Dear All,

It seems REMUSN3V1 is vulnerable to Key recovery attacks. This vulnerability comes from the very simple Key Derivation function (L <= (Key Xor (N||032))).

This KDF produces an output of size 128 bit. A birthday attack will find a collision with an offline precomputed value of L with 2^64 complexity. Once a collision happens, Key recovery is trivial.

This is in contradiction with the security claim of table 3.2 claiming 128 bit security for key recovery for REMUSN3V1.

Best regards,

Alexandre Mège
Dear all,

We have a follow-up on Alexandre's observation. By viewing the key recovery attack as a state recovery attack one can actually construct forgery attacks on REMUS-N1 (primary version), REMUS-N3, and REMUS-M1.

On a closer inspection, we found that the same attack also works on TGIF-N1 (primary version) and TGIF-M1 as well. Basically, the attack recovers the nonce-based key L, which can be used to construct valid forgeries.

The forgery attack works as follows:

Suppose the key derivation function KDF_K takes a nonce value N as input and outputs a nonce-based key L.

1. For i = 1 to $2^{a}$:
   i. Sample $L^{i}$ in without replacement manner from $\{0,1\}^{128}$.
   ii. Simulate the encryption of (A,M) using $L^{i}$ as the nonce-based key, where $|A| = n$ and $M = M_1 || M_2 || M_3 || M_4$, where each $|M_k| = n$ for $k \in \{1,2,3,4\}$. Response: $(C^i,T^i)$ where $C^i = C^i_1 || C^i_2 || C^i_3 || C^i_4$. Store $(L^{i},C^i,T^i)$ in a list H.

2. Sort H on (C,T).

3. For j = 1 to $2^{128-a}$:
   i. Query $(N^j,A,M)$ to the encryption oracle of AE. Response: $(C^j,T^j)$. Search $(C^j,T^j)$ in H (binary search would suffice).
   ii. Suppose there exist index i $\in$ H such that $(C^j,T^j) = (C^i,T^i)$ then it would mean that $L^j = L^i$ with very high probability (matching on all 5 blocks of ciphertext and tag helps in avoiding false positives).

4. Now, the adversary can easily construct forgeries given the nonce-based key $L^j$ for nonce value $N^j$.

The query complexity is $2^{128-a}$ and the total time complexity is $2^{a} + a2^{a} + a2^{128-a}$, where the first, second, and third terms correspond to time complexity of step 1, 2, and 3, respectively. We ignore the time complexity of step 4, as it can be made negligible.

Clearly, for good choice of a, say $a=90$, we get query complexity approx. $2^{38}$ and time complexity $< 2^{100}$ (using a $< 128$). This is well within the data and time limit set by NIST. Note that, the forgery attack is in nonce-respecting model.

We remark that, here we count the direct block cipher calls within time complexity as is the norm for ideal cipher model (REMUS and TGIF use ICM). This is also plausible in real scenario where the adversary can actually make block cipher evaluations on its own by devoting sufficient time. In this regard, we also note that according to NIST's requirement, the adversary can not be restricted to at most $2^{64}$ and $2^{60}$ offline queries (as is directed on page 22 of REMUS specification) in REMUS-N1/M1/TGIF-N1/M1 and REMUS-N3, respectively.

This is especially required from REMUS-N1 and TGIF-N1, which are the primary variants in their respective submissions.

In summary, it seems that REMUS-N1/N3/M1 and TGIF-N1/M1 do not satisfy NIST requirements.

The N1/N3/M1 variants of REMUS (and TGIF) have an inherent weakness:
insufficient randomness in the initial state (key,input). Although the key is derived using nonce for each encryption query, the adversary can easily fix a constant value as the initial input. So, to create an initial state collision the adversary just needs to collide the initial key.

Thanks and Regards,
Nilanjan Datta, Ashwin Jha, Mridul Nandi