Breaking REMUS and TGIF

N.Datta, A.Jha, A.Mege*, M.Nandi

Indian Statistical Institute, Kolkata, India
*Airbus Defence and Space, Elancourt, France

NIST Lightweight Workshop, 2019

Nov 05, 2019
Call for submission Draft

- Cryptanalytic attacks on the AEAD algorithm shall require at least $2^{112}$ computations on a classical computer in a single-key setting.

- The limits on the input sizes (plaintext, associated data, and the amount of data that can be processed under one key) for this member shall not be smaller than $2^{50} - 1$ bytes.
NIST LwC Security Requirements

- $D$ (data complexity): the maximum amount of data processed under one key.
- $T$ (time complexity): total number of computations done.

Minimum security requirements from an AEAD scheme $\Psi$

If $D < 2^{50}$ bytes and $T < 2^{112}$, then $\Psi$ is secure.
A Note on the Data Complexity

The Data Limit (Data Complexity of an attack)

- Quantifies the online (queries to the AEAD scheme) resource requirements.
- Includes the total number of blocks (among all messages/ciphertexts and associated data) processed through the underlying primitive for a fixed master key.

The Computation Time (Time Complexity of an attack)

- Quantifies the offline resource requirements, and includes the total time required to process the offline evaluations of the underlying block cipher.
- The number of primitive evaluations is taken as the time complexity of evaluations.
A Note regarding the Time Complexity

The direct evaluations of the primitives have been considered within time complexity in multiple papers:

- The time-memory trade-off attack by Hellman [Hellman, 80],
- Related-key attacks on AES-256 [Biryukov+, 09],
- Attacks on hash functions [Kelsey+ 05, 06, Guo+ 14, Andreeva+ 16],
- Attacks on HMAC and NMAC [Peyrin+ 12, Leurent+ 13, Peyrin+ 14, Guo+ 14, Dinur+ 17],
- Attacks on Even-Mansour ciphers [Dunkelman+ 12, Dinur+ 13, Dinur+ 14, Dunkelman+ 15], and
- Multi-key attacks on Even-Mansour cipher [Mouha+ 15].

In fact, this also makes sense in real scenario, where the adversary can actually make block cipher evaluations on its own by devoting sufficient time.
Main Observation on REMUS and TGIF

- REMUS-N1, REMUS-N3, REMUS-M1, TGIF-N1, and TGIF-M1 restrict the number of offline evaluations of the underlying block cipher to less than $2^{64}$.

- This clearly violates the NIST LwC requirements as stated above, as the adversary is allowed to make beyond $2^{64}$ (anything below $2^{112}$ is valid) block cipher evaluations.

- This is especially required from REMUS-N1 and TGIF-N1, which are the primary variants in their respective submissions.
Revisiting the Multi-Key Attack [Mouha+ 15]

Figure: An ideal block cipher $E_K$ in the multi-key setting.
Revisiting the Multi-Key Attack [Mouha+ 15]

Make the Off-line Queries

- Choose $K^0, \ldots, K^{T-1}$ without replacement.
- For $i = 0, \ldots, (T - 1)$, simulate the encryption of $M$ using $K^i$, and store the response $(K^i, C^i)$ in a list $H$.

Make the On-line Queries

- Query $M$ under $D$ many independent keys. Let the outputs be $\hat{C}^0, \ldots, \hat{C}^{D-1}$.
- If $C^i = \hat{C}^j$ (matching occurs), recover the key $K^i$ (with high probability).

Matching occurs with probability $DT/2^n$. 
Specification of REMUS-N1 and TGIF-N1

M.Nandi (ISI, Kolkata)
Specification of REMUS-M1 and TGIF-M1
Specification of REMUS-M1 and TGIF-M1

\[ S \xrightarrow{n} M[1] \xrightarrow{n} E \xrightarrow{n} \rho \xrightarrow{n} C[1] \]

\[ (N, 34, \overline{0}) \xrightarrow{L \oplus 34} (N, 34, \overline{1}) \xrightarrow{2L \oplus 34} (N, 34, m-1) \xrightarrow{2^{m-1}L \oplus 34} \text{pad}(M[n]) \]

\[ \text{pad}(M[n]) \xrightarrow{2^{w_M}L \oplus w_M} (N, w_M, \overline{a + m}) \xrightarrow{0^n} E \xrightarrow{n} \rho \xrightarrow{n} C[m] \]

\[ w_M \in [46, 47] \]
Key Derivation Functions for REMUS -N1/M1 and TGIF -N1/M1

Choice of Parameters
- Block and key size is set to \( n = 128 \).
- Nonce size is also set to \( r = 128 \).

The Key Derivation Function
Takes a nonce \( N \) as input and outputs a nonce-based key \( L \):

\[
KDF_K(N) := E_K(N).
\]
Algorithm 1: Find the Nonce-based Key for REMUS-N1/M1 and TGIF-N1/M1

Step 1: Make the Off-line Queries
- Choose $L^0, \ldots, L^{2^t-1}$ without replacement.
- For $i = 0, \ldots, (2^t - 1)$, simulate the encryption of $(A, M)$ using $L^i$ as the nonce-based key, where $|A| = |M| = n$. Response: $(C^i, \tau^i)$. Store $(L^i, C^i, \tau^i)$ in a list $H$.

Step 2: Sort the List
Sort entries in $H$ on second and third coordinates, i.e. $(C, \tau)$. 

M.Nandi (ISI, Kolkata)  

Breaking REMUS and TGIF
Algorithm 1: Find the Nonce-based Key for REMUS -N1/M1 and TGIF -N1/M1

Step 3: Make the On-line Queries and Find Matching

- Choose distinct nonces \( \hat{\mathcal{N}}^0, \ldots, \hat{\mathcal{N}}^{2^t-1} \).
- For \( j = 0, \ldots, 2^d - 1 \), query \((\hat{\mathcal{N}}^j, A, M)\) to the encryption oracle of AEAD. Let the response be \((\hat{C}^j, \hat{\tau}^j)\).
- Search \((\hat{C}^j, \hat{\tau}^j)\) in \( H \). If \( \exists i \in H \) such that \((\hat{C}^j, \hat{\tau}^j) = (C^i, \tau^i)\) then \( \hat{L}^j = L^i \) with very high probability.
Specification of REMUS-N3

KDF3

\[ N \xrightarrow{n} KDF3 \xrightarrow{n} L \]

\[ A[1] \xrightarrow{n} (N, 132, 1) \xrightarrow{n} 2L \oplus 132 \xrightarrow{n} A[2] \]

\[ (N, 132, 2) \xrightarrow{n} 2^2L \oplus 132 \xrightarrow{n} \text{pad}(A[n]) \]

\[ (N, w_A, n) \xrightarrow{n} 2^L \oplus w_A \xrightarrow{n} \]

\[ S \]

\[ 0^n \]

\[ S \xrightarrow{n} E \]

\[ \rho \]

\[ C[1] \xrightarrow{n} \]

\[ C[2] \xrightarrow{n} \]

\[ C[m] \xrightarrow{n} \]

\[ T \]

\[ M[1] \xrightarrow{n} (N, 130, a+1) \xrightarrow{n} 2^{a+1}L \oplus 130 \xrightarrow{n} M[2] \]

\[ (N, 130, a+2) \xrightarrow{n} 2^{a+2}L \oplus 130 \xrightarrow{n} \text{pad}(M[n]) \]

\[ (N, w_M, a + m) \xrightarrow{n} 2^{a+m}L \oplus w_M \xrightarrow{n} \]

\[ T \]

\[ w_A \in [140, 141] \]
Choice of Parameters

- Block and key size is set to $n = 128$.
- Nonce size is set to $r = 96$.

The Key Derivation Function

Takes a nonce $N$ as input and outputs a nonce-based key $L$:

$$KDF_K(N) := K \oplus N \| 0^{32}.$$
Extended Algorithm 1: Find the Nonce-based Key for REMUS-N3

- Set the following parameters: \( t \geq 32, \ d = n - t \).

- Define \( L^i := 0^d || \langle i \rangle_t \), where \( \langle i \rangle_t \) denotes the \( t \)-bit representation of integer \( i \).

- Define \( \hat{N}^j = \langle j \rangle_d || 0^{r-d} \). Note that \( r - d \geq 0 \) due to \( t \geq 32 \).

- Invoke Algorithm 1 with this modified \( L^i \)'s and \( \hat{N}^j \)'s.
Key Recovery Attack against REMUS-N3

- Use Algorithm 1 to obtain a nonce-based key pair \((N', L')\).
- Recover the master key \(K = L' \oplus N' \| 0^{32}\).
Forgery against REMUS -N1/N3/M1 and TGIF -N1/M1

Nonce-respecting forgery attacks

- Use Algorithm 1 to obtain a nonce-based key pair \((N', L')\).
- Construct valid forgeries of the form \((N', A', C', T')\), where \(A'\) and \(C'\) can be chosen arbitrarily, and the tag is computed using \(L', A'\) and \(C'\).

- This attack is applicable on REMUS-N1 (primary version), REMUS-N3, and REMUS-M1 as well as TGIF-N1 (primary version) and TGIF-M1.
Complexity of the Attack

- Data complexity, \( D \approx 2^{d+5.6} \) bytes. The factor of 5.6 is due to the fact that each encryption query consists of \( 3 \approx 2^{1.6} \) blocks of data and each block contains \( 2^{4} \) bytes.

- Total time complexity, \( T \approx 2^{t+5.6} + t \cdot 2^{t} + t \cdot 2^{n-t} \).

**Choices of \( d \) and \( t \)**

- The algorithm works for all choices of \( t \geq 32 \), as \( d + t = 128 \).
- Set \( t = 90 \), which gives \( d = 38 \).
- For this choice of \( t \), we obtain \( D \approx 2^{43.6} \) bytes and \( T \approx 2^{97.5} \), which clearly falls within the NIST LwC minimum data and time limit.
Possible Improvements

- Use a hash table instead of a list.

- Improve data complexity by using empty message and empty AD. However, this may lead to some false positives which can be eliminated by making constant number of checking queries.

- Note: We do not use the empty message and AD case, as such inputs seldom occur in real scenario.
Inherent Weakness of REMUS-N1/N3/M1 and TGIF-N1/M1

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.
- To create an initial state collision, the adversary just needs to collide the initial key.

- Use of nonce in the beginning of AD processing would have prevented the above attack.
- This attack is not possible for REMUS-N2/M2 and TGIF-N2/M2 due to the larger state.
Thank You..!! Questions??