mathcal{S} = \mathbb{Z}_n$ (for some $n \in \mathbb{N}_{\geqslant 2}$) and where $N \leftarrow \mathrm{GenN}(\mathrm{st})$ outputs $N = \mathrm{st}$ and updates the state as $\mathrm{st} \leftarrow \mathrm{st} + 1$. As nonces are not required in ITSAKE, one could also take a stateless nonce generator where both $\mathcal{N}$ and $\mathcal{S}$ are singleton sets.

3 The protocol ITSAKE

3.1 The general case

We now describe our information-theoretically secure AKE protocol ITSAKE in its full generality. It aims to establish a session key between any pair of two distinct parties A and B among a set $\{P_1, \ldots, P_M\}$ of $M \geqslant 2$ parties. Each successful run of the protocol establishes an n-bit key sk in the session key space $\mathcal{K}_\mathrm{s} = \mathbb{F}_2^n$. It is built from the following building blocks:

- an integer $n > 0$ representing the size of the session keys established;
- an integer $\tau > 0$ representing the maximum number of keys that can be established between two parties in some direction before the master keys must be renewed;
- an integer $\ell > \log_2(\tau)$;
- a nonce generator GenN with nonce space $\mathcal{N}$ and state space $\mathcal{S}$;
- a real $\varepsilon > 0$ representing the security bound of authentication;
- an ε-strongly secure τ-time MACC $\Sigma^{\mathrm{init}} = (\mathrm{KeyGen}^{\mathrm{init}}, \mathrm{Mac}^{\mathrm{init}}, \mathrm{Vrfy}^{\mathrm{init}})$ with key space $\mathcal{K}_{\mathrm{MACC}}^{\mathrm{init}}$, message space $\mathcal{M}^{\mathrm{init}} \supseteq \{P_1, \ldots, P_M\} \times \mathbb{F}_2^\ell \times \mathcal{N}$, and tag space $\mathcal{T}^{\mathrm{init}}$, used by the initiator to authenticate its messages;
- an ε-strongly secure τ-time MACC $\Sigma^{\mathrm{resp}} = (\mathrm{KeyGen}^{\mathrm{resp}}, \mathrm{Mac}^{\mathrm{resp}}, \mathrm{Vrfy}^{\mathrm{resp}})$ with key space $\mathcal{K}_{\mathrm{MACC}}^{\mathrm{resp}}$, message space $\mathcal{M}^{\mathrm{resp}} \supseteq \{P_1, \ldots, P_M\} \times \mathbb{F}_2^\ell \times \mathcal{N}$, and tag space $\mathcal{T}^{\mathrm{resp}}$, used by the responder to authenticate its messages;
- a function $\mathrm{der}\colon \{P_1, \ldots, P_M\} \times \{P_1, \ldots, P_M\} \times (\mathbb{F}_2^\ell \times \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{init}}) \times (\mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{resp}}) \to \mathbb{F}_2^n$.

In the initialization phase of ITSAKE, one runs for each ordered pair of distinct parties $(A, B) \in \{P_1, \ldots, P_M\} \times \{P_1, \ldots, P_M\}$ the following steps independently:

- $K_{AB}^{\mathrm{init,otp}} = K_{BA}^{\mathrm{resp,otp}} \xleftarrow{\mathrm{U}} \mathbb{F}_2^{\tau n}$
- $K_{AB}^{\mathrm{init,mac}} = K_{BA}^{\mathrm{resp,vrfy}} \leftarrow \mathrm{KeyGen}^{\mathrm{init}}$
- $K_{AB}^{\mathrm{init,vrfy}} = K_{BA}^{\mathrm{resp,mac}} \leftarrow \mathrm{KeyGen}^{\mathrm{resp}}$
- $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} \leftarrow 0^\ell \in \mathbb{F}_2^\ell$
- $\mathrm{CTR}_{AB}^{\mathrm{init,mac}} \leftarrow 0^\ell \in \mathbb{F}_2^\ell$
- $\mathrm{CTR}_{BA}^{\mathrm{resp}} \leftarrow 0^\ell \in \mathbb{F}_2^\ell$
- $\mathrm{st}_{AB}^{\mathrm{initN}} \leftarrow \mathcal{S}$
- $\mathrm{st}_{BA}^{\mathrm{respN}} \leftarrow \mathcal{S}$

Considering Alice (A) is the initiator and Bob (B) the responder, the key $K_{AB}^{\mathrm{init,otp}} = K_{BA}^{\mathrm{resp,otp}}$ will serve for a one-time pad (OTP), the key $K_{AB}^{\mathrm{init,mac}} = K_{BA}^{\mathrm{resp,vrfy}}$ will be used by Alice to authenticate her messages and by Bob to check the tags, and the key $K_{BA}^{\mathrm{resp,mac}} = K_{AB}^{\mathrm{init,vrfy}}$ will be used by Bob to authenticate his messages and by Alice to check the tags. Still supposing Alice is the initiator,

$\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ is a counter to keep track of which part of $K_{AB}^{\mathrm{init,otp}}$ to use for the OTP, the counter $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ will be used by Alice to know how many messages she has already authenticated as an initiator, and the counter $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ will serve Bob both to keep track of which part of $K_{BA}^{\mathrm{resp,otp}}$ to use and as an upper bound on how many messages he has already authenticated as a responder. The state $\mathrm{st}_{AB}^{\mathrm{initN}}$ will be used by Alice to run GenN when she is the initiator, and the state $\mathrm{st}_{BA}^{\mathrm{respN}}$ will be used by Bob to run GenN when he is the responder. The way states in $\mathcal{S}$ are selected in the initialization phase is left unspecified for the sake of generality. This could, for instance, be a constant public value.

The counters CTR all live in $\mathbb{F}_2^\ell$ and are interpreted as natural numbers in $\mathbb{N}_{<2^\ell}$ using big endian notation. In particular, inequalities have to be understood by interpreting each counter that way as an integer in $\mathbb{N}_{<2^\ell}$. In addition, the instruction CTR++ means that 1 is added to CTR if thought as a natural number. Note that, if $\mathrm{CTR} = 1^\ell \in \mathbb{F}_2^\ell$ (i.e., CTR represents the natural number $2^\ell - 1$), the result of CTR++ is $0^\ell \in \mathbb{F}_2^\ell$ (i.e., the natural number 0), although we will take care of never reaching such a situation (see Lemma 20.3).

As required by our security model, the protocol is designed to be bidirectional, that is, each of A and B can initiate a protocol run. This is done by including the keys, counters and states for both the pairs (A, B) and (B, A) in the master keys MK_{AB} and MK_{BA}, so that each party can play the role of initiator or responder. Specifically, the master keys are initially set on both sides as

$$\mathrm{MK}_{AB} = \Big(K_{AB}^{\mathrm{init,otp}}, K_{AB}^{\mathrm{init,mac}}, K_{AB}^{\mathrm{init,vrfy}}, \mathrm{CTR}_{AB}^{\mathrm{init,otp}}, \mathrm{CTR}_{AB}^{\mathrm{init,mac}}, \mathrm{st}_{AB}^{\mathrm{initN}}, \\ K_{AB}^{\mathrm{resp,otp}}, K_{AB}^{\mathrm{resp,mac}}, K_{AB}^{\mathrm{resp,vrfy}}, \mathrm{CTR}_{AB}^{\mathrm{resp}}, \mathrm{st}_{AB}^{\mathrm{respN}}\Big)$$

and

$$\mathrm{MK}_{BA} = \Big(K_{BA}^{\mathrm{init,otp}}, K_{BA}^{\mathrm{init,mac}}, K_{BA}^{\mathrm{init,vrfy}}, \mathrm{CTR}_{BA}^{\mathrm{init,otp}}, \mathrm{CTR}_{BA}^{\mathrm{init,mac}}, \mathrm{st}_{BA}^{\mathrm{initN}}, \\ K_{BA}^{\mathrm{resp,otp}}, K_{BA}^{\mathrm{resp,mac}}, K_{BA}^{\mathrm{resp,vrfy}}, \mathrm{CTR}_{BA}^{\mathrm{resp}}, \mathrm{st}_{BA}^{\mathrm{respN}}\Big).$$

Note that, although the counters and states can be publicly known, we include them in the master keys to give the session oracles access to them.

A run of the protocol ITSAKE is described in Figure 1 where Alice acts as the initiator and Bob as the responder. Let us make precise the following points:

- the master keys are indicated at the top of the figure. Some parts of these are not used by the protocol run (except if there is a key re-initialization). These unused parts of the master keys, which are required when the roles are swapped, are indicated in gray;
- for an initiator oracle π_i^s on Alice's side, A represents her identity P_i, B represents $\pi_i^s.\mathrm{pid}$ and st represents $\pi_i^s.\mathrm{st}$; while for a responder oracle π_j^t on Bob's side, B represents his identity P_j and $A = \pi_j^t.\mathrm{pid}$;
- 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 'sk $\leftarrow X$' for an oracle π_i^s means that '$\pi_i^s.\mathrm{sk} \leftarrow X$' and '$\pi_i^s.\alpha \leftarrow$ accepted' are executed;
- the variable st for an initiator oracle π_i^s represents the local state $\pi_i^s.\mathrm{st}$;
- an instruction **re-initialize master keys** for an oracle π_i^s with $\pi_i^s.\mathrm{pid} = P_j$ means that the execution of the query aborts, the master keys MK_{ij} and MK_{ji} must be re-generated as described above and, for all oracles π_h^u with $\{P_h, \pi_h^u.\mathrm{pid}\} = \{P_i, P_j\}$ and $\pi_h^u.\alpha = \text{negotiating}$, one sets $\pi_h^u.\alpha \leftarrow$ rejected;
- for given parties $A, B \in \{P_1, \ldots, P_M\}$ and tuples $m_A = (C_1, C_2, N_A, \sigma_A) \in \mathbb{F}_2^\ell \times \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{init}}$ and $m_B = (C_3, N_B, \sigma_B) \in \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{resp}}$, we sometimes use $\mathrm{der}(A, B, m_A, m_B)$ to denote $\mathrm{der}(A, B, C_1, C_2, N_A, \sigma_A, C_3, N_B, \sigma_B)$.

Alice	Bob
$\left(K_{AB}^{\text{init,otp}}, K_{AB}^{\text{init,mac}}, K_{AB}^{\text{init,vrfy}}, \text{CTR}_{AB}^{\text{init,otp}}, \text{CTR}_{AB}^{\text{init,mac}}, \text{st}_{AB}^{\text{initN}}, K_{AB}^{\text{resp,otp}}, K_{AB}^{\text{resp,mac}}, K_{AB}^{\text{resp,vrfy}}, \text{CTR}_{AB}^{\text{resp}}, \text{st}_{AB}^{\text{respN}}\right)$	$\left(K_{BA}^{\text{init,otp}}, K_{BA}^{\text{init,mac}}, K_{BA}^{\text{init,vrfy}}, \text{CTR}_{BA}^{\text{init,otp}}, \text{CTR}_{BA}^{\text{init,mac}}, \text{st}_{BA}^{\text{initN}}, K_{BA}^{\text{resp,otp}}, K_{BA}^{\text{resp,mac}}, K_{BA}^{\text{resp,vrfy}}, \text{CTR}_{BA}^{\text{resp}}, \text{st}_{BA}^{\text{respN}}\right)$

if $\text{CTR}_{AB}^{\text{init,mac}} \geqslant \tau$

re-initialize master keys

$N_A \leftarrow \text{GenN}\left(\text{st}_{AB}^{\text{initN}}\right)$

$\sigma_A \leftarrow \text{Mac}^{\text{init}}_{K_{AB}^{\text{init,mac}}, \text{CTR}_{AB}^{\text{init,mac}}}\left(A, \text{CTR}_{AB}^{\text{init,otp}}, N_A\right)$

$$m_A = \left(\text{CTR}_{AB}^{\text{init,mac}}, \text{CTR}_{AB}^{\text{init,otp}}, N_A, \sigma_A\right) \longrightarrow$$

$\text{CTR}_{AB}^{\text{init,mac}}$++

$\text{st} \leftarrow m_A$

if $\text{CTR}_{BA}^{\text{resp}} \geqslant \tau$

re-initialize master keys

if $m_A \notin \mathbb{F}_2^\ell \times \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\text{init}}$

abort

if $\text{CTR}_{AB}^{\text{init,mac}} \geqslant \tau$ **or** $\text{CTR}_{AB}^{\text{init,otp}} \geqslant \tau$

abort

if $\text{Vrfy}^{\text{init}}_{K_{BA}^{\text{resp,vrfy}}, \text{CTR}_{AB}^{\text{init,mac}}}\left(\left(A, \text{CTR}_{AB}^{\text{init,otp}}, N_A\right), \sigma_A\right) = 0$

abort

if $\text{CTR}_{AB}^{\text{init,otp}} > \text{CTR}_{BA}^{\text{resp}}$

$\mathit{length} \leftarrow n\left(\text{CTR}_{AB}^{\text{init,otp}} - \text{CTR}_{BA}^{\text{resp}}\right)$

$K_{BA}^{\text{resp,otp}}\left[n\text{CTR}_{BA}^{\text{resp}}, \mathit{length}\right] \leftarrow 0^{\mathit{length}}$

$\text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{AB}^{\text{init,otp}}$

$N_B \leftarrow \text{GenN}\left(\text{st}_{BA}^{\text{respN}}\right)$

$\sigma_B \leftarrow \text{Mac}^{\text{resp}}_{K_{BA}^{\text{resp,mac}}, \text{CTR}_{BA}^{\text{resp}}}\left(B, \text{CTR}_{AB}^{\text{init,mac}}, N_B\right)$

$$m_B = (\text{CTR}_{BA}^{\text{resp}}, N_B, \sigma_B) \longleftarrow$$

if $m_B \notin \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\text{resp}}$

abort

if $\text{CTR}_{BA}^{\text{resp}} \geqslant \tau$ **or** $\text{CTR}_{BA}^{\text{resp}} < \text{CTR}_{AB}^{\text{init,otp}}$

abort

$m_A = (C_1, C_2, N_A, \sigma_A) \leftarrow \text{st}$

if $\text{Vrfy}^{\text{resp}}_{K_{AB}^{\text{init,vrfy}}, \text{CTR}_{BA}^{\text{resp}}}\left(\left(B, C_1, N_B\right), \sigma_B\right) = 0$

abort

$\mathit{length} \leftarrow n\left(\text{CTR}_{BA}^{\text{resp}} - \text{CTR}_{AB}^{\text{init,otp}}\right)$

$K_{AB}^{\text{init,otp}}\left[n\text{CTR}_{AB}^{\text{init,otp}}, \mathit{length}\right] \leftarrow 0^{\mathit{length}}$

$\text{CTR}_{AB}^{\text{init,otp}} \leftarrow \text{CTR}_{BA}^{\text{resp}}$

$\text{sk} \leftarrow \text{der}\left(A, B, m_A, m_B\right) \oplus K_{AB}^{\text{init,otp}}\left[n\text{CTR}_{AB}^{\text{init,otp}}, n\right]$

$K_{AB}^{\text{init,otp}}\left[n\text{CTR}_{AB}^{\text{init,otp}}, n\right] \leftarrow 0^n$

$\text{CTR}_{AB}^{\text{init,otp}}$++

if $\text{CTR}_{AB}^{\text{init,otp}} \geqslant \tau$

re-initialize master keys

$\text{sk} \leftarrow \text{der}\left(A, B, m_A, m_B\right) \oplus K_{BA}^{\text{resp,otp}}\left[n\text{CTR}_{BA}^{\text{resp}}, n\right]$

$K_{BA}^{\text{resp,otp}}\left[n\text{CTR}_{BA}^{\text{resp}}, n\right] \leftarrow 0^n$

$\text{CTR}_{BA}^{\text{resp}}$++

Figure 1: The AKE protocol ITSAKE providing unconditional security.

3.2 An instantiation

We describe in this subsection a possible instantiation of our protocol ITSAKE. It depends on the following parameters:

- an even integer $n > 0$;
- an integer $\ell > 0$;
- an integer $w > 0$;
- a $\frac{1}{2^w}$-difference universal function $f\colon \mathbb{F}_2^{\ell'} \times \mathbb{F}_2^{2\ell+\frac{n}{2}} \to \mathbb{F}_2^w$ where $\ell' = \max\{2\ell + \frac{n}{2}, w\}$ (e.g. the one from Example 15 or Example 16 depending on whether $2\ell + \frac{n}{2}$ or w is larger).

This instantiation of ITSAKE will be denoted ITSAKE(f). Each run of the protocol aims to establish an n-bit session key between any pair of two distinct parties A and B among a set $\{P_1, \ldots, P_M\}$ of M parties, where we assume $2 \leqslant M \leqslant 2^\ell$. The integer w represents the security parameter. The protocol is bidirectional, i.e., both A and B can initiate the protocol. Between two given parties, the protocol is designed to establish a maximum of $2^\ell - 1$ keys in each direction before re-initialization of the master keys.

In ITSAKE(f), the building blocks of ITSAKE are instantiated as follows:

- $\tau = 2^\ell - 1$;
- the nonce generator GenN is a stateless uniform selection of an element of $\mathcal{N} = \mathbb{F}_2^{\frac{n}{2}}$;
- $\varepsilon = \frac{1}{2^w}$;
- $\Sigma^{\text{init}} = \Sigma^{\text{resp}} = \Sigma_f^\tau$ are the MACC from Proposition 17 instantiated with f;
- the function der$\colon \{P_1, \ldots, P_M\} \times \{P_1, \ldots, P_M\} \times (\mathbb{F}_2^\ell \times \mathbb{F}_2^\ell \times \mathbb{F}_2^{\frac{n}{2}} \times \mathbb{F}_2^w) \times (\mathbb{F}_2^\ell \times \mathbb{F}_2^{\frac{n}{2}} \times \mathbb{F}_2^w) \to \mathbb{F}_2^n$ sends $(A, B, C_1, C_2, N_A, \sigma_A, C_3, N_B, \sigma_B)$ to $N_A \| N_B$.

We propose such a choice for the function der in order to increase the key freshness and reduce the key control of each party. Notice that $\mathcal{K}^{\text{init}}_{\text{MACC}} = \mathcal{K}^{\text{resp}}_{\text{MACC}} = \mathbb{F}_2^{\ell'+\tau w}$, $\mathcal{M}^{\text{init}} = \mathcal{M}^{\text{resp}} = \mathbb{F}_2^{2\ell+\frac{n}{2}}$ and $\mathcal{T}^{\text{init}} = \mathcal{T}^{\text{resp}} = \mathbb{F}_2^w$. Since we assume $M \leqslant 2^\ell$, we can represent the identity of each party P_i as an element of $\mathbb{F}_2^\ell$ so that $\{P_1, \ldots, P_M\}$ embeds in $\mathbb{F}_2^\ell$ and $\mathcal{M}^{\text{init}}$ and $\mathcal{M}^{\text{resp}}$ effectively contain $\{P_1, \ldots, P_M\} \times \mathbb{F}_2^\ell \times \mathcal{N}$.

In view of the above, the initialization phase of ITSAKE(f) consists, for each ordered pair of distinct parties $(A, B) \in \{P_1, \ldots, P_M\} \times \{P_1, \ldots, P_M\}$, of the following steps:

- $K_{AB}^{\text{init,otp}} = K_{BA}^{\text{resp,otp}} \xleftarrow{\text{U}} \mathbb{F}_2^{\tau n}$
- $K_{AB}^{\text{init,mac}} = K_{BA}^{\text{resp,vrfy}} \xleftarrow{\text{U}} \mathbb{F}_2^{\ell'+\tau w}$
- $K_{AB}^{\text{init,vrfy}} = K_{BA}^{\text{resp,mac}} \xleftarrow{\text{U}} \mathbb{F}_2^{\ell'+\tau w}$
- $\text{CTR}_{AB}^{\text{init,otp}} \leftarrow 0^\ell$
- $\text{CTR}_{AB}^{\text{init,mac}} \leftarrow 0^\ell$
- $\text{CTR}_{BA}^{\text{resp}} \leftarrow 0^\ell$

The master keys are then initially set as

$$\text{MK}_{AB} = \Big(K_{AB}^{\text{init,otp}}, K_{AB}^{\text{init,mac}}, K_{AB}^{\text{init,vrfy}}, \text{CTR}_{AB}^{\text{init,otp}}, \text{CTR}_{AB}^{\text{init,mac}}, K_{AB}^{\text{resp,otp}}, K_{AB}^{\text{resp,mac}}, K_{AB}^{\text{resp,vrfy}}, \text{CTR}_{AB}^{\text{resp}}\Big) \in \mathbb{F}_2^{3\ell+4\ell'+2\tau n+4\tau w}$$

and

$$\text{MK}_{BA} = \Big(K_{BA}^{\text{init,otp}}, K_{BA}^{\text{init,mac}}, K_{BA}^{\text{init,vrfy}}, \text{CTR}_{BA}^{\text{init,otp}}, \text{CTR}_{BA}^{\text{init,mac}}, K_{BA}^{\text{resp,otp}}, K_{BA}^{\text{resp,mac}}, K_{BA}^{\text{resp,vrfy}}, \text{CTR}_{BA}^{\text{resp}}\Big) \in \mathbb{F}_2^{3\ell+4\ell'+2\tau n+4\tau w}.$$

Each master key is thus $(2n+4w)\cdot 2^\ell - 4w + 11\ell$ bits long if $\ell \geqslant \frac{w}{2} - \frac{n}{4}$, or $(2n+4w)\cdot 2^\ell - 2n + 3\ell$ bits long if $\ell \leqslant \frac{w}{2} - \frac{n}{4}$. For large ℓ, the master key length is approximately $2\tau(n+2w)$. The ratio

between the master key size and the maximal total length of derived session keys is thus about $1 + \frac{2w}{n}$ for large ℓ. Therefore, it mainly depends on the ratio $\frac{w}{n}$, as illustrated by the two examples below.

Let us discuss two different applications of ITSAKE(f). Since its security is unconditional and since the security of a solution is as weak as its weakest point, we recommend to use the established session keys with an ITS scheme. In that case, we need session keys to be at least as long as the messages. Hence, one needs n to be much larger than the security parameter w; and for ℓ large enough, the master key size is approximately $\tau \cdot 2n$, that is about the same size as the total amount of keying material established (since up to τ keys can be established in both directions). For instance, if two parties want to establish the necessary keying material to information-theoretically encrypt one million 1 kB blocks in each direction (so two million blocks in total), one can take $n = 2^{13}$, $\ell = 20$ and $w = 128$. In this scenario, the size of master keys is less than 2.22 GB for more than 2.14 GB of total length of session keys that can be established.

As discussed in the Introduction, it can also make sense to use the keys established via ITSAKE(f) with a computationally secure symmetric-key algorithm. In that case, one can e.g. set $w = n$. For a large ℓ, one has that the master key is about 3 times as big as the total size of the session keys that can be established. For instance, if two parties want to establish a billion keys in each direction (so two billion in total), each key consisting of 128 bits, one can take $n = w = 128$ and $\ell = 30$. In that case, each party has to store a master key of about 103 GB for more than 34.3 GB of session keys that can be established.

A run of the protocol ITSAKE(f) is described in Figure 2 where Alice (A) acts as the initiator and Bob (B) as the responder. It consists of a first message of $2\ell + \frac{n}{2} + w$ bits and a second message of $\ell + \frac{n}{2} + w$ bits.

We conclude this section by summarizing, and particularizing to the case of ITSAKE(f), the security theorems that will be proved in Section 4.

Corollary 19. Let n, ℓ, w and M be four positive integers such that n is even and $2 \leqslant M \leqslant 2^\ell$. Let also $f\colon \mathbb{F}_2^{\ell'} \times \mathbb{F}_2^{2\ell+\frac{n}{2}} \to \mathbb{F}_2^w$ be a $\frac{1}{2^w}$-difference universal function where $\ell' = \max\{2\ell + \frac{n}{2}, w\}$. Let $\mathcal{A}$ be an adversary attacking the protocol ITSAKE(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. One has the following bounds:

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{Correct}}_{\mathrm{ITSAKE}(f)}(\mathcal{A}) &\leqslant 8M(M-1)\frac{q^2}{2^w} \\
\mathrm{Adv}^{\mathrm{wSR}}_{\mathrm{ITSAKE}(f)}(\mathcal{A}) &\leqslant 2M(M-1)\frac{q}{2^w} \quad \text{if } \ell \geqslant \log_2(q+2) \\
\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{ITSAKE}(f)}(\mathcal{A}) &\leqslant 18M(M-1)\frac{q^2}{2^w} \\
\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{ITSAKE}(f)}(\mathcal{A}) &\leqslant 2M(M-1)\frac{q_\tau q}{2^w}
\end{aligned}$$

where $q_\tau = 4\left\lfloor \frac{q}{2^\ell - 1} \right\rfloor + 1$.

Proof. This is an immediate consequence of Theorems 26, 27, 28 and 30. □

4 Security theorems

This section is devoted to the statement and the proof of the security theorems of ITSAKE. We first start with the lemmas necessary for the main proofs. The correctness is then proved in Theorem 26, the weak synchronization robustness in Theorem 27, the entity authentication in Theorem 28, and finally the key indistinguishability in Theorem 30.

4.1 Technical lemmas

Given two distinct parties $A, B \in \{P_1, \ldots, P_M\}$ and an adversary $\mathcal{A}$ attacking ITSAKE, a *master key period for* $\{A, B\}$ is the period of time between the initialization phase and the first time an

Alice	Bob
$\left(K_{AB}^{\text{init,otp}}, K_{AB}^{\text{init,mac}}, K_{AB}^{\text{init,vrfy}}, \text{CTR}_{AB}^{\text{init,otp}}, \text{CTR}_{AB}^{\text{init,mac}}, K_{AB}^{\text{resp,otp}}, K_{AB}^{\text{resp,mac}}, K_{AB}^{\text{resp,vrfy}}, \text{CTR}_{AB}^{\text{resp}}\right)$	$\left(K_{BA}^{\text{init,otp}}, K_{BA}^{\text{init,mac}}, K_{BA}^{\text{init,vrfy}}, \text{CTR}_{BA}^{\text{init,otp}}, \text{CTR}_{BA}^{\text{init,mac}}, K_{BA}^{\text{resp,otp}}, K_{BA}^{\text{resp,mac}}, K_{BA}^{\text{resp,vrfy}}, \text{CTR}_{BA}^{\text{resp}}\right)$

if $\text{CTR}_{AB}^{\text{init,mac}} = 1^\ell \in \mathbb{F}_2^\ell$

re-initialize master keys

$N_A \xleftarrow{\text{U}} \mathbb{F}_2^{\frac{n}{2}}$

$\sigma_A \leftarrow f_{K_{AB}^{\text{init,mac}}[0,\ell']}\left(A \| \text{CTR}_{AB}^{\text{init,otp}} \| N_A\right) \oplus K_{AB}^{\text{init,mac}}\left[\ell' + w\text{CTR}_{AB}^{\text{init,mac}}, w\right]$

$$m_A = \text{CTR}_{AB}^{\text{init,mac}} \| \text{CTR}_{AB}^{\text{init,otp}} \| N_A \| \sigma_A \in \mathbb{F}_2^{2\ell + \frac{n}{2} + w} \longrightarrow$$

$\text{CTR}_{AB}^{\text{init,mac}}$++

$\text{st} \leftarrow m_A$

if $\text{CTR}_{BA}^{\text{resp}} = 1^\ell \in \mathbb{F}_2^\ell$

re-initialize master keys

if $m_A \notin \mathbb{F}_2^{2\ell + \frac{n}{2} + w}$

abort

decompose m_A as $m_A = C_1 \| C_2 \| N_A \| \sigma_A$

with $|C_1| = |C_2| = \ell$, $|N_A| = \frac{n}{2}$ and $|\sigma_A| = w$

if $C_1 = 1^\ell \in \mathbb{F}_2^\ell$ **or** $C_2 = 1^\ell \in \mathbb{F}_2^\ell$

abort

if $\sigma_A \neq f_{K_{BA}^{\text{resp,vrfy}}[0,\ell']}\left(A \| C_2 \| N_A\right) \oplus K_{BA}^{\text{resp,vrfy}}\left[\ell' + wC_1, w\right]$

abort

if $C_2 > \text{CTR}_{BA}^{\text{resp}}$

$K_{BA}^{\text{resp,otp}}\left[n\text{CTR}_{BA}^{\text{resp}}, n\left(C_2 - \text{CTR}_{BA}^{\text{resp}}\right)\right] \leftarrow 0^{n\left(C_2 - \text{CTR}_{BA}^{\text{resp}}\right)}$

$\text{CTR}_{BA}^{\text{resp}} \leftarrow C_2$

$N_B \xleftarrow{\text{U}} \mathbb{F}_2^{\frac{n}{2}}$

$\sigma_B \leftarrow f_{K_{BA}^{\text{resp,mac}}[0,\ell']}\left(B \| C_1 \| N_B\right) \oplus K_{BA}^{\text{resp,mac}}\left[\ell' + w\text{CTR}_{BA}^{\text{resp}}, w\right]$

$$m_B = \text{CTR}_{BA}^{\text{resp}} \| N_B \| \sigma_B \in \mathbb{F}_2^{\ell + \frac{n}{2} + w} \longleftarrow$$

if $m_B \notin \mathbb{F}_2^{\ell + \frac{n}{2} + w}$

abort

decompose m_B as $m_B = C_3 \| N_B \| \sigma_B$

with $|C_3| = \ell$, $|N_B| = \frac{n}{2}$ and $|\sigma_B| = w$

if $C_3 = 1^\ell \in \mathbb{F}_2^\ell$ **or** $C_3 < \text{CTR}_{AB}^{\text{init,otp}}$

abort

if $\sigma_B \neq f_{K_{AB}^{\text{init,vrfy}}[0,\ell']}\left(B \| \text{st}[0,\ell] \| N_B\right) \oplus K_{AB}^{\text{init,vrfy}}\left[\ell' + wC_3, w\right]$

abort

$K_{AB}^{\text{init,otp}}\left[n\text{CTR}_{AB}^{\text{init,otp}}, n\left(C_3 - \text{CTR}_{AB}^{\text{init,otp}}\right)\right] \leftarrow 0^{n\left(C_3 - \text{CTR}_{AB}^{\text{init,otp}}\right)}$

$\text{CTR}_{AB}^{\text{init,otp}} \leftarrow C_3$

$\text{sk} \leftarrow \left(\text{st}[2\ell, \frac{n}{2}] \| N_B\right) \oplus K_{AB}^{\text{init,otp}}\left[n\text{CTR}_{AB}^{\text{init,otp}}, n\right]$

$K_{AB}^{\text{init,otp}}\left[n\text{CTR}_{AB}^{\text{init,otp}}, n\right] \leftarrow 0^n$

$\text{CTR}_{AB}^{\text{init,otp}}$++

if $\text{CTR}_{AB}^{\text{init,otp}} = 1^\ell \in \mathbb{F}_2^\ell$

re-initialize master keys

$\text{sk} \leftarrow (N_A \| N_B) \oplus K_{BA}^{\text{resp,otp}}\left[n\text{CTR}_{BA}^{\text{resp}}, n\right]$

$K_{BA}^{\text{resp,otp}}\left[n\text{CTR}_{BA}^{\text{resp}}, n\right] \leftarrow 0^n$

$\text{CTR}_{BA}^{\text{resp}}$++

Figure 2: The AKE protocol ITSAKE(f) providing unconditional security.

instruction **re-initialize master keys** is triggered by an oracle π_i^s with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$, between two consecutive such re-initializations, or between the last such (re-)initialization and the end of the run of $\mathcal{A}$.

Our first lemma is quite easy to prove, but it will be used many times throughout the various proofs. For that reason, we may sometimes omit to make an explicit reference to it when using it.

Lemma 20. Consider an adversary $\mathcal{A}$ attacking ITSAKE: 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 Subsection 2.2. Let $A, B \in \{P_1, \ldots, P_M\}$ be two distinct parties. Then, the following properties hold.

1. For each oracle π_i^s that has accepted, with $P_i = A$, $\pi_i^s.\text{pid} = B$ and $\pi_i^s.\rho = \text{initiator}$, the queries NewSessionI(π_i^s, P_j) and Send(π_i^s, m_B) (for some message m_B) have been done during the same master key period for $\{A, B\}$.
2. At any time during the experiment, one has $K_{AB}^{\text{init,mac}} = K_{BA}^{\text{resp,vrfy}}$ and $K_{AB}^{\text{init,vrfy}} = K_{BA}^{\text{resp,mac}}$ and these keys do not evolve during a master key period for $\{A, B\}$.
3. The counters $\text{CTR}_{AB}^{\text{init,otp}}$, $\text{CTR}_{AB}^{\text{init,mac}}$ and $\text{CTR}_{BA}^{\text{resp}}$ are non-decreasing during a master key period for $\{A, B\}$. In particular, they are never updated from $2^\ell - 1$ to 0 except during a master key re-initialization.
4. At any time during the experiment (except between an update of $K_{AB}^{\text{init,otp}}$ and the immediate next update of $\text{CTR}_{AB}^{\text{init,otp}}$, or between an update of $K_{BA}^{\text{resp,otp}}$ and the immediate next update of $\text{CTR}_{BA}^{\text{resp}}$), denoting by K the key $K_{AB}^{\text{init,otp}} = K_{BA}^{\text{resp,otp}}$ generated at the beginning of the current master key period for $\{A, B\}$, one has

$$K_{AB}^{\text{init,otp}} = 0^{nc} \,\|\, K[nc, n(\tau - c)] \qquad \text{where } c = \text{CTR}_{AB}^{\text{init,otp}},$$
$$K_{BA}^{\text{resp,otp}} = 0^{nc'} \,\|\, K[nc', n(\tau - c')] \qquad \text{where } c' = \text{CTR}_{BA}^{\text{resp}}.$$

Proof.

1. If there had been a re-initialization of the master keys for $\{A, B\}$ between the queries NewSessionI(π_i^s, P_j) and Send(π_i^s, m_B), then π_i^s would have rejected (and the Send query would not have been allowed). Therefore, this case is not possible.
2. No query of $\mathcal{A}$ can ever change $K_{AB}^{\text{init,mac}}$, $K_{BA}^{\text{resp,vrfy}}$, $K_{AB}^{\text{init,vrfy}}$ or $K_{BA}^{\text{resp,mac}}$, except if it provokes a change of master key period for $\{A, B\}$. Since the equalities $K_{AB}^{\text{init,mac}} = K_{BA}^{\text{resp,vrfy}}$ and $K_{AB}^{\text{init,vrfy}} = K_{BA}^{\text{resp,mac}}$ hold at the (re-)initialization phase, they remain true at all time.
3. Outside a master key re-initialization, the counter $\text{CTR}_{AB}^{\text{init,otp}}$ can only be updated as

$$\text{CTR}_{AB}^{\text{init,otp}} \leftarrow \text{CTR}_{BA}^{\text{resp}}$$
$$\text{CTR}_{AB}^{\text{init,otp}}{+}{+}$$

during a Send query. Moreover, such an update is realized on the condition that the received $\text{CTR}_{BA}^{\text{resp}}$ satisfies $\text{CTR}_{BA}^{\text{resp}} < \tau \leqslant 2^\ell - 1$ and $\text{CTR}_{BA}^{\text{resp}} \geqslant \text{CTR}_{AB}^{\text{init,otp}}$. These conditions ensure that these instructions are non-decreasing for $\text{CTR}_{AB}^{\text{init,otp}}$.

The only way $\text{CTR}_{AB}^{\text{init,mac}}$ can be updated outside a master key re-initialization is during a NewSessionI query. It is then updated as

$$\text{CTR}_{AB}^{\text{init,mac}}{+}{+}$$

under the condition that $\text{CTR}_{AB}^{\text{init,mac}} < \tau \leqslant 2^\ell - 1$, thus increasing $\text{CTR}_{AB}^{\text{init,mac}}$.

Finally, outside a master key re-initialization, the counter $\text{CTR}_{BA}^{\text{resp}}$ can only be updated during a NewSessionR query for which $\text{CTR}_{BA}^{\text{resp}}$ and the received $\text{CTR}_{AB}^{\text{init,otp}}$ satisfy $\text{CTR}_{BA}^{\text{resp}} < \tau$ and $\text{CTR}_{AB}^{\text{init,otp}} < \tau$, and this is realized as

$$\textbf{if } \text{CTR}_{AB}^{\text{init,otp}} > \text{CTR}_{BA}^{\text{resp}}$$
$$\quad \text{CTR}_{BA}^{\text{resp}} \leftarrow \text{CTR}_{AB}^{\text{init,otp}}$$
$$\text{CTR}_{BA}^{\text{resp}}{+}{+}.$$

The instruction under the **if** condition updates $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ to $\max\{\mathrm{CTR}_{BA}^{\mathrm{resp}}, \mathrm{CTR}_{AB}^{\mathrm{init,otp}}\}$ and is clearly non-decreasing. Moreover, this updated value of $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ must satisfy $\mathrm{CTR}_{BA}^{\mathrm{resp}} < \tau \leqslant 2^{\ell} - 1$, making the last instruction non-decreasing.

4. Since the counters $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ are initialized to 0, the statement is clearly true at the beginning of the master key period for $\{A, B\}$. We first consider $K_{AB}^{\mathrm{init,otp}}$. Outside a master key re-initialization, the key $K_{AB}^{\mathrm{init,otp}}$ is always updated just before $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ is updated, and $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ is always updated just after $K_{AB}^{\mathrm{init,otp}}$ is updated. Moreover, when such updates are performed, they are of the form

$$\begin{aligned} K_{AB}^{\mathrm{init,otp}}\left[n\mathrm{CTR}_{AB}^{\mathrm{init,otp}}, nr\right] &\leftarrow 0^{nr} \\ \mathrm{CTR}_{AB}^{\mathrm{init,otp}} &\leftarrow \mathrm{CTR}_{AB}^{\mathrm{init,otp}} + r \end{aligned}$$

for some integer r. They are well-defined since $0 \leqslant r \leqslant \tau - \mathrm{CTR}_{AB}^{\mathrm{init,otp}}$. Indeed, in one case, one has $r = c' - \mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ where c' is the received value of $\mathrm{CTR}_{BA}^{\mathrm{resp}}$. Due to the **if** conditions, $0 \leqslant r < \tau$. In the other case, one has $r = 1$ and $0 < r \leqslant \tau - \mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ since, at that time, $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} = c' < \tau$. Therefore, if

$$\begin{aligned} K_{AB}^{\mathrm{init,otp}} &= 0^{nc} \,\|\, K[nc, n(\tau - c)] \\ &= 0^{nc} \,\|\, K[nc, nr] \,\|\, K[n(c+r), n(\tau - c - r)] \end{aligned}$$

with $c = \mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ before these updates, then, after the updates, one has

$$K_{AB}^{\mathrm{init,otp}} = 0^{n(c+r)} \,\|\, K[n(c+r), n(\tau - (c+r))]$$

and $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} = c + r$. Therefore, these updates do not break the validity of the announced equality. The identity linking $K_{BA}^{\mathrm{resp,otp}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ is proved similarly.

□

The next lemma bounds the probability that a message m_A output by an initiator is accepted by its responder in another master key period (earlier or later).

Lemma 21. Consider an adversary $\mathcal{A}$ attacking ITSAKE 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 Subsection 2.2. 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. We denote by OtherMKP the event that a message m_A is output by a $\mathrm{NewSessionI}(\pi_i^s, P_j)$ query and that, during another master key period for $\{P_i, P_j\}$ (earlier or later), a $\mathrm{NewSessionR}(\pi_j^t, P_i, m_A)$ query leads to $\pi_j^t.\alpha = \text{accepted}$. One has

$$\Pr(\mathrm{OtherMKP}) \leqslant 8M(M-1)q^2\varepsilon.$$

Proof. Given two distinct parties $A, B \in \{P_1, \ldots, P_M\}$, we notice that there are at most

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

Therefore, there are at most $4q$ oracles π_i^s with $\{P_i, \pi_i^s.\mathrm{pid}\} = \{A, B\}$. Moreover, a master key re-initialization can only be triggered once by each oracle (indeed, during a master key re-initialization, negotiating oracles between A and B automatically reject). Thus, there are at most $4q$ master key periods for $\{A, B\}$, omitting the $(4q+1)^{\text{th}}$ in which no session oracle between A and B can be initiated.

For an ordered pair $(A, B) = (P_i, P_j)$ of distinct parties and for an integer $p \in [2, 4q]$, let $\mathrm{OtherMKP}_{A,B,p}^{\mathrm{later}}$ be the event that $\mathcal{A}$ makes a $\mathrm{NewSessionI}(\pi_i^s, B)$ query during the $p'^{\,\text{th}}$ master key period for $\{A, B\}$ for some $p' < p$, that this query outputs a message m_A and that a

NewSessionR(π_j^t, A, m_A) query is made during the pth master key period for $\{A, B\}$ leading to $\pi_j^t.\alpha =$ accepted. Similarly, let $\text{OtherMKP}_{A,B,p}^{\text{earlier}}$ be the event that $\mathcal{A}$ makes a NewSessionR(π_j^t, A, m_A) query during the p'^{th} master key period for $\{A, B\}$ for some $p' < p$ leading to $\pi_j^t.\alpha =$ accepted and that a NewSessionI(π_i^s, B) query is made during the pth master key period for $\{A, B\}$ which outputs the message m_A. One has that

$$\text{OtherMKP} = \left(\bigvee_{(A,B)} \bigvee_{p=2}^{4q} \text{OtherMKP}_{A,B,p}^{\text{later}} \right) \vee \left(\bigvee_{(A,B)} \bigvee_{p=2}^{4q} \text{OtherMKP}_{A,B,p}^{\text{earlier}} \right)$$

so that, by the union bound,

$$\Pr\left(\text{OtherMKP}\right) \leqslant \sum_{(A,B)} \sum_{p=2}^{4q} \Pr\left(\text{OtherMKP}_{A,B,p}^{\text{later}}\right) + \sum_{(A,B)} \sum_{p=2}^{4q} \Pr\left(\text{OtherMKP}_{A,B,p}^{\text{earlier}}\right).$$

For A, B and p as above, we construct an adversary $\mathcal{B}_{A,B,p}^{\text{later}}$ attacking the MACC Σ^{init} in the experience $\text{MaccStrongForge}_{\mathcal{B}_{A,B,p}^{\text{later}},\Sigma^{\text{init}}}^{\tau\text{-time},\, q\text{-vrfy}}$ as in Proposition 13. This adversary starts by initializing the oracles of the parties $\{P_1, \ldots, P_M\}$ as usual and runs $\mathcal{A}$. At each master key (re-)initialization, up to (but not including) the pth one for $\{A, B\}$, it generates itself the master key MK_{ij} of the parties as prescribed by the protocol. Having all master keys, it can answer $\mathcal{A}$'s queries as its challenger would do. At the start of the pth master key period for $\{A, B\}$, it stops the execution of $\mathcal{A}$. Then, it considers the key of its challenger as the key $K_{AB}^{\text{init,mac}} = K_{BA}^{\text{resp,vrfy}}$ for the pth master key period for $\{A, B\}$. When this stop occurs, $\mathcal{B}_{A,B,p}^{\text{later}}$ queries its $\mathcal{O}_{\text{Vrfy}}$ oracle with the triples $(C_1, (A, C_2, N_A), \sigma_A) \in \mathbb{N}_{<\tau} \times \mathcal{M}^{\text{init}} \times \mathcal{T}^{\text{init}}$ for all oracles π_i^s with $P_i = A$, $\pi_i^s.\text{pid} = B$, $\pi_i^s.\rho =$ initiator and first message of its transcript of the form $(C_1, C_2, N_A, \sigma_A)$. We remark that $\mathcal{B}_{A,B,p}^{\text{later}}$ does not query its $\mathcal{O}_{\text{Mac}}$ oracle and queries at most q times its oracle $\mathcal{O}_{\text{Vrfy}}$ since there are at most q oracles π_i^s as above. Moreover, as far as $\mathcal{B}_{A,B,p}^{\text{later}}$ is running, it perfectly simulates $\mathcal{A}$. By construction, if $\text{OtherMKP}_{A,B,p}^{\text{later}}$ occurs in a normal execution of $\mathcal{A}$, then $\mathcal{B}_{A,B,p}^{\text{later}}$ wins in the corresponding simulation. Therefore,

$$\Pr\left(\text{OtherMKP}_{A,B,p}^{\text{later}}\right) \leqslant \Pr\left(\text{MaccStrongForge}_{\mathcal{B}_{A,B,p}^{\text{later}},\Sigma^{\text{init}}}^{\tau\text{-time},\, q\text{-vrfy}} = 1\right) \leqslant q\varepsilon$$

since Σ^{init} is ε-strongly secure (see Proposition 13).

Similarly, we construct an adversary $\mathcal{B}_{A,B,p}^{\text{earlier}}$ attacking the MACC Σ^{init} in the experience $\text{MaccStrongForge}_{\mathcal{B}_{A,B,p}^{\text{earlier}},\Sigma^{\text{init}}}^{\tau\text{-time},\, q\text{-vrfy}}$. This adversary begins by initializing the oracles of the parties $\{P_1, \ldots, P_M\}$ as usual and runs $\mathcal{A}$. At each master key (re-)initialization, up to (but not including) the pth one for $\{A, B\}$, it generates itself the master key MK_{ij} of the parties as prescribed by the protocol. Having all master keys, it can answer $\mathcal{A}$'s queries as its challenger would do. At the start of the pth master key period for $\{A, B\}$, it stops the execution of $\mathcal{A}$. Then, it considers the key of its challenger as the key $K_{AB}^{\text{init,mac}} = K_{BA}^{\text{resp,vrfy}}$ for the pth master key period for $\{A, B\}$. When this stop occurs, $\mathcal{B}_{A,B,p}^{\text{earlier}}$ queries its $\mathcal{O}_{\text{Vrfy}}$ oracle with the triples $(C_1, (A, C_2, N_A), \sigma_A) \in \mathbb{N}_{<\tau} \times \mathcal{M}^{\text{init}} \times \mathcal{T}^{\text{init}}$ for all oracles π_j^t with $P_j = B$, $\pi_j^t.\text{pid} = A$, $\pi_j^t.\rho =$ responder and first message of its transcript of the form $(C_1, C_2, N_A, \sigma_A) \in \mathbb{N}_{<\tau} \times \mathbb{N}_{<\tau} \times \mathcal{N} \times \mathcal{T}^{\text{init}}$. We remark that $\mathcal{B}_{A,B,p}^{\text{earlier}}$ does not query its $\mathcal{O}_{\text{Mac}}$ oracle and queries at most q times its oracle $\mathcal{O}_{\text{Vrfy}}$ since there are at most q oracles π_j^t as above. Moreover, as far as $\mathcal{B}_{A,B,p}^{\text{earlier}}$ is running, it perfectly simulates $\mathcal{A}$. Moreover, if $\text{OtherMKP}_{A,B,p}^{\text{earlier}}$ occurs in a normal execution of $\mathcal{A}$, this means that $\mathcal{A}$ made a query of the form NewSessionR(π_j^t, A, m_A) before the pth master key period for $\{A, B\}$, that this query led to $\pi_j^t.\alpha =$ accepted, and that $\mathcal{A}$ made a NewSessionI(π_i^s, B) query during the pth master key period for $\{A, B\}$ which output m_A. Since $\pi_j^t.\alpha =$ accepted, m_A must be on the form $m_A = (C_1, C_2, N_A, \sigma_A) \in \mathbb{N}_{<\tau} \times \mathbb{N}_{<\tau} \times \mathcal{N} \times \mathcal{T}^{\text{init}}$. Moreover, since NewSessionI(π_i^s, B) output m_A, the tag σ_A has been obtained as $\sigma_A \leftarrow \text{Mac}_{k,C_1}^{\text{init}}(A, C_2, N_A)$ where k is the key $K_{AB}^{\text{init,mac}} = K_{BA}^{\text{resp,vrfy}}$ used during the pth master key period for $\{A, B\}$. Since our MACCs are assumed to be correct (see Definition 11), this means that $\text{Vrfy}_{k,C_1}^{\text{init}}((A, C_2, N_A), \sigma_A) = 1$.

Therefore, if $\mathrm{OtherMKP}^{\mathrm{earlier}}_{A,B,p}$ occurs in a normal execution of $\mathcal{A}$, then $\mathcal{B}^{\mathrm{earlier}}_{A,B,p}$ wins in the corresponding simulation, implying that

$$\Pr\left(\mathrm{OtherMKP}^{\mathrm{earlier}}_{A,B,p}\right) \leqslant \Pr\left(\mathrm{MaccStrongForge}^{\tau\text{-time},\, q\text{-vrfy}}_{\mathcal{B}^{\mathrm{earlier}}_{A,B,p},\Sigma^{\mathrm{init}}} = 1\right) \leqslant q\varepsilon$$

since Σ^{init} is ε-strongly secure (see Proposition 13).

One thus has

$$\begin{aligned}\Pr\left(\mathrm{OtherMKP}\right) &\leqslant \sum_{(A,B)}\sum_{p=2}^{4q}\Pr\left(\mathrm{OtherMKP}^{\mathrm{later}}_{A,B,p}\right) + \sum_{(A,B)}\sum_{p=2}^{4q}\Pr\left(\mathrm{OtherMKP}^{\mathrm{earlier}}_{A,B,p}\right)\\ &\leqslant \sum_{(A,B)}\sum_{p=2}^{4q} q\varepsilon + \sum_{(A,B)}\sum_{p=2}^{4q} q\varepsilon\\ &\leqslant 8M(M-1)q^2\varepsilon\end{aligned}$$

as desired. □

The above lemma bounds the probability that an adversary breaks, in a certain sense, authentication across different master key periods. We also need to study the event of breaking authentication inside a master key period. We say that an adversary $\mathcal{A}$ attacking ITSAKE makes an *init-forgery* to an oracle π^t_j with $P_j = B$, $\pi^t_j.\mathrm{pid} = A$ and $\pi^t_j.\rho =$ responder when the following conditions are satisfied:

(a) $\mathcal{A}$ makes a NewSessionR query to the oracle π^t_j which is (at the end of this query) two-sided uncorrupted,

(b) this query passes the $\mathrm{Vrfy}^{\mathrm{init}}$ test of the oracle in the sense that the condition of the form "**if** $\mathrm{Vrfy}^{\mathrm{init}}_{K^{\mathrm{resp,vrfy}}_{BA},C}(m,\sigma) = 0$" specified by the protocol (see Figure 1) is reached and one has $\mathrm{Vrfy}^{\mathrm{init}}_{K^{\mathrm{resp,vrfy}}_{BA},C}(m,\sigma) = 1$,

(c) and the counter-message-tag triple (C, m, σ) that passes this $\mathrm{Vrfy}^{\mathrm{init}}$ test has not been obtained earlier in the same master key period for $\{A, B\}$ by an instruction $\sigma \leftarrow \mathrm{Mac}^{\mathrm{init}}_{K^{\mathrm{init,mac}}_{AB},C}(m)$ of an oracle π^s_i with $P_i = A$, $\pi^s_i.\mathrm{pid} = B$ and $\pi^s_i.\rho =$ initiator.

Similarly, we say that $\mathcal{A}$ makes a *resp-forgery* to an oracle π^s_i with $P_i = A$, $\pi^s_i.\mathrm{pid} = B$ and $\pi^s_i.\rho =$ initiator when the following conditions are satisfied:

(a) $\mathcal{A}$ makes a Send query to the oracle π^s_i which is (at the time of this query) two-sided uncorrupted,

(b) this query passes the $\mathrm{Vrfy}^{\mathrm{resp}}$ test of the oracle in the sense that the condition of the form "**if** $\mathrm{Vrfy}^{\mathrm{resp}}_{K^{\mathrm{init,vrfy}}_{AB},C}(m,\sigma) = 0$" specified by the protocol (see Figure 1) is reached and one has $\mathrm{Vrfy}^{\mathrm{resp}}_{K^{\mathrm{init,vrfy}}_{AB},C}(m,\sigma) = 1$,

(c) and the counter-message-tag triple (C, m, σ) that passes this $\mathrm{Vrfy}^{\mathrm{resp}}$ test has not been obtained earlier in the same master key period for $\{A, B\}$ by an instruction $\sigma \leftarrow \mathrm{Mac}^{\mathrm{resp}}_{K^{\mathrm{resp,mac}}_{BA},C}(m)$ of an oracle π^t_j with $P_j = B$, $\pi^t_j.\mathrm{pid} = A$ and $\pi^t_j.\rho =$ responder.

We say that $\mathcal{A}$ makes a *forgery* to an oracle π^s_i if $\mathcal{A}$ makes either a resp-forgery or an init-forgery to π^s_i.

Lemma 22. Consider an adversary $\mathcal{A}$ attacking ITSAKE 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 Subsection 2.2. As long as $\mathcal{A}$ has not made any forgery, for every oracle π^s_i that has accepted and which was two-sided uncorrupted when it accepted, there exists an oracle π^t_j activated in the same master key period for $\{P_i, \pi^s_i.\mathrm{pid}\}$ such that π^s_i has π^t_j as strong partner.

Proof. We denote by P the master key period for $\{P_i, \pi_i^s.\text{pid}\}$ during which π_i^s accepts.

Let us first treat the case $\pi_i^s.\rho = \text{initiator}$ and we denote $P_i = A$ and $\pi_i^s.\text{pid} = P_j = B$. Since $\pi_i^s.\alpha = \text{accepted}$, the second message of its transcript must be of the form $m_B = (C_3, N_B, \sigma_B)$ with $C_3 \in \mathbb{F}_2^\ell$, $N_B \in \mathcal{N}$, $\sigma_B \in \mathcal{T}^{\text{resp}}$ and $\text{Vrfy}^{\text{resp}}_{k_2,C_3}((B, C_1, N_B), \sigma_B) = 1$ where C_1, stored in $\pi_i^s.\text{st}$, was the value of the counter $\text{CTR}^{\text{init,mac}}_{AB}$ when π_i^s created its first message, and k_2 is the key $K^{\text{init,vrfy}}_{AB} = K^{\text{resp,mac}}_{BA}$ used during P. Since $\mathcal{A}$ has not made any resp-forgery, there exists a responder oracle π_j^t that produced earlier in P the tag $\sigma_B \leftarrow \text{Mac}^{\text{resp}}_{k_2,C_3}(B, C_1, N_B)$ and such that $\pi_j^t.\text{pid} = A$. Thus, it must have output the message m_B. Moreover, it must have been activated with a query $\text{NewSessionR}(\pi_j^t, A, m_A)$ where m_A is of the form $m_A = (C_1, C_2, N_A, \sigma_A)$ with $C_1, C_2 \in \mathbb{F}_2^\ell$, $N_A \in \mathcal{N}$, $\sigma_A \in \mathcal{T}^{\text{init}}$ and $\text{Vrfy}^{\text{init}}_{k_1,C_1}((A, C_2, N_A), \sigma_A) = 1$, where k_1 is the key $K^{\text{resp,vrfy}}_{BA} = K^{\text{init,mac}}_{AB}$ used during P, and C_1 is as the one stored in $\pi_i^s.\text{st}$ since it was used to compute the tag σ_B. Since $\mathcal{A}$ has not made any init-forgery, there exists an initiator oracle $\pi_i^{s'}$ that produced earlier in P the tag $\sigma_A \leftarrow \text{Mac}^{\text{init}}_{k_1,C_1}(A, C_2, N_A)$ and such that $\pi_i^{s'}.\text{pid} = B$. Thus, $\pi_i^{s'}$ must have output the message m_A. Since both π_i^s and $\pi_i^{s'}$ started to negotiate during P and used the same value C_1 for the counter $\text{CTR}^{\text{init,mac}}_{AB}$, by Lemma 20.3, this implies that $\pi_i^{s'} = \pi_i^s$ (i.e. $s' = s$). Therefore, one has $\text{T}_i^s = (m_A, m_B) = \text{T}_j^t$ 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.

It remains to treat the case $\pi_i^s.\rho = \text{responder}$. We denote $P_i = B$ and $\pi_i^s.\text{pid} = P_j = A$. Since $\pi_i^s.\alpha = \text{accepted}$, the first message of its transcript must be of the form $m_A = (C_1, C_2, N_A, \sigma_A)$ with $C_1, C_2 \in \mathbb{F}_2^\ell$, $N_A \in \mathcal{N}$, $\sigma_A \in \mathcal{T}^{\text{init}}$ and $\text{Vrfy}^{\text{init}}_{k_1,C_1}((A, C_2, N_A), \sigma_A) = 1$ where k_1 is the key $K^{\text{init,mac}}_{AB} = K^{\text{resp,vrfy}}_{BA}$ used during P. Thus, since $\mathcal{A}$ has not made any init-forgery, there exists an initiator oracle π_j^t that produced earlier in P the tag $\sigma_A \leftarrow \text{Mac}^{\text{init}}_{k_1,C_1}(A, C_2, N_A)$ and such that $\pi_j^t.\text{pid} = B$. Moreover, the message output by the query $\text{NewSessionI}(\pi_j^t, B)$ must be m_A. Therefore, π_j^t first produced the message m_A and then it was delivered to π_i^s which accepted and output a second message m_B. This means that π_i^s has π_j^t as strong partner, which concludes the proof. □

Lemma 23. Consider an adversary $\mathcal{A}$ attacking ITSAKE 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 Subsection 2.2. Let (A, B) be an ordered pair of distinct parties and let P be a master key period for $\{A, B\}$. At any time during P, as long as the conditions

1. A is not corrupted with respect to B,
2. B is not corrupted with respect to A,
3. $\mathcal{A}$ has not made any forgery to an oracle π_i^s with $\{P_i, \pi_i^s.\text{pid}\} = \{A, B\}$ during P

are satisfied, one has

$$\text{CTR}^{\text{init,otp}}_{AB} \leqslant x^{\mathsf{P}}_{B\leftarrow A} \quad \text{and} \quad \text{CTR}^{\text{resp}}_{BA} \leqslant x^{\mathsf{P}}_{B\leftarrow A}$$

where $x^{\mathsf{P}}_{B\leftarrow A}$ is the number of oracles π_j^t that accepted so far during P with $P_j = B$, $\pi_j^t.\text{pid} = A$ and $\pi_j^t.\rho = \text{responder}$.

Proof. At the start of the period P, the numbers $\text{CTR}^{\text{init,otp}}_{AB}$, $\text{CTR}^{\text{resp}}_{BA}$ and $x^{\mathsf{P}}_{B\leftarrow A}$ are all equal to 0 and thus the inequality

$$\max\{\text{CTR}^{\text{init,otp}}_{AB}, \text{CTR}^{\text{resp}}_{BA}\} \leqslant x^{\mathsf{P}}_{B\leftarrow A} \tag{1}$$

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 made during P.

For NewSessionR queries, only those of the form $\text{NewSessionR}(\pi_j^t, A, m_A)$ with $P_j = B$ may have an impact. Moreover, for such a query to have an impact, the first four **if** tests must be passed and in particular one must have $m_A = (C_1, C_2, N_A, \sigma_A)$ for some $C_1, C_2 \in \mathbb{F}_2^\ell$, $N_A \in \mathcal{N}$ and $\sigma_A \in \mathcal{T}^{\text{init}}$ with $\text{Vrfy}^{\text{init}}_{K^{\text{resp,vrfy}}_{BA},C_1}((A, C_2, N_A), \sigma_A) = 1$. During such a query, a new responder oracle of B with intended partner A accepts, therefore $x^{\mathsf{P}}_{B\leftarrow A}$ is increased by 1. Furthermore, $\text{CTR}^{\text{resp}}_{BA}$ is increased to $\max\{\text{CTR}^{\text{resp}}_{BA}, C_2\} + 1$ (due to the fifth **if** test and the last instruction) while $\text{CTR}^{\text{init,otp}}_{AB}$ does not change. By conditions 1, 2, 3, and by definition of an init-forgery to π_j^t, we know that, in that case, there exists an oracle π_i^s that previously produced the tag $\sigma_A \leftarrow \text{Mac}^{\text{init}}_{K^{\text{init,mac}}_{AB},C_1}(A, C_2, N_A)$

during the same master key period P and such that $P_i = A$, $\pi_i^s.\mathrm{pid} = B$ and $\pi_i^s.\rho = \text{initiator}$. By Lemma 20.3, this implies that $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ is at least C_2 at the time of the considered NewSessionR query. Therefore, the left-hand side of (1) 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, the first three **if** tests must be passed and in particular one must have $m_B = (C_3, N_B, \sigma_B)$ for some $C_3 \in \mathbb{F}_2^\ell$, $N_B \in \mathcal{N}$ and $\sigma_B \in \mathcal{T}^{\mathrm{resp}}$ with $C_3 \geqslant \mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ and $\mathrm{Vrfy}^{\mathrm{resp}}_{K_{AB}^{\mathrm{init,vrfy}},C_3}((B, C_1, N_B), \sigma_B) = 1$ where C_1, stored in $\pi_i^s.\mathrm{st}$, was the value of the counter $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ when π_i^s created its first message. During such a Send query, $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ is increased to $C_3 + 1$ while $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ and $x_{B\leftarrow A}^{\mathsf{P}}$ do not change. By conditions 1, 2, 3, and by definition of a resp-forgery to π_i^s, we know that, in that case, there exists an oracle π_j^t that previously produced the tag $\sigma_B \leftarrow \mathrm{Mac}^{\mathrm{resp}}_{K_{BA}^{\mathrm{resp,mac}},C_3}(B, C_1, N_B)$ during the same master key period P and such that $P_j = B$, $\pi_j^t.\mathrm{pid} = A$ and $\pi_j^t.\rho = \text{responder}$. After this tag was produced, both $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ and $x_{B\leftarrow A}^{\mathsf{P}}$ were increased by 1, so that, by Lemma 20.3, $\mathrm{CTR}_{BA}^{\mathrm{resp}} \geqslant C_3 + 1$ at the time of the considered Send query. Therefore, the left-hand side of (1) remains unchanged with this Send query, which concludes the proof of (1). □

Corollary 24. Consider an adversary $\mathcal{A}$ attacking ITSAKE 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 Subsection 2.2. 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. Let $\{A, B\}$ be a fixed unordered pair of distinct parties. As long as the conditions

1. A is not corrupted with respect to B,
2. B is not corrupted with respect to A,
3. $\mathcal{A}$ has not made any forgery to an oracle π_i^s with $\{P_i, \pi_i^s.\mathrm{pid}\} = \{A, B\}$

are satisfied, there are at most

$$q_\tau = 4\left\lfloor \frac{q}{\tau} \right\rfloor + 1$$

master key periods for $\{A, B\}$.

Proof. By assumption, we can apply Lemma 23 for any master key period P for $\{A, B\}$. Therefore, at any time during period P,

$$\mathrm{CTR}_{AB}^{\mathrm{init,otp}} \leqslant x_{B\leftarrow A}^{\mathsf{P}} \quad \text{and} \quad \mathrm{CTR}_{BA}^{\mathrm{resp}} \leqslant x_{B\leftarrow A}^{\mathsf{P}}$$

and moreover, reversing the role of A and B,

$$\mathrm{CTR}_{BA}^{\mathrm{init,otp}} \leqslant x_{A\leftarrow B}^{\mathsf{P}} \quad \text{and} \quad \mathrm{CTR}_{AB}^{\mathrm{resp}} \leqslant x_{A\leftarrow B}^{\mathsf{P}}.$$

In order for a re-initialization of the master keys for $\{A, B\}$ to occur, one of the six counters $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$, $\mathrm{CTR}_{BA}^{\mathrm{init,otp}}$, $\mathrm{CTR}_{AB}^{\mathrm{resp}}$, $\mathrm{CTR}_{BA}^{\mathrm{resp}}$, $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$, or $\mathrm{CTR}_{BA}^{\mathrm{init,mac}}$ must be at least τ. Therefore, at any change of master key period for $\{A, B\}$, one has

$$\tau \leqslant x_{B\leftarrow A}^{\mathsf{P}} \quad \text{or} \quad \tau \leqslant x_{A\leftarrow B}^{\mathsf{P}} \quad \text{or} \quad \tau \leqslant \mathrm{CTR}_{AB}^{\mathrm{init,mac}} \quad \text{or} \quad \tau \leqslant \mathrm{CTR}_{BA}^{\mathrm{init,mac}}.$$

By assumption, there are at most q oracles π_j^t with $P_j = B$, $\pi_j^t.\mathrm{pid} = A$ and $\pi_j^t.\rho = \text{responder}$, and there are at most q oracles π_i^s with $P_i = A$, $\pi_i^s.\mathrm{pid} = B$ and $\pi_i^s.\rho = \text{responder}$. Thus, the inequality $\tau \leqslant x_{B\leftarrow A}^{\mathsf{P}}$ can occur for at most $\lfloor \frac{q}{\tau} \rfloor$ master key periods and the inequality $\tau \leqslant x_{A\leftarrow B}^{\mathsf{P}}$ can occur for at most $\lfloor \frac{q}{\tau} \rfloor$ master key periods. Since $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ increases only during a $\mathrm{NewSessionI}(\pi_i^s, B)$ query (where $P_i = A$), and does it by at most 1, the inequality $\tau \leqslant \mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ can occur for at most $\lfloor \frac{q}{\tau} \rfloor$ master key periods. Similarly, the inequality $\tau \leqslant \mathrm{CTR}_{BA}^{\mathrm{init,mac}}$ can occur for at most $\lfloor \frac{q}{\tau} \rfloor$ master key periods. There are thus at most $4\lfloor \frac{q}{\tau} \rfloor$ changes of master key periods for $\{A, B\}$ in total, i.e., at most $4\lfloor \frac{q}{\tau} \rfloor + 1$ master key periods for $\{A, B\}$ counting the initial one. □

Before proving the security properties of ITSAKE, we need one more lemma to bound the probability for an adversary to make a forgery.

Lemma 25. Consider an adversary $\mathcal{A}$ attacking ITSAKE 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 Subsection 2.2. 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. We denote by Forge the event that $\mathcal{A}$ makes a forgery to some session oracle π_i^s. One has

$$\Pr(\text{Forge}) \leqslant 2M(M-1)q_\tau q\varepsilon$$

where $q_\tau = 4\left\lfloor \frac{q}{\tau} \right\rfloor + 1$.

Proof. For an ordered pair $(A, B) = (P_i, P_j)$ of distinct parties and a positive integer p, we denote by $\text{Forge}^{\text{init}}_{A,B,p}$ the event that $\mathcal{A}$ makes an init-forgery to an oracle π_j^t with $\pi_j^t.\text{pid} = A$ and $\pi_j^t.\rho =$ responder during the pth master key period for $\{A, B\}$. Similarly, we denote by $\text{Forge}^{\text{resp}}_{A,B,p}$ the event that $\mathcal{A}$ makes a resp-forgery to an oracle π_i^s with $\pi_i^s.\text{pid} = B$ and $\pi_i^s.\rho =$ initiator during the pth master key period for $\{A, B\}$. One thus has

$$\text{Forge} = \left(\bigvee_{(A,B)} \bigvee_{p=1}^{+\infty} \text{Forge}^{\text{init}}_{A,B,p} \right) \vee \left(\bigvee_{(A,B)} \bigvee_{p=1}^{+\infty} \text{Forge}^{\text{resp}}_{A,B,p} \right).$$

The first forgery to a session oracle between A and B cannot occur during the pth master key period for $\{A, B\}$ for $p > q_\tau$. Indeed, if it had been the case, no corruption between A and B could have occurred before, and so a forgery must have occurred before by Corollary 24. Hence,

$$\text{Forge} = \left(\bigvee_{(A,B)} \bigvee_{p=1}^{q_\tau} \text{Forge}^{\text{init}}_{A,B,p} \right) \vee \left(\bigvee_{(A,B)} \bigvee_{p=1}^{q_\tau} \text{Forge}^{\text{resp}}_{A,B,p} \right).$$

If we prove that

$$\Pr\left(\text{Forge}^{\text{init}}_{A,B,p}\right) \leqslant q\varepsilon \tag{2}$$

and

$$\Pr\left(\text{Forge}^{\text{resp}}_{A,B,p}\right) \leqslant q\varepsilon \tag{3}$$

for all such A, B and p, we would get by the union bound that

$$\begin{aligned}
\Pr(\text{Forge}) &\leqslant \left(\sum_{(A,B)} \sum_{p=1}^{q_\tau} \Pr\left(\text{Forge}^{\text{init}}_{A,B,p}\right) \right) + \left(\sum_{(A,B)} \sum_{p=1}^{q_\tau} \Pr\left(\text{Forge}^{\text{resp}}_{A,B,p}\right) \right) \\
&\leqslant \left(\sum_{(A,B)} \sum_{p=1}^{q_\tau} q\varepsilon \right) + \left(\sum_{(A,B)} \sum_{p=1}^{q_\tau} q\varepsilon \right) \\
&= 2M(M-1)q_\tau q\varepsilon
\end{aligned}$$

as desired. It thus remains to prove (2) and (3).

In order to prove (3), we construct an adversary $\mathcal{B}^{\text{resp}}_{A,B,p}$ attacking the ε-strongly secure τ-time MACC Σ^{resp} in the experience $\text{MaccStrongForge}^{\tau\text{-time},\, q\text{-vrfy}}_{\mathcal{B}^{\text{resp}}_{A,B,p}, \Sigma^{\text{resp}}}$ from Proposition 13 and show that $\Pr\left(\text{Forge}^{\text{resp}}_{A,B,p}\right) \leqslant \Pr\left(\text{MaccStrongForge}^{\tau\text{-time},\, q\text{-vrfy}}_{\mathcal{B}^{\text{resp}}_{A,B,p}, \Sigma^{\text{resp}}} = 1\right) \leqslant q\varepsilon$. This algorithm $\mathcal{B}^{\text{resp}}_{A,B,p}$ starts by initializing the oracles of the parties $\{P_1, \ldots, P_M\}$ as usual and runs $\mathcal{A}$. At each master key (re-)initialization, except for the initialization of the pth master key period for $\{A, B\}$, it generates itself the master key of the parties as prescribed by the protocol. For the initialization of the pth master key period for $\{A, B\}$, $\mathcal{B}^{\text{resp}}_{A,B,p}$ generates MK_{AB} and MK_{BA} as prescribed by the protocol except that it does not run $\text{KeyGen}^{\text{resp}}$ to generate $K^{\text{init,vrfy}}_{AB} = K^{\text{resp,mac}}_{BA} \in \mathcal{K}^{\text{resp}}_{\text{MACC}}$, this key is supposed to be the one of its challenger. Having almost all keys, $\mathcal{B}^{\text{resp}}_{A,B,p}$ can answer $\mathcal{A}$'s NewSessionI, RevealKey, RevealState and Test queries as in their definitions. It answers the other queries as follows (where $P_i = A$ and $P_j = B$):

- For NewSessionR queries, $\mathcal{B}^{\text{resp}}_{A,B,p}$ answers as prescribed by the protocol except those of the form NewSessionR(π^t_j, A, m_A) made during the pth master key period for $\{A, B\}$. In that case, it answers it as prescribed by the protocol except that it uses its oracle $\mathcal{O}_{\text{Mac}}$ on input $(B, \text{CTR}^{\text{init,mac}}_{AB}, N_B)$ (using the notation of Figure 1) to generate the tag σ_B. Before that query, it actually makes as many as necessary queries to $\mathcal{O}_{\text{Mac}}$ with the garbage input $(A, \text{CTR}^{\text{init,mac}}_{AB}, N_B)$ in order for the next query to be the $(\text{CTR}^{\text{resp}}_{BA})^{\text{th}}$ one and the generated tag to be the correct σ_B. Note that, due to the first and third **if** tests in a NewSessionR answer, $\mathcal{B}^{\text{resp}}_{A,B,p}$ uses its oracle $\mathcal{O}_{\text{Mac}}$ at most τ times. Moreover, by Lemma 20.3 and since $\text{CTR}^{\text{resp}}_{BA}$ is increased after these $\mathcal{O}_{\text{Mac}}$ queries, $\mathcal{B}^{\text{resp}}_{A,B,p}$ is never asked to do a zth query to $\mathcal{O}_{\text{Mac}}$ and then a yth query to it with $y \leqslant z$. This answer is thus well-defined and makes $\mathcal{B}^{\text{resp}}_{A,B,p}$ correctly simulates the answer to the NewSessionR queries.
- For Send queries, $\mathcal{B}^{\text{resp}}_{A,B,p}$ answers as prescribed by the protocol except those of the form Send(π^s_i, m_B) with $\pi^s_i.\text{pid} = B$ made during the pth master key period for $\{A, B\}$. In that case, it answers it as prescribed by the protocol except that it uses its oracle $\mathcal{O}_{\text{Vrfy}}$ to verify the tag σ_B. Note that, by assumption, it uses at most q times its oracle $\mathcal{O}_{\text{Vrfy}}$.
- For Corrupt queries, $\mathcal{B}^{\text{resp}}_{A,B,p}$ returns the required master key to $\mathcal{A}$ if it possesses it. Otherwise, i.e., for Corrupt(A, B) or Corrupt(B, A) queries made during the pth master key period for $\{A, B\}$, it aborts.

We notice that, while $\mathcal{B}^{\text{resp}}_{A,B,p}$ is running, it perfectly simulates $\mathcal{A}$. Therefore, if $\text{Forge}^{\text{resp}}_{A,B,p}$ occurs in a run of $\mathcal{A}$, this means that, during the corresponding simulation of $\mathcal{A}$ by $\mathcal{B}^{\text{resp}}_{A,B,p}$, either $\text{Forge}^{\text{resp}}_{A,B,p}$ occurs or $\mathcal{B}^{\text{resp}}_{A,B,p}$ has stopped before. Such a stop can only happen during the pth master key period for $\{A, B\}$ with a Corrupt(A, B) or Corrupt(B, A) query. If such a query occurs, $\text{Forge}^{\text{resp}}_{A,B,p}$ could not occur afterwards since all oracles π^s_i with $\pi^s_i.\text{pid} = B$ will no longer be two-sided uncorrupted. Therefore, in order to prove (3), we need to show that if $\text{Forge}^{\text{resp}}_{A,B,p}$ occurs during a simulation of $\mathcal{A}$ by $\mathcal{B}^{\text{resp}}_{A,B,p}$, then $\mathcal{B}^{\text{resp}}_{A,B,p}$ wins. But the event $\text{Forge}^{\text{resp}}_{A,B,p}$ exactly means that a Send(π^s_i, m_B) query is made during the pth master key period for $\{A, B\}$ for which $\pi^s_i.\text{pid} = B$ and π^s_i is two-sided uncorrupted, that the $\text{Vrfy}^{\text{resp}}$ test is passed (and thus an entry (C, m, σ) is added to $\mathcal{V}$ in the terminology of Proposition 13) and that this triple (C, m, σ) has not been added to $\mathcal{Q}$ before. In other words, this means that $\mathcal{B}^{\text{resp}}_{A,B,p}$ wins, concluding the proof of (3). Notice that the garbage messages used as intermediate $\mathcal{O}_{\text{Mac}}$ queries in the answer to a NewSessionR query start with party A as first component in order to ensure not to add an undesired triple to $\mathcal{Q}$.

The inequality (2) is proved analogously, by constructing an adversary $\mathcal{B}^{\text{init}}_{A,B,p}$ attacking the ε-strongly secure τ-time MACC Σ^{init} in the experience $\text{MaccStrongForge}^{\tau\text{-time},\, q\text{-vrfy}}_{\mathcal{B}^{\text{init}}_{A,B,p}, \Sigma^{\text{init}}}$ and proving that $\Pr\left(\text{Forge}^{\text{init}}_{A,B,p}\right) \leqslant \Pr\left(\text{MaccStrongForge}^{\tau\text{-time},\, q\text{-vrfy}}_{\mathcal{B}^{\text{init}}_{A,B,p}, \Sigma^{\text{init}}} = 1\right) \leqslant q\varepsilon$. This part of the proof is analogous to the previous one except that, to answer a NewSessionI(π^s_i, B) query made during the pth master key period for $\{A, B\}$, $\mathcal{B}^{\text{init}}_{A,B,p}$ never needs to use its $\mathcal{O}_{\text{Mac}}$ oracle on garbage messages (due to how $\text{CTR}^{\text{init,mac}}_{AB}$ evolves). In view of this analogy, we omit details here. □

4.2 Correctness

We can now prove the correctness of ITSAKE. As we show below, default of correctness can only occur when two mutual partner oracles accept in different master key periods.

Theorem 26 (Correctness of ITSAKE)**.** Let $\mathcal{A}$ be an adversary attacking the correctness of the protocol ITSAKE 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. One has

$$\text{Adv}^{\text{Correct}}_{\text{ITSAKE}}(\mathcal{A}) \leqslant 8M(M-1)q^2\varepsilon.$$

Proof. Let us first show that if two oracles π^s_i and π^t_j are mutual partners and accepted during the same master key period for $\{P_i, P_j\}$, then $\pi^s_i.\text{sk} = \pi^t_j.\text{sk}$. From Definition 3, we know that both oracles have accepted with the same transcript and $\pi^s_i.\text{pid} = P_j$, $\pi^t_j.\text{pid} = P_i$ and $\pi^s_i.\rho \neq \pi^t_j.\rho$.

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 and m_B of the form $m_A = (C_1, C_2, N_A, \sigma_A) \in \mathbb{F}_2^\ell \times \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{init}}$ and $m_B = (C_3, N_B, \sigma_B) \in \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{resp}}$. Moreover, due to the instruction $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} \leftarrow \mathrm{CTR}_{BA}^{\mathrm{resp}}$ in the query $\mathrm{Send}(\pi_i^s, m_B)$, we know that $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} = C_3$ when π_i^s accepted, that is, at the time of the instruction $\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{AB}^{\mathrm{init,otp}}\left[n\mathrm{CTR}_{AB}^{\mathrm{init,otp}}, n\right]$. In view of Lemma 20.4, this means that

$$\pi_i^s.\mathrm{sk} = \mathrm{der}\,(A, B, m_A, m_B) \oplus K\,[nC_3, n]$$

where K is the key $K_{AB}^{\mathrm{init,otp}} = K_{BA}^{\mathrm{resp,otp}}$ generated at the beginning of the considered master key period.

Since $\mathrm{T}_i^s = \mathrm{T}_j^t$, we know that π_j^t received the message m_A and output the message m_B. Therefore, when π_j^t accepted, that is, at the time of the instruction $\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{BA}^{\mathrm{resp,otp}}\,[n\mathrm{CTR}_{BA}^{\mathrm{resp}}, n]$, we know that $\mathrm{CTR}_{BA}^{\mathrm{resp}} = C_3$. In view of Lemma 20.4, since π_i^s and π_j^t accepted during the same master key period for $\{A, B\}$, this means that

$$\pi_j^t.\mathrm{sk} = \mathrm{der}\,(A, B, m_A, m_B) \oplus K\,[nC_3, n]$$

and so $\pi_i^s.\mathrm{sk} = \pi_j^t.\mathrm{sk}$.

Denoting OtherMKP the event as in Lemma 21, we know by that lemma that

$$\Pr(\mathrm{OtherMKP}) \leqslant 8M(M-1)q^2\varepsilon$$

and it remains to prove that $\mathrm{Adv}_{\mathrm{ITSAKE}}^{\mathrm{Correct}}(\mathcal{A}) \leqslant \Pr(\mathrm{OtherMKP})$. If π_i^s and π_j^t are mutual partners with $\pi_i^s.\mathrm{sk} \neq \pi_j^t.\mathrm{sk}$, we know from above that they accepted in a different master key period for $\{P_i, P_j\}$. Since the first message of the transcript of π_i^s and π_j^t are the same, this implies that OtherMKP occurs and so $\mathrm{Adv}_{\mathrm{ITSAKE}}^{\mathrm{Correct}}(\mathcal{A}) \leqslant \Pr(\mathrm{OtherMKP})$. □

4.3 Weak synchronization robustness

When we discuss the weak synchronization robustness, we must prevent the adversary to initiate $\tau + 1$ times a protocol run between the same parties A and B in the same direction. Indeed, otherwise, the adversary can simply first query $\mathrm{NewSessionI}(\pi_i^s, B)$ for each $s \in \{1, \ldots, \tau\}$ where $P_i = A$, which turns $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ to τ. Then, it launches an honest protocol run between A and B, which starts with the query $\mathrm{NewSessionI}(\pi_i^{\tau+1}, B)$ that launches a master key re-initialization and makes $\pi_i^{\tau+1}$ reject. Hence, $\mathcal{A}$ wins the game $\mathrm{wSR}_{\mathcal{A},\mathrm{ITSAKE}}$ that way. To prevent this trivial attack, we impose q to be less than τ.

Theorem 27 (Weak synchronization robustness of ITSAKE)**.** Let $\mathcal{A}$ be an adversary attacking the weak synchronization robustness of the protocol ITSAKE 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. If $\tau > q$, one has

$$\mathrm{Adv}_{\mathrm{ITSAKE}}^{\mathrm{wSR}}(\mathcal{A}) \leqslant 2M(M-1)q\varepsilon.$$

Proof. We first prove that if an uninterrupted honest run of the protocol is executed between the oracles π_i^s and π_j^t such that $\mathrm{CTR}_{AB}^{\mathrm{init,mac}} < \tau$, $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} < \tau$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}} < \tau$ at the start of this honest run where $P_i = A$, $P_j = B$ and $\pi_i^s.\rho =$ initiator, then, at the end of the honest run, one has $\pi_i^s.\alpha = \pi_j^t.\alpha =$ accepted and $\pi_i^s.\mathrm{sk} = \pi_j^t.\mathrm{sk}$.

We denote by C_1, C_2 and C_3 the respective values of $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$, $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ at the start of the honest run. We also denote by P the master key period for $\{A, B\}$ during which the honest run starts, by k_1 the key $k_1 = K_{AB}^{\mathrm{init,mac}} = K_{BA}^{\mathrm{resp,vrfy}}$ during P, by k_2 the key $k_2 = K_{BA}^{\mathrm{resp,mac}} = K_{AB}^{\mathrm{init,vrfy}}$ during P, and by K the key $K_{AB}^{\mathrm{init,otp}} = K_{BA}^{\mathrm{resp,otp}}$ generated at the beginning of P. Note that k_1 and k_2 are constant during P (see Lemma 20.2), while the evolution of $K_{AB}^{\mathrm{init,otp}}$ and $K_{BA}^{\mathrm{resp,otp}}$ in terms of K is described by Lemma 20.4.

The honest run starts with the query NewSessionI(π_i^s, P_j). Since $\mathrm{CTR}_{AB}^{\mathrm{init,mac}} = C_1 < \tau$, there is no master key re-initialization during that query. Then, a nonce $N_A \in \mathcal{N}$ is selected and a tag $\sigma_A \in \mathcal{T}^{\mathrm{init}}$ is generated as $\sigma_A \leftarrow \mathrm{Mac}_{k_1,C_1}^{\mathrm{init}}(A, C_2, N_A)$. The output message is $m_A = (C_1, C_2, N_A, \sigma_A)$ and $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ is then updated to $C_1 + 1$.

The immediate next query of $\mathcal{A}$ is NewSessionR(π_j^t, P_i, m_A). Since $\mathrm{CTR}_{BA}^{\mathrm{resp}} = C_3 < \tau$, there is no master key re-initialization during that query. Moreover, one has $m_A \in \mathbb{F}_2^\ell \times \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{init}}$, $C_1 < \tau$, $C_2 < \tau$ and $\mathrm{Vrfy}_{k_1,C_1}^{\mathrm{init}}((A, C_2, N_A), \sigma_A) = 1$ since our MACCs are required to be always correct (see Definition 11). Therefore, π_j^t does not abort. After these checks, the instructions under the last **if** statement have the effect of turning $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ to $\max\{C_2, C_3\}$ and updating $K_{BA}^{\mathrm{resp,otp}}$ according to Lemma 20.4. A nonce $N_B \in \mathcal{N}$ is then selected and a tag $\sigma_B \in \mathcal{T}^{\mathrm{resp}}$ is generated as $\sigma_B \leftarrow \mathrm{Mac}_{k_2,\max\{C_2,C_3\}}^{\mathrm{resp}}(B, C_1, N_B)$. The output message is $m_B = (\max\{C_2, C_3\}, N_B, \sigma_B)$ and π_j^t accepts with $\pi_j^t.\mathrm{sk} = \mathrm{der}(A, B, m_A, m_B) \oplus K[n \max\{C_2, C_3\}, n]$ (see Lemma 20.4). Then, $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ is updated to $\max\{C_2, C_3\} + 1$ and $K_{BA}^{\mathrm{resp,otp}}$ is updated accordingly.

The last query of the honest run by $\mathcal{A}$ is Send(π_i^s, m_B). By construction and using our assumptions about the counters, one has $m_B \in \mathbb{F}_2^\ell \times \mathcal{N} \times \mathcal{T}^{\mathrm{resp}}$, $\max\{C_2, C_3\} < \tau$, $\max\{C_2, C_3\} \geqslant \mathrm{CTR}_{AB}^{\mathrm{init,otp}} = C_2$ and $\mathrm{Vrfy}_{k_2,\max\{C_2,C_3\}}^{\mathrm{resp}}((B, C_1, N_B), \sigma_B) = 1$. Hence the oracle π_i^s does not abort. The next three instructions following the **if** tests (see Figure 1) update $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ to $\max\{C_2, C_3\}$, and update $K_{AB}^{\mathrm{init,otp}}$ accordingly. Then π_i^s accepts with $\pi_i^s.\mathrm{sk} = \mathrm{der}(A, B, m_A, m_B) \oplus K[n \max\{C_2, C_3\}, n]$ (see Lemma 20.4). Afterwards, $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ is updated to $\max\{C_2, C_3\} + 1$ and $K_{AB}^{\mathrm{init,otp}}$ is modified accordingly. Finally, if $\max\{C_2, C_3\} + 1 \geqslant \tau$, the master keys are re-initialized. Even if it occurs, it does not impact π_i^s nor π_j^t since both of them have already accepted.

We thus have proved that after an uninterrupted execution of an honest run of the protocol between oracles π_i^s and π_j^t starting with $\max\{\mathrm{CTR}_{AB}^{\mathrm{init,mac}}, \mathrm{CTR}_{AB}^{\mathrm{init,otp}}, \mathrm{CTR}_{BA}^{\mathrm{resp}}\} < \tau$ (where $P_i = A$, $P_j = B$ and $\pi_i^s.\rho = \text{initiator}$), one has $\pi_i^s.\alpha = \pi_j^t.\alpha = \text{accepted}$ and $\pi_i^s.\mathrm{sk} = \mathrm{der}(A, B, m_A, m_B) \oplus K[n \max\{C_2, C_3\}, n] = \pi_j^t.\mathrm{sk}$.

Denoting by Forge the event that $\mathcal{A}$ makes a forgery to some session oracle as in Lemma 25, we know by that lemma that

$$\Pr(\mathrm{Forge}) \leqslant 2M(M-1)q\varepsilon$$

since $q_\tau = 4\left\lfloor \frac{q}{\tau} \right\rfloor + 1 = 1$ as $q < \tau$. It thus remains to prove that $\mathrm{Adv}_{\mathrm{ITSAKE}}^{\mathrm{wSR}}(\mathcal{A}) \leqslant \Pr(\mathrm{Forge})$.

In view of what we have proved above and of Definition 6, it suffices to prove that, given two distinct parties A and B, if $\mathcal{A}$ does not make any forgery to a session oracle and if A is not corrupted with respect to B nor vice versa, then any of $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$, $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ is less than τ. Since there are at most q queries NewSessionI(π_i^s, B) where $P_i = A$, one has $\mathrm{CTR}_{AB}^{\mathrm{init,mac}} \leqslant q < \tau$. For the other two counters, one has $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} \leqslant x_{B \leftarrow A}^{\mathsf{P}}$ and $\mathrm{CTR}_{BA}^{\mathrm{resp}} \leqslant x_{B \leftarrow A}^{\mathsf{P}}$ using the notation and result of Lemma 23. Since $x_{B \leftarrow A}^{\mathsf{P}} \leqslant q < \tau$, this concludes the proof. □

4.4 Entity authentication

We now prove the entity authentication of ITSAKE. The requirement of at most one partner in the negation of condition 3(c) of Definition 9 is ensured by the non-occurrence of OtherMKP as in Lemma 21, while the existence of a strong partner is ensured by the non-occurrence of Forge as in Lemma 25. Hence, the advantage of the adversary is bounded by the sum of the bounds provided in these two lemmas.

Theorem 28 (Entity authentication of ITSAKE)**.** Let $\mathcal{A}$ be an adversary attacking the entity authentication of the protocol ITSAKE 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. One has

$$\mathrm{Adv}_{\mathrm{ITSAKE}}^{\mathrm{EntAuth}}(\mathcal{A}) \leqslant 18M(M-1)q^2\varepsilon.$$

Proof. If $q = 0$, $\mathcal{A}$ cannot win. Hence, we can assume that $q > 0$. As in Lemma 21, we denote by OtherMKP the event that a message m_A is output by a NewSessionI(π_i^s, P_j) query and that, during another master key period for $\{P_i, P_j\}$ (earlier or later), a NewSessionR(π_j^t, P_i, m_A) query

leads to $\pi_j^t.\alpha = \text{accepted}$. By that lemma, we know that

$$\Pr(\text{OtherMKP}) \leqslant 8M(M-1)q^2\varepsilon.$$

In addition, as in Lemma 25, we denote by Forge the event that $\mathcal{A}$ makes a forgery to some session oracle π_i^s. By that lemma, we know that

$$\Pr(\text{Forge}) \leqslant 2M(M-1)q_\tau q\varepsilon$$

where $q_\tau = 4\left\lfloor \frac{q}{\tau} \right\rfloor + 1 \leqslant 5q$. Combining these results, we obtain that

$$\begin{aligned}\Pr(\text{OtherMKP} \vee \text{Forge}) &\leqslant \Pr(\text{OtherMKP}) + \Pr(\text{Forge}) \\ &\leqslant 8M(M-1)q^2\varepsilon + 2M(M-1)q_\tau q\varepsilon \\ &= 2M(M-1)\left(q_\tau q + 4q^2\right)\varepsilon \\ &\leqslant 18M(M-1)q^2\varepsilon.\end{aligned}$$

It thus remains to prove that $\mathrm{Adv}^{\mathrm{EntAuth}}_{\mathrm{ITSAKE}}(\mathcal{A}) \leqslant \Pr(\text{OtherMKP} \vee \text{Forge})$. In order to do so, we suppose that neither OtherMKP nor Forge occurs, and we show that $\mathcal{A}$ cannot win; that is, no oracle accepts maliciously. By Lemma 22, as $\mathcal{A}$ has not made any forgery, we already know that any oracle that has accepted and which was two-sided uncorrupted when it accepted admits a strong partner. It remains to prove that it cannot have two different partners. Let us consider two distinct parties $P_i = A$ and $P_j = B$.

We start with the case of a responder oracle π_j^t and denote by P the master key period for $\{A, B\}$ during which π_j^t accepts. If π_i^s and $\pi_i^{s'}$ were two distinct oracles such that π_j^t has both of them as partners, the first message of T_j^t, T_i^s and $\mathrm{T}_i^{s'}$ would be the same. In particular, the value of $\mathrm{CTR}_{AB}^{\mathrm{init,mac}}$ would be the same when $\mathcal{A}$ queried $\mathrm{NewSessionI}(\pi_i^s, B)$ and $\mathrm{NewSessionI}(\pi_i^{s'}, B)$. Since this counter was increased at the end of both of these queries and in view of Lemma 20.3, this would mean that one of these oracles, say π_i^s, started to negotiate in a master key period different from P. Since this would contradict the non-occurrence of OtherMKP, this is not possible.

We conclude with the case of an initiator oracle π_i^s and denote by P the master key period for $\{A, B\}$ during which π_i^s accepts. If π_j^t and $\pi_j^{t'}$ were two distinct oracles such that π_i^s has both of them as partners, one would have $\mathrm{T}_i^s = \mathrm{T}_j^t = \mathrm{T}_j^{t'}$. 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 $\mathrm{CTR}_{BA}^{\mathrm{resp}}$. Since $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ was increased when these oracles accepted and this counter is non-decreasing during a master key period for $\{A, B\}$ (see Lemma 20.3), this would mean that one of them, say π_j^t, accepted during a master key period other than P. Since $\mathrm{T}_i^s = \mathrm{T}_j^t$ and by Lemma 20.1, this would contradict the non-occurrence of OtherMKP. This situation is thus not possible, which concludes the proof. □

4.5 Key indistinguishability

We conclude by proving the key indistinguishability of ITSAKE. In order to do so, we reduce it to the well-known security of the one-time pad, expressed as in the following proposition.

Proposition 29. Given a positive integer n, let $\mathrm{IndOTP}_{\mathcal{A}}$ be the following game between an adversary $\mathcal{A}$ and its challenger:

1. The challenger uniformly chooses a key $K \in \mathbb{F}_2^n$.
2. The challenger uniformly and independently chooses a bit $b \in \{0, 1\}$.
3. $\mathcal{A}$ queries a message $m \in \mathbb{F}_2^n$ to its challenger. If $b = 0$, the challenger gives a uniformly chosen $c \in \mathbb{F}_2^n$ to $\mathcal{A}$; if $b = 1$ the challenger gives $c = m \oplus K \in \mathbb{F}_2^n$ to $\mathcal{A}$.
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).

One has

$$\Pr\left(\mathrm{IndOTP}_{\mathcal{A}} = 1\right) = \frac{1}{2}.$$

Theorem 30 (Key indistinguishability of ITSAKE)**.** Let $\mathcal{A}$ be an adversary attacking the key indistinguishability of the protocol ITSAKE 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. One has

$$\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{ITSAKE}}(\mathcal{A}) \leqslant 2M(M-1)q_\tau q\varepsilon$$

where $q_\tau = 4\left\lfloor \frac{q}{\tau} \right\rfloor + 1$.

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

Game 0. This is the original game $G_0 = \mathrm{KeyInd}_{\mathcal{A},\mathrm{ITSAKE}}$ as defined in Definition 10. One thus has

$$\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{ITSAKE}}(\mathcal{A}) = 2\left|\Pr(G_0 = 1) - \frac{1}{2}\right|. \tag{4}$$

Game 1. This game G_1 is the same as G_0 except that, as soon as $\mathcal{A}$ makes a forgery to some session oracle π_i^s, then the experiment aborts and outputs a uniformly chosen value in $\{0, 1\}$. As in Lemma 25, we denote this event by Forge. By this lemma, one has

$$\Pr(\mathrm{Forge}) \leqslant 2M(M-1)q_\tau q\varepsilon.$$

Moreover, by definition of G_1, one has $\Pr(G_1 = 1 \,|\, \mathrm{Forge}) = \frac{1}{2}$ and $\Pr(G_1 = 1 \,|\, \neg\mathrm{Forge}) = \Pr(G_0 = 1 \,|\, \neg\mathrm{Forge})$. Therefore,

$$\begin{aligned}
\Pr(G_1 = 1) &= \Pr(G_1 = 1 \,|\, \mathrm{Forge}) \Pr(\mathrm{Forge}) + \Pr(G_1 = 1 \,|\, \neg\mathrm{Forge}) \Pr(\neg\mathrm{Forge}) \\
&= \frac{1}{2}\Pr(\mathrm{Forge}) + \Pr(G_0 = 1 \,|\, \neg\mathrm{Forge}) \Pr(\neg\mathrm{Forge}) \\
&= \frac{1}{2}\Pr(\mathrm{Forge}) + \Pr(G_0 = 1) - \Pr(G_0 = 1 \,|\, \mathrm{Forge}) \Pr(\mathrm{Forge})
\end{aligned}$$

and thus,

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

This yields

$$2\left|\Pr(G_0 = 1) - \frac{1}{2}\right| \leqslant 2M(M-1)q_\tau q\varepsilon + 2\left|\Pr(G_1 = 1) - \frac{1}{2}\right|. \tag{5}$$

Game 2. The *accepting counter value* of an accepting initiator oracle π_i^s with $P_i = A$ and $\pi_i^s.\mathrm{pid} = B$ is the value in $[0, \tau - 1]$ of the counter $\mathrm{CTR}_{AB}^{\mathrm{init,otp}}$ when π_i^s accepted, i.e., at the time of the instruction $\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{AB}^{\mathrm{init,otp}}[n\mathrm{CTR}_{AB}^{\mathrm{init,otp}}, n]$ (see Figure 1). Similarly, the *accepting counter value* of an accepting responder oracle π_i^s with $P_i = B$ and $\pi_i^s.\mathrm{pid} = A$ is the value in $[0, \tau - 1]$ of the counter $\mathrm{CTR}_{BA}^{\mathrm{resp}}$ when π_i^s accepted, i.e., at the time of the instruction $\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{BA}^{\mathrm{resp,otp}}[n\mathrm{CTR}_{BA}^{\mathrm{resp}}, n]$ (see again Figure 1). The game G_2 is the same as G_1 except that, at the beginning, the challenger guesses uniformly and independently

- an ordered pair (A, B) of distinct parties,
- an integer $p \in [1, q_\tau]$,
- and an integer $C \in [0, \tau - 1]$,

and, once $\mathcal{A}$ has concluded, if $\mathcal{A}$ has made its Test query on π_i^s which accepted during the $p'^{\,\mathrm{th}}$ master key period for $\{P_i, \pi_i^s.\mathrm{pid}\}$ with accepting counter value C' and with $(P_i, \pi_i^s.\mathrm{pid}, \pi_i^s.\rho, p', C') \notin \{(A, B, \mathrm{initiator}, p, C), (B, A, \mathrm{responder}, p, C)\} = S_{ABpC}$, then the experiment aborts and outputs a uniform value in $\{0, 1\}$ (otherwise, the output of the experiment is as in G_1). Let T be the random variable that takes the value $(P_i, \pi_i^s.\mathrm{pid}, \pi_i^s.\rho, p', C')$ if $\mathcal{A}$ makes its Test query on π_i^s and

if π_i^s accepted during the $p'^{\text{ th}}$ master key period for $\{P_i, \pi_i^s.\text{pid}\}$ with accepting counter value C' (and $T = \perp$ if $\mathcal{A}$ does not make any Test query). From the description of G_2 and of T, one has

$$\Pr(G_2 = 1 \,|\, T \notin S_{ABpC}) = \frac{1}{2}$$

and

$$\Pr(G_2 = 1 \,|\, T \in S_{ABpC}) = \Pr(G_1 = 1 \,|\, T \in S_{ABpC}).$$

Furthermore, since the result of G_1 is independent of the choice of A, B, p and C, the condition $T \in S_{ABpC}$ reduces to $T \neq \perp$ and $p' \leqslant q_\tau$ in a run of G_1, i.e.

$$\Pr(G_1 = 1 \,|\, T \in S_{ABpC}) = \Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau).$$

By condition 4(a) in Definition 10, one has $\Pr(G_1 = 1 \,|\, T = \perp) = \frac{1}{2}$. Moreover, if $T \neq \perp$ but $p' > q_\tau$, by the rules of G_1 and Corollary 24, we know that the oracle π_i^s on which the Test query was issued was not two-sided uncorrupted when it accepted. Thus, by condition 4(b) in Definition 10, one has $\Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' > q_\tau) = \frac{1}{2}$. One can then compute

$$\begin{aligned}
&\Pr(G_1 = 1) \\
&= \Pr(G_1 = 1 \,|\, T = \perp) \Pr(T = \perp) + \Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' > q_\tau) \Pr(T \neq \perp \wedge p' > q_\tau) \\
&\quad + \Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) \Pr(T \neq \perp \wedge p' \leqslant q_\tau) \\
&= \frac{1}{2} \Pr(T = \perp) + \frac{1}{2} \Pr(T \neq \perp \wedge p' > q_\tau) + \Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) \Pr(T \neq \perp \wedge p' \leqslant q_\tau) \\
&= \frac{1}{2}(1 - \Pr(T \neq \perp \wedge p' \leqslant q_\tau)) + \Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) \Pr(T \neq \perp \wedge p' \leqslant q_\tau) \\
&= \frac{1}{2} + \Pr(T \neq \perp \wedge p' \leqslant q_\tau) \left(\Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) - \frac{1}{2} \right)
\end{aligned}$$

and

$$\begin{aligned}
\Pr(G_2 = 1) &= \Pr(G_2 = 1 \,|\, T \notin S_{ABpC}) \Pr(T \notin S_{ABpC}) + \Pr(G_2 = 1 \,|\, T \in S_{ABpC}) \Pr(T \in S_{ABpC}) \\
&= \frac{1}{2} \Pr(T \notin S_{ABpC}) + \Pr(G_1 = 1 \,|\, T \in S_{ABpC}) \Pr(T \in S_{ABpC}) \\
&= \frac{1}{2} \left(1 - \Pr(T \in S_{ABpC})\right) + \Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) \Pr(T \in S_{ABpC}) \\
&= \frac{1}{2} + \Pr(T \in S_{ABpC}) \left(\Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) - \frac{1}{2} \right).
\end{aligned}$$

Using $\Pr(T \in S_{ABpC}) = \Pr(T \in S_{ABpC} \,|\, T \neq \perp \wedge p' \leqslant q_\tau) \Pr(T \neq \perp \wedge p' \leqslant q_\tau)$ together with $\Pr(T \in S_{ABpC} \,|\, T \neq \perp \wedge p' \leqslant q_\tau) = \frac{1}{M(M-1)q_\tau\tau}$, and putting the last two results together, we obtain

$$\begin{aligned}
\Pr(G_2 = 1) &= \frac{1}{2} + \frac{1}{M(M-1)q_\tau\tau} \Pr(T \neq \perp \wedge p' \leqslant q_\tau) \left(\Pr(G_1 = 1 \,|\, T \neq \perp \wedge p' \leqslant q_\tau) - \frac{1}{2} \right) \\
&= \frac{1}{2} + \frac{1}{M(M-1)q_\tau\tau} \left(\Pr(G_1 = 1) - \frac{1}{2} \right).
\end{aligned}$$

This means that

$$\left| \Pr(G_1 = 1) - \frac{1}{2} \right| = M(M-1)q_\tau\tau \cdot \left| \Pr(G_2 = 1) - \frac{1}{2} \right|. \tag{6}$$

So far, we have proved that

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{KeyInd}}_{\mathrm{ITSAKE}}(\mathcal{A}) &= 2 \left| \Pr(G_0 = 1) - \frac{1}{2} \right| && \text{(by (4))} \\
&\leqslant 2M(M-1)q_\tau q\varepsilon + 2 \left| \Pr(G_1 = 1) - \frac{1}{2} \right| && \text{(by (5))} \\
&= 2M(M-1)q_\tau q\varepsilon + 2M(M-1)q_\tau\tau \cdot \left| \Pr(G_2 = 1) - \frac{1}{2} \right| && \text{(by (6))}
\end{aligned}$$

and it remains to prove that $\Pr(G_2 = 1) = \frac{1}{2}$.

Let us now construct an adversary $\mathcal{B}$ against the indistinguishability of the one-time pad such that

$$\Pr(G_2 = 1) = \Pr(\mathrm{IndOTP}_{\mathcal{B}} = 1),$$

where $\mathrm{IndOTP}_{\mathcal{B}}$ is the experiment as in Proposition 29. The adversary $\mathcal{B}$ starts by uniformly and independently choosing an ordered pair (A, B) of distinct parties, an integer $p \in [1, q_\tau]$ and an integer $C \in [0, \tau - 1]$. It then initializes the oracles of the parties $\{P_1, \ldots, P_M\}$ as usual and runs $\mathcal{A}$. At each master key (re-)initialization, except for the initialization of the pth master key period for $\{A, B\}$, it generates itself the MK_{ij} of the parties as prescribed by the protocol. For the initialization of the pth master key period for $\{A, B\}$, it generates MK_{AB} and MK_{BA} as prescribed by the protocol except that it does not generate $K_{AB}^{\mathrm{init,otp}} = K_{BA}^{\mathrm{resp,otp}} \in \mathbb{F}_2^{\tau n}$ but only $K_{AB}^{\mathrm{init,otp}}[0, nC] = K_{BA}^{\mathrm{resp,otp}}[0, nC] \in \mathbb{F}_2^{nC}$ and $K_{AB}^{\mathrm{init,otp}}[nC + n, n(\tau - C - 1)] = K_{BA}^{\mathrm{resp,otp}}[nC + n, n(\tau - C - 1)] \in \mathbb{F}_2^{n(\tau - C - 1)}$ uniformly. The remaining n-bit part $K_{AB}^{\mathrm{init,otp}}[nC, n] = K_{BA}^{\mathrm{resp,otp}}[nC, n] \in \mathbb{F}_2^n$ is not generated but is considered by $\mathcal{B}$ as the key of its challenger. $\mathcal{B}$ sets $K_{AB}^{\mathrm{init,otp}}[nC, n] \leftarrow \bot$ and $K_{BA}^{\mathrm{resp,otp}}[nC, n] \leftarrow \bot$ to indicate it does not know their value yet, and it also initializes two variables m_{init} and m_{resp} as $\bot$. It answers $\mathcal{A}$'s queries as follows:

- NewSessionI(π_i^s, P_j): $\mathcal{B}$ answers exactly as prescribed by the protocol.
- NewSessionR(π_i^s, P_j, m): $\mathcal{B}$ answers exactly as prescribed by the protocol except if $(P_i, P_j) = (B, A)$ and the current master key period for $\{A, B\}$ is the pth one. In that case, $\mathcal{B}$ answers as prescribed by the protocol except that it replaces the instruction

  $$\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{BA}^{\mathrm{resp,otp}}\,[n\mathrm{CTR}_{BA}^{\mathrm{resp}}, n]$$

  where $m_A = m$ is the message of the query, by

  if $\mathrm{CTR}_{BA}^{\mathrm{resp}} = C$
  $\pi_i^s.\alpha \leftarrow$ accepted
  $m_{\mathrm{resp}} \leftarrow \mathrm{der}\,(A, B, m_A, m_B)$
  else
  $\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{BA}^{\mathrm{resp,otp}}\,[n\mathrm{CTR}_{BA}^{\mathrm{resp}}, n]\,.$

  In all cases, if this provokes a forgery to π_i^s, then $\mathcal{B}$ aborts the experiment and outputs a uniform value in $\{0, 1\}$.
- Send(π_i^s, m): $\mathcal{B}$ answers exactly as prescribed by the protocol except if $(P_i, \pi_i^s.\mathrm{pid}) = (A, B)$ and the current master key period for $\{A, B\}$ is the pth one. In that case, $\mathcal{B}$ answers as prescribed by the protocol except that it replaces the instruction

  $$\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{AB}^{\mathrm{init,otp}}\left[n\mathrm{CTR}_{AB}^{\mathrm{init,otp}}, n\right]$$

  where $m_B = m$, by

  if $\mathrm{CTR}_{AB}^{\mathrm{init,otp}} = C$
  $\pi_i^s.\alpha \leftarrow$ accepted
  $m_{\mathrm{init}} \leftarrow \mathrm{der}\,(A, B, m_A, m_B)$
  else
  $\mathrm{sk} \leftarrow \mathrm{der}\,(A, B, m_A, m_B) \oplus K_{AB}^{\mathrm{init,otp}}\left[n\mathrm{CTR}_{AB}^{\mathrm{init,otp}}, n\right].$

  In all cases, if this provokes a forgery to π_i^s, then $\mathcal{B}$ aborts the experiment and outputs a uniform value in $\{0, 1\}$.
- RevealKey(π_i^s): If $\mathcal{B}$ has $\pi_i^s.\mathrm{sk} \neq \bot$, it simply returns it to $\mathcal{A}$. Otherwise, it aborts the experiment and outputs a uniform value in $\{0, 1\}$.
- RevealState(π_i^s): $\mathcal{B}$ always knows $\pi_i^s.\mathrm{st}$ and it simply returns it to $\mathcal{A}$.

- Corrupt(P_i, P_j): If $\mathcal{B}$ knows MK_{ij}, that is, if no part of MK_{ij} consists of $\perp$, it returns it to $\mathcal{A}$. Otherwise, it aborts the experiment and outputs a uniform value in $\{0,1\}$.
- Test(π_i^s): If $\mathcal{B}$ has $\pi_i^s.\mathrm{sk} \neq \perp$, it aborts the experiment and outputs a uniform value in $\{0,1\}$. Otherwise, $\mathcal{B}$ queries the message m_{init} to its challenger if $\pi_i^s.\rho = \text{initiator}$, and it queries the message m_{resp} to its challenger if $\pi_i^s.\rho = \text{responder}$. The challenger returns $c \in \mathbb{F}_2^n$ to $\mathcal{B}$ who returns it to $\mathcal{A}$. (Notice that $\mathcal{B}$ does not choose the test bit b_{test} but considers it as the bit hidden by its challenger in the game $\mathrm{IndOTP}_{\mathcal{B}}$).

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

In view of how NewSessionR queries are answered, $\mathcal{B}$ always has $\pi_i^s.\mathrm{sk} \neq \perp$ for responder oracles π_i^s that have accepted, except if $(P_i, \pi_i^s.\mathrm{pid}) = (B, A)$ and if π_i^s accepted during the pth master key period for $\{A, B\}$ with accepting counter value C. By Lemma 20.3, there is at most one such oracle. Similarly, in view of how Send queries are answered, $\mathcal{B}$ always has $\pi_i^s.\mathrm{sk} \neq \perp$ for initiator oracles π_i^s that have accepted, except if $(P_i, \pi_i^s.\mathrm{pid}) = (A, B)$ and if π_i^s accepted during the pth master key period for $\{A, B\}$ with accepting counter value C. Again by Lemma 20.3, there is at most one such oracle. Therefore, if $\mathcal{A}$ queries Test(π_i^s) on an oracle π_i^s that has accepted and for which, in $\mathcal{B}$'s view, $\pi_i^s.\mathrm{sk} = \perp$, then m_{init} if $\pi_i^s.\rho = \text{initiator}$ (respectively, m_{resp} if $\pi_i^s.\rho = \text{responder}$) is an element of $\mathbb{F}_2^n$ and $\mathcal{B}$ correctly simulates the answer to this Test query. Indeed, the challenger's key is considered as the value of $K_{AB}^{\mathrm{init,otp}}[nC, n] = K_{BA}^{\mathrm{resp,otp}}[nC, n]$ at the start of the pth master key period for $\{A, B\}$ and the value of $K_{AB}^{\mathrm{init,otp}}[nC, n]$ (respectively, of $K_{BA}^{\mathrm{resp,otp}}[nC, n]$) had not yet been modified when $\mathrm{CTR}_{AB}^{\mathrm{init}} = C$ (respectively, when $\mathrm{CTR}_{BA}^{\mathrm{resp}} = C$) by virtue of Lemma 20.4. This shows that $\mathcal{B}$ is well defined and, as long as it is running, it perfectly simulates an execution of $\mathcal{A}$ in G_2.

Let Z be the event that, in the course of an execution of $\mathcal{A}$ in G_2, 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 or G_2 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 $\mathrm{IndOTP}_{\mathcal{B}}$, 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_2 by $\mathcal{B}$. This would imply that $\Pr(Z) = \Pr(Z')$ and $\Pr(G_2 = 1 \,|\, \neg Z) = \Pr(\mathrm{IndOTP}_{\mathcal{B}} = 1 \,|\, \neg Z')$. It is clear that $\Pr(G_2 = 1 \,|\, Z) = \frac{1}{2} = \Pr(\mathrm{IndOTP}_{\mathcal{B}} = 1 \,|\, Z')$. Therefore, one would have

$$\begin{aligned}
\Pr(G_2 = 1) &= \Pr(G_2 = 1 \,|\, Z)\Pr(Z) + \Pr(G_2 = 1 \,|\, \neg Z)\Pr(\neg Z) \\
&= \Pr(\mathrm{IndOTP}_{\mathcal{B}} = 1 \,|\, Z')\Pr(Z') + \Pr(\mathrm{IndOTP}_{\mathcal{B}} = 1 \,|\, \neg Z')\Pr(\neg Z') \\
&= \Pr(\mathrm{IndOTP}_{\mathcal{B}} = 1) \\
&= \frac{1}{2}
\end{aligned}$$

by Proposition 29, which would conclude the proof.

If Z occurs because of the new rule of G_1, then Z' occurs in view of the way NewSessionR and Send queries are answered. If Z occurs because of the new rule of G_2, i.e., if $\mathcal{A}$ has made its Test query on π_i^s which accepted during the $p'^{\,\mathrm{th}}$ master key period for $\{P_i, \pi_i^s.\mathrm{pid}\}$ with accepting counter value C' and with $(P_i, \pi_i^s.\mathrm{pid}, \pi_i^s.\rho, p', C') \notin \{(A, B, \text{initiator}, p, C), (B, A, \text{responder}, p, C)\}$, this means that, by the way NewSessionR and Send queries are answered, $\pi_i^s.\mathrm{sk} \neq \perp$ when π_i^s accepted. Thus 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 $\mathcal{B}$ acts once $\mathcal{A}$ has concluded.

It remains to prove that, if Z' occurs, then Z occurs in the simulation of G_2 by $\mathcal{B}$. We suppose by contradiction that Z does not occur. If Z' occurs by the way $\mathcal{B}$ 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 $\mathcal{B}$ answers $\mathcal{A}$'s queries. By the rule of G_1, Z' cannot occur in a NewSessionR or Send query. It thus remains to treat the cases of RevealKey, Corrupt and Test queries.

Since Z does not occur, we know, by condition 4(a) of Definition 10, that $\mathcal{A}$ issued a Test(π_i^s) query to some oracle π_i^s. Moreover, by the rule of G_2, one has

$$(P_i, \pi_i^s.\text{pid}, \pi_i^s.\rho) \in \{(A, B, \text{initiator}), (B, A, \text{responder})\},$$

and π_i^s accepted during the pth master key period for $\{A, B\}$ with accepting counter value C. By the way $\mathcal{B}$ answers to NewSessionR and Send queries, and since $\mathcal{B}$ never changes the value of $\pi_i^s.\text{sk}$ after acceptance, one has $\pi_i^s.\text{sk} = \perp$ at the time of the query Test(π_i^s). This already shows that Z' cannot occur during a Test query.

By condition 4(b) of Definition 4, we know that π_i^s was two-sided uncorrupted when it accepted. By the new rule of G_1 and by Lemma 22, this implies that there exists an oracle π_j^t, also activated during the pth master key period for $\{A, B\}$, such that π_i^s has π_j^t as strong partner.

If $\pi_i^s.\rho =$ initiator, it has received the last message of its transcript, and so π_i^s and π_j^t are mutual partners. Since they have the same second message of their transcripts, this means that π_j^t also accepted with accepting counter value C. Therefore, π_i^s and π_j^t are the only two oracles that have accepted and for which $\mathcal{B}$ does not know the corresponding session key. By condition 4(d) of Definition 4, $\mathcal{A}$ did not query RevealKey(π_i^s); and by condition 4(e) of Definition 4, $\mathcal{A}$ did not query RevealKey(π_j^t). Therefore, Z' cannot occur because of a RevealKey query in this case. Moreover, since π_i^s was two-sided uncorrupted when it accepted, and since it has π_j^t as strong partner, π_j^t accepted before π_i^s did and π_j^t was also two-sided uncorrupted when it accepted. This means that the first query Corrupt(A, B) or Corrupt(B, A) (if it exists) has been done either after the pth master key period for $\{A, B\}$, or during this p^{th} master key period for $\{A, B\}$ when both $\text{CTR}_{AB}^{\text{init,otp}}$ and $\text{CTR}_{BA}^{\text{resp}}$ were larger than C. In the first case, the master keys were known to $\mathcal{B}$ at that time. In the second case, $K_{AB}^{\text{init,otp}}[nC, n]$ and $K_{BA}^{\text{resp,otp}}[nC, n]$ were already zeroized to 0^n and thus $\mathcal{B}$ knew both MK_{AB} and MK_{BA}. Therefore, Z' cannot occur because of a Corrupt query in this case.

It remains to cover the case where $\pi_i^s.\rho =$ responder and $\pi_j^t.\rho =$ initiator. We already know that π_i^s was two-sided uncorrupted when it accepted. After this acceptance, $\mathcal{B}$ knows $K_{BA}^{\text{resp,otp}}[nC, n]$ because it has been zeroized and therefore $\mathcal{B}$ knows the whole key MK_{BA} when $\mathcal{A}$ queries Corrupt(B, A). If a Corrupt(A, B) query is issued after π_i^s has accepted, one must have $\text{T}_i^s = \text{T}_j^t$ when it occurs by condition 4(c) of Definition 4. Since in this case the second message of T_i^s is faithfully delivered from π_i^s to π_j^t, we know that one of the following cases occurs:

- π_j^t rejects because $C < \text{CTR}_{AB}^{\text{init,otp}}$,
- π_j^t rejects because of the $\text{Vrfy}^{\text{resp}}$ (which can only occur if the pth master key period for $\{A, B\}$ is already over),
- π_j^t accepts with accepting counter value C.

In all three cases, $\mathcal{B}$ knows the value of $K_{AB}^{\text{init,otp}}[nC, n]$ after this Send query (either because of zeroization, or because of master key re-initialization). Therefore, when $\mathcal{A}$ queries Corrupt(A, B), $\mathcal{B}$ already knows MK_{AB}. Thus, Z' cannot occur during a Corrupt query. In addition, by condition 4(d) of Definition 4, $\mathcal{A}$ does not query RevealKey(π_i^s). Therefore, the only way Z' could occur is because of a query RevealKey($\pi_j^{t'}$) made to an initiator oracle $\pi_j^{t'}$ with $\pi_j^{t'}.\text{pid} = P_i = B$ that accepted during the pth master key period for $\{A, B\}$ with accepting counter value C. By what we have proved above, $\pi_j^{t'}$ must be two-sided uncorrupted when it accepted, because otherwise, Z' would occur via a Corrupt query. By Lemma 22 and by the rule of G_1, we know that there exists a oracle $\pi_i^{s'}$ activated during the pth master key period for $\{A, B\}$ such that $\pi_j^{t'}$ has $\pi_i^{s'}$ as strong partner. Since $\text{T}_j^{t'}$ and $\text{T}_i^{s'}$ have the same second message, $\pi_i^{s'}$ accepted with accepting counter value C. Therefore, one has $\pi_i^{s'} = \pi_i^s$ (i.e. $s' = s$) and $\text{T}_j^{t'} = \text{T}_i^s$. This means that $\pi_j^{t'}$ and π_i^s are mutual partners. By condition 4(e) of Definition 4, this implies that RevealKey($\pi_j^{t'}$) could not have been issued, concluding the proof. □

References

[1] M. Atici and D.R. Stinson, Universal hashing and multiple authentication, *Advances in Cryptology – CRYPTO 1996, Springer Lecture Notes in Computer Science* **1109**, 16–30 (1996).

[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, *IACR 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] C.H. Bennett and G. Brassard, Quantum cryptography: Public key distribution and coin tossing, reprinted in 2014 in *Theoretical Computer Science* **560**, 7–11 (1984).

[7] 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).

[8] G. Brassard, N. Lütkenhaus, T. Mor and B.C. Sanders, Limitations on practical quantum cryptography, *Physical Review Letters* **85**, 1330 (2000).

[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] H.-P. Chen, M. Mohammad and L.B. Kish, Current injection attack against the KLJN secure key exchange, *Metrology and Measurement Systems* **23**, 173–181 (2016).

[11] 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).

[12] S. Ferdous and L.B. Kish, Transient attacks against the Kirchhoff–Law–Johnson–Noise (KLJN) secure key exchanger, *Applied Physics Letters* **122**, 143503 (2023).

[13] M.J. Fischer, M.S. Paterson and C. Rackoff, Secret bit transmission using a random deal of cards, in Distributed Computing and Cryptography, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **2**, 173–181 (1991).

[14] M.J. Fischer and R.N. Wright, An efficient protocol for unconditionally secure secret key exchange, *SODA 1993: Proceedings of the fourth annual ACM-SIAM symposium on discrete algorithms*, 475–483 (1993).

[15] E.N. Gilbert, F.J. MacWilliams and N.J.A. Sloane, Codes which detect deception, *The Bell System Technical Journal* **53**, 405–424 (1974).

[16] P.-A. Jacqmin and J. Liénardy, LP2+: a robust symmetric-key AKE protocol with perfect forward secrecy, and an advocacy for thorough security proofs, *IACR Cryptology ePrint Archive* **2025/983**, (2025).

[17] 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).

[18] N. Jain, E. Anisimova, I. Khan, V. Makarov, C. Marquardt and G. Leuchs, Trojan-horse attacks threaten the security of practical quantum cryptography, *New Journal of Physics* **16**, 123030 (2014).

[19] L.B. Kish, Totally secure classical communication utilizing Johnson(-like) noise and Kirchoff's law, *Physics Letters A* **352**, 178–182 (2006).

[20] L.B. Kish and C.G. Granqvist, On the security of the Kirchhoff-law–Johnson-noise (KLJN) communicator, *Quantum Information Processing* **13**, 2213–2219 (2014).

[21] H. Krawczyk, LFSR-based hashing and authentication, *Advances in Cryptology – CRYPTO 1994, Springer Lecture Notes in Computer Science* **839**, 129–139 (1994).

[22] L. Lydersen, C. Wiechers, C. Wittmann, D. Elser, J. Skaar and V. Makarov, Hacking commercial quantum cryptography systems by tailored bright illumination, *Nature Photonics* **4**, 686–689 (2010).

[23] U.M. Maurer, Perfect cryptographic security from partially independent channels, *STOC 1991: Proceedings of the twenty-third annual ACM symposium on Theory of Computing*, 561–571 (1991).

[24] U.M. Maurer, Information-theoretically secure secret-key agreement by NOT authenticated public discussion, *Advances in Cryptology – EUROCRYPT 1997, Springer Lecture Notes in Computer Science* **1233**, 209–225 (1997).

[25] M.Y. Melhem and L.B. Kish, A static-loop-current attack against the Kirchhoff-Law-Johnson-Noise (KLJN) secure key exchange system, *Applied Sciences* **9**, 666 (2019).

[26] M.N. Wegman and J.L. Carter, New hash functions and their use in authentication and set equality, *Journal of Computer and System Sciences* **22**, 265–279 (1981).

[27] A.D. Wyner, The wire-tap channel, *The Bell System Technical Journal* **54**, 1355–1387 (1975).