Designing Efficient Dyadic Operations for Cryptographic Applications

Gustavo Banegas\textsuperscript{1}, Paulo S. L. M. Barreto\textsuperscript{2}, Edoardo Persichetti\textsuperscript{3} and Paolo Santini\textsuperscript{4}

August 19, 2018

\textsuperscript{1}Technische Universiteit Eindhoven, Netherlands
\textsuperscript{2}University of Washington at Tacoma, USA
\textsuperscript{3}Florida Atlantic University, USA
\textsuperscript{4}Università Politecnica delle Marche, Italy
### NIST proposals

November 2017: NIST posts 82 submissions from 260 people.

<table>
<thead>
<tr>
<th></th>
<th>Signatures</th>
<th>KEM/Encryption</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice-based</td>
<td>4</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>Code-based</td>
<td>5</td>
<td>19</td>
<td>24</td>
</tr>
<tr>
<td>MQ-based</td>
<td>7</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>Hash-based</td>
<td>4</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td>3</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>23</strong></td>
<td><strong>59</strong></td>
<td><strong>82</strong></td>
</tr>
</tbody>
</table>
DAGS: Key Encapsulation from Dyadic GS Codes

- It is a code-based KEM;
- It uses Generalized Srivastava codes;
- It has short keys
DAGS: Key Encapsulation from Dyadic GS Codes

- It is a code-based KEM;
- It uses Generalized Srivastava codes;
- It has short keys, much smaller than Classic McEliece;
- As the name suggest it uses dyadic operations.
Introduction

What did we do?

- Improve code-based cryptographic schemes that use Quasi-Dyadic (QD) operations;
- Analyze the multiplication of dyadic matrices using: “Standard”, Karatsuba and Fast Walsh-Hadamard Transformation;
- Apply LUP decomposition to dyadic case.
What are dyadic matrices?

Given a ring $\mathcal{R}$ and a vector $h = (h_0, h_1, \ldots, h_{n-1}) \in \mathcal{R}$ with $n = 2^r$, $r \in \mathbb{N}$, called the order.

A dyadic matrix is the symmetric matrix with components $\Delta_{ij} = h_i \oplus j$, where $\oplus$ stands for bitwise exclusive-or.

We use $\Delta(h)$ to denote dyadic matrix.

The product of two dyadic matrices is a dyadic matrix.

Quasi-dyadic matrix

A quasi-dyadic matrix is a block matrix whose blocks are dyadic.
What are dyadic matrices?

Given a ring $\mathcal{R}$ and a vector $h = (h_0, h_1, \ldots, h_{n-1}) \in \mathcal{R}$ with $n = 2^r, \ r \in \mathbb{N}$, called the order.

A dyadic matrix is the symmetric matrix with components $\Delta_{ij} = h_i \oplus j$, where $\oplus$ stands for bitwise exclusive-or.

We use $\Delta(h)$ to denote dyadic matrix.

The product of two dyadic matrices is a dyadic matrix.

Quasi-dyadic matrix

A quasi-dyadic matrix is a block matrix whose blocks are dyadic.

In particular, we focus on the special case of quasi-dyadic matrices with elements belonging to a field $\mathbb{F}$ of characteristic 2.
A dyadic permutation

A *dyadic permutation* is a dyadic matrix $\Pi^i \in \Delta(\{0, 1\}^n)$ given by $\Pi^i = \Delta(\pi^i)$ where $\pi^i$ is the $i$-th unit vector.

**Example**

Suppose $n = 4$, and $i = 1$. So, we have $\pi^1 = (0, 1, 0, 0)$ and $\Pi^1$ is equal to:

$$
\Pi^1 = \begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
$$
The element of a matrix \( C \) in position \((i, j)\) is obtained as the multiplication between the \(i\)-th row of \( A \) and the \(j\)-th column of \( B \).
Standard Multiplication

The element of a matrix $C$ in position $(i, j)$ is obtained as the multiplication between the $i$-th row of $A$ and the $j$-th column of $B$. However, $A$ and $B$ are dyadic matrices and they are symmetric. So, the product is equivalent to the inner product between $i$-th row of $A$ and the $j$-th of $B$.

The schoolbook matrix multiplication takes $2^{3r}$ multiplications. Because $\Delta(a)\Delta(b)$ is dyadic we only need the top row.
Standard Multiplication

input : \( r \in \mathbb{N}, \ n = 2^r \) and \( a, b \in \mathbb{F}^n \)
output: \( c \in \mathbb{F}^n \) such that \( \Delta(c) = \Delta(a)\Delta(b) \)
\( c \leftarrow (0, 0, \ldots, 0) \);
\( c_0 \leftarrow a_0b_0 \);
for \( i \leftarrow 1 \) to \( n - 1 \) do
  \( c_0 \leftarrow c_0 + a_ib_i \);
  \( i^{(b)} \leftarrow \text{binary representation of } i \);
  for \( j \leftarrow 0 \) to \( n - 1 \) do
    \( j^{(b)} \leftarrow \text{binary representation of } j \);
    \( \pi^{(b)} \leftarrow i^{(b)} \oplus j^{(b)} \);
    \( \pi \leftarrow \text{integer representation of } \pi^{(b)} \);
    \( c_i \leftarrow c_i + a_ib_\pi \);
  end
end
return \( c \);
Standard Multiplication

**input**: $r \in \mathbb{N}$, $n = 2^r$ and $a, b \in \mathbb{F}^n$

**output**: $c \in \mathbb{F}^n$ such that $\Delta(c) = \Delta(a)\Delta(b)$

$c \leftarrow (0, 0, \ldots, 0)$;

$c_0 \leftarrow a_0b_0$;

for $i \leftarrow 1$ to $n - 1$ do

\[ c_0 \leftarrow c_0 + a_ib_i; \]

$i(b) \leftarrow$ binary representation of $i$;

for $j \leftarrow 0$ to $n - 1$ do

\[ j(b) \leftarrow$ binary representation of $j; \]

\[ \pi(b) \leftarrow i(b) \oplus j(b); \]

\[ \pi \leftarrow$ integer representation of $\pi(b); \]

\[ c_i \leftarrow c_i + a_ib_\pi; \]

end

end

return $c$;

**Complexity estimated in:**

\[ C_{std} = r(2^{2r} - 2^r) + 2^{2r} C_{mul} + (2^{2r} - 2^r) C_{sum} \]
Dyadic Convolution

What is dyadic convolution?
The dyadic convolution of two vectors $a, b \in \mathbb{F}$, denoted by $a \triangle b$, is the unique vector of $\mathbb{F}$ such that $\Delta(a \triangle b) = \Delta(a) \Delta(b)$.

Sylvester-Hadamard Matrices

$$H_0 = \begin{bmatrix} 1 \end{bmatrix},$$

$$H_r = \begin{bmatrix} H_{r-1} & H_{r-1} \\ H_{r-1} & -H_{r-1} \end{bmatrix}, \quad r > 0.$$
What do we achieve?
Computing $c$ such that $\Delta(a)\Delta(b) = \Delta(c)$ involves only three multiplications of vectors by Sylvester-Hadamard matrices.
What do we achieve?
Computing \( c \) such that \( \Delta(a) \Delta(b) = \Delta(c) \) involves only three multiplications of vectors by Sylvester-Hadamard matrices.

For this we propose two algorithms. First, we need to compute \( aH_r \), where \( a \) is a vector and \( H_r \) a Sylvester-Hadamard matrix. Second, we perform the multiplication
Dyadic Convolution

input : $r \in \mathbb{N}$, $n = 2^r$ and $a \in \mathbb{F}^n$
output: $aH_r$
$v \leftarrow 1$
for $j \leftarrow 1$ to $n$ do
  $w \leftarrow v$
  $v \leftarrow (v << 1)$;
  /* left shift by one position */
  for $i \leftarrow 0$ to $n - 1$ by $v$ do
    for $l \leftarrow 0$ to $w$ do
      $s \leftarrow a_{i+l}$;
      $q \leftarrow a_{i+l+w}$;
      $a_{i+l} \leftarrow s + q$;
      $a_{i+l+w} \leftarrow s - q$;
    end
  end
return $a$;

Algorithm 1: The fast Walsh-Hadamard transform (FWHT)
Dyadic Convolution

**input**: \( r \in \mathbb{N}, n = 2^r \) and \( a, b \in \mathbb{F}^n \)

**output**: \( a \triangledown b \in \mathbb{F}^n \) such that \( \Delta(a)\Delta(b) = \Delta(a \triangledown b) \)

\( c \leftarrow (0, 0, \ldots, 0); \)
\( \tilde{c} \leftarrow (0, 0, \ldots, 0); \)

Compute \( \tilde{a} \leftarrow aH_r \) via previous algorithm;

Compute \( \tilde{b} \leftarrow bH_r \) via previous algorithm;

for \( j \leftarrow 0 \) to \( n - 1 \) do
  \( \tilde{c} \leftarrow \tilde{a}_j\tilde{b}_j; \)
end

Compute \( c \leftarrow \tilde{c}H_r \) via previous algorithm;

\( c \leftarrow (c >> r); \)

/* right shift by \( r \) positions */

return \( c; \)

**Algorithm 2**: Dyadic convolution via the FWHT
Consider a vector $\mathbf{a}$ and its halves defined as:

$$
\mathbf{a}_0 = \left[ a_0, a_1, \cdots, a_{\frac{n}{2} - 1} \right]
$$
$$
\mathbf{a}_1 = \left[ a_{\frac{n}{2}}, a_{\frac{n}{2} + 1}, \cdots, a_{n-1} \right].
$$

Some straightforward computations show that the following relations hold:

$$
\mathbf{c}_0 = \mathbf{a}_0 \mathbf{b}_0 + \mathbf{a}_1 \mathbf{b}_1
$$
$$
\mathbf{c}_1 = (\mathbf{a}_0 + \mathbf{a}_1)(\mathbf{b}_0 + \mathbf{b}_1) + \mathbf{c}_0
$$

We can summarize the complexity of this method as:

$$
C_{\text{Kar}} = 3^r \cdot C_{\text{mul}} + 4 \cdot [3^r - 2^r] \cdot C_{\text{sum}}
$$
Dyadic Matrices Inverse

Inverse of dyadic matrices can be defined as:
The inverse $\Delta(a)^{-1}$ is a dyadic matrix $\Delta(b)$. We can compute $b$ as follows:

1. Compute $\tilde{b}$ with $\text{diag}(\tilde{b}) = [\text{diag}(aH_r)]^{-1}$;
2. Compute $b' = \tilde{b}H_r$;
3. For each entry in $b'$ shift right $r$ positions, the result is $b$. 
### Improving DAGS

**Table: Cost of Multiplication between Dyadic Matrices**

<table>
<thead>
<tr>
<th></th>
<th>Standard</th>
<th>Karatsuba</th>
<th>Dyadic Convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_{2^5}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r = 4$</td>
<td>4,833</td>
<td>2,194</td>
<td>3,899</td>
</tr>
<tr>
<td>$r = 5$</td>
<td>21,285</td>
<td>5,909</td>
<td>12,045</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^6}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r = 4$</td>
<td>5,833</td>
<td>2,194</td>
<td>4,899</td>
</tr>
<tr>
<td>$r = 5$</td>
<td>23,231</td>
<td>6,223</td>
<td>13,568</td>
</tr>
</tbody>
</table>
Improving DAGS

Table: Comparison of Inversion Methods

<table>
<thead>
<tr>
<th></th>
<th>Original DAGS</th>
<th>LUP Inversion</th>
<th>LUP + Karatsuba</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAGS 1</td>
<td>1,318,973,209</td>
<td>321,771</td>
<td>108,117</td>
</tr>
<tr>
<td>DAGS 3</td>
<td>2,211,076,311</td>
<td>557,822</td>
<td>198,199</td>
</tr>
<tr>
<td>DAGS 5</td>
<td>17,925,330,712</td>
<td>654,713</td>
<td>431,890</td>
</tr>
</tbody>
</table>
Questions

Thank you for your attention.
gustavo@cryptme.in