Algorithm 1: Masked AES Implementation (with masked Sbox LUT)

Input: a 16-Byte plaintext $(p[0], \ldots, p[15])$,
and a 16-Byte master key $(k_m[0], \ldots, k_m[15])$
Masks: a 16-Byte Input mask before MixColumns $(m[0], \ldots, m[15])$
a 16-Byte Output mask after MixColumns $(m_2[0], \ldots, m_2[15])$
a 1-Byte Input mask for SubBytes (m_{in})
and a 1-Byte Output mask for SubBytes (m_{out})
Output: a 16-Byte ciphertext $(c[0], \ldots, c[15])$
Function `maskedAES(`p,k_m`):`

```
// INITIALIZATION
m_in ← GetRandomBytes(1)
m_out ← GetRandomBytes(1)
m ← GetRandomBytes(16)
m_2 ← MixColumns(m)
for i ← 0 to 256 by 1 do
    S_m[i ⊕ m_in] = S[i] ⊕ m_out;  // Masked SBox LUT computation
// AES COMPUTATION
for i ← 0 to 15 by 1 do
    k[i] ← k_m[i]                          // Charge master key
    state[i] ← p[i]                        // Load Plaintext
    state[i] ← state[i] ⊕ k[i]              // Key Addition
    state[i] ← state[i] ⊕ m_in[i]           // Apply m_in
for round ← 1 to 9 by 1 do
    for i ← 0 to 15 by 1 do
        state[i] ← S_m[state[i]]            // Masked SBox LUT
    state ← ShiftRows(state)
    for i ← 0 to 15 by 1 do
        state[i] ← m[i] ⊕ m_out             // Remove m_out and apply m
    state ← MixColumns(state)
    k ← KeyScheduling(k)
    for i ← 0 to 15 by 1 do
        k[i] ← k[i] ⊕ m_2[i] ⊕ m_in         // Apply m_2 and m_in to k
    for i ← 0 to 15 by 1 do
        state[i] ← state[i] ⊕ k[i]          // Key addition
// final round with no MixColumns
for i ← 0 to 15 by 1 do
    state[i] ← S_m[state[i]]                // Masked LUT (Sbox)
state ← ShiftRows(state)
k ← KeyScheduling(k)
for i ← 0 to 15 by 1 do
    state[i] ← state[i] ⊕ m_out             // Apply m_out
for i ← 0 to 15 by 1 do
    state[i] ← state[i] ⊕ k[i]              // Key addition
return state
```