Q1. What is the exact attack scenario that is considered? In particular:

a) what are the exact capabilities of the relevant adversaries in terms of time complexity, number of queries, types of allowed queries?

b) are the various building blocks only considered in the context of the protocol, or is their security to be considered individually, as implied by criterion A of Document 4, Section 9.1?

A: a) The attacks considered are stated in the form of attack-resistance requirements in clause 6.1 of Documents 1 and 4. When the design started, *key recovery* attacks were required to withstand active (chosen input) attacks, see Document 1, clause 6.1, item 2. For attacks attempting to *distinguish* the overall Milenage *f*-set from random functions, there was initially no firm requirement on whether to consider active attacks or not since the previous Milenage-128 did not require this either (obviously, passive attacks must be infeasible). However, when the design was finished, a formal security proof covering also active attack (*adaptively* chosen inputs) was produced for the overall framework (see ref [51] of Document 4). Note that this proof also covers the *f1* MAC (which the old proof for Milenage-128 omitted).

The proof is dependent of the strength of the kernel (MDPH-AES-256 and Rijndael-256-256) as follows. First, the proof establishes that if the kernel is "perfectly random", then any (active) attack on the complete Milenage f-set using up to q chosen inputs, will not enable the attacker to distinguish Milenage-256 from a completely random function, except with an advantage bounded by roughly

$$\sim q^2\, 2^{-256}. \qquad \text{(Eq 1)}$$

When one then replaces the perfect kernel by either the AES-MHDP construct (Milenage256-A) or Rijndael-256-256 (Milenage256-R), this bound will increase slightly. For Milenage256-R, it will increase by ε_R, which expresses how well Rijndael-256-256 mimics a perfectly random function. For Milenage256-A it will similarly increase by ε_A, which now expresses how well the AES-based MDPH mimics a perfectly random function. However, the situation is more complex since, additionally, ε_A also depends on ε_{AES}, expressing how well AES-128-256 mimics a perfect *128-bit* random function.

Neither ε_R nor ε_{AES} are of course known, but there is good confidence they are very small. In any case, even if they are zero, there is no way to prove a security better than that stated by (Eq 1), implying that attack resistance cannot be guaranteed beyond $q \sim 2^{128}$ queries. This is in fact an *optimal* result for any construction using a *permutation* as the kernel.

Further, for Milenage-256A, the additional need to invoke the MDPH-construct makes it impossible to guarantee security beyond $q \sim 2^{121}$. This is due to the fact that even if one makes an assumption that AES-128-256 is "perfect" it cannot be excluded that there is a slight loss in security due the MDPH-construct itself.

Note also that the above points reiterate why the security assurance for Milenage-256R is both “cleaner” and likely stronger. Relative to the optimal bound (Eq 1), Mileange-256A experiences a reduction in security due to the fact that the MDPH-construct itself is not perfectly random *and* that AES-128-256, as used in the MDPH-construct, does not generate perfectly random outputs. Mileange-256R only experiences a reduction due to the imperfect randomness of Rijndael-256-256.

b) The goal has been to make the overall Milenage-256 *f*-set secure. Note that the security proof referenced above does establish this (up to the stated bound on q). The only source of insecurity that

cannot be ruled out (cryptographically speaking) is potential insecurity in AES-128-256 and/or Rijndael-256-256.

Q2: The level of security that should be achieved by the new constructions seems unclear. The key sizes of all constructions are increased to 256 bits, which also seems to be the targeted security level of TUAK. However, as acknowledged in Document 4, the respective security levels (as PRFs) of MDPH-AES-256 and Rijndael-256-256 are 121 and 128 bits. Why is there such a difference? As an example, xoring two instances of Rijndael-256-256 (using keys that could be derived from the same 256-bit key) would then give 256 bits of security.

A: The values 2^{121} and 2^{128} are bounds on attack complexity for *distinguishing attacks*. There is no reason to expect that the security against key recovery is much less than 256 bits. It could indeed be possible to increase also the distinguishing attack resistance to higher values, and SAGE did consider such constructions initially. The appropriate way to do this would be to construct a new kernel, replacing Rijndael-256-256 and MDPH-AES-256 by something which is a *provably* secure pseudo random function. SAGE looked at several such constructs but found none to be better than the final specification, some even had questionable properties. One of those constructs was indeed the XOR-construct (for which proofs do exist). However, SAGE could not see *any* motive to define an additional key derivation function and to also *double* the number of kernel applications[1] just to allow attackers making more than 2^{128} chosen queries. (Making that number of queries would require completely unrealistic timeframes and resources.)

Q3: Section 9.1 mentions the following:

> "Related key attacks (RKA) directly targeting the underlying Rijndael/AES block ciphers need not, and have not, been considered due to the MILENAGE operational context in which it is judged extremely difficult to mount such attacks in practice. It should however be noted that if an attacker is able to manipulate inputs to MILENAGE-256-A, it does influence the keys that are input to the AES-instances used inside the MDPH-construct."

It should be noted that, if an attacker is able to manipulate inputs to MILENAGE-256-A, it is actually capable of *choosing* half the bits of the keys to the underlying AES instances. Besides, due to the structure of the mode, each key will be queried on two related points. This allows mounting attacks with related input and partially chosen keys. Since AES-256 has important issues in the chosen-key setting [0,1], has the impact of such attacks been considered?

[0] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In CRYPTO 2009, Proceedings, volume 5677, pages 231–249.

[1] Xiaoyang Dong and Shun Li and Phuong Pham. Chosen-Key Distinguishing Attacks on Full AES-192, AES-256, Kiasu-BC, and More.

Cryptology ePrint Archive, Paper 2023/1095, https://eprint.iacr.org/2023/1095.

A: Firstly, SAGE would like to address this claim from your question:

>> "It is actually capable of *choosing* half the bits of the keys"

[1] The required 500msec time to produce the outputs would no longer be ensured.

By looking on Figure 1 below we cannot see how the IN values can indeed be chosen – these are the inputs to the key in MDPH. Relatedly, the outputs to the first kernel execution (TEMP value, in figure 1) are internal. We need more clarification from your side about this.

Secondly, differential attacks presented in the mentioned academic papers are not applicable to the case of MILENAGE-A for the following reasons.

1. Following the MILENAGE framework depicted in Figure 1 one can indeed insert input differentials to INx vectors of the second call to the PRF and observe OUTx from that. However, note that in practice not all bits of IN are possible to manipulate, and not all bits of OUT are available to an attacker. This greatly limits the landscape for mounting an attack scenario.
2. Assume now that an attacker can indeed insert the full 256-bit INx and observe the full 256-bit OUTx from MILENAGE256-A (this is of course not possible in practice since the attack model assumes an attacker can only input RAND values and observe the final OUTx values – but for sake of discussion, let's consider this case). Following the MDPH construction depicted on Figure 2, and utilized by MILENAGE256-A, we see that an attacker can manipulate only a half of the key bits, while the second half as well as the input blocks (initial AES states) values remain secret.
3. In the referred literature, an attacker performing a differential attack would need to be able to insert differentials in both the secret key and the plaintext (i.e., AES state), and then also be able to observe the differential from the whole output block. As we have seen from above, the only point of insertion of such a differential into an instance of AES-128-256 is the first half of the 256-bit key, while we still have 256 secret bits (128 bits of the state and 128 bits of the second half of the key are both unknown to an attacker). This situation can be compared to inserting a difference to a 128-bit plaintext having a 256-bit key, where the only difference is that it is the first half of the key that serves as the "plaintext" for the purpose of a one-way compression.
4. However:
    a. The optimal differentials presented in the literature involve bits of both halves of the key (and the plaintext in [1]) – see the differential trail from both papers. Most importantly, those efficient differentials that involve a small number of active SBoxes span over both halves of the key.
    b. In case of MDPH, the fact that there are strict limitations to a possible key differential, described above, forces the number of active SBoxes to increase by a lot.
    c. SAGE has not seen anywhere, nor could SAGE derive an efficient differential trail that use a small number of active SBoxes, following these strict differential constraints, that would lead to any possible attack, even theoretical.

There is even more confidence that the MDPH construct is secure. During our work on the new MILENAGE, SAGE considered many possible design variants. "Consideration" was not limited to studying literature only, SAGE also carried out real cryptanalysis and simulations, many of them.

This way, at one point AES-128-256 was studied as a tweakable function where the 128-bit tweak T is mixed with the 256-bit key in different ways. One of the mixing functions that was also studied looked as follows:

AES-128-256((K0||K1) + (T||0); Msg)

This function is exactly the case of MDPH where an attacker can manipulate the first half of the 256-bit key by adding a differential, in this case through the tweak value T. SAGE used the tool from [X,Y] to do simulations and perform that analysis, from where the following table was derived, as an example:

Truncated rounds of AES-128-256	4 Rounds	5 Rounds	6 Rounds	7 Rounds
Any key difference is possible (should coincide with [X])	$3/2^{-18}$	$3/2^{-18}$	$5/2^{-30}$	$5/2^{-30}$
Adding a half-key difference	$5/2^{-30}$	$13/2^{-83}$	$15/2^{-101}$	$23/2^{-154}$

Where the first value is the minimum number of active SBoxes in a differential byte-trail, and the second number is the differential probability of a first-found binary-trail corresponding to the shortest byte-trail. One can see that having that strict constraint that a differential can only be inserted in the first half of the key, leads to the situation when the number of active SBoxes grows much faster than if the attacker would be allowed to insert differentials into the whole 256-bit key.

After 14 rounds, it was found that there must be at least 54 active SBoxes resulting to the probability of any possible trail to be at most 2^{-324}. This can be compared with only 24 active SBoxes for 14 rounds in case a full key differential is allowed.

SAGE did not study the latest paper from 2023 at the time of the work on MILENAGE-256, but SAGE still believe that at least 54 active SBoxes is a large enough margin to be quite confident on the full strength of MDPH. Moreover, that paper [1] talks about “chosen-key” attack and it is not yet clear how to sketch such an attack scenario for MILENAGE-256-A/R.

Finally, recall that SAGE offered the MILENAGE-256R option in part because its cleaner, stronger security assurances mean the above MDPH-related topics are avoided.

[X] David Gerault, Pascal Lafourcade, Marine Minier, Christine Solnon, Computing AES related-key differential characteristics with constraint programming, Artificial Intelligence, 2020, volume 278, pp.103183, ISSN 0004-3702, https://doi.org/10.1016/j.artint.2019.103183.

[Y] INRIA Gitlab: AES-Cryptanalysis-CP-XOR-2019 (available on 2023-01-19). https://gitlab.inria.fr/source_code/aes-cryptanalysis-cp-xor-2019

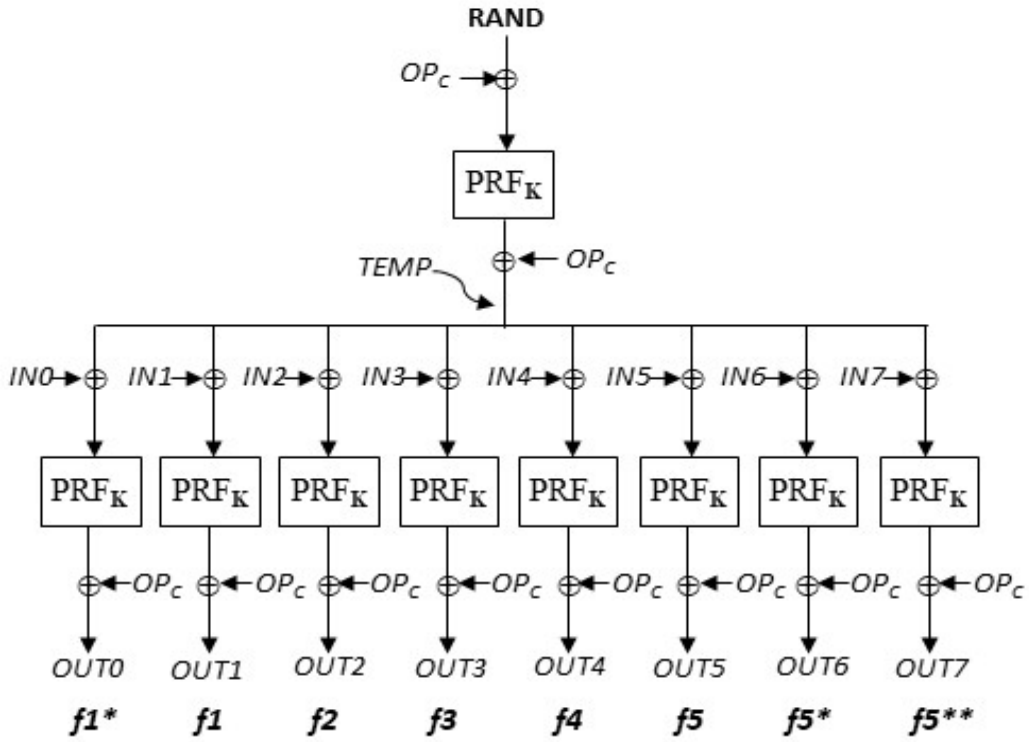

Figure 1: Overview of the *f*-algorithm set.

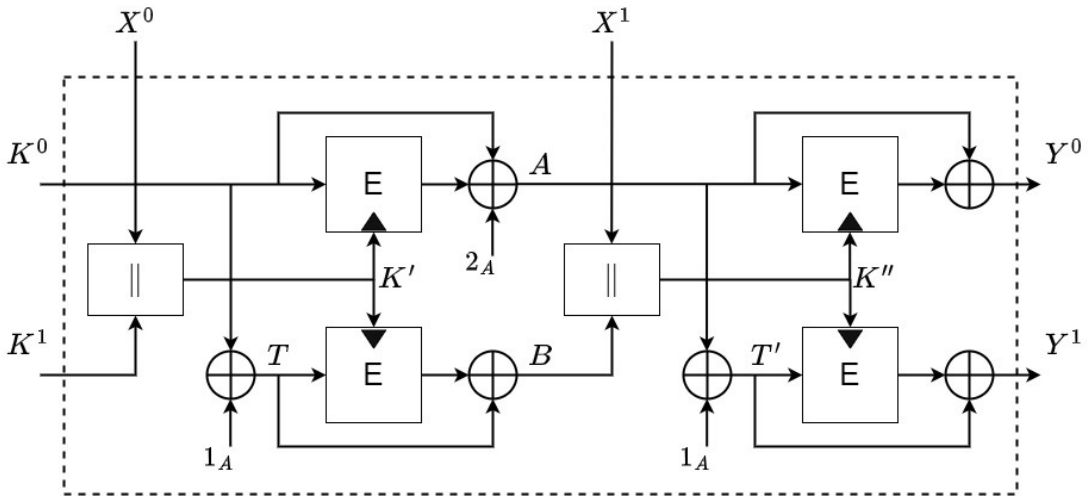

Figure 2: The MDPH^E construction. The values K', K'' are intermediate internal keys, and T, T', A, B are intermediate internal values. The triangles signify the key-inputs to the block cipher E.

Differential trails from [0]: https://eprint.iacr.org/2009/241.pdf

i	Plaintext		Subkey	Ciphertext			
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		0f 0e 0f 0e 07 07 07 07 07 07 07 07 09 09 09 09	01 01 01 01 00 00 00 00 00 00 00 00 00 00 00 00			
i	After SB	After MC	Subkey	i	After SB	After MC	Subkey
1	30 5c e1 b0 7c b5 ed 72 a6 d6 c2 16 82 eb 29 03	65 00 02 00 1f 25 1f 00 1f 00 e2 00 21 00 21 33	37 00 37 00 1f 00 1f 00 1f 00 1f 00 21 00 21 00	2	1b 00 07 00 00 12 00 00 00 00 1a 00 00 00 00 16	0c 00 0e 00 07 00 07 00 07 00 07 00 09 00 09 00	0f 01 0e 00 07 00 07 00 07 00 07 00 09 00 09 00

NOTE: in the first two rounds the plain halves of the 256-bit key are used, we see nonzero differentials in both rounds.

Differential trail from [1] https://eprint.iacr.org/2023/1095.pdf

NOTE: Both halves of the Key and also the plaintext (initial AES state) differentials are nonzero.

Round	State differences	Key differences
Plaintext	8E474700 00000000 8E474700 00??0000	
0	00000000 00000000 00000000 00??0000 00000000 00000000 A6C46262 00000000	8E474700 0C000000 8E474700 00000000
1	00C46262 A6000000 00000000 00000000 00000000 ???????? ???????? ????????	00C46262 A6000000 A6C46262 00000000

…

Ciphertext	??4747C9 ??000000 ??000000 ??000000	C94747C9 00000000 00000000 00000000