

Quantum Security of the UMTS-AKA Protocol and its Primitives, Milenage and TUAK

Abstract. The existence of a quantum computer is one of the most significant threats cryptography has ever faced. However, it seems that real world protocols received little attention so far with respect to their future security. Indeed merely relying upon post-quantum primitives may not suffice in order for a security protocol to be resistant in a full quantum world. In this paper, we consider the fundamental UMTS key agreement used in 3G but also in 4G (LTE), and in the (recently deployed) 5G technology. We analyze the protocol in a quantum setting, with quantum communications (allowing superposition queries by the involved parties), and where quantum computation is granted to the adversary. We prove that, assuming the underlying symmetric-key primitive is quantum-secure, the UMTS key agreement is also quantum-secure. We also give a quantum security analysis of the underlying primitives, namely Milenage and TUAK. To the best of our knowledge this paper provides the first rigorous proof of the UMTS key agreement in a strong quantum setting. Our result shows that in the quantum world to come, the UMTS technology remains a valid scheme in order to secure the communications of billions of users.

Keywords: Quantum cryptography · AKA protocol · 3G/4G/5G · Security model

1 Introduction

The UMTS-AKA protocol was specified in 1999 by the 3GPP [2] with a specific architecture in mind: a *client* is a subscriber of a telecommunication *operator* for mobile services (messages, Internet use, calls, ...) and the service is delivered through an intermediate local network operator, simply called *server*. Since then, this protocol has been, and still is, the foundation of all mobile communications in the world. Indeed, the 3G standard was replaced in 2010 by the 4G one (a.k.a. LTE for Long Term Evolution), but only one single bit of the UMTS-AKA protocol has been added to the previous version, its purpose being for parties to know whether the session keys must be used to protect the data (3G), or as input to a subsequent derivation function in order to compute more session keys (4G). Last year, the 5G specifications were released and, again, the UMTS-AKA remains the foundation which the 5G-AKA is based on. Some properties are added (related to the user's privacy, and the network's "awareness") but the core of the AKA is unchanged. These adaptations are motivated by the need for new telecommunication services, and not because of some security concerns.

Presently the UMTS-AKA protocol is used all over the world to secure the communications of any individual using a mobile phone.

Regarding security, the main purpose of such specifications is first for the client and the operator to authenticate to each other (through the intermediary of the server), and then to provide integrity and confidentiality of the future communications (voice and data) between the client and the server. To obtain such a protocol, the basic idea of the 3GPP is for the operator and the client to share a common long-term secret key sk_s . Regarding servers, they are considered as trusted for delivering services but not for keeping long-term secrets, hence the idea of designing a *three-party authenticated key agreement* between the client, the server and the operator. To add more constraints, the client side is managing a USIM card, whose computational capabilities are limited, and in particular is not able to generate good pseudo-random numbers.

More precisely, the main idea of the UMTS-AKA is as follows:

- the operator first generates a set of Authentication Vectors (AV) thanks to the shared key sk_s , some randomness R and a counter Sqn synchronized with the client. Such vectors are then sent to the server;
- using the received AV, the server interacts with the client to (i) permit the client to authenticate the operator (through an authentication message denoted $Auth$), (ii) authenticate the client (using a message authenticated code denoted Mac), and (iii) generate shared session keys for integrity (key IK) and confidentiality (key CK);
- those session keys are then used to secure the communication between the client and the server.

The security of this UMTS-AKA protocol was proved in a classical (i.e., non-quantum) setting by Alt, Fouque, Macario-Rat, Onete, Richard [5] (only considering the two first steps, and not treating the security of the subsequent communication). However, if the communication between the operator, the server and the client are now quantum, superposition queries are allowed, and if the adversary is granted with quantum computations, then the result given by Alt et al. no longer holds.

Consequently, in this paper, we treat the following important question: what is the exact security of the UMTS-AKA protocol in a full quantum world?

1.1 Quantum Threat and Current Research

Indeed, numerous computer scientists and experts consider as imminent the advent of quantum computers. The possibility that, in a near future, all our computational capabilities will be replaced by these new computers is today not so ludicrous, as proved by the recent IBM roadmap¹. Hence, quantum superposition and quantum entanglement will one day be used by any honest computing party to execute their part of a protocol, but also by any attacker against such

¹ See <https://research.ibm.com/blog/ibm-quantum-roadmap>

a protocol. And it is today well-known that such a quantum machine is one of the most significant threats cryptography has ever faced.

In 2016 the NIST has initiated a process to design, evaluate, and eventually standardize quantum-resistant public-key algorithms. This initiative was motivated by the well-known quantum insecurity of the most popular asymmetric schemes still in use today.

Regarding secret-key cryptography, since the seminal works of Kuwakado and Morii [27, 28], and Kaplan, Leurent, Leverrier, and Naya-Plasencia [25], the analysis of cryptographic primitives in a (post-)quantum setting has produced an increasing amount of results [6–8, 12, 13, 15, 17–19, 22, 23, 35]. Several of these results grant superposition queries to the adversary.

Most work concerning the analysis of (real-world) protocols in a (post-)quantum setting focuses on the performances, and the implementation issues (e.g., [21, 26, 32–34]). However they do not consider the possibilities provided to an attacker by superposition queries.

In this regard, Ebrahimi, Chevalier, Kaplan, and Minelli [20] and Music, Chevalier, and Kashefi [31] stand out by including in their analysis the powers granted by such quantum queries. They focus their attention respectively on oblivious transfer and Yao’s protocols [37]. We observe that Music et al. modified Yao’s protocol so that the attack they present can actually apply. Although this result demonstrates the separation between adversaries with and without superposition access, this seems to be based on a rather convoluted example of such a possibility.

In contrast, our paper focuses on the analysis of a quantum version of the real-world UMTS authenticated key agreement.

Regarding the quantum threat on the 5G protocol, Mitchell [30] raises two main issues: the size of the user symmetric key K (128 bits), and the asymmetric scheme used to encrypt the user permanent identifier. A single run of the 5G AKA (in fact a few data extracted from such a run) is enough for an attacker to execute the Grover’s algorithm and retrieve K . Moreover the asymmetric encryption scheme can be straightforwardly broken with the Shor’s algorithm. Mitchell recommends to use 256-bit permanent and session keys, and corresponding symmetric-key algorithms. Note that the 4G and 5G key derivation functions already compute 256-bit session keys, which are then truncated to 128 bits. Therefore these functions already comply with this change. It remains to define (in the 4G and 5G specifications) the corresponding symmetric-key functions. Although this issue must be taken into account, we observe that Grover’s algorithm is not the only threat against symmetric-key functions [8, 11, 12, 25, 27, 28]. Therefore the alternate algorithms must be chosen with care. Moreover Mitchell points also out that a quantum attack could target the function (undefined in the 5G specification) used by an operator to compute the multiple subscriber keys K . If all these keys derive from a single operator master key, then one attack may endanger a wide set of subscribers. Finally Mitchell recommends to use a post-quantum asymmetric encryption scheme instead of the current (classic) algorithm. Overall, Mitchell focuses on specific cryptographic primitives used in

5G but does not provide an extensive security analysis of the whole protocol in a quantum setting.

1.2 UMTS-AKA in a Full Quantum Setting

As sketched above, we made the choice in our work to study the security of the UMTS-AKA protocol in the strong full quantum setting (and not only the primitives it is made of). This means that both honest parties and adversaries are given quantum capabilities and quantum communications (i.e., superposition queries). Such a research is still in its infancy, but brings some important specificities that should be taken into account. While it could be surprising to put our study in such a strong setting, since this will not emerge overnight, this is indeed motivating by several arguments that we explain in this section.

On the difficulty and necessary time to design new versions. The UMTS-AKA protocol is used since the beginning of our century. In addition, it is the basis of the recently published 5G standard that is currently under deployment for the next decades. Hence, any new result related to the security of the UMTS-AKA protocol has also an impact on the security of 5G networks. Moreover, experience shows that technology may have a rather long lifespan. For instance 2G mobile networks are still alive in most parts of Europe, Africa, Central and South America² although their first deployments date from the early 90s, and are now outdated with the 3G and 4G technologies. The cryptographic components of the UMTS-AKA technology are allocated among servers, smartphones, and microchips (USIM) and it well-known that the life-cycle to replace cryptographic schemes takes time from design and security analysis, to (flawless) implementation and certification. No one can ensure that the 3G/4G technology will be replaced before quantum machines be fully functional, without speaking of the retroactive threat posed by a (future quantum) adversary that harvests data now in order to decrypt later. Additionally, making a new standard in the telecommunication setting is a very long process. As evidence, the work for next 6G telecommunication network has already started³ while it is planned to be only deployed in 10 to 15 years. No one could say today whether this generation will be fully quantum or not. Yet knowing if the UMTS-AKA can be used in this stronger setting is an important question that should be answered as soon as possible to be prepared to any choice for such future, and make the right decision as of today.

Strong security implies weak security. When analysing symmetric-key functions, superposition queries are granted to the adversary (as recalled in the previous section). When the symmetric-key functions are incorporated into

² See <https://www.gsma.com/mobileeconomy/>

³ See for example <https://www.6gworld.com/wp-content/uploads/2021/06/6G-Vision-Enabling-Technologies-David-Soldani-.pdf>.

an interactive protocol this implies quantum communications for both honest parties and adversaries. Moreover such a security model straightforwardly incorporates the case when honest messages are classical, which corresponds then to a weaker security configuration. Therefore, our setting allows us to be as general as possible, and directly proves the security of the UMTS-AKA protocol against a quantum-capable adversary, the exchanges between the client, the server and the operator being either classical or quantum.

One step on the security of a quantum Internet. Finally, the simplicity of the UMTS-AKA protocol and the way it uses symmetric cryptography and anti-replay techniques (using random queries and synchronized sequences) make it a good example for how to make a security proof for fully quantum protocols. Hence, the techniques that we introduce in this paper can be used to prove the security of the core protocols deployed in the current Internet, such as TLS, IPSec, SSH, etc. The discussion on the definition of a future quantum Internet has already begun⁴ and its security must be based on the fully quantum versions of all those protocols.

1.3 Our Contribution

In this paper, we first define a security model that does not rely on unconditional security and allows honest parties and attackers to use superposition messages. In particular, assuming that we are in a quantum world, we provide a discussion on what can be put into superposition by the client, the server and the operator during a honest execution of such *quantum UMTS-AKA* protocol. More precisely, the quantum version of the UMTS-AKA that we describe assumes quantum computations as well as quantum communications (i.e., the messages that are exchanged between parties are in a superposition of quantum states).

We then formally prove that this quantum version of the UMTS-AKA is as secure in this quantum model as it is in the classical setting [5]. As the quantum version we built is an extension of the classical one, the attack by Zhang is still valid [41]. It consists in using the extra authentication vectors of a corrupted server by the attacker elsewhere to make an unauthorized authentication. As a supplementary contribution, we also exhibit a new attack against the state confidentiality of a mobile user. This attack holds in the quantum but also in the classical setting.

We finally study the security of the underlying primitives that can instantiate the UMTS-AKA, Milenage and TUAK. We show an attack on Milenage as a qPRF, however the context of the UMTS-AKA makes this attack unrealistic and we further prove the security of Milenage based on the security of AES. We also show a reduction from the security of TUAK to the security of Keccak-f.

We believe that our work will pave the way for the systematic analysis and the better understanding of the currently deployed real-world protocols with regard to the quantum threat, and help design secure quantum networks.

⁴ See for example <https://www.energy.gov/articles/quantum-internet-future-here>.

2 The AKA Protocol

In this section, we present the standard protocol AKA used in the 3G and 4G infrastructure, and which is also the basis of the 5G AKA. The we describe a quantum variant supposed to be run between quantum computers and through superposition-allowing channels.

2.1 The Classic Version of AKA

The AKA procedure consists of an exchange of Message Authentication Codes (MAC) and key derivations from a fresh random value, and static secret keys shared by the *client* (Mobile Equipment/User Subscriber Identity Module, ME/USIM) and the *operator* (Home Location Register, HLR). It is executed as a challenge-response through a *server* (Visited Location Register, VLR).

Elements of the protocol. The main elements are the following.

Pseudo-random functions. The protocol defines seven functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ which take as input the secret keys and the random value R . f_1 and f_1^* take as additional inputs the sequence number (Sqn_C or $Sqn_{Op,C}$), and an Authentication Management Field (AMF). f_1 outputs the MAC tag Mac_S sent to the client. f_2 outputs the MAC tag Mac_C sent to the server. f_3 and f_4 output respectively the session keys CK (Confidentiality Key) and IK (Integrity Key). f_5 outputs an Anonymity Key (AK) used to mask Sqn . f_1^* and f_5^* intervene in place of f_1 and f_5 during the re-synchronization procedure. In practice, these functions are instantiated by either Milenage [1] or TUAK [3].

Static secret keys. The client key (sk_C) is known by the client and the operator. From the latter and the operator key (sk_{Op}) an intermediate value ($sk_{Op,C}$) is derived as $sk_{Op,C} = KD(sk_C, sk_{Op})$.⁵ The client stores $sk_{Op,C}$ while sk_{Op} is known only to the operator. We note $sk_s = sk_C || sk_{Op,C}$.

Random value. A fresh random value R is generated during each session, and used as challenge for the client. It aims at protecting the client and the operator from replay attacks.

Sequence number. The client and the operator keep respectively the secret values Sqn_C and $Sqn_{Op,C}$, which number the executions of the protocol, and prevent replay attacks against the client. A re-synchronization procedure may be executed in case of discrepancy.

We consider that the values Sqn_C do not repeat (otherwise the client would accept values from a previous execution of the protocol). To be sure the operators do not allow for a repetition of Sqn_C , we restrict the number of authentication vector all operators can produce with the same sk_C to be at most $\frac{2^{|Sqn_C|}}{\Delta}$ (since a client uses the same Sqn_C for every operator), where Δ , chosen by the operator, defines an acceptance interval.

⁵ sk_{Op} is denoted as OP_C in Milenage and TOP_C in TUAK.

Description of an execution. The main steps are the following (see Figure 1).

2.2 Quantum AKA Protocol

In this section we describe a quantum variant of the AKA protocol. More precisely, we consider a version within a full quantum world, where both honest and dishonest parties are given quantum capabilities. Hence, each party is able to perform quantum computations, and communication allows transferring qubits in superposition. Since the values can still be computed classically (measurements are still possible), this quantum variant is obviously an extension of the classical AKA.

Modelization of quantum communications. We base our modelization of quantum communications on the work of Yao [38]. For a communication between Alice and Bob, the total state $|a\rangle|c\rangle|b\rangle$ is composed of three parts:

- $|a\rangle$ is Alice’s secret part and modifiable only by her;
- $|b\rangle$ is Bob’s secret part and only modifiable by him;
- $|c\rangle$ is the state of the channel, it is modifiable by everyone, or only Alice and Bob if the channel is secure (like the ones between operators and servers).

Other modelizations. Other modelizations of quantum communications exists, like [16], where communications are made using EPR-pairs, pairs of entangled qubits shared by Alice and Bob. While this englobes our modelization, it implies making and storing those pairs. The latter seems highly unpractical.

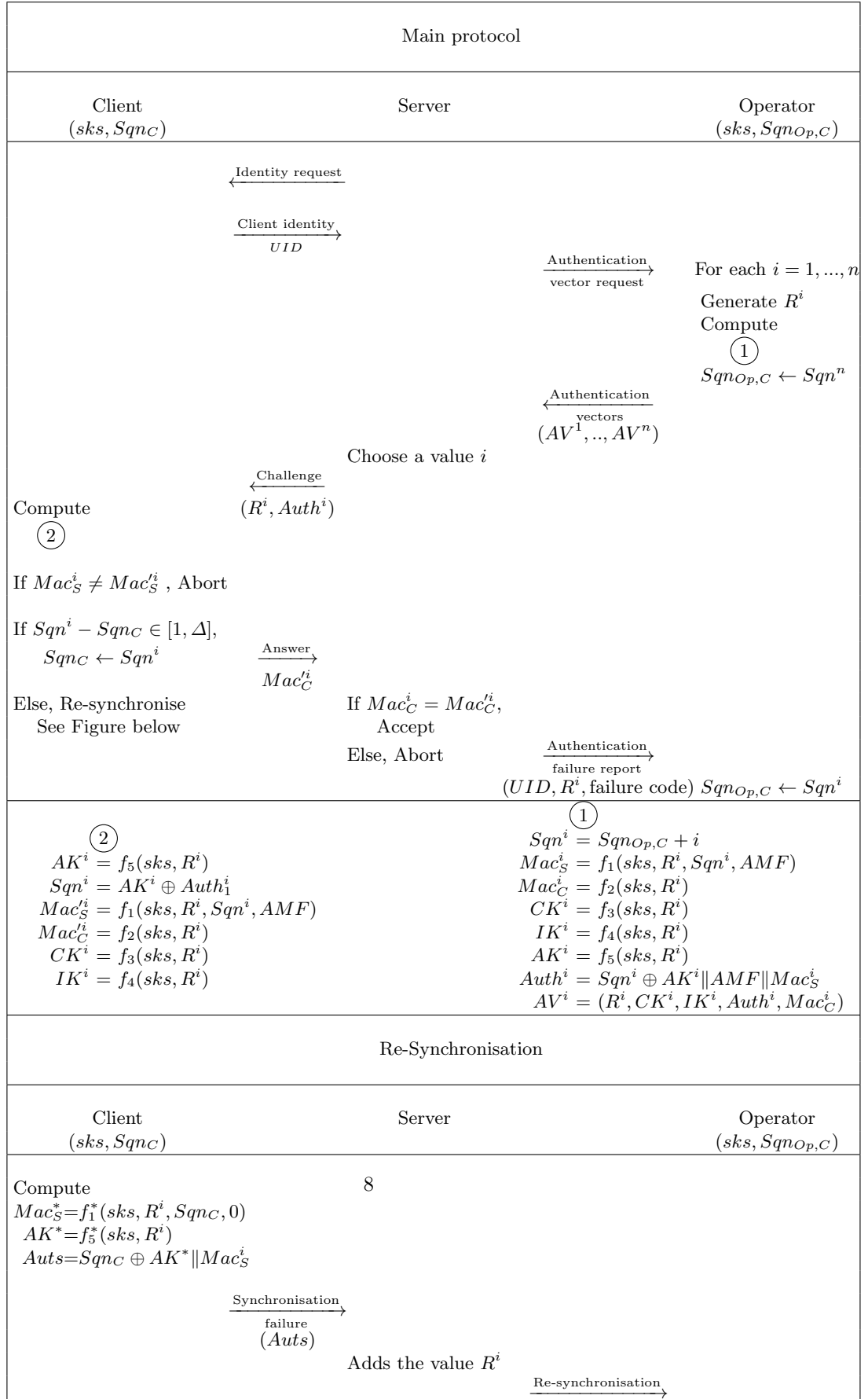
Superposition or not. We detail below whether the parameters of the protocol may be used as values in superposition in the quantum variant.

Static secret keys. These keys are long-term values whose lifespan depends only on the time needed by an attacker to find them. Having long-term keys maintained in superposition seems obviously unpractical. From now on, we consider client and operator keys (sk_C, sk_{Op}) to be in a classical state. Consequently $sk_{Op,C}$ is also in classical state.

Sequence number. We consider sequence numbers to be in classical state.

Random value. The random R does not need to be stored from one session to another. Therefore it is a matter of choice to keep it in classical state or superposition state. As the latter is a generalization of the former, we allow for the random values to be used in superposition. The operator can generate authentication vectors in superposition, and the attacker is allowed to send superposition queries to client and server parties.

Confidentiality and integrity keys. As R is allowed to be in superposition, we expect the session keys CK and IK to be in superposition too.



Challenge Generator. In the classical case [5], R is expected to be an unpredictable value from $\{0, 1\}^{|R|}$. In a quantum setting, R are elements of $\mathbb{C}^{2^{|R|}}$. Therefore our expectations of its randomness are meant to change. While the function G now has to resist quantum attacks, one could think the generation of R would face a similar challenge. Yet our security proofs only involve few properties of such a generator.

Expectations. We expect the function that generates $R = \sum_{i=0}^{2^{|R|}-1} a_i |i\rangle$ to have the following properties:

- a newly generated R is not entangled to any other R ;
- for all $i \in \{0, 1, \dots, 2^{|R|} - 1\}$, $\mathbb{E}[|a_i|^2] = 2^{-|R|}$.

We want the random generator to produce independent uncorrelated values as we want to take care of minimal long-term values (we have the keys for black-boxing G and Sqn for resistance toward replay attack). We want also the random generator to have a good distribution to make use of the full space of possible challenges.

Useful property for security proofs in later sections. Then we have the following property: with n values R^1, \dots, R^n , the probability to measure a collision R^1, \dots, R^n is less than $\frac{n^2}{2^{|R|}}$.

Possible construction. Note that $|R\rangle = \frac{1}{2^{|R|/2}} \sum_{i=0}^{2^{|R|}-1} |i\rangle$ is a possible option of generation that respects our expectations. While this challenge generation does not contain any randomness, it still has the properties we need. Indeed, as the further computations of G entangle R with other elements of the protocol that are uncomputable by the adversary, the underlying measurement results in the variability of the protocol.

Description of a quantum execution. The main steps of the quantum version of the protocol are the following (see Figure 2).

Generation of the challenge. For $1 \leq i \leq n$, the operator generates each superposition $\sum_{R^i} a_{R^i} |R^i\rangle$, increments $Sqn^i = Sqn_{Op,C} + i$, and computes the parameters entangled with R^i . The operator gets the superposition:

$$\sum_{R^i} a_{R^i} |R^i, Mac_S^i, Mac_C^i, CK^i, IK^i, AK^i\rangle$$

$$\begin{aligned} Mac_S^i &= f_1(sk_s, R^i, Sqn^i, AMF) & Mac_C^i &= f_2(sk_s, R^i) \\ CK^i &= f_3(sk_s, R^i) & IK^i &= f_4(sk_s, R^i) \\ AK^i &= f_5(sk_s, R^i) \end{aligned}$$

The operator computes $Auth^i = |Sqn^i \oplus AK^i\rangle |AMF\rangle |Mac_S^i\rangle$ and $AV^i = |R^i, CK^i, IK^i, Auth^i, Mac_C^i\rangle$, and sends the authentication vectors AV^i to the server.

Challenge-Response. The server picks a vector i and sends the corresponding challenge to the client. The client computes

$$\begin{aligned} AK^i &= f_5(sks, R^i) & Sqn^i &= AK^i \oplus Auth_1^i \\ Mac_S^i &= f_1(sks, R^i, Sqn^i, AMF) & Mac_C^i &= f_2(sks, R^i) \\ CK^i &= f_3(sks, R^i) & IK^i &= f_4(sks, R^i) \end{aligned}$$

At this point, several qubits are shared between the operator, the server and the client. They are entangled, and every state value depends only on the value R^i it was computed from. The quantum state of the system can be represented as

$$\sum_{R^i} a_{R^i} \left(\begin{array}{c} R^i \\ Mac_S^i \\ Mac_S^i \\ Mac_C^i \\ CK^i \\ IK^i \\ AK^i \\ Sqn_{Op,C} \end{array} , \begin{array}{c} R^i \\ Mac_S^i \\ Mac_C^i \\ CK^i \\ IK^i \\ AK^i \oplus Sqn_{Op,C} \\ AK^i \end{array} , \begin{array}{c} R^i \\ Mac_S^i \\ Mac_C^i \\ CK^i \\ IK^i \\ AK^i \end{array} \right)$$

The first, second and third columns correspond respectively to the client's, server's, and operator's view.

The end of the protocol goes as follows. The client measures $Mac_S^i \oplus Mac_S^i$ to verify that it equals 0, and sends an abort message if not. Then it measures Sqn^i and checks if the challenge is fresh, i.e., $Sqn^i \in [Sqn_C + 1, Sqn_C + \Delta]$. Finally the client answers the challenge, and sets Sqn_C to Sqn^i . The server checks the answer by measuring $Mac_C^i \oplus Mac_C^i$, and proceeds as in the classical protocol.

Re-synchronisation. If $Sqn^i \notin [Sqn_C + 1, Sqn_C + \Delta]$, the client triggers a re-synchronization procedure. First it computes the superposition $\sum_{R^i} a_{R^i} |R^i, Mac_S^*, AK^*\rangle$ where R^i corresponds to the current challenge and

$$AK^* = f_5^*(sks, R^i) \quad Mac_S^* = f_1^*(sks, R^i, Sqn_C, AMF)$$

The client generates $Auts = |(Sqn_C \oplus AK^*)||Mac_S^*\rangle$ and sends it to the server which forwards it as $\sum_{R^i} a_{R^i} |R^i, Auts\rangle$ to the operator. Then the operator computes

$$\begin{aligned} AK^* &= f_5^*(sks, R^i) & Sqn_C &= AK^* \oplus Auts_1 \\ Mac_S^* &= f_1^*(sks, R^i, Sqn_C, AMF) \end{aligned}$$

and verifies if the measure of $Mac_S^* \oplus Mac_S^*$ equals 0. If so, it updates $Sqn_{Op,C}$ to Sqn_C , otherwise it aborts the procedure.

Notes on the measurements. During the protocol, we measure Sqn_C as a value in classical state. We also need to measure $Mac_S \oplus Mac_S'$ and $Mac_C \oplus Mac_C'$ as it decides if the protocol continues or is interrupted.

Compared to the classical protocol, allowing for answers in superposition may mislead us into think that an adversary can have a higher probability to

get an accepting state by entering random values. As seen in the description of the protocol, every value is determined by R^i . For each R^i , the probability of acceptance being identical, the same holds regarding the superposition of R^i . Another way to see this phenomenon is to pass the deciding measurements, the answer needs to be measured as the expected superposition which is entangled with R .

3 Adversarial Model and Building Blocks

In this section, we define the quantum adversarial model for a quantum variant of the UMTS-AKA protocol, and some building blocks we need. We here adapt the work of Alt et al. [5] (done in a classical context) to our quantum setting.

3.1 Oracles

The adversary interacts with the system by means of the following oracles, in addition to the functions G and KD through their specifications.

CreateClient(Op) $\rightarrow (sk_C, ID_C, Sqn_C)$: This oracle (used by the challenger) creates a client C with unique identifier ID_C , the client's secret keys sk_C and $sk_{Op,C} = KD(sk_C, sk_{Op})$ and the sequence number Sqn_C . The tuples $(ID_C, sk_C, sk_{Op,C}, Sqn_C)$ are associated with the client ID_C and with the corresponding operator Op (i.e., each "copy" of Op in each server does this). The operator sets $Sqn_{Op,C} := Sqn_C$ and then keeps track of $Sqn_{Op,C}$. The adversary is given ID_C .

CreateOperator() $\rightarrow (sk_{Op}, ID_{Op})$: This oracle (used by the challenger) creates an operator Op with unique identifier ID_{Op} , the operator's secret keys sk_{Op} . The adversary is given ID_{Op} .

CreateServer(Op) $\rightarrow (ID_S)$: This oracle (used by the challenger) creates a server S with unique identifier ID_S . The challenger makes a secure channel between the new server and the operator Op . The adversary is given ID_S .

NewInstance(P) $\rightarrow (P_j, m)$: This oracle instantiates the new instance P_j , of party P , which is either a client or a server. Furthermore, the oracle also outputs a message m , which is either the first message in an honest protocol session (if P is a server) or \perp (if P is a client).

Execute(C, i, S, j) $\rightarrow \tau$: Selects the client C and the server S , creates (fresh) instances C_i, S_j , allocates the necessary quantum memory for these instances, then runs the protocol between them using the allocated quantum memory. The adversary has access to the transcript of the protocol instance τ via the channel state.

As for the immediate description of the protocol, there is no use of the results CK, IK , the adversary gets maximum control of the transcript by making the client uncompute its quantum values of the protocol (except R as it comes from the challenge sent by the server) and the challenger uncompute the quantum values of the protocol in the sever and in the operator (except

R , as it links the quantum state of the protocol, and the ones concerned by a *RevealSession* or *Test* query, see below) in the clients, servers and operators. Then the only remaining value in superposition are the R in the client and in the operator which are sent to the attacker (it already has access to it).⁶

(One could consider the whole system to be a simulation and then executing the protocol on superpositions of clients, servers and operators but this is not our interest here. As such, this oracle takes classical values but returns a script written on qubits and most often in superposition, as specified in Section 2.2).

Send(P, i, m) $\rightarrow m'$: Simulates sending the message m to the instance P_i of P . The output is a response message m' (which is set to \perp in case of an error or an abort). As our protocol is meant to run on superposition allowing channels, the messages m and m' use qubits and can be in superposition.

RevealSession(P, i) $\rightarrow \{K, \perp\}$: This oracle can be used just after an execution of the protocol. If the party has not terminated in an accepting state, this oracle outputs \perp , else it outputs the session keys $K = (CK, IK)$ computed by P_i on the last execution involving P_i (as explained in Section 2.2 the session keys are expected to be a superposition of values).

CorruptClient(C) $\rightarrow (sk_C, \{sk_{Op,C}\})$: This oracle corrupts C and returns the long-term client key sk_C and $sk_{Op,C}$ (not in superposition), but not sk_{Op} as the client is not given the operator keys.

CorruptServer(S): This oracle corrupts the server S and gives the adversary access to a special oracle *OpAccess*.

OpAccess(S, C) $\rightarrow m$: For a corrupted server S , this oracle gives the adversary access to the server's local copy of all the operators, in particular returning the message that the operator Op would output to this server for initializing an execution of the protocol with the client C . (We do not consider superpositions of parties, and the output message can be a superposition.)

RevealState(C, i, b) $\rightarrow Sqn_C$: For a client C , if $b = 0$, then this oracle reveals the current state of C_i , else, if $b = 1$, then the oracle returns the state the operator stores for C (Sqn_C is not a superposition).

Test(P, i) $\rightarrow K$: This oracle can be used just after an execution of the protocol. It is initialized with a secret random bit b and secret random functions f'_3 and f'_4 whose outputs are in the same space as CK and IK respectively. It returns \perp if the instance P_i is not fresh or if it has not terminated in an accepting state (with a session key CK, IK). If $b = 0$, then the oracle returns $CK = f_3(R), IK = f_4(R)$, else it returns $CK' = f'_3(R), IK' = f'_4(R)$. We assume that the adversary makes a single *Test* query.

Partners. Two instances P_i and P'_j are partners if:

- one is from a server and one is from a client;
- both instances are in accepting state;

⁶ This process separates the used memory of previous executions entangled with the attacker and the memory in new executions of the protocol.

- they share the same entangled superposition $sid = (R, AK \oplus Sqn_C = AK \oplus Sqn_{Op,C})$.

Note that sid is a superposition included in the challenge transmitted by the server to the client. sid still contains information corresponding to the operator and the intended client as it is computed using their keys.

The sid are entangled values not only in themselves but also between the partnered instances as seen in the description of the quantum protocol. Then, deciding whether two instances are partnered can be done by xoring their sid . The result is 0 if they are effectively partnered and not 0 with overwhelming probability otherwise.

3.2 Targeted Notions

We here only give a short description of each security properties. Detailed definitions (together with proofs) are given in Section 4.

Indistinguishability of the output of the protocol: a man-in-the-middle (MitM) attacker should not be able to distinguish (with significant probability) the output of an execution of the protocol from random values, even with corruptions of other clients and servers (strong indistinguishability).

Resistance to client impersonation: a MitM attacker should not be able to make an uncorrupted server accept an execution (with significant probability) without relaying expected messages, even with corruptions of other clients and servers (strong resistance).

Resistance to server impersonation: a MitM attacker should not be able to make an uncorrupted client accept an execution (with significant probability) without relaying expected messages.

Soundness: a malicious server should not be able to make more authentication (with significant probability) with uncorrupted clients than what operators have allowed.

State-confidentiality: a malicious server should not be able to recover the values $sk_{Op}, sk_{Op,C}, sk_C$ or Sqn_C significantly faster than guessing them.

Communications between operators and servers. Many elements that could easily break the security of the authentication are communicated between the operators and the servers. As such we consider the channels of communications between the operators and the servers to be secure.

3.3 Pseudo-random Function

As seen in Section 2, the protocol's security depends on the functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ implemented with either Milenage or TUAK. We need to assess how they impact the protocol security. For convenience, we denote $G = (f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*)_{sk_{Op,C}, sk_C}$.

$$G : (R, Sqn_{Op,C}, AMF) \mapsto (Mac_S, Mac_C, CK, IK, AK, Mac_S^*, AK^*)$$

Following Alagic, Broadbent, Fefferman, Gagliardini, Schaffner, Jules [4] and Zhandry [39], we estimate the protocol's security based on the security of G . As such, the latter is defined by the ability to distinguish a group of implementations of G from random functions with the property that only the outputs of f_1 and f_1^* depend on $Sqn_{Op,C}$ and AMF .

Quantum advantage on a group of pseudo-random functions. More precisely, we define the game for the quantum pseudo-randomness of G as follows.

1. The challenger generates random independent keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ to key the pseudo-random functions $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$, and $7n_{Op}n_C$ random functions.⁷ The challenger gives the attacker either the random functions or the pseudo-random ones as a quantum black box $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}}$ such that $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}}(|x\rangle|y\rangle) = |x\rangle|y \oplus f_{i,sk_{C,j},sk_{Op,C,j,j'}}(x)\rangle$.
2. The attacker can use the given function in quantum circuits.
3. The attacker guesses if the given black box is an implementation of the pseudo-random functions or random ones, and wins if it is right.

The pseudo-random function is $(q, t, n_{Op}, n_C, \epsilon)$ -indistinguishable if no attacker \mathcal{A} running in time t with q uses of any black box $O_{f_{i,sk_{C,j},sk_{Op,C,j,j'}}$ has an advantage $\text{Adv}(\mathcal{A}) = 2(\mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2})$ more than ϵ . We also define $\text{Adv}(G)_{n_{Op},n_C}^{q,t} = \sup_{\mathcal{A}} \{\text{Adv}(\mathcal{A})\}$.

Note that our definition is similar to what is usually depicted as advantage on a secret-key quantum pseudo-function. The main difference is the reuse of sk_C with different $sk_{Op,C}$. This notion makes related-key attacks happen faster than the standard case having the full keys being random when the key difference is on $sk_{Op,C}$.

3.4 Pseudo-random Key-derivation

Through corruption, one can reasonably expect some keys sk_C and $sk_{Op,C}$ to get leaked. Such an event can occur but should not reveal anything about any sk_{Op} or unlearned sk_C and $sk_{Op,C}$. As said in Section 2, the $sk_{Op,C}$ are obtained through a key derivation function KD . We define the advantage on KD against both of these issues.

Advantage on a pseudo-random key-derivation function. We define the game about the pseudo-randomness of KD as a key-derivation function as follows.

1. The challenger generates random independent keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,1}, \dots, sk_{Op,n_{Op}}$. The challenger generates the derived keys $sk_{Op,C,1,1} = KD(sk_{C,1}, sk_{Op,1}), \dots, sk_{Op,C,n_C,n_{Op}} = KD(sk_{C,n_C}, sk_{Op,n_{Op}})$. The challenger gives the attacker the client keys and either the derived keys or random values of the same length.

⁷ This is the amount of generated functions since for each couple (C, Op) there is a generation of G .

2. The attacker guesses if the given keys are generated with KD or random values. The attacker wins if the guess is right.

The pseudo-random key derivation is $(t, n_{Op}, n_C, \epsilon)$ -indistinguishable if no attacker \mathcal{A} running in time t has an advantage $\text{Adv}(\mathcal{A}) = 2(\mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2})$ more than ϵ . We also define $\text{Adv}(KD)_{n_{Op}, n_C}^t = \sup_{\mathcal{A}} \{\text{Adv}(\mathcal{A})\}$.

Note that guessing a value sk_{Op} is an attack. For one operator, this game can be seen as a known-plaintext game for the pseudo-random function $sk_C \mapsto KD(sk_C, sk_{Op})$.

4 Quantum Security of the UMTS-AKA Protocol

In this section, we present detailed definitions of the security properties by means of games executed between a challenger and an adversary. Then, we prove the adversary's advantages on those games to be negligible with the use of secure primitives. Sketch of proofs are provided in this section, and full proofs in Appendix B. In this section, we consider a generic secure pseudo random function G . As said above, it can be instantiated using either Milenage or TUAk. The exact quantum security of those two instances are studied in Sections 5 and 6 respectively.

4.1 Session Keys Indistinguishability

Freshness: session keys indistinguishability. An instance P_i is fresh if the adversary has not used the oracles *CorruptClient*, *CorruptServer* or *RevealSession* on P_i or on any of its partners.

Session keys indistinguishability game. We define the session keys indistinguishability game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target, a server or a client, and an operator.
3. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker.
4. The attacker uses the *Test* query on a fresh instance of the target.
5. The attacker is an active MtM for executions of the protocol whose participants are decided by the attacker (second phase).
6. The attacker guesses the secret bit b of the *Test* query. The attacker wins if the guess is right and the target instance is still fresh.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -strongly key-indistinguishable if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server (outside an execution of the protocol) has an advantage $\text{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2}$ more than ϵ .

Theorem 1 (Session Keys Indistinguishability). *The advantage on the session keys indistinguishability game for the UMTS-AKA protocol is bounded as follows:*

$$\text{Adv}(\mathcal{A}) \leq \frac{1}{2^{\text{Mac}_S}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t + \frac{(q_{exec} + q_{Op})^2}{2^{|R|}}$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof (sketch). We proceed by a sequence of games.

We first replace KD by a perfect key derivation KD' on which the adversary has no advantage. The loss is exactly the same as the one of an adversary against the pseudo-randomness of the key derivation, given in Section 3. Then we replace the pseudo-random function G by a perfect pseudo-random function G' . We can build a quantum adversary against G with a total of $q_{req} = 10q_{exec} + 4q_{res} + 5q_{Op}$ requests, corresponding to the number of queries needed to simulate the games. Indeed, during an execution, the operator and the client each computes f_1, f_2, f_3, f_4 and f_5 . Then, for a re-synchronisation, the client and the operator each computes f_1^* and f_5^* and for a corrupted request, the operator computes f_1, f_2, f_3, f_4 and f_5 . The loss between both game is then $\text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$.

Our next step is to reduce the configuration to one client, one server and one operator. Regarding clients and operators, the attacker does not have benefits from having other such entities, as the keyed G' are uncorrelated (G' and KD' are now perfect) and the attacker has only one target : the corresponding loss is null. Regarding servers, as the values exchanged do not depend on the server, we essentially make the attacker unable to corrupt them. Since corruption of servers only gives access to authentication vectors which would be given during an execution and reveal, the queries from a corrupt server are now counted as additional executions of the protocol: $q'_{exec} = q_{exec} + q_{Op}$. The difference between the games with several or one single server is given by the modification of this value q'_{exec} .

We finally consider that the challenger now generates the R^i before the game, and measures the property of R^i being all different. With our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q_{exec}^2}{2^{|R|}}$.

In the last game, the attacker has to distinguish between values in superposition generated by a perfect pseudo-random function from different R^i and true random values. But even with the access on a superposition, none of the quantum states of the authentication vector carries information that could grant an advantage for the adversary. Another possibility for the attacker is to send a different challenge by guessing the output Mac_S with a value Sqn_C never seen before. This happens with probability $\frac{1}{2^{\text{Mac}_S}}$, which concludes the proof. \square

4.2 Client-impersonation Resistance

Freshness: client-impersonation resistance. An instance P_i is considered fresh if the adversary has not used the oracles *CorruptClient* or *CorruptServer* on P_i or on any of its partners.

Relay: client-impersonation. The attacker is considered to be a simple relay during an execution of the protocol between a server instance S_j and a client instance C_i if the following events happened in the following order:

1. use of the oracle $Send(S, j, m)$ where m is the UID of the client C , initializing the execution of the protocol;
2. use of the oracle $Send(C, i, m')$ where m' is a projection of the output of $Send(S, j, m)$;
3. use of the oracle $Send(S, j, m'')$ where m'' is a projection of the output of $Send(C, i, m')$.

Client-Impersonation resistance game. We define the client-impersonation resistance game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target, a server, and an operator.
3. The attacker corrupts any number of clients and servers except the target.
4. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker.
5. The attacker wins if he successfully breaks the client-impersonation resistance for the target server, i.e., makes the target server instance accept, is still fresh and either the server instance is not partnered with the intended client, the server instance is partnered with a non intended client or the attacker has not been a simple relay.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -strongly client-impersonation resistant if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server has an advantage $Adv(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 2 (Client-impersonation resistance). *The advantage on the client-impersonation resistance game for the UMTS-AKA protocol is bounded as follows:*

$$Adv(\mathcal{A}) \leq \frac{(q_{exec} + q_{Op})^2}{2^{|R|}} + Adv(G)_{n_{Op}, n_C}^{q_{req}, t} + Adv(KD)_{n_{Op}, n_C}^t + \frac{q_{exec}}{2^{|MacC|}}$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof (sketch). As for the proof for the indistinguishability property, we proceed by a sequence of games, and start by replacing KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage. The loss here corresponds to $Adv(KD)_{n_{Op}, n_C}^t$ and $Adv(G)_{n_{Op}, n_C}^{q_{req}, t}$ respectively, where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ (with an argument similar to the one given in Section 4.1). Then we reduce the configuration to one client, one server and one operator with a similar result in the loss. The only exception is that to win the next games, we need to mark those additional executions as not valid. Then

$q_{valid} = q_{exec}$ gives the number of valid executions for winning (w.l.o.g., the first executions are the marked ones and the valid ones are at the end). We give the last game as for the indistinguishability one by generating R^i before the game, and measuring the property of R^i being all different. Regarding this last game, since the uncorrupted server needs the first and last messages in that order to be able to accept the execution, the attacker cannot ask the answer to the client. Otherwise, it will be a simple relay and then has to guess values Mac_C generated by a perfect pseudo-random function with a R^i different from what he has seen as none of the quantum state of the protocol executions carries any information useful to the attacker. Hence, the probability for an attacker to win this game is equal to $\frac{q_{valid}}{2^{|Mac_C|}}$, which concludes the proof. \square

4.3 Weak Server-Impersonation Resistance

Freshness: server-impersonation resistance. An instance P_i is considered fresh if the adversary has not used the oracles *CorruptClient* or *CorruptServer* on P_i or on any of its partners.

Relay: server-impersonation. The attacker is considered to be a simple relay during an execution of the protocol between a server instance S_j and a client instance C_i if the following events happened in the following order:

1. use of the oracle $Send(S, j, m)$ where m is the *UID* of the client C , initializing the execution of the protocol;
2. use of the oracle $Send(C, i, m')$ where m' is a projection of the output of $Send(S, j, m)$.⁸

Weak server-impersonation resistance game. We define the weak server-impersonation resistance game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target, a client, and an operator.
3. The attacker corrupts any number of clients except the target.
4. The attacker is an active MitM for executions of the protocol whose participants are decided by the attacker.
5. The attacker wins if he successfully breaks the server-impersonation resistance for the target client, i.e.s makes the target client instance accept, is still fresh and either the client instance is not partnered with the intended server, or is partnered with an other server, or the attacker has not been a simple relay.

The protocol is $(q_{exec}, q_{res}, t, \epsilon)$ -weakly server-impersonation resistant if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol and q_{res} re-synchronizations has an advantage $\text{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

⁸ We do not consider the last message of the protocol in the client-impersonation game as it does not interfere with the client acceptance.

Theorem 3 (Weak server-impersonation resistance). *The advantage on the server-impersonation resistance game for the UMTS-AKA protocol is bounded as follows:*

$$\text{Adv}(\mathcal{A}) \leq \frac{q_{exec}}{2^{|Mac_S|}} + \frac{q_{exec}^2}{2^{|R|}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

where $q_{req} = 10q_{exec} + 4q_{res}$.

Proof (sketch). Again we start our games by first replacing KD with a perfect key derivation KD' and G with a perfect pseudo-function G' , with $q_{req} = 10q_{exec} + 4q_{res}$, and then reducing the configuration to one client and one operator (the case of servers is here treated at the end of the proof). The result is again similar to what was done for the indistinguishability property in Section 4.1. We define the next game as the same as the previous one but the defender now generates R^i before the game and measures the property of R^i being all different. In the last game, we finally consider that there is only one server. As servers are incorruptible, they can only partner with a client that shares the same R they got from an operator. Then, as the R are all different, an unintended server will not partner.

Since the uncorrupted client of the last game needs the second message to accept, the attacker cannot ask the answer to the server or it will be a simple relay. It has then to guess values Mac_S generated by a perfect pseudo-random function with a Sqn_C different from what he has seen, as none of the quantum state of the protocol executions carries any information useful to the attacker. Hence, the winning probability for an attacker is $\frac{q_{exec}}{2^{|Mac_S|}}$, which concludes the proof. \square

4.4 Soundness

Soundness game. We define the soundness game as follows.

1. The challenger generates n_C clients and n_{Op} operators and their respective keys. Then the challenger generates q_{Op} authentication vectors and reveal them to the attacker.
2. The attacker is the server for executions of the protocol with the clients decided by the attacker.
3. The attacker wins if he makes $q_{Op} + 1$ executions of the protocol accept.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -server-sound if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol and q_{res} re-synchronizations has an advantage $\text{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 4 (Soundness). *The advantage on the soundness game for the UMTS-AKA protocol is bounded as follows:*

$$\text{Adv}(\mathcal{A}) \leq \frac{q_{exec} + q_{Op}}{2^{|Mac_S|}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof (sketch). We proceed as before by replacing the structural elements KD and G with perfect ones. Again, one can easily use the same argument used in Section 4.1 with $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ for G . Our next step is to reduce the configuration to one client and one operator. As previously, a difference would mean an advantage over KD' or G' . In the resulting game, the attacker has to guess values Mac_S generated by a perfect pseudo-random function with a Sqn_C different from what he has seen, which comes with probability less or equal to $\frac{q_{exec} + q_{Op}}{2^{|Mac_S|}}$. This concludes the proof. \square

4.5 State Confidentiality

Freshness: state confidentiality. An entity P is fresh if the adversary has not used the oracles *CorruptClient* or *RevealState* on any instance of P .

State confidentiality game. We define the state confidentiality game as follows.

1. The challenger generates n_{Op} operators, n_S servers and n_C clients and their respective keys.
2. The attacker fixes a target, a client, and an operator.
3. The attacker corrupts any number of clients except the target.
4. The attacker is the server for executions of the protocol with the clients decided by the attacker.
5. The attacker sends a tuple $(sk_C, sk_{Op,C}, sk_{Op}, n, Sqn_{C,n})$, wins if he guesses right any of the values sk_C , $sk_{Op,C}$, sk_{Op} or $Sqn_{C,t}$ which is the value Sqn_C at the end of the execution n of the target client, and the client is still fresh.

The protocol is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -state-confidential if no attacker \mathcal{A} running in time t with q_{exec} executions of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server has an advantage $Adv(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins})$ more than ϵ .

Theorem 5 (State confidentiality). *The advantage on the state confidentiality game for the UMTS-AKA protocol is bounded as follows:*

$$Adv(\mathcal{A}) \leq Adv(G)_{n_{Op}, n_C}^{q_{req}, t} + Adv(KD)_{n_{Op}, n_C}^t + \frac{8\Delta q_{exec}}{2^{|Sqn_C|}} + \frac{1}{2^{|sk_C|}} + \frac{1}{2^{|sk_{Op,C}|}} + \frac{1}{2^{|sk_{Op}|}}$$

where $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$.

Proof (sketch). As previously, our first step is to replace structural elements KD and G with perfect ones. The number of requests related to $Adv(G)_{n_{Op}, n_C}^{q_{req}, t}$ is here equal to $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$. Our second step is to reduce the configuration to one client and one operator with a similar argument and result, as before. Eventually, in the resulting final game, the attacker can only win by either sending the correct value for sk_C , $sk_{Op,C}$ or sk_{Op} or getting a right couple $(n, Sqn_{C,n})$. Regarding the keys, the attacker can only guess them since G' and KD' are perfect, hence the values $\frac{1}{2^{|sk_C|}}$, $\frac{1}{2^{|sk_{Op,C}|}}$ and $\frac{1}{2^{|sk_{Op}|}}$.

The case of the couple $(n, Sqn_{C,n})$ is more tricky. We here display a new theoretical attack against the state confidentiality of a mobile user. This attack gives us our limit below. It holds in the quantum but also in the classical setting, and is of independent interest.

In fact, the attacker being the server, it controls and knows the differences $Sqn_{C,n+1} - Sqn_{C,n}$ of the value Sqn_C at times n and n' (which by design is between 1 and Δ). From executions and resynchronisations, the attacker only gets $Sqn_{Op,C,n} \oplus AK^n$ and $Sqn_{C,n} \oplus AK^{*n}$. Considering G' and KD' as perfect, as it was done in the first game above, the attacker can only learn $Sqn_{C,n} \oplus Sqn_{C,n'}$ when R^n is the same as $R^{n'}$ (which can happen since the attacker can use an old authentication vector and get the result from the resynchronisation procedure). But the attacker can indeed learn the value of all the bits of $Sqn_{C,n}$ that changes throughout the game. For example, $Sqn_C \oplus (Sqn_C + 1)$ is of the form $0 \dots 01 \dots 1$ where the number of 1 is one more than the 2-adic valuation of $Sqn_C + 1$ (the details of this property change with the value of the difference between the successive values but the principle remains the same). As we expect around $\log(\Delta q_{exec}) + 3$ bits to be affected, the probability for the attacker to guess a good couple (n, Sqn_C) is at most $\frac{8\Delta q_{exec}}{2^{|Sqn_C|}}$.

This concludes this proof. \square

In the above security analysis, we have idealized the core pseudo-random function and our purpose in the sequel is to study the quantum security of the two standardized implementations: Milenage and TUAK.

5 Quantum Security of Milenage

In the last section, we proved security properties of the UMTS-AKA protocol assuming the security of the primitives underneath. We resolve this point in this section, discussing the quantum security of Milenage, and the next one, TUAK.

5.1 Description of Milenage

Milenage is a construction based on a block cipher noted E that acts on 128-bit messages (KASUMI for 3G and AES for 4G/5G).

Key Derivation. The keys $sk_{Op,C}$ are computed as $sk_{Op,C} = E_{sk_C}(sk_{Op}) \oplus sk_{Op}$.

Functions $f_1, f_2, f_3, f_4, f_5, f_1^, f_5^*$.* From the input R, Sqn, AMF , intermediate values T and I are computed as $T = E_{sk_C}(R \oplus sk_{Op,C})$ and $I = Sqn \parallel AMF \parallel Sqn \parallel AMF$.

Then five values are computed as follows:

- $out_1 = E_{sk_C}(T \oplus rot_{64}(I \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_2 = E_{sk_C}(c_2 \oplus rot_0(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_3 = E_{sk_C}(c_3 \oplus rot_{32}(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$
- $out_4 = E_{sk_C}(c_4 \oplus rot_{64}(T \oplus sk_{Op,C})) \oplus sk_{Op,C}$

- $out_5 = E_{sk_C}(c_5 \oplus rot_{96}(T \oplus sk_{O_p,C})) \oplus sk_{O_p,C}$
- f_1 outputs the first 64 bits of out_1 .
- f_1^* outputs the last 64 bits of out_1 .
- f_2 outputs the last 64 bits of out_2 .
- f_3 outputs out_3 .
- f_4 outputs out_4 .
- f_5 outputs the first 48 bits of out_2 .
- f_5^* outputs the first 48 bits of out_5 .

5.2 Attack on Milenage as a Pseudo-Random Function

We present the main algorithmic components before the complete attack, Grover’s search and Simon’s algorithm respectively.

Theorem 6 (Theorem 2 in [14]). *Let \mathcal{A} be a quantum algorithm that uses no measurements, let $f : X \rightarrow \{0, 1\}$ be a boolean function that tests if an output of \mathcal{A} is “good”. Let a be the success probability of \mathcal{A} . Let O_0 be the “inversion around zero” operator that does: $O_0|x\rangle = (-1)^{x \neq 0}|x\rangle$ and O_f a quantum oracle for f : $O_f|x\rangle = (-1)^{f(x)}|x\rangle$. Let $\theta_a = \arcsin \sqrt{a}$ and $0 < \theta_a \leq \frac{\pi}{2}$. Let $t = \left\lfloor \frac{\pi}{4\theta_a} \right\rfloor$. Then by measuring $(\mathcal{A}O_0\mathcal{A}^\dagger O_f)^t \mathcal{A}|0\rangle$, we obtain a “good” result with success probability greater than $\max(1 - a, a)$.*

Theorem 7 (Theorem 2 in [29]). *Suppose that $f : \{0, 1\}^n \rightarrow X$ has a period s , i.e. $f(x \oplus s) = f(x)$ for all $x \in \{0, 1\}^n$ and satisfies*

$$\max_{t \notin \{0, s\}} \mathbb{P}(f(x \oplus s) = f(x)) \leq \frac{1}{2}$$

When we apply Simon’s algorithm to f , it returns s with a probability at least $1 - 2^{-n} \cdot (3/4)^{cn}$. It is running in cn queries to f and time cn^2 .

Simon’s algorithm can be used without measurements and can answer whether s exists or not by making the classical post-processing of the Simon’s algorithm into quantum. This procedure has been widely described in [10, 29]. For our use, $c = 5$ will ensure a failing probability lower than 2^{-n} , which makes this algorithm look exact up to 2^n uses. The final algorithm has a time complexity in $125n^3$.

Idea of the attack. The attack relies on the following property of Milenage.

For any value R_0 and R_1 for R , $E(R_0 \oplus sk_{O_p,C}) = A||B$, $E(R_1 \oplus sk_{O_p,C}) = C||D$, if $A \oplus B = C \oplus D$ (probability 2^{-64} to happen), then for $x = SQN||AMF$ and $b \in \{0, 1\}$, $F : b||x \mapsto f_1(R_b, Sqn, AMF)$ had a period $1||(A \oplus C)$. (This should not happen for a random function.)

We will fix R_0 and search for a R_1 such that F has a period.

(Then querying F on $0||x$ will give the value of F on $1||(x \oplus A \oplus C)$, which also breaks the security of f_1 as a MAC function.)

A pseudo-random function is $(q_{exec}, q_{res}, q_{Op}, t, \epsilon)$ -strongly in AKA-indistinguishable if no attacker \mathcal{A} running in time t with q_{exec} instances of the protocol, q_{res} re-synchronizations and q_{Op} authentication vectors asked by a corrupted server (outside an execution of the protocol) has an advantage $\text{Adv}(\mathcal{A}) = \mathbb{P}(\mathcal{A} \text{ wins}) - \frac{1}{2}$ more than ϵ .

From the structure of UMTS-AKA, an execution of the protocol uses only G twice: once for generating the authentication vectors and once to verify them. Note that in order to get an interesting answer from the client, the attacker needs to pass through the client verification with a different element. The verification of the server is not an oracle since it exclusively uses the authentication vector it is given, which the attacker has also access to). For simplicity, we give the attacker the values $out_1, out_2, out_3, out_4, out_5$ instead of $f_1, f_2, f_3, f_4, f_5, f_1^*, f_5^*$ with no loss of generality (we actually give more to the attacker this way). We recall that Sqn and AMF are classical values. Then we can replace the previous game with the following (named \mathbb{G}_0) which is obviously exactly equivalent to the one above. There is no difference in winning one or the other game.

1. The challenger generates random independent keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ for the studied pseudo-random functions $out_1, out_2, out_3, out_4, out_5$ and $5n_{Op}n_C$ random functions. The challenger will use either the random functions or the pseudo-random ones as a quantum black box $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}$ such that $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}(|x\rangle|y\rangle) = |x\rangle|y \oplus out_{i, sk_{C,j}, sk_{Op,C,j,j'}}(x)\rangle$.
2. The challenger now gives the attacker authentication vectors and the associated answers (i.e. elements $\sum_R a_R |R, out_1, out_2, out_3, out_4, out_5\rangle$) and access to black-box verification functions F_1 and F_1^* that take quantum superpositions of states $|x, y\rangle$ and measures whether $y = f_1(x, Sqn, AMF)$ and $y = f_1^*(x, Sqn, AMF)$ respectively and give the result on a classic bit.
3. The attacker is free to interact with those elements.
4. The attacker guesses whether the challenger is using G or random functions and wins if the guess is right.

The number of given authentication vectors is bounded by $q_{vect} = q_{exec} + q_{Op}$, the number of calls to F_1 is bounded by $q_{F_1} = q_{exec}$ and, the number of calls to F_1^* is bounded by $q_{F_1^*} = q_{res}$.

We now show that Milenage is G -in-AKA indistinguishable by proving the following theorem.

Theorem 8 (G -in-AKA indistinguishability). *The advantage on the G -in-AKA indistinguishability game for Milenage is bounded as follows:*

$$\text{Adv}(\mathcal{A}) \leq \frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|R|}} + \frac{50q_{vect}^2}{2^{|R|}} + \frac{q_{F_1} + q_{F_1^*}}{|Mac_S|} + \text{Adv}(E)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

where $q_{req} = 6q_{vect} + 2q_{F_1} + 2q_{F_1^*}$.

We here only give a sketch of proof and the full version is in Appendix 8.

Proof (sketch). The first step is to replace structural elements E and KD with a perfect ones. Here, the number of related queries is $q_{req} = 6q_{vect} + 2q_{F_1} + 2q_{F_1^*}$. Our second step is to reduce the game to one client. With all client has a different key and the block cipher is now considered perfect, the clients act independently. A third one step is to forbid the use of verification functions. We show that for every R different than the initial elements, an attacker cannot a positive response from the verification function with probability more than $\frac{24q_{vect}(q_{F_1}+q_{F_1^*})}{2^{|R|}} + \frac{q_{F_1}+q_{F_1^*}}{|MacS|}$. The final step is to distinguish the initial elements from random. Here we can consider the initial elements as a superposition of classical states on which the security is determined by the absence of internal collision, which happens with probability lower than $\frac{50q_{vect}^2}{2^{|R|}}$, which concludes the proof. \square

Note on AES. AES has been studied in the classical case for over 20 years. While the same cannot be said for the quantum case, the work of Bonnetain et al. [13] seems convincing that AES can be considered quantum-safe for now.

Security of the Key Derivation. As stated in 3.4, the model for evaluating the key derivation is a known-plaintext model on AES. It can be considered safe.

Note on the advantage on E . The use of fixed superposition may mislead us to think only the classical security of E is needed. However, an example for a quantum attack happening is the one-time pad distinguisher from [9] (Proposition 7.1). This distinguisher happens with only one query on $H|0\rangle$ against a one-time pad (where a classical distinguisher would need at least two queries) as G is a kind of one-time pad when E is a one-time pad.

6 Quantum Security of TUAKE

Description of TUAKE. In this section we discuss the security of TUAKE, which is based on a permutation noted P (in practice, it is Keccak-f[1600] [3]). The following description concerns the highest size of elements (keys and MACs are 256-bit long). The lower sizes only have some tweaks on the *INSTANCE* value and only output a subset of their higher size counterparts. We note $rev(M)$ the message M reversed, $a * M$ to be the message M repeated a times and *ALGONAME* to be the 56-bit value of the ASCII representation of “TUAKE1.0”.

Key derivation. *INSTANCE* is set to 00000001,
 IN is $rev(sk_{Op})\|rev(INSTANCE)\|rev(ALGONAME)\|192 * (0)\|$
 $rev(sk_C)\|5 * (1)\|314 * (0)\|(1)\|512 * (0)$,
 $sk_{Op,C}$ is defined as the reverse of the first reversed 256 bits of $P(IN)$.

Functions $f_1, f_2, f_3, f_4, f_5, f_1^, f_5^*$.*

For f_1 , *INSTANCE* is 00100001,
 IN is $rev(sk_{Op})\|rev(INSTANCE)\|rev(ALGONAME)\|rev(R)\|rev(AMF)\|$

$rev(Sqn)\|rev(sk_C)\|5 * (1)\|314 * (0)\|(1)\|512 * (0)$,

f_1 outputs the reverse of the first reversed 256 bits of $P(IN)$.

For f_1^* , $INSTANCE$ is 10100001,

IN is $rev(sk_{Op})\|rev(INSTANCE)\|rev(ALGONAME)\|rev(R)\|rev(AMF)\|rev(Sqn)\|rev(sk_C)\|5 * (1)\|314 * (0)\|(1)\|512 * (0)$,

f_1^* outputs the reverse of the first reversed 256 bits of $P(IN)$.

For f_2, f_3, f_4, f_5 , $INSTANCE$ is set to 01100111,

IN is $rev(sk_{Op})\|rev(INSTANCE)\|rev(ALGONAME)\|rev(R)\|64 * (0)\|rev(sk_C)\|5 * (1)\|314 * (0)\|(1)\|512 * (0)$,

f_2 outputs the reverse of the first 256 bits of $P(IN)$,

f_3 outputs the reverse of the bits 256 to 511 of $P(IN)$,

f_4 outputs the reverse of the bits 512 to 767 of $P(IN)$,

f_5 outputs the reverse of the bits 768 to 815 of $P(IN)$.

For f_5^* , $INSTANCE$ is set to 11000001,

IN is $rev(sk_{Op})\|rev(INSTANCE)\|rev(ALGONAME)\|rev(R)\|64 * (0)\|rev(sk_C)\|5 * (1)\|314 * (0)\|(1)\|512 * (0)$,

f_5^* outputs the reverse of the bits 768 to 815 of $P(IN)$.

Security Arguments of TUAK. From a security perspective, we look at P as a pseudo-random function taking on input the key $sk_{Op,C}$ on position 0 to 255 and the key sk_C on position 512 to 767. Any attack on P would be considered as an attack against the permutation.

With this mindset, we can obviously reduce the security of TUAK to the one of P with at most $4q$ queries.

Note on Keccak-f. We now have to give some evidence that the underlying permutation, namely Keccak-f, can be considered as quantum-secure. As Keccak is a permutation on 1600 bits, it means that the best general attack as a qPRF is to try and fail at finding a collision which takes at least 2^{533} computations to do [40] compared to the 2^{256} computations to brute-force the keys. Keccak-f has been studied in the classical case since the NIST SHA-3 competition in 2008. While the same cannot be said for the quantum case, the fact that no practical classical attack breaks more than 8 rounds out of 24 [24, 42] (the best classical attack on the full permutation to our knowledge uses 2^{1573} computations [36]) seems convincing that Keccak-f can be considered quantum-safe for now.

7 Conclusion

The UMTS-AKA protocol was designed in 1999 and intended to a classical world. With the advent of quantum computers, that multiple experts consider as imminent, it is necessary to take a new look at such protocols deployed in real life, which are used to protect the communication of billions of users. Assessing their security anew is therefore implied by the coming quantum era.

In this paper we have focused our attention on the UMTS authenticated key agreement, designed for the 3G telecommunication technology, and still at the

basis of both 4G and 5G standards. This protocol is used every day by numerous users to protect their voice and data mobile communications.

We have defined a quantum version of the genuine AKA protocol. Starting from the work of Alt et al. we have derived a stronger quantum model which grants the adversary quantum computations as well as superposition queries. Then we have provided detailed security proofs of the quantum UMTS-AKA, showing that the quantum security of the protocol relies upon that of the underlying pseudo-random functions (f_1, \dots, f_5^*) . Therefore, under the assumption that they are quantum-secure, the UMTA-AKA remains a secure scheme to protect users' communications. To the best of our knowledge this paper provides the first rigorous proof of the UMTS-AKA in a strong quantum setting.

As supplementary contributions, we also exhibit a new attack against the state confidentiality of a mobile user. This attack holds in the quantum as well as in the classical setting. We also describe a quantum existential forgery attack against the standalone f_1 function when instantiated with Milenage. We analyzed the quantum security of the underlying primitives of UMTS-AKA, Milenage and TUAK, and conclude that, if keyed with a 256-bit key, they are quantum secure as core functions of the UMTS-AKA protocol.

As a next work we aim at studying in a quantum setting the new properties of user's privacy and network's "awareness" that are provided in the 5G technology. With in mind the threat posed by quantum computing, the goal of the NIST's post-quantum competition is to motivate the design of quantum-secure asymmetric schemes. Likewise we hope that our work will contribute to the systematic analysis of the currently deployed protocols with regard to the quantum threat they will soon face, and help design secure quantum networks.

References

1. 3GPP: 3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions $f_1, f_1^*, f_2, f_3, f_4, f_5$ and f_5^* ; Document 2: Algorithm specification (1999), rev. 16.0.0
2. 3GPP: 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security architecture (1999), rev. 16.0.0
3. 3GPP: Specification of the TUAK algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions $f_1, f_1^*, f_2, f_3, f_4, f_5$ and f_5^* ; Document 1: Algorithm specification (2015), rev. 16.0.0
4. Alagic, G., Broadbent, A., Fefferman, B., Gagliardini, T., Schaffner, C., Jules, M.S.: Computational security of quantum encryption. In: Information Theoretic Security – 9th International Conference, ICITS 2016
5. Alt, S., Fouque, P., Macario-Rat, G., Onete, C., Richard, B.: A cryptographic analysis of UMTS/LTE AKA. In: Applied Cryptography and Network Security – 14th International Conference, ACNS 2016
6. Anand, M.V., Targhi, E.E., Tabia, G.N., Unruh, D.: Post-quantum security of the cbc, cfb, ofb, ctr, and XTS modes of operation. In: Post-Quantum Cryptography – 7th International Workshop, PQCrypto 2016
7. Bhaumik, R., Bonnetain, X., Chailloux, A., Leurent, G., Naya-Plasencia, M., Schrottenloher, A., Seurin, Y.: QCB: efficient quantum-secure authenticated encryption. IACR Cryptol. ePrint Arch. (2020)

8. Bonnetain, X.: Quantum key-recovery on full AEZ. In: Selected Areas in Cryptography – SAC 2017 – 24th International Conference, 2017
9. Bonnetain, X.: Hidden Structures and Quantum Cryptanalysis. (Structures cachées et cryptanalyse quantique). Ph.D. thesis, Sorbonne University, France (2019)
10. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: The offline simon’s algorithm. In: Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I
11. Bonnetain, X., Leurent, G., Naya-Plasencia, M., Schrottenloher, A.: Quantum linearization attacks. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I
12. Bonnetain, X., Naya-Plasencia, M.: Hidden shift quantum cryptanalysis and implications. In: Advances in Cryptology – ASIACRYPT 2018 – 24th International Conference on the Theory and Application of Cryptology and Information Security, 2018
13. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. **2019**
14. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics (2002)
15. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Advances in Cryptology – ASIACRYPT 2017 – 23rd International Conference on the Theory and Applications of Cryptology and Information Security, 2017
16. Cleve, R., Buhrman, H.: Substituting quantum entanglement for communication. Physical Review A
17. Czakajkowski, J., Bruinderink, L.G., Hülsing, A., Schaffner, C., Unruh, D.: Post-quantum security of the sponge construction. In: Post-Quantum Cryptography – 9th International Conference, PQCrypto 2018
18. Czakajkowski, J., Bruinderink, L.G., Hülsing, A., Schaffner, C., Unruh, D.: Post-quantum security of the sponge construction. IACR Cryptol. ePrint Arch. (2017)
19. Czakajkowski, J., Hülsing, A., Schaffner, C.: Quantum indistinguishability of random sponges. In: Advances in Cryptology – CRYPTO 2019 – 39th Annual International Cryptology Conference, 2019
20. Ebrahimi, E., Chevalier, C., Kaplan, M., Minelli, M.: Superposition attack on OT protocols. IACR Cryptol. ePrint Arch. **2020**
21. Ghosh, S., Kate, A.: Post-quantum forward-secure onion routing – (future anonymity in today’s budget). In: Applied Cryptography and Network Security – 13th International Conference, ACNS 2015
22. Hosoyamada, A., Iwata, T.: On tight quantum security of HMAC and NMAC in the quantum random oracle model. In: Advances in Cryptology – CRYPTO 2021 – 41st Annual International Cryptology Conference, CRYPTO 2021
23. Hosoyamada, A., Sasaki, Y.: Quantum collision attacks on reduced SHA-256 and SHA-512. In: Advances in Cryptology – CRYPTO 2021 – 41st Annual International Cryptology Conference, CRYPTO 2021
24. Jean, J., Nikolic, I.: Internal differential boomerangs: Practical analysis of the round-reduced keccak- f f permutation. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers

25. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Advances in Cryptology – CRYPTO 2016 – 36th Annual International Cryptology Conference, 2016
26. Knip, Q.M., Müller, W., Redlich, J.: Post-quantum cryptography in wireguard VPN. In: Security and Privacy in Communication Networks – 16th EAI International Conference, SecureComm 2020
27. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: IEEE International Symposium on Information Theory, ISIT 2010
28. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012
29. Leander, G., May, A.: Grover meets simon - quantumly attacking the fx-construction. In: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II
30. Mitchell, C.J.: The impact of quantum computing on real-world security: A 5g case study. *Comput. Secur.*
31. Music, L., Chevalier, C., Kashefi, E.: Dispelling myths on superposition attacks: Formal security model and attack analyses. In: Provable and Practical Security – 14th International Conference, ProvSec 2020
32. Paul, S., Scheible, P.: Towards post-quantum security for cyber-physical systems: Integrating PQC into industrial M2M communication. In: Computer Security – ESORICS 2020 – 25th European Symposium on Research in Computer Security
33. Schanck, J.M., Whyte, W., Zhang, Z.: Circuit-extension handshakes for tor achieving forward secrecy in a quantum world. *Proc. Priv. Enhancing Technol.* **2016**
34. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security (2020)
35. Song, F., Yun, A.: Quantum security of NMAC and related constructions – PRF domain extension against quantum attacks. In: Advances in Cryptology – CRYPTO 2017 – 37th Annual International Cryptology Conference, 2017
36. Yan, H., Lai, X., Wang, L., Yu, Y., Xing, Y.: New zero-sum distinguishers on full 24-round keccak-f using the division property. *IET Inf. Secur.*
37. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science, 1986
38. Yao, A.C.: Quantum circuit complexity. In: 34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993
39. Zhandry, M.: How to construct quantum random functions. *J. ACM*
40. Zhandry, M.: A note on the quantum collision and set equality problems. *Quantum Inf. Comput.*
41. Zhang, M.: Provably-secure enhancement on 3gpp authentication and key agreement protocol. *IACR Cryptol. ePrint Arch.* **2003**
42. Zhao, Z., Chen, S., Wang, M., Wang, W.: Improved cube-attack-like cryptanalysis of reduced-round ketje-jr and keccak-mac. *Inf. Process. Lett.*

A Full Description of AKA

Generation of the challenge. First the server requests from the client an *UID*, which is an IMSI (International Mobile Subscriber Identify) or a TMSI (Temporary Mobile Subscriber Identity), a value agreed in a previous session. The

server forwards it to the operator. The operator produces n Authentication Vectors (AV), sends them to the server, and updates $Sqn_{Op,C}$ to $Sqn_{Op,C} + n$. Each vector AV^i , $1 \leq i \leq n$, is computed from a fresh random value R^i as follows:

$$\begin{aligned}
Sqn^i &= Sqn_{Op,C} + i \\
Mac_S^i &= f_1(sk_s, R^i, Sqn^i, AMF) \\
Mac_C^i &= f_2(sk_s, R^i) \\
CK^i &= f_3(sk_s, R^i) \\
IK^i &= f_4(sk_s, R^i) \\
AK^i &= f_5(sk_s, R^i) \\
Auth^i &= Sqn^i \oplus AK^i || AMF || Mac_S^i \\
AV^i &= (R^i, CK^i, IK^i, Auth^i, Mac_C^i)
\end{aligned}$$

Challenge-Response. The server picks a vector AV^i and sends the corresponding R^i to the client. In turn, the latter computes

$$\begin{aligned}
AK^i &= f_5(sk_s, R^i) \\
Sqn^i &= AK^i \oplus Auth_1^i \\
Mac_S^i &= f_1(sk_s, R^i, Sqn^i, AMF) \\
Mac_C^i &= f_2(sk_s, R^i) \\
CK^i &= f_3(sk_s, R^i) \\
IK^i &= f_4(sk_s, R^i)
\end{aligned}$$

The client verifies that $Mac_S^i = Mac_S^i$, sends an abort message if not, and checks if the challenge is fresh, i.e., $Sqn^i \in [Sqn_C + 1, Sqn_C + \Delta]$ (where Δ , chosen by the operator, defines an acceptance interval). Next it sets Sqn_C to Sqn^i , and replies to the server with Mac_C^i . If this answer is incorrect, the server sends to the operator a report containing the R^i of the used AV^i , the client's UID , and a failure code. In such a case the operator updates $Sqn_{Op,C}$ to Sqn^i .

Re-synchronisation. If $Sqn^i \notin [Sqn_C + 1, Sqn_C + \Delta]$ the client triggers a re-synchronization procedure (see Figure 1). First it computes

$$\begin{aligned}
AK^* &= f_5^*(sk_s, R^i) \\
Mac_S^* &= f_1^*(sk_s, R^i, Sqn_C, 0) \\
Auts &= (Sqn_C \oplus AK^*) || Mac_S^*
\end{aligned}$$

where R^i corresponds to the current challenge. The client sends $Auts$ to the server which forwards it along with R^i to the operator. Then the operator computes

$$\begin{aligned}
AK^* &= f_5^*(sk_s, R^i) \\
Sqn_C &= AK^* \oplus Auts_1 \\
Mac_S^* &= f_1^*(sk_s, R^i, Sqn_C, 0)
\end{aligned}$$

and verifies if $Mac_S^* = Mac_S^*$. If so, it updates $Sqn_{Op,C}$ to Sqn_C , otherwise it aborts the procedure.

B Detailed proofs

In this section we provide the detailed proofs of the different security properties defined in Section 3 for the quantum variant of the AKA protocol.

B.1 Proof of Indistinguishability

Proof.

GAME \mathbb{G}_0 : This is the game of the definition.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \text{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: An attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD . To put it in another way, one can use an attack that is successful against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' . More formally, using the best adversary \mathcal{A}_0 for \mathbb{G}_0 , we can build an adversary \mathcal{A}_{KD} for the game against KD .

\mathcal{A}_{KD} will take the given values as keys sk_C and $sk_{Op,C}$ to generate the clients and operators. Then it will run the adversary \mathcal{A}_0 with the generated clients and operators then if \mathcal{A}_0 is right, \mathcal{A}_{KD} will answer “the keys are generated” and if \mathcal{A}_0 is wrong, \mathcal{A}_{KD} will answer “the keys are random”.

In the case the tuple of keys is the generated one, \mathcal{A}_0 works normally. If it is the random one \mathcal{A}_0 will work as an adversary for the game \mathbb{G}_1 . The probability of the described events is reported as follows:

\mathcal{A}_0	right	wrong
keys generated	$\frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t})}{2}$
random	$\leq \frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\geq \frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$

$$\text{Adv}(KD)_{n_{Op}, n_C}^t \geq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

GAME \mathbb{G}_2 : We define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but we replace G with a perfect pseudo-random function G' on which there is no advantage.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: An attacker for \mathbb{G}_1 either attacks \mathbb{G}_2 or uses a property of G . To put it in another way, one can use an attack against \mathbb{G}_1 but fails against \mathbb{G}_2 to distinguish between G and G' . More formally, using the best adversary \mathcal{A}_1 for \mathbb{G}_1 , we can build a quantum adversary \mathcal{A}_G for the game against

G with a total of $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests¹⁰ for keyed G as a black box.

\mathcal{A}_G will take the black boxes to generate the clients and operators. Then it will run the adversaries \mathcal{A}_1 with the generated clients and operators then if \mathcal{A}_1 is right, \mathcal{A}_G will answer “the functions are generated” and if \mathcal{A}_1 is wrong, \mathcal{A}_G will answer “the functions are random”.

In the case the functions are the generated one, \mathcal{A}_1 works normally. If they are the random one \mathcal{A}_1 will work as an adversary for the game \mathbb{G}_2 . The probability of the described events is reported as follows:

	\mathcal{A}_1		
		right	wrong
functions	generated	$\frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t})}{2}$
	random	$\leq \frac{\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})}{2}$	$\geq \frac{1 - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})}{2}$

$$\text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} \geq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) - \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client, one server and one operator.

GAME \mathbb{G}_3 : We define the game \mathbb{G}_3 as the same as \mathbb{G}_2 but there is only one client and one operator.

TRANSITION $\mathbb{G}_2 \rightarrow \mathbb{G}_3$: As G' and KD' are perfect, the keyed G' are uncorrelated. Thus the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_3 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$ or $\mathbb{G}_1 \rightarrow \mathbb{G}_2$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, q_{Op}, t})$$

GAME \mathbb{G}_4 : We define the game \mathbb{G}_4 as the same as \mathbb{G}_3 but there is only one server now.

TRANSITION $\mathbb{G}_3 \rightarrow \mathbb{G}_4$: As servers are interchangeable in the sense that the values exchanged do not depend on the server, we essentially make the attacker unable to corrupt servers. Since corruption of servers only gives access to authentication vectors which would be given during an execution and reveal, the queries from a corrupt server are now counted as additional executions of the protocol: $q'_{exec} = q_{exec} + q_{Op}$.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t})$$

¹⁰ This is the number of queries needed to simulate the games. During an execution, the operator and the client each computes f_1, f_2, f_3, f_4 and f_5 . For a re-synchronisation, the client and the operator each computes f_1^* and f_5^* . For a corrupted request, the operator computes f_1, f_2, f_3, f_4 and f_5 .

GAME \mathbb{G}_5 : We define the game \mathbb{G}_5 as the same as \mathbb{G}_4 but the challenger now generates the R^i before the game and measures the property of R^i being all different.

TRANSITION $\mathbb{G}_4 \rightarrow \mathbb{G}_5$: With our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q_{exec}^2}{2^{|R|}}$.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t}) &= \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t} \text{ and some } R^i \text{ are equal}) \\ &\quad + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t} \text{ and the } R^i \text{ are all different}) \end{aligned}$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, t}) \leq \frac{(q'_{exec})^2}{2^{|R|}} + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_5^{q'_{exec}, q_{res}, t})$$

ENDING: The attacker has access to a superposition of authentication vectors such that none of the states contained in the superposition holds any significant information. It has to distinguish between values generated by a perfect pseudo-random function from different R^i and true random values. To attack a client, the attacker can also try to send a different challenge which means guessing the output Mac_S of a perfect pseudo-random function with a value Sqn_C never seen before.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_5^{q'_{exec}, q_{res}, t}) \leq \frac{1}{2} + \frac{1}{2^{Mac_S}}$$

Then by recalling the previous transitions, we get:

$$\text{Adv}(\mathcal{A}) \leq \frac{1}{2^{Mac_S}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t + \frac{(q_{exec} + q_{Op})^2}{2^{|R|}}$$

□

B.2 Proof of Client-impersonation resistance

Proof.

GAME \mathbb{G}_0 : This is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \text{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: An attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section B.1 with $q_{req} =$

$10q_{exec} + 5q_{Op} + 4q_{res}$ requests¹¹.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \text{Adv}(KD)_{n_{Op}, n_C}^t + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client, one server and one operator.

GAME \mathbb{G}_2 : We define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: As G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

GAME \mathbb{G}_3 : We define the game \mathbb{G}_3 as the same as \mathbb{G}_2 but there is only one server now.

TRANSITION $\mathbb{G}_2 \rightarrow \mathbb{G}_3$: As servers are interchangeable in the sense that the values exchanged do not depend on the server, we essentially make the attacker unable to corrupt servers. Since corruption of servers only gives access to authentication vectors which would be given during an execution and reveal, the queries from a corrupt server are now counted as additional executions of the protocol: $q'_{exec} = q_{exec} + q_{Op}$. For the next games, those additional executions will be marked as not valid for winning the game, $q_{valid} = q_{exec}$ is the number of valid executions for winning. The first executions are the marked ones and the valid ones are last, doing so does not remove generality.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t})$$

GAME \mathbb{G}_4 : We define the game \mathbb{G}_4 as the same as \mathbb{G}_3 but the challenger now generates the R^i before the game and measures the property of R^i being all different.

TRANSITION $\mathbb{G}_3 \rightarrow \mathbb{G}_4$: With our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q'_{exec}}{2|R|}$.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t} \text{ and some } R^i \text{ are equal}) + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t} \text{ and the } R^i \text{ are all different})$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q'_{exec}, q_{res}, q_{valid}, t}) \leq \frac{(q'_{exec})^2}{2|R|} + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q'_{exec}, q_{res}, q_{valid}, t})$$

¹¹ This is the number of queries needed to simulate the games. During an execution, the operator and the client each computes f_1, f_2, f_3, f_4 and f_5 . For a re-synchronisation, the client and the operator each computes f_1^* and f_5^* . For a corrupted request, the operator computes f_1, f_2, f_3, f_4 and f_5 .

ENDING: The attacker has access to a superposition of authentication vectors such that none of the states contained in the superposition holds any significant information. Since the uncorrupted server needs the first and the last message in that order to be able to accept the execution, the attacker cannot ask the answer to the client or it will be a simple relay and then has to guess values Mac_C generated by a perfect pseudo-random function with a R^i different from what he has seen.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, q_{valid}, t}) = \frac{q_{valid}}{2^{|Mac_C|}}$$

Then by recalling the previous transitions, we get:

$$\text{Adv}(\mathcal{A}) \leq \frac{(q_{exec} + q_{Op})^2}{2^{|R|}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t + \frac{q_{exec}}{2^{|Mac_C|}}$$

□

B.3 Proof of Server-impersonation resistance

Proof.

GAME \mathbb{G}_0 : This is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, t}) = \text{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: An attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section B.1 with $q_{req} = 10q_{exec} + 4q_{res}$ requests ¹².

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \text{Adv}(KD)_{n_{Op}, n_C}^t + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client, one server and one operator.

GAME \mathbb{G}_2 : We define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: As G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients

¹² This is the number of queries needed to simulate the games. During an execution, the operator and the client each computes f_1, f_2, f_3, f_4 and f_5 . For a re-synchronisation, the client and the operator each computes f_1^* and f_5^* . There is no server corruption (i.e. $q_{Op} = 0$).

or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

GAME \mathbb{G}_3 : We define the game \mathbb{G}_3 as the same as \mathbb{G}_2 but the challenger now generates the R^i before the game and measures the property of R^i being all different.

TRANSITION $\mathbb{G}_2 \rightarrow \mathbb{G}_3$: With our specifications on the generation of R^i , the probability of not measuring this property is about $\frac{q_{exec}}{2^{|R|}}$.

$$\begin{aligned} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t}) &= \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t} \text{ and some } R^i \text{ are equal}) \\ &\quad + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t} \text{ and the } R^i \text{ are all different}) \end{aligned}$$

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t}) \leq \frac{(q_{exec})^2}{2^{|R|}} + \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, t})$$

GAME \mathbb{G}_4 : We define the game \mathbb{G}_4 as the same as \mathbb{G}_3 but there is only one server.

TRANSITION $\mathbb{G}_3 \rightarrow \mathbb{G}_4$: As servers are incorruptible, they can only partner with a client that shares the same R they got from an operator. Then, as the R are all different, an unintended server will not partner.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{exec}, q_{res}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, t})$$

ENDING: The attacker has access to a superposition of authentication vectors such that none of the states contained in the superposition holds any significant information. Since the uncorrupted client needs the second message to accept, the attacker cannot ask the answer to the server or it will be a simple relay and then has to guess values Mac_S generated by a perfect pseudo-random function with a Sqn_C different from what he has seen.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_4^{q_{exec}, q_{res}, q_G}) = \frac{q_{exec}}{2^{|Mac_S|}}$$

Then by recalling the previous transitions, we get:

$$\text{Adv}(\mathcal{A}) \leq \frac{q_{exec}}{2^{|Mac_S|}} + \frac{q_{exec}^2}{2^{|R|}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

□

B.4 Proof of Soundness

Proof.

GAME \mathbb{G}_0 : This is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \text{Adv}(\mathcal{A})$$

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: An attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section B.1 with $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests¹³.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \text{Adv}(KD)_{n_{Op}, n_C}^t + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client and one operator.

GAME \mathbb{G}_2 : We define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: As G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

ENDING: Then the attacker has to guess values Mac_S generated by a perfect pseudo-random function with a Sqn_C different from what he has seen.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) \leq \frac{q_{exec} + q_{Op}}{2^{|Mac_S|}}$$

Then by recalling the previous transitions, we get:

$$\text{Adv}(\mathcal{A}) \leq \frac{q_{exec} + q_{Op}}{2^{|Mac_S|}} + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

□

B.5 Proof of State-confidentiality

Proof.

GAME \mathbb{G}_0 : This is the game defined just above.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) = \text{Adv}(\mathcal{A})$$

¹³ This is the number of queries needed to simulate the games. During an execution, the operator and the client each computes f_1, f_2, f_3, f_4 and f_5 . For a re-synchronisation, the client and the operator each computes f_1^* and f_5^* . For a corrupted request, the operator computes f_1, f_2, f_3, f_4 and f_5 .

Our first step is to replace structural elements with perfect ones.

GAME \mathbb{G}_1 : We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 but we replace KD with a perfect key derivation KD' and G with a perfect pseudo-function G' on which there is no advantage.

TRANSITION $\mathbb{G}_0 \rightarrow \mathbb{G}_1$: An attacker for \mathbb{G}_0 either attacks \mathbb{G}_1 or uses a property of KD or G . To put it in another way, one can use an attack against \mathbb{G}_0 but fails against \mathbb{G}_1 to distinguish between KD and KD' or between G and G' . One can easily use the same argument used in Section B.1 with $q_{req} = 10q_{exec} + 5q_{Op} + 4q_{res}$ requests¹⁴.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{exec}, q_{res}, q_{Op}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) + \text{Adv}(KD)_{n_{Op}, n_C}^t + \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t}$$

Our next step is to reduce the configuration to one client and one operator.

GAME \mathbb{G}_2 : We define the game \mathbb{G}_2 as the same as \mathbb{G}_1 but there is only one client and one operator.

TRANSITION $\mathbb{G}_1 \rightarrow \mathbb{G}_2$: As G' and KD' are perfect, the keyed G' are uncorrelated. Therefore the attacker does not have benefits from having other clients or operators. An attacker for \mathbb{G}_2 could simply simulate other uncorrupted clients by reusing the target client and corrupted clients by generating random keys for an attacker in \mathbb{G}_2 . As in $\mathbb{G}_0 \rightarrow \mathbb{G}_1$, a difference would mean an advantage over KD' or G' .

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{exec}, q_{res}, q_{Op}, t}) = \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t})$$

ENDING: The attacker can only win by sending the correct value for $sk_C, sk_{Op, C}$ or sk_{Op} or getting a right couple $(n, Sqn_{C, n})$.

For obtaining the secret keys, since G' and KD' are perfect, the attacker can only guess them.

Since the attacker is the server, it controls and knows the differences $Sqn_{C, n+1} - Sqn_{C, n}$ of the value Sqn_C at times n and n' (which by design is between 1 and Δ).

From executions and resynchronisations, the attacker only gets $Sqn_{Op, C, n} \oplus AK^n$ and $Sqn_{C, n} \oplus AK^{*n}$. Since G' and KD' are perfect, the attacker can only learn $Sqn_{C, n} \oplus Sqn_{C, n'}$ when R^n is the same as $R^{n'}$ (which can happen since the attacker can use an old authentication vector and get the result from the resynchronisation procedure).

The attacker then can learn the value of all the bits of $Sqn_{C, n}$ that changes through the game. For example, $Sqn_C \oplus (Sqn_C + 1)$ is of the form $0 \dots 01 \dots 1$ where the number of 1 is one more than the 2-adic valuation of $Sqn_C + 1$ (the details of this property change with the value of the difference between the successive values but the principle remains the same). As we expect around

¹⁴ This is the number of queries needed to simulate the games. During an execution, the operator and the client each computes f_1, f_2, f_3, f_4 and f_5 . For a re-synchronisation, the client and the operator each computes f_1^* and f_5^* . For a corrupted request, the operator computes f_1, f_2, f_3, f_4 and f_5 .

$\log(\Delta q_{exec}) + 3$ bits to be affected, the probability for the attacker of guessing a good couple (n, Sqn_C) is at most $\frac{8\Delta q_{exec}}{2^{|Sqn_C|}}$.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_2^{q_{exec}, q_{res}, q_{Op}, t}) \leq \frac{8\Delta q_{exec}}{2^{|Sqn_C|}} + \frac{1}{2^{|sk_C|}} + \frac{1}{2^{|sk_{Op,C}|}} + \frac{1}{2^{|sk_{Op}|}}$$

Then by recalling the previous transitions, we get:

$$\text{Adv}(\mathcal{A}) \leq \text{Adv}(G)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t + \frac{8\Delta q_{exec}}{2^{|Sqn_C|}} + \frac{1}{2^{|sk_C|}} + \frac{1}{2^{|sk_{Op,C}|}} + \frac{1}{2^{|sk_{Op}|}}$$

□

B.6 Proof of Theorem 8

Use of E. We define the game \mathbb{G}_1 as the same as \mathbb{G}_0 except in Milenage we use a perfect quantum pseudo-random permutation E' instead of E and a perfect key derivation KD' instead of KD .

Lemma 1

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_0^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) \leq \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_1^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) + \text{Adv}(E)_{n_{Op}, n_C}^{q_{req}, t} + \text{Adv}(KD)_{n_{Op}, n_C}^t$$

The attacker either uses a property of E with at most $q_{req} = 6q_{vect} + 2q_{F_1} + 2q_{F_1^*}$ queries or it attacks \mathbb{G}_1 .

Use of multiple clients. As we now deal with a perfect block cipher, the different elements obtained through the different clients are unrelated. The total advantage is the sum of the advantages against each client. We show that for one client, the advantage is less than $\frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|R|}} + \frac{50q_{vect}^2}{2^{|R|}} + \frac{q_{F_1} + q_{F_1^*}}{|MacS|}$. The resources of the attacker (access to authentication vectors, calls to F_1 and F_1^*) are spread between the clients ($q_{vect} = \sum_{clients} q_{vect, client}$, $q_{F_1} = \sum_{clients} q_{F_1, client}$ and, $q_{F_1^*} = \sum_{clients} q_{F_1^*, client}$).

Then the total advantage is less than $\sum_{clients} \frac{24q_{vect, client}(q_{F_1, client} + q_{F_1^*, client})}{2^{|R|}} + \frac{50q_{vect, client}^2}{2^{|R|}} + \frac{q_{F_1, client} + q_{F_1^*, client}}{|MacS|}$, which is still less than $\frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|R|}} + \frac{50q_{vect}^2}{2^{|R|}} + \frac{q_{F_1} + q_{F_1^*}}{|MacS|}$. We only consider the case of one client for the rest of the proof.

Use of verification functions. We define an intermediate game $\mathbb{G}_{2, R', Sqn, AMF}$ as the following:

1. The challenger generates random independent keys in pure states $sk_{C,1}, \dots, sk_{C,n_C}$ and $sk_{Op,C,1,1}, \dots, sk_{Op,C,n_C,n_{Op}}$ for the studied pseudo-random function $out_1, out_2, out_3, out_4, out_5$ and $5n_{Op}n_C$ random functions. The challenger will use the pseudo-random ones as a quantum black box $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}$ such that $O_{f_i, sk_{C,j}, sk_{Op,C,j,j'}}(|x\rangle|y\rangle) = |x\rangle|y \oplus out_{i, sk_{C,j}, sk_{Op,C,j,j'}}(x)\rangle$.

2. The challenger now gives the attacker authentication vectors and the associated answers (i.e. elements $\sum_R a_R |R, out_1, out_2, out_3, out_4, out_5\rangle$) and access to black-box verification functions F_1 and F_1^* that take quantum superpositions of states $|x, y\rangle$ and measures whether $y = f_1(x, Sqn, AMF)$ and $y = f_1^*(x, Sqn, AMF)$ respectively and gives the result on a classical bit.
3. The challenger gives the attacker a value $\sum_{R',h} a_{R',h} |R', h\rangle$ such that R' is measured to never equal to a previous R^i .
4. The attacker is free to interact with those elements.
5. The attacker guesses the value of any part of $G(R', Sqn, AMF)$ ($f_1, f_1^*, out_2, out_3, out_4, out_5$) and wins if the guess is right.

Lemma 2 For every R' ,

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) \leq \frac{24q_{vect}(q_{F_1} + q_{F_1^*})}{2^{|R|}} + \frac{q_{F_1} + q_{F_1^*}}{|Mac_S|}$$

Proof. As we want to consider all possible R' , the attacker uses F_1 or F_1^* either on an R' or on a previous R^i . However, the attacker gains nothing by querying F_1 or F_1^* on a previous R^i (the result is already known). Then we consider the attacker to only query F_1 or F_1^* on a R' . Then on an R' , either the result is “true” and we can declare the attacker to win, or the result is false and it continues. Either way, the measurement can be done at the end of the game with the final guess. Then having q_{F_1} access to F_1 and $q_{F_1^*}$ access to F_1^* only multiply the probability of success.

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, q_{F_1}, q_{F_1^*}, t}) \leq q_{F_1} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, 1, 0, t}) + q_{F_1^*} \mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_{2,R'}^{q_{vect}, 0, 1, t})$$

There are two cases, either one of the internal value of the computation matches one that appeared for one of the previous R^i , which happen with probability less than $\frac{12q_{vect}}{2^{|R|}}$,¹⁵ or there is no collision and the attacker has to guess a random value such that there is no internal collision between R' and any R^i .

Note that there is no gain to take more R' or a smaller part. In the first case, the attacker would have to spread the calls to F_1 or F_1^* between the different values R' to attack. In the second case, F_1 and F_1^* would only give a partial answer and the advantage of the attacker compared to a perfect G would not grow.

Randomness of the initial vectors. We showed that vectors outside the initial ones cannot be used efficiently. We now focus on the initial vectors. The verification function are trivial on those ones, and we can then dispose of them. We define the game \mathbb{G}_3 as distinguishing the authentication vectors given to the attacker from vectors generated by a perfect pseudo-random function.

Lemma 3

$$\mathbb{P}(\mathcal{A} \text{ wins } \mathbb{G}_3^{q_{vect}, t}) \leq \frac{1}{2} + \frac{50q_{vect}^2}{2^{|R|}}$$

¹⁵ There is two values for R' as only Out_1 is computed

Proof. There are two cases, either one of the internal value of the computation matches one that appeared for one of the previous R^i or there is no collision. The first one happens with probability less than $\frac{25q_{vect}^2}{2^{|R|}}$. The second one means the attacker has only the fact that there is no internal collision between the computations of $out_1, out_2, out_3, out_4, out_5$ (which gives the same advantage).

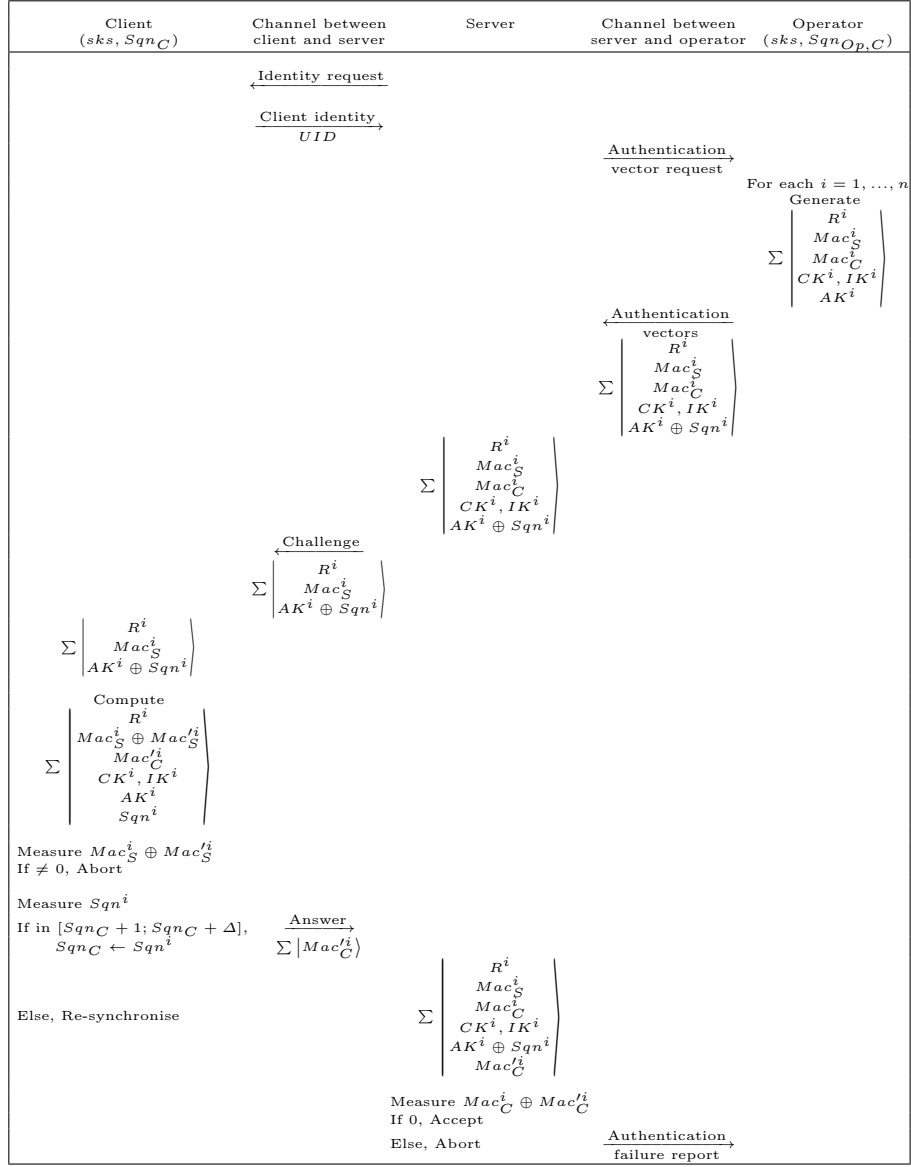


Fig. 2. Quantum version of the AKA protocol