# 1 Syntax

[BLMR]'s proposed syntax involves a tuple

$$\mathsf{UpEnc} = (\mathsf{UpEnc.KeyGen}, \mathsf{UpEnc.Enc}, \mathsf{UpEnc.Dec}, \mathsf{UpEnc.ReKeyGen}, \mathsf{UpEnc.ReEnc})$$

, where $\mathsf{UpEnc.ReKeyGen}$ takes two keys (generated by $\mathsf{UpEnc.KeyGen}$) and generates a constant size "rekeying token", which will be passed into $\mathsf{UpEnc.ReEnc}$ together with the ciphertext encrypted under the first key, which will in turn output the ciphertext encrypted under the other key.

I don't think that this is necessary. I propose a syntax of the form

$$\mathsf{UpEnc} = (\mathsf{UpEnc.KeyGen}, \mathsf{UpEnc.Enc}, \mathsf{UpEnc.Dec}, \mathsf{UpEnc.ReEnc})$$

, where $\mathsf{UpEnc.ReEnc}$ instead takes the two keys and the ciphertext. I argue the reason behind this in a separate report.

We can define the specific syntax for the tuple of algorithms we propose

1. $KeyGen : \{\epsilon\} \to \{0,1\}^k$, a key generation algorithm returning a $k$ bit long key, this should be randomized.

2. $Enc : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$, an encryption algorithm that takes a key and a string and returns a string, this should be randomized.

3. $Dec : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$, a decryption algorithm that takes a key and a string and returns a string, this should not be randomized.

4. $ReEnc : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$, an encryption algorithm that takes an old key, a new key, an encryption of a string under the old key and returns an encryption of the string under the new key. This should be randomized.

# 2 Correctness

This is the correctness condition

| | |
|---|---|
| $\underline{\mathbf{G}^{\mathrm{corr}}_{\mathsf{UpEnc},A}}$<br>$i \leftarrow 0$<br>$j \leftarrow 0$<br>$j^* \leftarrow\!\!\$\ A^{\textsc{Set},\textsc{KeyGen},\textsc{Enc}}()$<br>$(M, C, k) \leftarrow M_{j^*}$<br>**if** $k = \epsilon$ **then**<br>  **return** `false`<br>**end**<br>$M' \leftarrow \mathsf{UpEnc.Dec}(C, K_k)$<br>**return** $(M' = M)$<br>$\underline{\textsc{KeyGen}()}$<br>$K_i \leftarrow\!\!\$\ \mathsf{UpEnc.KeyGen}$<br>$i \leftarrow i + 1$<br>**return** $K_i$ | $\underline{\textsc{Set}(M)}$<br>$M_i \leftarrow (M, \epsilon, \epsilon)$<br>$j \leftarrow j + 1$<br>$\underline{\textsc{Enc}(i', j')}$<br>**if** $(j' > j) \vee (i' > i)$ **then**<br>  **return** $\perp$<br>**end**<br>$(M, C, k) \leftarrow M_{j'}$<br>**if** $k = \epsilon$ **then**<br>  $C \leftarrow\!\!\$\ \mathsf{UpEnc.Enc}(\mathsf{K_{i'}}, \mathsf{M})$<br>**else**<br>  $C \leftarrow\!\!\$\ \mathsf{UpEnc.ReEnc}(K_k, K_{i'}, C)$<br>**end**<br>$M_{j'} \leftarrow (M, C, i')$<br>**return** $C$ |

Then, we can say that $\mathsf{UpEnc}$ satisfies the correctness condition if, for all adversaries $A$ we have that:

$$\Pr[\mathbf{G}^{\mathrm{corr}}_{\mathsf{UpEnc},A}] = 0$$

## 3 Rivest (1997)

$\underline{\mathbf{G}^{\mathrm{ind}}_{\mathsf{AONT}}(A)}$

    $b \leftarrow\!\!\$ \{0,1\}$
    $b' \leftarrow\!\!\$ A^{\mathrm{LR}}$
    **return** $(b = b')$

$\underline{\mathrm{LR}(M, N, i)}$

    **if** $|M| \neq |N|$ **then**
        **return** $\perp$
    **end**
    $(m_1, m_2 \ldots m_s) \leftarrow M$
    $(n_1, n_2 \ldots n_s) \leftarrow N$
    $n_i \leftarrow \epsilon$
    $m_i \leftarrow \epsilon$
    **if** $b = 0$ **then**
        $y \leftarrow\!\!\$ \mathsf{AONT.Transform}(m_1, m_2 \ldots m_s)$
    **else**
        $y \leftarrow\!\!\$ \mathsf{AONT.Transform}(n_1, n_2 \ldots n_s)$
    **end**
    **return** $y$

Then we say that the indistinguishability adversary $A$ has $l$-AONT-IND advantage:

$$\mathbf{Adv}^{\text{aont-ind}}_{\mathsf{AONT}}(A) = 2 \cdot \Pr\left[\, \mathbf{G}^{\mathrm{ind}}_{\mathsf{AONT}}(A) \,\right] - 1$$

# 4 Boyko (1999)/ Canetti et. al (2000)

$\underline{\mathbf{G}^{\text{leak}}_{\mathsf{AONT},l}(A)}$

$b \leftarrow_{\$} \{0, 1\}$
$b' \leftarrow_{\$} A^{\text{LR}}$
**return** $(b = b')$

$\underline{\text{LR}(M, N, S)}$

**if** $|M| \neq |N|$ **then**
  **return** $\perp$
**end**
$(m_1, m_2 \ldots m_s) \leftarrow M$
$(n_1, n_2 \ldots n_s) \leftarrow N$
**if** $b = 0$ **then**
  $y \leftarrow_{\$} \mathsf{AONT.Transform}(m_1, m_2 \ldots m_s)$
**else**
  $y \leftarrow_{\$} \mathsf{AONT.Transform}(n_1, n_2 \ldots n_s)$
**end**
**if** $(|S| \neq |y|) \vee (\mathsf{Hamm}(S) > (|y| - l))$ **then**
  **return** $\perp$
**else**
  $y \leftarrow y \,\&\, S$
**end**
**return** $y$

*Note that $|M|$ is the length of the string $M$ in bits, & is a bitwise AND and* $\mathsf{Hamm}(M)$ *takes the hamming weight of $M$*

Then we say that the leakage adversary $A$ has $l$-AONT-LEAK advantage:

$$\mathbf{Adv}^{\text{aont-leak}}_{\mathsf{AONT},l}(A) = 2 \cdot \Pr\left[\mathbf{G}^{\text{leak}}_{\mathsf{AONT},l}(A)\right] - 1$$

# 5 Leakage Resilience Model

$\underline{\mathbf{G}^{\mathrm{lr}}_{\mathsf{AONT},m}(A)}$

  $b \leftarrow_{\$} \{0,1\}$
  $b' \leftarrow_{\$} A^{\mathrm{LR}}$
  **return** $(b = b')$

$\underline{\mathrm{LR}(M, N, C)}$

  **if** $|M| \neq |N|$ **then**
  | **return** $\perp$
  **end**
  $(m_1, m_2 \ldots m_s) \leftarrow M$
  $(n_1, n_2 \ldots n_s) \leftarrow N$
  **if** $b = 0$ **then**
  | $y \leftarrow_{\$} \mathsf{AONT.Transform}(m_1, m_2 \ldots m_s)$
  **else**
  | $y \leftarrow_{\$} \mathsf{AONT.Transform}(n_1, n_2 \ldots n_s)$
  **end**
  **if** $(C \notin \mathcal{C}_{|y|,(|y|-m)})$ **then**
  | **return** $\perp$
  **else**
  | **return** $C(y)$
  **end**

*Note that $\mathcal{C}_{n,m}$ is the set of boolean circuits taking $n$ inputs and $m$ outputs, expressed in a string in some reasonable encoding. Then, for $C \in \mathcal{C}_{n,m}$, when we run $C(S)$ for some binary string $S$ of length $n$, $C$ will take as input the bits of $S$ and return a $m$ bit long string.*

Then we say that the leakage resilience adversary $A$ has $m$-AONT-LR advantage:

$$\mathbf{Adv}^{\text{aont-lr}}_{\mathsf{AONT},m}(A) = 2 \cdot \Pr\left[\mathbf{G}^{\mathrm{lr}}_{\mathsf{AONT},m}(A)\right] - 1$$

# 6 Relationship between Notions

## 6.1 AONT.bl-AONT-L $\Longrightarrow$ AONT-IND

**Theorem 6.1** *For any AONT-IND adversary $A$, we can construct* AONT.bl-*AONT-L adversary $B$, running in the same time and making the same number of queries, such that*

$$\mathbf{Adv}^{\text{aont-ind}}_{\mathsf{AONT}}(A) \leq \mathbf{Adv}^{\text{aont-leak}}_{\mathsf{AONT},\mathsf{AONT.bl}}(B)$$

Here is the adversary (the full proof is omitted for now):

$\underline{B^{\mathrm{LR}}}$

$b \leftarrow_{\$} A^{\textsc{SimLR}}$

**return** $b$

$\underline{\textsc{SimLR}(M, N, i)}$

$mask \leftarrow \epsilon$

$s \leftarrow \lceil \frac{|M|}{\mathsf{AONT.bl}} \rceil$

**for** $j = 1, 2, \ldots s$ **do**

**if** $j \neq i$ **then**

$mask \leftarrow mask || 1^{\mathsf{AONT.bl}}$

**else**

$mask \leftarrow mask || 0^{\mathsf{AONT.bl}}$

**end**

**end**

**return** $\mathrm{LR}(M, N, mask)$

### 6.2 $l$-AONT-L $\Longrightarrow$ $l$-AONT-LR

**Theorem 6.2** *For any $l$-AONT-L adversary $A$, we can construct $l$-AONT-LR adversary $B$, running in the same time and making the same number of queries, such that*

$$\boldsymbol{Adv}^{\textit{aont-leak}}_{\mathsf{AONT},l}(A) \leq \boldsymbol{Adv}^{\textit{aont-lr}}_{\mathsf{AONT},l}(B)$$

Here is the adversary (the full proof is omitted for now):

$\underline{B^{\mathrm{LR}}}$

$b \leftarrow_{\$} A^{\textsc{SimLR}}$

**return** $b$

$\underline{\textsc{SimLR}(M, N, mask)}$

**return** $\mathrm{LR}(M, N, C_{mask})$

$\underline{C_{mask}(X)}$

**return** $mask \,\&\, X$

### 6.3 AONT-IND $\not\Rightarrow$ $\mathsf{AONT.bl}$-AONT-L

Consider the following AONT, $\mathsf{XOR} - \mathsf{AONT}$ with some sufficiently large block length and number of messages to prevent exhaustive searches. It is a modified version of the "package transform", with some padding.

| $\underline{\mathsf{AONT.Transform}(m_1, m_2, \ldots m_s)}$ | $\underline{\mathsf{AONT.Inverse}(m'_1, m'_2 \ldots m'_{s'})}$ |
|---|---|
| **if** $\exists i.\, \lvert m_i \rvert \neq \mathsf{AONT.bl}$ **then**<br>  **return** $\perp$<br>**end**<br>$K \leftarrow_\$ \{0,1\}^{\mathsf{AONT.bl}}$<br>$K' \leftarrow_\$ \{0,1\}^{\mathsf{AONT.bl}}$<br>$m'_{s+1} \leftarrow K'$<br>**for** $i = 1, 2 \ldots s$ **do**<br>  $m'_i \leftarrow m_i \oplus E(K', \langle i \rangle_{\mathsf{AONT.bl}})$<br>  $h_i \leftarrow E(K, m'_i \oplus \langle i \rangle_{\mathsf{AONT.bl}})$<br>  $m'_{s+1} \leftarrow m'_{s+1} \oplus h_i$<br>**end**<br>$b_1 \vert\vert b_2 \ldots \vert\vert b_{\mathsf{AONT.bl}} \leftarrow m'_{s+1}$<br>**for** $i = 1, 2 \ldots s$ **do**<br>  $m'_{s+i} \leftarrow_\$ \{0,1\}^{\mathsf{AONT.bl}}$<br>  $m'_{s+i} \leftarrow m'_{s+i}\ \&\ 0^{\mathsf{AONT.bl}-1} b_i$<br>**end**<br>**return** $(m'_1, m'_2 \ldots m'_{2s-1}, m'_{2s}, K)$ | **if** $(\exists i.\, \lvert m'_i \rvert \neq \mathsf{AONT.bl}) \vee (s' \leq 2)$ **then**<br>  **return** $\perp$<br>**end**<br>**for** $i = 1, 2, \ldots s$ **do**<br>  $b_i \leftarrow m_{s+i} \mod 2$<br>**end**<br>$K' \leftarrow b_1 \vert\vert b_2 \ldots \vert\vert b_{\mathsf{AONT.bl}}$<br>$K \leftarrow m'_{s'}$<br>$s \leftarrow s' - 2$<br>**for** $(i = 1, 2, \ldots s)$ **do**<br>  $h_i \leftarrow E(K, m'_i \oplus i)$<br>  $K' \leftarrow K' \oplus h_i$<br>**end**<br>**for** $(i = 1, 2, \ldots s)$ **do**<br>  $m_i \leftarrow E(K', i) \oplus m'_i$<br>**end**<br>**return** $(m_1, m_2 \ldots m_s)$ |

Then, consider the following $\mathsf{AONT.bl} - AONT - L$ adversary $A$ which has $\mathbf{Adv}^{\text{aont-leak}}_{\mathsf{AONT},l}(A) = 1$, making only 1 query and running as efficiently as the $\mathsf{AONT.Inverse}$ function.

$\underline{A}$

$S \leftarrow 0^{256 \cdot \mathsf{AONT.bl}}$
**for** $i = 1, 2, \ldots \mathsf{AONT.bl}$ **do**
  $S \leftarrow S \vert\vert 0^{\mathsf{AONT.bl}-1} 1$
**end**
$S \leftarrow S \vert\vert 1^{\mathsf{AONT.bl}}$
$M \leftarrow \mathrm{LR}(0^{256 \cdot \mathsf{AONT.bl}}, 1^{256 \cdot \mathsf{AONT.bl}}, S)$
$X \leftarrow \mathsf{AONT.Inverse}(M)$
**if** $M = 0^{256 \cdot \mathsf{AONT.bl}}$ **then**
  **return** 0
**else**
  **return** 1
**end**

Notice that the hamming weight of $S$ will be $|S| - \mathsf{AONT.bl}$, and so LR will not return $\perp$. In addition, since the bits of the AONT that were not leaked were all "padding" bits, the value $X$ is exactly the value that was transformed by the LR. Therefore, the adversary does not run in time much longer than the inverse function, and has advantage 1.

Now we show that this is still $\mathsf{AONT} - \mathsf{IND}$ secure. We do this by showing that for any $\mathsf{AONT} - \mathsf{IND}$ adversary $A$ against this scheme, we have an adversary $B$ against the package transform (call this the $\mathsf{Package}$ scheme) such that

$$\mathbf{Adv}^{\text{aont-ind}}_{\mathsf{AONT}}(A) \geq \mathbf{Adv}^{\text{aont-ind}}_{\mathsf{Package}}(B)$$