Faster Implementation of Scalar Multiplication on Koblitz Curves

Diego F. Aranha — *University of Brasilia*
Armando Faz Hernández — *CINVESTAV-IPN*
Julio López — *University of Campinas*
Francisco Rodríguez Henríquez — *CINVESTAV-IPN*

Second International Conference on Cryptology and Information Security in Latin America 2012

Outline

Contribution

We are able to perform a scalar multiplication on a Koblitz Curve in less than 10^5 clock cycles on a desktop user processor.

Key points to achieve this result are:

- Native support of polynomial multiplication over $\mathbb{F}_2$ and use of vectorized instructions.

Contribution

We are able to perform a scalar multiplication on a Koblitz Curve in less than 10^5 clock cycles on a desktop user processor.

Key points to achieve this result are:

- Native support of polynomial multiplication over $\mathbb{F}_2$ and use of vectorized instructions.
- Optimized implementation of binary field arithmetic.

Contribution

We are able to perform a scalar multiplication on a Koblitz Curve in less than 10^5 clock cycles on a desktop user processor.
Key points to achieve this result are:

- Native support of polynomial multiplication over $\mathbb{F}_2$ and use of vectorized instructions.
- Optimized implementation of binary field arithmetic.
- Improvements on point addition implementation.

Contribution

We are able to perform a scalar multiplication on a Koblitz Curve in less than 10^5 clock cycles on a desktop user processor.

Key points to achieve this result are:

- Native support of polynomial multiplication over $\mathbb{F}_2$ and use of vectorized instructions.
- Optimized implementation of binary field arithmetic.
- Improvements on point addition implementation.
- Data-dependent precision on computation of $w\tau$-NAF recoding.

Contribution

We are able to perform a scalar multiplication on a Koblitz Curve in less than 10^5 clock cycles on a desktop user processor.

Key points to achieve this result are:

- Native support of polynomial multiplication over $\mathbb{F}_2$ and use of vectorized instructions.
- Optimized implementation of binary field arithmetic.
- Improvements on point addition implementation.
- Data-dependent precision on computation of $w\tau$-NAF recoding.
- Application of Frobenius endomorphism in a similar way to s-GLV decomposition.

Koblitz Curves

Koblitz curves, E_a, are elliptic curves over $\mathbb{F}_2$ defined by the following equation:

$$E_a\colon\ y^2 + xy = x^3 + ax^2 + 1 \tag{1}$$

where $a \in \{0, 1\}$.

Koblitz Curves

Koblitz curves, E_a, are elliptic curves over $\mathbb{F}_2$ defined by the following equation:

$$E_a\colon\ y^2 + xy = x^3 + ax^2 + 1 \qquad (1)$$

where $a \in \{0, 1\}$.
For cryptographic purposes, we can work in elliptic curves over extension fields, $E_a(\mathbb{F}_{2^m})$ such that,

$$\#E_a(\mathbb{F}_{2^m}) = f \cdot r \qquad (2)$$

where $f = 2^{2-a}$ and r is a large prime.

Koblitz Curves

One important remark on Koblitz curves is the presence of an endomorphic function, say τ. Its evaluation involves application of Frobenius automorphism in $\mathbb{F}_2$ over each coordinate of a given point:

$$\begin{aligned} \tau\colon (x,y) &\rightarrow (x^2,y^2) \\ \mathcal{O} &\rightarrow \mathcal{O} \end{aligned} \tag{3}$$

Koblitz Curves

One important remark on Koblitz curves is the presence of an endomorphic function, say τ. Its evaluation involves application of Frobenius automorphism in $\mathbb{F}_2$ over each coordinate of a given point:

$$\begin{array}{rcl} \tau\colon (x, y) & \rightarrow & (x^2, y^2) \\ \mathcal{O} & \rightarrow & \mathcal{O} \end{array} \quad (3)$$

This endomorphism holds, for every point $P \in E_a$, the following equation:

$$\tau^2(P) + 2P = \mu\tau(P) \qquad \mu = (-1)^{1-a} \quad (4)$$

is also known as the *characteristic equation*.

Koblitz Curves

- Computation of scalar multiplication can be improved by recoding the scalar into τ-adic representation, which allows to replace point doublings by τ applications.

Koblitz Curves

- Computation of scalar multiplication can be improved by recoding the scalar into τ-adic representation, which allows to replace point doublings by τ applications.
- If some extra memory is available, we can think about an $w\tau$-adic representation, which produces a sparse non-zero coefficient expansion of the scalar, enabling the use of precomputed points.

Koblitz Curves

- Computation of scalar multiplication can be improved by recoding the scalar into τ-adic representation, which allows to replace point doublings by τ applications.
- If some extra memory is available, we can think about an $w\tau$-adic representation, which produces a sparse non-zero coefficient expansion of the scalar, enabling the use of precomputed points.
- We can fix w value in such a way that optimizes time/memory to get high speed.

Vector instructions

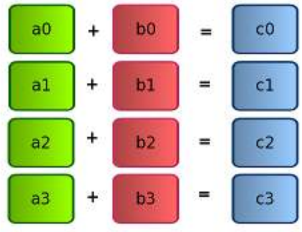
- Nowadays vector instructions are present in contemporary desktop processors.

Vector instructions

- Nowadays vector instructions are present in contemporary desktop processors.
- Latest architectures have special register and instruction sets that are able to perform one single operation over a set of data. Resulting in a vector-wise processing.

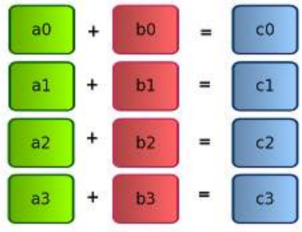
Vector instructions

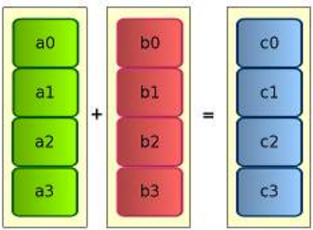
- Nowadays vector instructions are present in contemporary desktop processors.
- Latest architectures have special register and instruction sets that are able to perform one single operation over a set of data. Resulting in a vector-wise processing.



Vector instructions

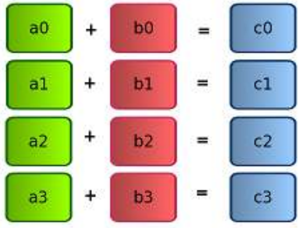
- Nowadays vector instructions are present in contemporary desktop processors.
- Latest architectures have special register and instruction sets that are able to perform one single operation over a set of data. Resulting in a vector-wise processing.

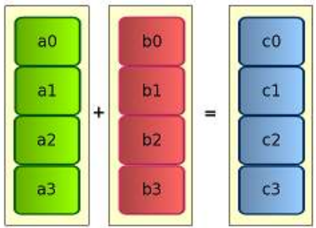




Vector instructions

- Nowadays vector instructions are present in contemporary desktop processors.
- Latest architectures have special register and instruction sets that are able to perform one single operation over a set of data. Resulting in a vector-wise processing.





- In this work, we exploit capabilities of AVX and SSE instruction sets on a Sandy Bridge micro-architecture to develop binary field arithmetic.

Vector instructions

Relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.

Vector instructions

Relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** Processes four parallel shifts per each 64-bit integer in the register.

Vector instructions

Relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** Processes four parallel shifts per each 64-bit integer in the register.
- **128-bit shifts.** Processes two parallel shifts per each 128-bit data in the register, with the restriction that only shifts by multiples of 8 bits are supported.

Vector instructions

Relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** Processes four parallel shifts per each 64-bit integer in the register.
- **128-bit shifts.** Processes two parallel shifts per each 128-bit data in the register, with the restriction that only shifts by multiplies of 8 bits are supported.
- **Memory alignment.** This instruction concatenates two vector registers and shifts by an 8-bit multiple. It is useful to handle misaligned data.

Vector instructions

Relevant vector instructions:

- **Bit-wise XOR, AND, OR.** These instructions operate with 256-bit registers performing bit-wise operations.
- **64-bit shifts.** Processes four parallel shifts per each 64-bit integer in the register.
- **128-bit shifts.** Processes two parallel shifts per each 128-bit data in the register, with the restriction that only shifts by multiples of 8 bits are supported.
- **Memory alignment.** This instruction concatenates two vector registers and shifts by an 8-bit multiple. It is useful to handle misaligned data.
- **Carry-less multiplier.**

Carry-less multiplier

- The instruction PCLMULQDQ, included in AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.

Carry-less multiplier

- The instruction PCLMULQDQ, included in AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.
- Unlike integer multiplier, this instruction performs intermediate additions without carry bits, hence its name.

Carry-less multiplier

- The instruction PCLMULQDQ, included in AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.
- Unlike integer multiplier, this instruction performs intermediate additions without carry bits, hence its name.
- Recent applications of this instruction on binary field arithmetic have shown that increases throughput on high speed implementations such as GCM for authenticated encryption, elliptic curves and η-pairings

Carry-less multiplier

- The instruction PCLMULQDQ, included in AES-NI instruction set, performs a polynomial multiplication over $\mathbb{F}_2[x]$.
- Unlike integer multiplier, this instruction performs intermediate additions without carry bits, hence its name.
- Recent applications of this instruction on binary field arithmetic have shown that increases throughput on high speed implementations such as GCM for authenticated encryption, elliptic curves and η-pairings
- Native support of this operation produces relevant speed-up in the computation of scalar multiplication on Koblitz curves.

Binary field arithmetic

- **Addition.** This is the fastest and simplest operation that can be performed just with bit-wise XOR using 256-bit registers.

Binary field arithmetic

- **Addition.** This is the fastest and simplest operation that can be performed just with bit-wise XOR using 256-bit registers.
- **Multiplication.** The most performance-critical operation is computed in two stages, first a polynomial multiplication and then the result of that is followed by a modular reduction.

Binary field arithmetic

- **Addition.** This is the fastest and simplest operation that can be performed just with bit-wise XOR using 256-bit registers.
- **Multiplication.** The most performance-critical operation is computed in two stages, first a polynomial multiplication and then the result of that is followed by a modular reduction.
  - **Polynomial multiplication**. Karatsuba approach was applied in 64-bit granularity, thus 13 polynomial multiplications and 32 additions are computed.

    Our algorithm processes all polynomial multiplications independently, so maximizes pipeline occupancy level.

    In order to improve register allocation, operands are stored in an interleaved form.

Binary field arithmetic

- **Multiplication.**
  - **Modular reduction.** After a polynomial multiplication or squaring is processed, a double length element needs to be reduced to obtain a field element.

Binary field arithmetic

- **Multiplication.**
  - **Modular reduction.** After a polynomial multiplication or squaring is processed, a double length element needs to be reduced to obtain a field element.
    Despite of inefficient choice of the standarized irreducible pentanomial for modular reduction:

$$f(z) = z^{283} + z^{12} + z^{7} + z^{5} + 1$$

Binary field arithmetic

- **Multiplication.**
  - **Modular reduction.** After a polynomial multiplication or squaring is processed, a double length element needs to be reduced to obtain a field element.
    Despite of inefficient choice of the standarized irreducible pentanomial for modular reduction:

    $$f(z) = z^{283} + z^{12} + z^7 + z^5 + 1$$

    we take advantage of this factorization:

    $$f(z) = z^{283} + (z^7 + 1)(z^5 + 1)$$

    to formulate a faster modular reduction.

Binary field arithmetic

- **Multiplication.**
  - **Modular reduction.** After a polynomial multiplication or squaring is processed, a double length element needs to be reduced to obtain a field element.

    Despite of inefficient choice of the standarized irreducible pentanomial for modular reduction:

    $$f(z) = z^{283} + z^{12} + z^{7} + z^{5} + 1$$

    we take advantage of this factorization:

    $$f(z) = z^{283} + (z^{7} + 1)(z^{5} + 1)$$

    to formulate a faster modular reduction.

    The new optimized reduction algorithm is based on bit and byte-wise shifting and memory alignment instructions.

Binary field arithmetic

- **Squaring.** Squaring is a cheap operation due to Frobenius map in binary fields.
  This operation can be accomplished using look-up table based instructions provided in Supplemental SSE3 extension set.

Binary field arithmetic

- **Squaring.** Squaring is a cheap operation due to Frobenius map in binary fields.
  This operation can be accomplished using look-up table based instructions provided in Supplemental SSE3 extension set.
- **Multi-squaring.** Given a field element a, evaluating

$$b = a^{2^k}$$

  for a fixed value k, is a time-memory trade-off. In which a look-up table of $16\lceil \frac{m}{4} \rceil$ field elements is stored. Resulting faster than repeatedly squarings when $k > 6$.
  We can increase performance when 256-bit XOR instruction is present with respect to 128-bit XOR instructions.

Binary field arithmetic

- **Inversion.** The friendliest approach to compute the most costly binary field operation is using Itoh-Tsujii algorithm.

Binary field arithmetic

- **Inversion.** The friendliest approach to compute the most costly binary field operation is using Itoh-Tsujii algorithm.
  Given a field element a, we use the following identity to compute its inverse:
  $$a^{-1} = \left(a^{2^{m-1}-1}\right)^2$$

Binary field arithmetic

- **Inversion.** The friendliest approach to compute the most costly binary field operation is using Itoh-Tsujii algorithm.
  Given a field element a, we use the following identity to compute its inverse:
  $$a^{-1} = \left(a^{2^{m-1}-1}\right)^2$$
  The term $a^{2^{m-1}-1}$ is obtained by sequentially computing intermediate terms of the form:
  $$\left(a^{2^i-1}\right)^{2^j} \cdot \left(a^{2^j-1}\right) \qquad i, j \in [0, \lambda]$$
  where i, j are elements of an addition chain of λ length.

Binary field arithmetic

- **Inversion.** The friendliest approach to compute the most costly binary field operation is using Itoh-Tsujii algorithm.
  Given a field element a, we use the following identity to compute its inverse:
  $$a^{-1} = \left(a^{2^{m-1}-1}\right)^2$$
  The term $a^{2^{m-1}-1}$ is obtained by sequentially computing intermediate terms of the form:
  $$\left(a^{2^i-1}\right)^{2^j} \cdot \left(a^{2^j-1}\right) \qquad i, j \in [0, \lambda]$$
  where i, j are elements of an addition chain of λ length.
  This sequence of powers is done by using multi-squaring operations.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

We notice that the lazy reduction technique can be applied to save 2 modular reductions when sum of products are computed.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

We notice that the lazy reduction technique can be applied to save 2 modular reductions when sum of products are computed.
Thus, mixed point addition is computed by:

- 8 unreduced multiplications.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

We notice that the lazy reduction technique can be applied to save 2 modular reductions when sum of products are computed.
Thus, mixed point addition is computed by:

- 8 unreduced multiplications.
- 5 unreduced squarings.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

We notice that the lazy reduction technique can be applied to save 2 modular reductions when sum of products are computed.
Thus, mixed point addition is computed by:

- 8 unreduced multiplications.
- 5 unreduced squarings.
- 11 modular reductions.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

We notice that the lazy reduction technique can be applied to save 2 modular reductions when sum of products are computed.
Thus, mixed point addition is computed by:

- 8 unreduced multiplications.
- 5 unreduced squarings.
- 11 modular reductions.
- 10 additions.

Elliptic curve arithmetic

Lazy reduction. The formula for mixed point addition in López-Dahab coordinates requires:

- 8 multiplications.
- 5 squarings.
- 8 additions.

We notice that the lazy reduction technique can be applied to save 2 modular reductions when sum of products are computed.
Thus, mixed point addition is computed by:

- 8 unreduced multiplications.
- 5 unreduced squarings.
- 11 modular reductions.
- 10 additions.

Due to the choice of irreducible pentanomial, saving one modular reduction represents approximately 15% of a field multiplication,

$w\tau$-NAF recoding

Recoding. Recoding an integer scalar k to a $w\tau$-NAF expansion allows to apply Frobenius endomorphism in the computation of scalar multiplication.

$w\tau$-NAF recoding

Recoding. Recoding an integer scalar k to a $w\tau$-NAF expansion allows to apply Frobenius endomorphism in the computation of scalar multiplication.

$$k \in \mathbb{Z} \rightarrow \sum_{i=0}^{l-1} u_i \tau^i, \qquad u_i \in \{\alpha_j\}_{j \in \{1,3,5,\dots\}}$$

such that expansion preserves characteristics of $w\tau$-NAF expansion.

$w\tau$-NAF recoding

Recoding. Recoding an integer scalar k to a $w\tau$-NAF expansion allows to apply Frobenius endomorphism in the computation of scalar multiplication.

$$k \in \mathbb{Z} \to \sum_{i=0}^{l-1} u_i \tau^i, \qquad u_i \in \{\alpha_j\}_{j \in \{1,3,5,\dots\}}$$

such that expansion preserves characteristics of $w\tau$-NAF expansion.

- This conversion is an iterative algorithm, such that for every iteration, scalar k is reduced by some constant until becomes equal to zero.

$w\tau$-NAF recoding

Recoding. Recoding an integer scalar k to a $w\tau$-NAF expansion allows to apply Frobenius endomorphism in the computation of scalar multiplication.

$$k \in \mathbb{Z} \rightarrow \sum_{i=0}^{l-1} u_i \tau^i, \qquad u_i \in \{\alpha_j\}_{j \in \{1,3,5,\dots\}}$$

such that expansion preserves characteristics of $w\tau$-NAF expansion.

- This conversion is an iterative algorithm, such that for every iteration, scalar k is reduced by some constant until becomes equal to zero.
- Due to deterministic nature of algorithm, code was completely unrolled to handle only the required precision in current iteration.

Scalar multiplication on Koblitz curves

Once that scalar $k \in \mathbb{Z}$ is recoded to a $w\tau$-NAF expansion, then scalar multiplication is computed as follows:

$$Q = [k]P$$

Scalar multiplication on Koblitz curves

Once that scalar $k \in \mathbb{Z}$ is recoded to a $w\tau$-NAF expansion, then scalar multiplication is computed as follows:

$$\begin{aligned} Q &= [k]P \\ &= \sum_{i=0}^{l-1} u_i \tau^i P \end{aligned}$$

Scalar multiplication on Koblitz curves

Once that scalar $k \in \mathbb{Z}$ is recoded to a $w\tau$-NAF expansion, then scalar multiplication is computed as follows:

$$\begin{aligned} Q &= [k]P \\ &= \sum_{i=0}^{l-1} u_i \tau^i P \\ &= u_0 P + u_1 \tau(P) + u_2 \tau^2(P) + \cdots + u_{l-1}\tau^{l-1}(P) \end{aligned}$$

Scalar multiplication on Koblitz curves

Once that scalar $k \in \mathbb{Z}$ is recoded to a $w\tau$-NAF expansion, then scalar multiplication is computed as follows:

$$\begin{aligned} Q &= [k]P \\ &= \sum_{i=0}^{l-1} u_i \tau^i P \\ &= u_0 P + u_1 \tau(P) + u_2 \tau^2(P) + \cdots + u_{l-1}\tau^{l-1}(P) \end{aligned}$$

We notice that $\tau^{\lfloor m/s \rfloor}$ function acts as an endomorphism in the context of s-GLV decomposition.

s-GLV method

Given ψ an endomorphism in $E(\mathbb{F}_q)$, then for any point P of order r, $\exists\, \xi \in \mathbb{Z}_r$ such that,

$$\psi(P) = [\xi]P$$

s-GLV method

Given ψ an endomorphism in $E(\mathbb{F}_q)$, then for any point P of order r, $\exists\, \xi \in \mathbb{Z}_r$ such that,

$$\psi(P) = [\xi]P$$

We can split the scalar k such that,

$$k = k_0 + k_1\xi + k_2\xi^2 + \cdots + k_{s-1}\xi^{s-1} \pmod{r}$$

with $|k_i| \approx \frac{1}{s}|k|$.

s-GLV method

Given ψ an endomorphism in $E(\mathbb{F}_q)$, then for any point P of order r, $\exists\, \xi \in \mathbb{Z}_r$ such that,

$$\psi(P) = [\xi]P$$

We can split the scalar k such that,

$$k = k_0 + k_1\xi + k_2\xi^2 + \cdots + k_{s-1}\xi^{s-1} \pmod r$$

with $|k_i| \approx \frac{1}{s}|k|$.
Thus, scalar multiplication is performed as:

$$\begin{aligned} Q &= [k]P \\ &= [k_0]P + [k_1]\psi + [k_2]\psi^2 + \cdots + [k_{s-1}]\psi^{s-1}(P) \end{aligned}$$

s-GLV method

Given ψ an endomorphism in $E(\mathbb{F}_q)$, then for any point P of order r, $\exists\, \xi \in \mathbb{Z}_r$ such that,

$$\psi(P) = [\xi]P$$

We can split the scalar k such that,

$$k = k_0 + k_1\xi + k_2\xi^2 + \cdots + k_{s-1}\xi^{s-1} \pmod{r}$$

with $|k_i| \approx \frac{1}{s}|k|$.
Thus, scalar multiplication is performed as:

$$\begin{aligned} Q &= [k]P \\ &= [k_0]P + [k_1]\psi + [k_2]\psi^2 + \cdots + [k_{s-1}]\psi^{s-1}(P) \end{aligned}$$

The latest equation can be done using an interleaved version of scalar multiplication, in which $m - \lfloor \frac{m}{s} \rfloor$ doublings are saved.

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function.

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function. Once that scalar is recoded into a $w\tau$-NAF expansion, spliting in s parts is straightforward, arising the following equation:

$$Q = [k]P$$

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function. Once that scalar is recoded into a $w\tau$-NAF expansion, spliting in s parts is straightforward, arising the following equation:

$$\begin{aligned} Q &= [k]P \\ &= \sum_{i=0}^{l-1} u_i \tau^i P \end{aligned}$$

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function. Once that scalar is recoded into a $w\tau$-NAF expansion, spliting in s parts is straightforward, arising the following equation:

$$
\begin{aligned}
Q &= [k]P \\
&= \sum_{i=0}^{l-1} u_i \tau^i P \\
&= \sum_{i=0}^{\lfloor m/s \rfloor - 1} u_i \tau^i P +
\end{aligned}
$$

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function. Once that scalar is recoded into a $w\tau$-NAF expansion, spliting in s parts is straightforward, arising the following equation:

$$
\begin{aligned}
Q &= [k]P \\
&= \sum_{i=0}^{l-1} u_i \tau^i P \\
&= \sum_{i=0}^{\lfloor m/s \rfloor - 1} u_i \tau^i P + \sum_{i=\lfloor m/s \rfloor}^{2\lfloor m/s \rfloor - 1} u_i \tau^{i-\lfloor m/s \rfloor} \psi(P) + \cdots +
\end{aligned}
$$

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function. Once that scalar is recoded into a $w\tau$-NAF expansion, spliting in s parts is straightforward, arising the following equation:

$$
\begin{aligned}
Q &= [k]P \\
&= \sum_{i=0}^{l-1} u_i \tau^i P \\
&= \sum_{i=0}^{\lfloor m/s \rfloor -1} u_i \tau^i P + \sum_{i=\lfloor m/s \rfloor}^{2\lfloor m/s \rfloor -1} u_i \tau^{i-\lfloor m/s \rfloor} \psi(P) + \cdots + \\
&\quad \sum_{i=m-\lfloor m/s \rfloor}^{l-1} u_i \tau^{i-m-\lfloor m/s \rfloor} \psi^{s-1}(P)
\end{aligned}
$$

Scalar Multiplication on Koblitz curves

Under context of s-GLV method, let $\psi \equiv \tau^{\lfloor m/s \rfloor}$ an endomorphic function. Once that scalar is recoded into a $w\tau$-NAF expansion, spliting in s parts is straightforward, arising the following equation:

$$
\begin{aligned}
Q &= [k]P \\
&= \sum_{i=0}^{l-1} u_i \tau^i P \\
&= \sum_{i=0}^{\lfloor m/s \rfloor - 1} u_i \tau^i P + \sum_{i=\lfloor m/s \rfloor}^{2\lfloor m/s \rfloor - 1} u_i \tau^{i-\lfloor m/s \rfloor} \psi(P) + \cdots + \\
&\quad \sum_{i=m-\lfloor m/s \rfloor}^{l-1} u_i \tau^{i-m-\lfloor m/s \rfloor} \psi^{s-1}(P)
\end{aligned}
$$

The latest equation can be done using an interleaved version of scalar multiplication, saving $m - \lfloor \frac{m}{s} \rfloor$ applications of τ.

Results

Elliptic curve. In order to address 128-bit security level and compliance with standards, we chose the Koblitz curve K283 proposed by NIST.

$$E_0(\mathbb{F}_{2^{283}}) : \; y^2 + xy = x^3 + 1$$

$$\mathbb{F}_{2^{283}} \equiv \mathbb{F}_2[z]/(f(z))$$

where $f(z) = z^{283} + z^{12} + z^7 + z^5 + 1$.

Results

Elliptic curve. In order to address 128-bit security level and compliance with standards, we chose the Koblitz curve K283 proposed by NIST.

$$E_0(\mathbb{F}_{2^{283}}) : \; y^2 + xy = x^3 + 1$$

$$\mathbb{F}_{2^{283}} \equiv \mathbb{F}_2[z]/(f(z))$$

where $f(z) = z^{283} + z^{12} + z^7 + z^5 + 1$.

Platform. We use a Sandy Bridge architecture present on an Intel processor Core-i7 2600K at 3.4GHz. Some features:

- SSE4.2 and AVX instruction sets.
- PCLMULQDQ instruction included in AES-NI instruction set.
- GCC and ICC compilers were used under Linux environment.

Results

Elliptic curve. In order to address 128-bit security level and compliance with standards, we chose the Koblitz curve K283 proposed by NIST.

$$E_0(\mathbb{F}_{2^{283}}) : \; y^2 + xy = x^3 + 1$$

$$\mathbb{F}_{2^{283}} \equiv \mathbb{F}_2[z]/(f(z))$$

where $f(z) = z^{283} + z^{12} + z^7 + z^5 + 1$.

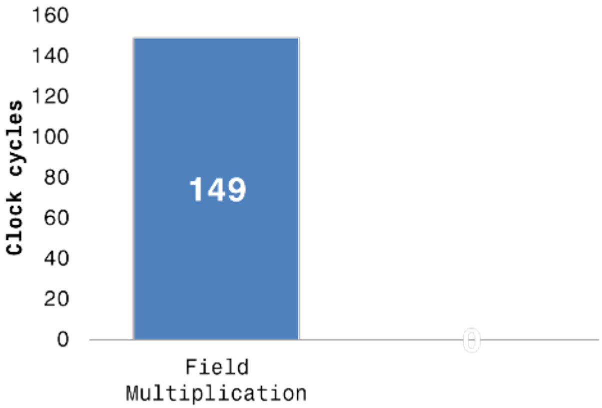
Platform. We use a Sandy Bridge architecture present on an Intel processor Core-i7 2600K at 3.4GHz. Some features:

- SSE4.2 and AVX instruction sets.
- PCLMULQDQ instruction included in AES-NI instruction set.
- GCC and ICC compilers were used under Linux environment.

The benchmarking was conducted using performance guidelines suggested by EBACS website.

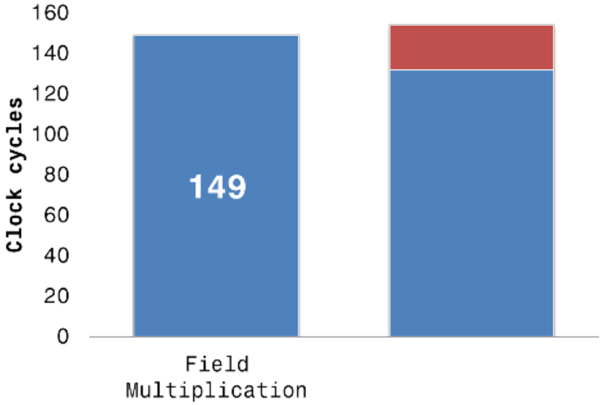
Results on binary field operations

Field multiplication $\mathbb{F}_{2^{283}}$.



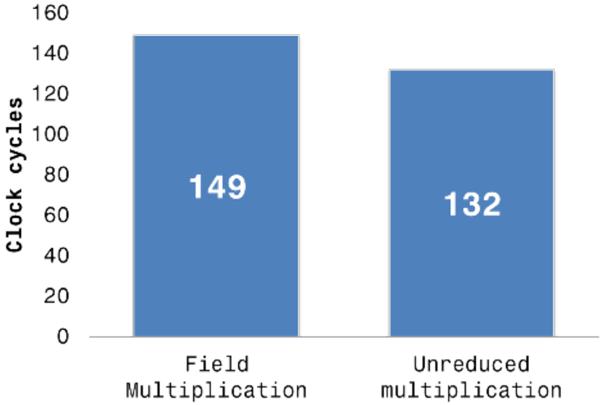
Results on binary field operations

Field multiplication $\mathbb{F}_{2^{283}}$.



Results on binary field operations

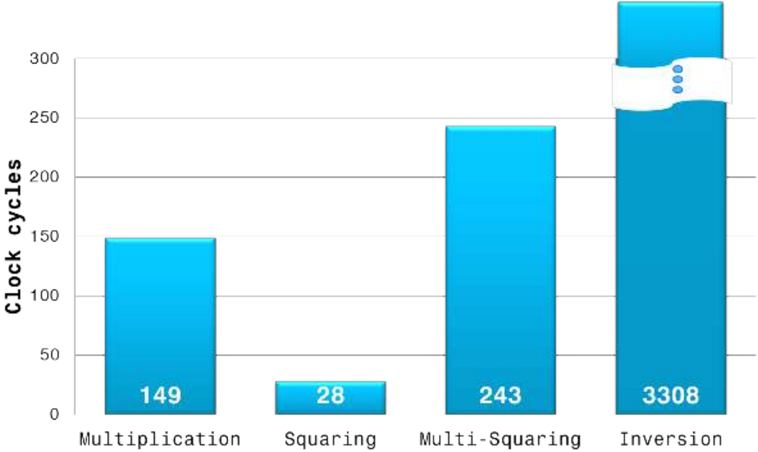
Field multiplication $\mathbb{F}_{2^{283}}$.



Saving of 15% per unreduced multiplication using lazy reduction technique.

Results on binary field operations

Comparison between binary field operations $\mathbb{F}_{2^{283}}$.



Results on Scalar Multiplication on Koblitz curves

Timings are reported in three different scenarios:

- **Unknown point**

$$[k]P$$

  P is unknown and k is generated at random, e.g. ECDH shared secret computing.

Results on Scalar Multiplication on Koblitz curves

Timings are reported in three different scenarios:

- **Unknown point**

$$[k]P$$

P is unknown and k is generated at random, e.g. ECDH shared secret computing.

- **Fixed point.**

$$[k]Q$$

Q is known in advance and k is generated at random, e.g. ECDSA signature.

Results on Scalar Multiplication on Koblitz curves

Timings are reported in three different scenarios:

- **Unknown point**

$$[k]P$$

P is unknown and k is generated at random, e.g. ECDH shared secret computing.

- **Fixed point.**

$$[k]Q$$

Q is known in advance and k is generated at random, e.g. ECDSA signature.

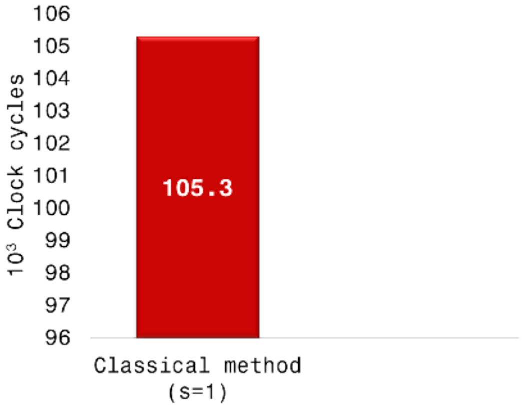
- **Multiple point multiplication.**

$$[k]P + [k']Q$$

P is unknown, Q is known in advance and k and k' are generated online at random, e.g. ECDSA verification.

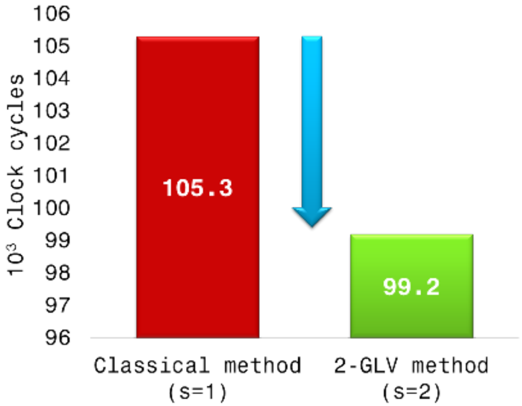
Results on Scalar Multiplication on Koblitz curves

Unknown point scenario.



Results on Scalar Multiplication on Koblitz curves

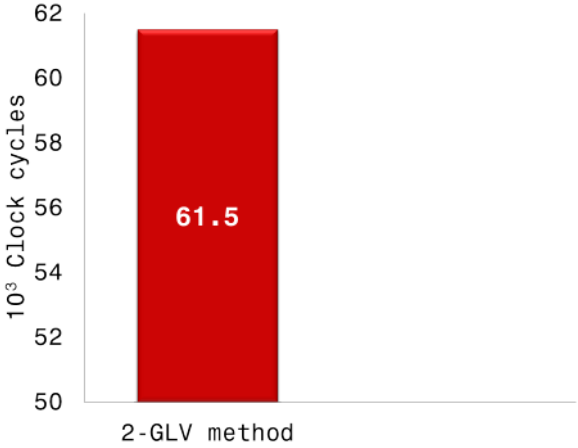
Unknown point scenario.



Breaking the barrier of 10^5 clock cycles to compute a scalar multiplication of a random point.

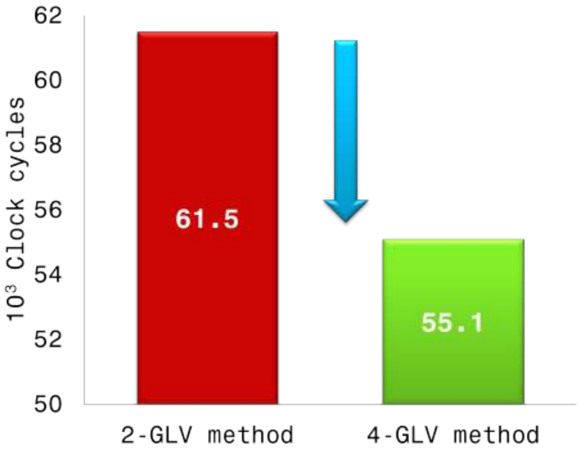
Results on Scalar Multiplication on Koblitz curves

Fixed point scenario. Using a table of 64 precomputed points ($w = 8$).



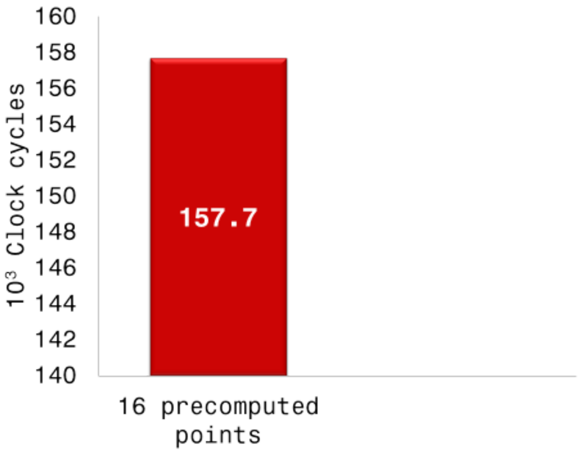
Results on Scalar Multiplication on Koblitz curves

Fixed point scenario. Using a table of 64 precomputed points ($w = 8$).



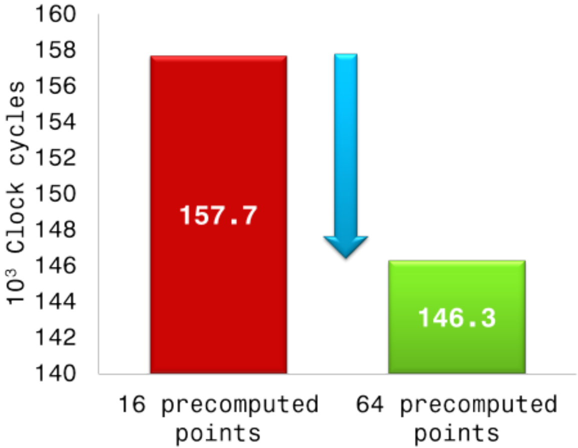
Results on Scalar Multiplication on Koblitz curves

Multiple point multiplication scenario. Using 2-GLV method.



Results on Scalar Multiplication on Koblitz curves

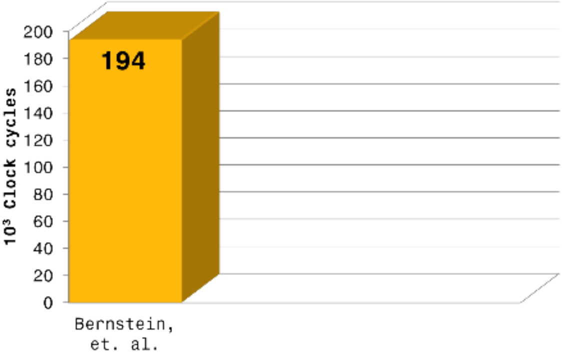
Multiple point multiplication scenario. Using 2-GLV method.



Comparison to related work at 128-bit security level

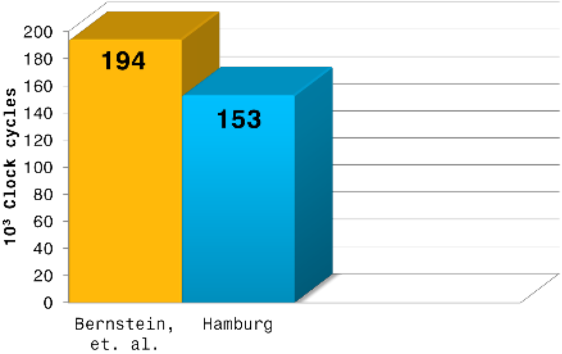
Comparison to related work at 128-bit security level

1. Twisted Edwards curve over $\mathbb{F}_{2^{255}-19}$. [BDL+11]



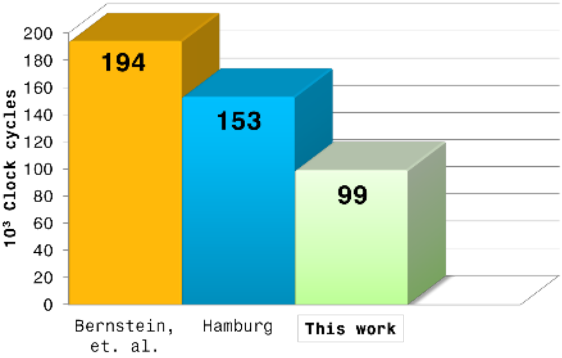
Comparison to related work at 128-bit security level

1. Twisted Edwards curve over $\mathbb{F}_{2^{255}-19}$. [BDL+11]
2. Twisted Edwards curve over $\mathbb{F}_{2^{252}-2^{232}-1}$. [Ham12]



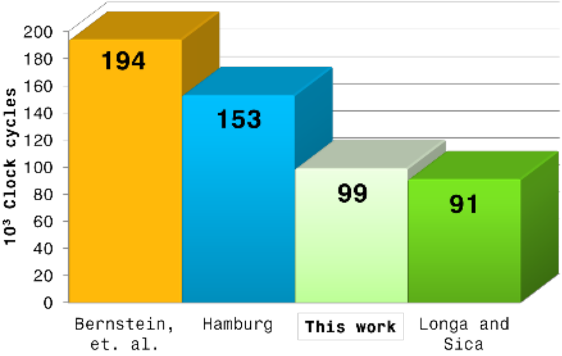
Comparison to related work at 128-bit security level

1. Twisted Edwards curve over $\mathbb{F}_{2^{255}-19}$. [BDL+11]
2. Twisted Edwards curve over $\mathbb{F}_{2^{252}-2^{232}-1}$. [Ham12]
3. Koblitz curve K283 by NIST recommendation.



Comparison to related work at 128-bit security level

1. Twisted Edwards curve over $\mathbb{F}_{2^{255}-19}$. [BDL+11]
2. Twisted Edwards curve over $\mathbb{F}_{2^{252}-2^{232}-1}$. [Ham12]
3. Koblitz curve K283 by NIST recommendation.
4. 4-GLV/GLS curve in TE form over $\mathbb{F}_{p^2}$, $p = 2^{127} - 5997$. [LS12]



Performance estimates

	Longa and Sica [LS12]	**This work**
Finite field	$\mathbb{F}_{p^2}$, $p = 2^{127} - 5997$	$\mathbb{F}_{2^{283}}$

Performance estimates

	Longa and Sica [LS12]	**This work**
Finite field	$\mathbb{F}_{p^2}$, $p = 2^{127} - 5997$	$\mathbb{F}_{2^{283}}$
Operation counting	742M+225S+1I+767A	407M + 1092S + 3I

Performance estimates

	Longa and Sica [LS12]	**This work**
Finite field	$\mathbb{F}_{p^2}$, $p = 2^{127} - 5997$	$\mathbb{F}_{2^{283}}$
Operation counting	742M+225S+1I+767A	407M + 1092S + 3I
64-bit multiplications	4(3(742)+2(225)) =10,704	13(407) = 5,291

Performance estimates

	Longa and Sica [LS12]	**This work**
Finite field	$\mathbb{F}_{p^2}$, $p = 2^{127} - 5997$	$\mathbb{F}_{2^{283}}$
Operation counting	742M+225S+1I+767A	407M + 1092S + 3I
64-bit multiplications	4(3(742)+2(225)) =10,704	13(407) = 5,291
Multiplier (latency)	Integer (3 cc)	Carry-less (8 cc)

Performance estimates

	Longa and Sica [LS12]	**This work**
Finite field	$\mathbb{F}_{p^2}$, $p = 2^{127} - 5997$	$\mathbb{F}_{2^{283}}$
Operation counting	742M+225S+1I+767A	407M + 1092S + 3I
64-bit multiplications	4(3(742)+2(225)) =10,704	13(407) = 5,291
Multiplier (latency)	Integer (3 cc)	Carry-less (8 cc)

This estimate shows that performing scalar multiplication on a Koblitz curve should be considerated faster than a prime curve equipped with endomorphisms, if sufficient support to binary field multiplication is present.

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.
- New optimization techniques were applied to binary field arithmetic to get maximum performance.

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.
- New optimization techniques were applied to binary field arithmetic to get maximum performance.
- Adaptation of s-GLV decomposition method to Koblitz curves via $\tau^{\lfloor m/s \rfloor}$ endomorphism.

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.
- New optimization techniques were applied to binary field arithmetic to get maximum performance.
- Adaptation of s-GLV decomposition method to Koblitz curves via $\tau^{\lfloor m/s \rfloor}$ endomorphism.
- This implementation provides a trade-off between side-channel protection and standards compliance.

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.
- New optimization techniques were applied to binary field arithmetic to get maximum performance.
- Adaptation of s-GLV decomposition method to Koblitz curves via $\tau^{\lfloor m/s \rfloor}$ endomorphism.
- This implementation provides a trade-off between side-channel protection and standards compliance.
- **Future work.**

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.
- New optimization techniques were applied to binary field arithmetic to get maximum performance.
- Adaptation of s-GLV decomposition method to Koblitz curves via $\tau^{\lfloor m/s \rfloor}$ endomorphism.
- This implementation provides a trade-off between side-channel protection and standards compliance.
- **Future work.**
  - Provide side-channel resistant against timing and memory cache attacks.

Conclusion

- We present fast timings for computation of scalar multiplication on binary curves at 128-bit security level.
- New optimization techniques were applied to binary field arithmetic to get maximum performance.
- Adaptation of s-GLV decomposition method to Koblitz curves via $\tau^{\lfloor m/s \rfloor}$ endomorphism.
- This implementation provides a trade-off between side-channel protection and standards compliance.
- **Future work.**
  - Provide side-channel resistant against timing and memory cache attacks.
  - Submit source code to EBACS website.

latincrypt
SANTIAGO DE CHILE 2012

Thanks!, Muchas Gracias!, Obrigado!

References I

D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang.
High-Speed High-Security Signatures.
In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 124–142. Springer, 2011.

Mike Hamburg.
Fast and compact elliptic-curve cryptography.
Cryptology ePrint Archive, Report 2012/309, 2012.
`http://eprint.iacr.org/`.

Patrick Longa and Francesco Sica.
Four-Dimensional Gallant-Lambert-Vanstone Scalar Multiplication.
Advances in Cryptology - ASIACRYPT 2012, 2012.