

1 Syntax

[BLMR]'s proposed syntax involves a tuple

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

, where UpEnc.ReKeyGen takes two keys (generated by UpEnc.KeyGen) and generates a constant size “rekeying token”, which will be passed into 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

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

, where 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. $\text{KeyGen} : \{\epsilon\} \rightarrow \{0, 1\}^k$, a key generation algorithm returning a k bit long key, this should be randomized.
2. $\text{Enc} : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, an encryption algorithm that takes a key and a string and returns a string, this should be randomized.
3. $\text{Dec} : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, a decryption algorithm that takes a key and a string and returns a string, this should not be randomized.
4. $\text{ReEnc} : \{0, 1\}^k \times \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^*$, 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

$\underline{\mathbf{G}_{\text{UpEnc}, A}^{\text{CORT}}}$ $i \leftarrow 0$ $j \leftarrow 0$ $j^* \leftarrow \$ A^{\text{SET}, \text{KEYGEN}, \text{ENC}}()$ $(M, C, k) \leftarrow M_{j^*}$ $\mathbf{if } k = \epsilon \mathbf{ then}$ $ \mathbf{return false}$ \mathbf{end} $M' \leftarrow \text{UpEnc.Dec}(C, K_k)$ $\mathbf{return } (M' = M)$ $\underline{\text{KEYGEN}()}$ $K_i \leftarrow \$ \text{UpEnc.KeyGen}$ $i \leftarrow i + 1$ $\mathbf{return } K_i$	$\underline{\text{SET}(M)}$ $M_i \leftarrow (M, \epsilon, \epsilon)$ $j \leftarrow j + 1$ $\underline{\text{ENC}(i', j')}$ $\mathbf{if } (j' > j) \vee (i' > i) \mathbf{ then}$ $ \mathbf{return } \perp$ \mathbf{end} $(M, C, k) \leftarrow M_{j'}$ $\mathbf{if } k = \epsilon \mathbf{ then}$ $ C \leftarrow \$ \text{UpEnc.Enc}(K_{i'}, M)$ \mathbf{else} $ C \leftarrow \$ \text{UpEnc.ReEnc}(K_k, K_{i'}, C)$ \mathbf{end} $M_{j'} \leftarrow (M, C, i')$ $\mathbf{return } C$
--	--

Then, we can say that UpEnc satisfies the correctness condition if, for all adversaries A we have that:

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

3 Rivest (1997)

<pre> G_{AONT}^{ind}(A) $b \leftarrow_{\\$} \{0, 1\}$ $b' \leftarrow_{\\$} A^{\text{LR}}$ return ($b = b'$) LR(M, N, i) if $M \neq N$ then return \perp end $(m_1, m_2 \dots m_s) \leftarrow M$ $(n_1, n_2 \dots n_s) \leftarrow N$ $n_i \leftarrow \epsilon$ $m_i \leftarrow \epsilon$ if $b = 0$ then $y \leftarrow_{\\$} \text{AONT.Transform}(m_1, m_2 \dots m_s)$ else $y \leftarrow_{\\$} \text{AONT.Transform}(n_1, n_2 \dots n_s)$ end return y </pre>
--

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

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

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

$\underline{\mathbf{G}_{\text{AONT},l}^{\text{leak}}(A)}$ $b \leftarrow_{\$} \{0, 1\}$ $b' \leftarrow_{\$} A^{\text{LR}}$ $\mathbf{return} (b = b')$ $\underline{\text{LR}(M, N, S)}$ $\mathbf{if} \ M \neq N \ \mathbf{then}$ $\quad \ \mathbf{return} \ \perp$ \mathbf{end} $(m_1, m_2 \dots m_s) \leftarrow M$ $(n_1, n_2 \dots n_s) \leftarrow N$ $\mathbf{if} \ b = 0 \ \mathbf{then}$ $\quad \ y \leftarrow_{\$} \text{AONT.Transform}(m_1, m_2 \dots m_s)$ \mathbf{else} $\quad \ y \leftarrow_{\$} \text{AONT.Transform}(n_1, n_2 \dots n_s)$ \mathbf{end} $\mathbf{if} \ (S \neq y) \vee (\text{Hamm}(S) > (y - l)) \ \mathbf{then}$ $\quad \ \mathbf{return} \ \perp$ \mathbf{else} $\quad \ y \leftarrow y \ \& \ S$ \mathbf{end} $\mathbf{return} \ y$

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

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

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

5 Leakage Resilience Model

$\mathbf{G}_{\text{AONT},m}^{\text{lr}}(A)$ $b \leftarrow_{\$} \{0, 1\}$ $b' \leftarrow_{\$} A^{\text{LR}}$ $\text{return } (b = b')$ <hr/> $\text{LR}(M, N, C)$ $\text{if } M \neq N \text{ then}$ $\quad \text{return } \perp$ end $(m_1, m_2 \dots m_s) \leftarrow M$ $(n_1, n_2 \dots n_s) \leftarrow N$ $\text{if } b = 0 \text{ then}$ $\quad y \leftarrow_{\$} \text{AONT.Transform}(m_1, m_2 \dots m_s)$ else $\quad y \leftarrow_{\$} \text{AONT.Transform}(n_1, n_2 \dots n_s)$ end $\text{if } (C \notin \mathcal{C}_{ y , (y -m)}) \text{ then}$ $\quad \text{return } \perp$ else $\quad \text{return } C(y)$ 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:

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

6 Relationship between Notions

6.1 AONT.bl – AONT – L \implies 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*

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

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

$\underline{B^{\text{LR}}}$ $b \leftarrow_{\$} A^{\text{SIMLR}}$ $\mathbf{return } b$ $\underline{\text{SIMLR}(M, N, i)}$ $mask \leftarrow \epsilon$ $s \leftarrow \lceil \frac{ M }{\text{AONT.bl}} \rceil$ $\mathbf{for } j = 1, 2, \dots s \mathbf{ do}$ <div style="margin-left: 20px;"> $\mathbf{if } j \neq i \mathbf{ then}$ <div style="margin-left: 20px;"> $mask \leftarrow mask 1^{\text{AONT.bl}}$ </div> \mathbf{else} <div style="margin-left: 20px;"> $mask \leftarrow mask 0^{\text{AONT.bl}}$ </div> \mathbf{end} </div> \mathbf{end} $\mathbf{return } \text{LR}(M, N, mask)$
--

6.2 $l\text{-AONT} - \text{L} \implies l\text{-AONT} - \text{LR}$

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

$$\mathbf{Adv}_{\text{AONT},l}^{\text{aont-leak}}(A) \leq \mathbf{Adv}_{\text{AONT},l}^{\text{aont-lr}}(B)$$

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

$\underline{B^{\text{LR}}}$ $b \leftarrow_{\$} A^{\text{SIMLR}}$ $\mathbf{return } b$ $\underline{\text{SIMLR}(M, N, mask)}$ $\mathbf{return } \text{LR}(M, N, C_{mask})$ $\underline{C_{mask}(X)}$ $\mathbf{return } mask \ \& \ X$
--

6.3 $\text{AONT} - \text{IND} \not\equiv \text{AONT.bl} - \text{AONT} - \text{L}$

Rivest provides a construction of an AONT known as the package transform. It makes use of an arbitrary block cipher, with key of length AONT.bl , $E : \{0, 1\}^{\text{AONT.bl}} \times \{0, 1\}^{\text{AONT.bl}} \rightarrow \text{AONT.bl}$.

I modified this scheme slightly to achieve a scheme that is $\text{AONT} - \text{IND}$ secure but not $\text{AONT.bl} - \text{AONT} - \text{L}$ secure.

<p><u>AONT.Transform($m_1, m_2, \dots m_s$)</u></p> <pre> if $\exists i. m_i \neq \text{AONT.bl}$ then return \perp end $K \leftarrow_{\\$} \{0, 1\}^{\text{AONT.bl}}$ $K' \leftarrow_{\\$} \{0, 1\}^{\text{AONT.bl}}$ $m'_{s+1} \leftarrow K'$ for $i = 1, 2 \dots s$ do $m'_i \leftarrow m_i \oplus E(K', \langle i \rangle_{\text{AONT.bl}})$ $h_i \leftarrow E(K, m'_i \oplus \langle i \rangle_{\text{AONT.bl}})$ $m'_{s+1} \leftarrow m'_{s+1} \oplus h_i$ end $m'_{s+1} \leftarrow E(K, m'_{s+1})$ $m'_{s+2} \leftarrow m'_{s+1} \& 1^{\frac{\text{AONT.bl}}{2}} 0^{\frac{\text{AONT.bl}}{2}}$ $m'_{s+1} \leftarrow m'_{s+1} \& 0^{\frac{\text{AONT.bl}}{2}} 1^{\frac{\text{AONT.bl}}{2}}$ return $(m'_1, m'_2 \dots m'_s, m'_{s+1}, m'_{s+2}, K)$ </pre>	<p><u>AONT.Inverse($m'_1, m'_2 \dots m'_{s'}$)</u></p> <pre> if $(\exists i. m'_i \neq \text{AONT.bl}) \vee (s' \leq 2)$ then return \perp end $K \leftarrow m'_{s'}$ $K' \leftarrow E^{-1}(K, m'_{s'-1} \oplus m'_{s'-2})$ $s \leftarrow s' - 3$ for $(i = 1, 2, \dots s)$ do $h_i \leftarrow E(K, m'_i \oplus i)$ $K' \leftarrow K' \oplus h_i$ end for $(i = 1, 2, \dots s)$ do $m_i \leftarrow E(K', i) \oplus m'_i$ end return $(m_1, m_2 \dots m_s)$ </pre>
--	--

Notice that since this construction ensures that there are AONT.bl bits of “padding” in the second and third to last blocks of the pseudo-message, an AONT.bl leakage adversary can just choose to not receive the padding, and still be able to compute the inverse function in full, since the padding is always zero. On the other hand, suppose that there existed a AONT – IND adversary against AONT, A , then we can construct B , a PRF adversary against block cipher E .