**Abstract & Introduction**

Public Key Cryptography (PKC) protects data confidentiality when transmitting it through an unsecured channel, such as the Internet. We present an efficient design for both Curve448 Elliptic Curve Diffie-Hellman (ECDH) and Ed448 Edwards-curve Digital Signature Algorithm (EdDSA) algorithms:

- We implement ARMv7 highly optimized low-level finite field arithmetic.
- Our design outperforms previous ECDH implementations by more than 48%.
- Our Ed448 DSA shows a speedup of 11%, requiring 6 and 7.4 MCCs for sign and verify.

**Curve448 & Ed448**

A Montgomery Elliptic Curve Curve448 over a finite field \( \mathbb{F}_p \) is defined by:

\[
E_M/\mathbb{F}_p : v^2 = u^3 + Au^2 + u
\]

where the value of \( A \) is defined as 156326 and \( p = 2^{448} - 2^{224} - 1 \). Montgomery curves have their birationally analogue Edwards curves, where Curve448 can be represented by the solutions to the equation:

\[
E_E/\mathbb{F}_p : ax^2 + y^2 = 1 + dx^3y^3
\]

with \( d = -30801 \) and \( a = 1 \). The core of Curve448 and Ed448 is the scalar-point multiplication where \( P = [k] \cdot Q \) is the addition of point \( Q \) to itself \( k \) times.

**ARMv7-M Architecture**

The ARM Cortex-M4 processor’s architecture delivers a set of powerful instructions that are devoid of structural hazards.

**Proposed Design for Field Arithmetic**

We present a novel technique for multi-precision multiplication and squaring, with an emphasis on increasing row (inner loop) size and hence decreasing memory accesses for partial value accumulation.

**Performance Evaluation & Conclusions**

We compare our work with the best-known counterparts in the literature targeting the same platform for Curve448- and Ed448-based algorithms and present the latency results in number of clock cycles in Tables 2 and 3. We mark around 48.2% and 36.8% of speedup for X448 @24MHz and @168MHz, respectively. We report a speedup of 13.1%, 8.1%, and 12.4% for Ed448 EdDSA.

**References**


---

**Table 1.** ARMv7-M ISA for memory access and MAC instructions.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Functionality</th>
<th>Latency (IC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVW</td>
<td>memory</td>
<td>2</td>
</tr>
<tr>
<td>MOVZ</td>
<td>memory</td>
<td>1</td>
</tr>
<tr>
<td>VMPYV</td>
<td>memory</td>
<td>1</td>
</tr>
<tr>
<td>UMAUL</td>
<td>Rs1, Rs2 × Rs3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Figure 1.** Proposed architecture for 448-bit multi-precision multiplication. Black lines denote inner loop execution.

**Figure 2.** Proposed architecture for 448-bit multi-precision squaring. Red line denotes additional simulated lane for increased word size of the doubled operand.

**Figure 3.** Proposed design for the inner execution loop for multi-precision multiplication and squaring.