An FPGA-based Accelerator for Tate Pairing on Edwards Curves over Prime Fields

Marcin Rogawski    Ekawat Homsirikamol    Kris Gaj

Cryptographic Engineering Research Group (CERG)
Department of ECE, Volgenau School of Engineering
George Mason University, Fairfax, VA, USA

11th CryptArchi Workshop - Fréjus, June 23-26, 2013
Motivation and introduction
Efficient arithmetic in FPGAs
Pairing on Edwards curves
Tate pairing coprocessor
Results and conclusions

Pairing Based Cryptography
Prime fields

Pairing Based Cryptography

One-Round Key Agreement

Encryption

Cert(BOB, P_{BOB})_{TA}

PAIRING

Traditional PKC

A
B

C

A

C

B

A

C

B

Note: Cert − Certificate, TA − Trust Authority

PAIRING

Encryption

ALICE BOB

ID_{BOB}

P_{TA}

M

E

C

ALICE BOB

P_{TA}

M

E

C

ALICE BOB

Cryptoarchi ’13 M.Rogawski, E.Homsirikamol, K. Gaj
Motivation and introduction
Efficient arithmetic in FPGAs
Pairing on Edwards curves
Tate pairing coprocessor
Results and conclusions

Pairing Based Cryptography
Prime fields

Pairing Based Cryptography

**ADDER**

**MULTIPLIER**

Note: signed size(unsigned size)

Note: $M = ((A_{hi} \times B_{hi}) \times 2^{18} + (A_{hi} \times 2^{9} \times B_{lo}) + (A_{lo} \times B_{lo}) + (A_{hi} \times 2^{18} \times B_{lo}))$
Prime Fields

- Pairing transformations can be defined over multiple fields: binary - $\text{GF}(2^n)$, ternary - $\text{GF}(3^m)$, and prime fields - $\text{GF}(p)$
- Binary and ternary fields are generally hardware-friendly
- Prime fields are generally better for software implementations and for cross-platform solutions
- The National Security Agency (NSA Suite B Cryptography) and eCRYPT II recommend prime fields

Scope of this work

Efficient implementation of Pairing Based Cryptosystems over prime fields using internal resources of modern FPGAs, such as fast carry chains (carry logic) and DSP units.
Motivation and introduction
Efficient arithmetic in FPGAs
Pairing on Edwards curves
Tate pairing coprocessor
Results and conclusions

Field operations
Solinas primes and Barrett reduction

Hierarchy of Operations in Pairing Based Cryptography (PBC)

Cryptographic protocols and schemes

- Curve Operations
  - Scalar Multiplication

- Bilinear Operations
  - Pairings

- Group Operations
  - Point Addition
  - Point Doubling

- Extension Field Operations
  - Multiplication
  - Squaring

- Field Operations
  - Multiplication
  - Squaring
  - Addition
  - Subtraction

Hardware architecture recipe:
Optimizations on every level CAN NOT be conducted totally independently!
Novel Hybrid high-radix carry save adder with parallel prefix Kogge-Stone network

Functionality: $A + B = S + C = cout, R$

Block $N-1$
- $a(N*w-1)\ldots a(N*(w-1))$
- $b(N*w-1)\ldots b(N*(w-1))$
- $s(N*w-1)\ldots s(N*(w-1))$
- $c(N*w)$

Block 1
- $a(2*w-1)\ldots a(w)$
- $b(2*w-1)\ldots b(w)$
- $s(2*w-1)\ldots s(w)$
- $c(2*w)$

Block 0
- $a(w-1)\ldots a(0)$
- $b(w-1)\ldots b(0)$
- $s(w-1)\ldots s(0)$
- $c(w)$

High-Radix Carry Save Form
- $A = \{a(N*w-1), \ldots, a(0)\}$
- $B = \{b(N*w-1), \ldots, b(0)\}$
- $R = \{r(N*w-1), \ldots, r(0)\}$
- $S = \{s(N*w-1), \ldots, s(0)\}$
- $C = \{c(N*w), 0^{w-1}, c((N-1)*w), 0^{w-1} \ldots c(w), 0^{w} \}$

Carry Projection Unit
- $g(N-1) \rightarrow g(N-2) \rightarrow \ldots \rightarrow g(0)$
- $p(N-1) \rightarrow p(N-2) \rightarrow \ldots \rightarrow p(0)$

Kogge-Stone Adder's Parallel Prefix Network
Generic modular adder

\[ R = A + B \mod P, \quad R = \begin{cases} A + B - P, & \text{if } A + B \geq 2^n \lor A + B - P \geq 0 \\ A + B, & \text{otherwise} \end{cases} \]

\[ R = A - B \mod P, \quad R = \begin{cases} A - B + P, & \text{if } A - B < 0 \\ A - B, & \text{otherwise} \end{cases} \]
**Novel modular adder/subtractor**

**Field operations**
- Solinas primes and Barrett reduction

**Motivation and introduction**
- Efficient arithmetic in FPGAs
- Pairing on Edwards curves
- Tate pairing coprocessor

**Results and conclusions**

**Field operations**

**Functionality:** $(A + B) \mod P = R$

$A = \{a(N-1), ..., a(0)\}$, $B = \{b(N-1), ..., b(0)\}$, $P = \{p(N-1), ..., p(0)\}$, IP = two’s complement of $P$

80-bit: $n=521$ bits, $N=31$ words
120-bit: $n=1264$ bits, $N=75$ words
128-bit: $n=1493$ bits, $N=88$ words

w=17 bits
Novel Multiply-and-Add DSP-based multiplier 1/3

Functionality: $A \times B = (RR, RC)$, where $A = \{a(N-1), ..., a(0)\}$, $B = \{b(N-1), ..., 0\}$, $RR = \{rr(2M-1), ..., rr(0)\}$, $RC = \{rc(2M-1), ..., rc(0)\}$

Radix 2$^{17}$ operations

Radix 2$^{24}$ operations

80-bit: $N=31, M=22$
120-bit: $N=75, M=53$
128-bit: $N=88, M=62$
Novel Multiply-and-Add DSP-based multiplier 2/3

Three operational phases of the selected processing element:

Phase I: clock cycle 0
- "0" to 24
- ss(i) to cc(i)
- "0" to 24
- 1 to 24
- rc(i+M) to rr(i+M–1)

Phase II: clock cycle 1..M–2
- "0" to 24
- ss(i) to cc(i)
- "0" to 24
- 1 to 24
- rc(i+M) to rr(i+M–1)

Phase III: clock cycle M–1
- "0" to 24
- ss(i) to cc(i)
- "0" to 24
- 1 to 24
- rc(i+M) to rr(i+M–1)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Xilinx Virtex-6</th>
<th>Altera Stratix IV &amp; V</th>
</tr>
</thead>
<tbody>
<tr>
<td>#bits of A processed per clock cycle</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>#bits of B processed per clock cycle</td>
<td>24</td>
<td>36</td>
</tr>
<tr>
<td>#clock cycles per multiplication</td>
<td>$\lceil \frac{n}{24} \rceil$</td>
<td>$\lceil \frac{n}{36} \rceil$</td>
</tr>
<tr>
<td>#DSP units</td>
<td>$\lceil \frac{n}{17} \rceil$</td>
<td>$\lceil \frac{n}{36} \rceil$</td>
</tr>
<tr>
<td>Meaning of DSP unit</td>
<td>DSP48E1 slice</td>
<td>Half-DSP block</td>
</tr>
</tbody>
</table>
Novel double speed mode: One multiplication using two multipliers

Let $A, B$ be $M \times w$-bit numbers, and $B = B_H \times 2^{\frac{w \times M}{2}} + B_L$, then the multiplication of $A$ and $B$ can be computed as follows:

$A \times B = A \times (B_H \times 2^{\frac{w \times M}{2}} + B_L) = A \times B_H \times 2^{\frac{w \times M}{2}} + A \times B_L$.

Very similar idea to BiPartite, Kaihara et al. [CHES’05], TiPartite multiplication, Sakiyama et al. [Integration’11]

When to use it? When we conduct the computations of a single multiplication, but we have two multipliers available! Or two multiplications and we have four multipliers available!
Arithmetic for Special Primes

- Reductions modulo $2^n+1$ and modulo $2^n-1$ are very efficient. (Problem: Not every number of this form is prime!)
- Primes of a form $(2^a \pm 2^b \pm 1$ and $2^a \pm 2^b \pm 2^c \pm 1$) were introduced by Solinas [NSA’99], Solinas prime’s arithmetic is recommended by NIST for digital signature schemes [FIPS-186]!

Comment:

But it is not applicable for all primes in Solinas form!
(e.g.: $2^{520} + 2^{363} - 2^{360} - 1$)
Novel solution: Barrett-based reductor for Solinas primes

Algorithm 1 Barrett modular reduction [Crypto’86]

Require: \( x = (x_{2n-1} \ldots x_1, x_0)_2, p = (p_{n-1} \ldots p_1, p_0)_2 \)
\( (p_{n-1} \neq 0), \mu = \lfloor 2^{2n} / p \rfloor \)
Ensure: \( r = x \mod p \)

1: \( q_1 \leftarrow \lfloor x / 2^{n-1} \rfloor \)
2: \( q_2 \leftarrow q_1 \ast \mu \)
3: \( q_3 \leftarrow \lfloor q_2 / 2^{n+1} \rfloor \)
4: \( r_1 \leftarrow x \mod 2^{n+1} \)
5: \( r_2 \leftarrow q_3 \ast p \mod 2^{n+1} \)
6: \( r \leftarrow r_1 - r_2 \)
7: if \( r < 0 \) then
8: \( r \leftarrow r + 2^{n+1} \)
9: end if
10: while \( r \geq p \) do
11: \( r \leftarrow r - p \)
12: end while
13: return \( r \)

Comment: \( p \) can be chosen in such a way, that \( \mu = 2^{a_t-1} + 2^{a_{t-2}} + \ldots + 2^{a_1} + 2^0 \), where \( t \) is a relatively small number.
Definition: Elliptic curve over $GF(p)$ is a set of points fulfilling equation of the curve and special point $\infty$.

$$y^2 = x^3 + x + 1 \mod p \quad (p=23)$$

$\exists P(\text{generator}) : P, 2P, 3P, \ldots mP = \infty$
Edwards curves + application

- Edwards curve [AmericanMath’07] is a set of points which fulfill equation \( x^2 + y^2 = 1 + dx^2y^2 \mod p \)
- Generalized form \( ax^2 + y^2 = 1 + dx^2y^2 \mod p - a\)-twisted Edwards curves proposed by Bernstein et al. [AfricaCrypt’08]
- Extended projective formulae defined by Hisil et al. [AsiaCrypt’08]
- An elliptic curve is called supersingular, if its number of points is equal to \( p + 1 \)
- Edwards curves together with special prime number P-25519 were adopted to digital signatures by Bernstein et al. [CHES’11]
What is Pairing?

Pairing is a mathematical transformation which takes two arguments: two elliptic curve points \( P \) and \( Q \) from two algebraic groups \( G_1 \) and \( G_2 \) and it produces an element of the third algebraic group \( G_T \).

The most important properties of these \( G_1 \times G_2 \rightarrow G_T \) functions are:

- **bilinearity** \( \forall a, b \in \mathbb{Z}_p: e(aP, bQ) = e(aP, Q)^b = e(P, bQ)^a = e(P, Q)^{ab} \)

- non-degeneracy (function \( e(P, Q) \) never returns '1'), and

- efficiency in computations.

**Pairing on Edwards curves**

- Pairing on twisted supersingular \( k = 2 \) Edwards curve was defined by Das and Sarkar [Pairing'08]
- Pairing on ordinary Edwards curves was defined by Arene et al. [Journal of Cryptology'09]
- So far NO hardware architectures or software implementations for pairing on Edwards curves reported in literature
Security concerns:

- $p$ must be a large prime number, and $p \equiv 3 \pmod{4}$
- $p + 1$ must have a large prime divisor $r$ - the discrete logarithm problem in the elliptic curves must be hard (Pollard rho)
- Edwards curves parameters must be $a = 1$, $d = p - 1$
- $k$, so called embedding degree, is the smallest number, such that $p^{k-1}$ is divisible by $r$
- For aforementioned parameters, Edwards curve has $p + 1$ points, embedding degree $k = 2$, and it is called supersingular curve
- $p$ must be a large prime, the discrete logarithm problem in the $p^2$ must be hard (functional field sieve)
How to generate secure and computationally-friendly prime numbers? - part II

- $p$ and $r$ can have a special form - Menezes and Koblitz [ePrint’06] recommended Solinas primes [NSA’99] ($2^a \pm 2^b \pm 1$)
- **Observation:** Barrett reduction [Crypto’86] requires multiplication by constants: $p$ and $\mu = \lfloor \frac{2^{2\cdot n}}{p} \rfloor$, and $2^n > p$, and $n$ - number of bits of $p$.
- we search such for $p$ such that $\mu = 2^{a_t-1} \pm ... \pm 1$ and $t$ is relatively a small number ($t < 30$).
- we were looking for $r$ with a very low Hamming weight ($< 5\%$)

**Comment:**
The GMP library-based software implementation of the parameters generation algorithm requires significant amount of time to find friendly numbers
Parameters used in our work

<table>
<thead>
<tr>
<th>Security</th>
<th>Field order - ( p )</th>
<th>Prime divisor - ( r )</th>
<th># terms of ( \mu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>80-bits</td>
<td>( 2^{520} + 2^{363} - 2^{360} - 1 )</td>
<td>( 2^{160} + 2^3 - 1 )</td>
<td>12</td>
</tr>
<tr>
<td>120-bits</td>
<td>( 2^{1263} + 2^{1037} - 2^{1005} - 1 )</td>
<td>( 2^{258} + 2^{32} - 1 )</td>
<td>28</td>
</tr>
<tr>
<td>128-bits</td>
<td>( 2^{1492} + 2^{1237} - 2^{1224} - 1 )</td>
<td>( 2^{268} + 2^{13} - 1 )</td>
<td>30</td>
</tr>
<tr>
<td>191-bits</td>
<td>( 2^{3955} + 2^{3581} + 2^{3573} - 1 )</td>
<td>( 2^{382} + 2^8 - 1 )</td>
<td>21</td>
</tr>
</tbody>
</table>

Observations:
- The multiplication by \( p \) and \( \mu \) can be replaced by multi-operand addition!
- The prime divisor \( r \left( 2^a + 2^b - 1 \right) \) has always a form of **10..01..1** (computationally cheaper - check next slide!).
Motivation and introduction

Efficient arithmetic in FPGAs

Pairing on Edwards curves

Tate pairing coprocessor

Results and conclusions

General algorithm for the modified Tate pairing

**Algorithm 2**  Miller's algorithm for computing modified Tate pairing

Require: Points $P$ and $\phi(Q)$, prime divisor $r = (r_{l-1} \ldots r_0)$, field order $p$, and embedding degree $k$, $h_{P,Q}$ a rational function

Ensure: $F = e(P, \phi(Q))$

1: $F = 1, R = P$
2: for $i = l-2$ downto 0 do
3:   $G \leftarrow h_R, R(\phi(Q))$ and $R = 2R$ /* Algorithm 3: 14 multiplications */
4:   $F = F^2 * G$ /* Algorithm 5 and 6: 2+4 multiplications */
5: if $r_i = 1$ then
6:   $G \leftarrow h_R, P(\phi(Q))$ and $R = R + P$ /* Algorithm 4: 24 multiplications */
7:   $F = F * G$ /* Algorithm 5: 2 multiplications */
8: end if
9: end for
10: return $F \leftarrow F^{p^k - 1}$ /* Algorithm 7: next slide */

What if for the substantial number of $P$ and $Q$ the result of $e(P, Q) = 1$?

Distortion maps! $\phi(Q) = (x_Q i, \frac{1}{y_Q})$, where $i^2 = -1$. Consequently, $F$ and $G$ are the complex numbers.

Other names: twists or operations in the extension field $x^2 + 1$, in this case.
Final exponentiation (Alg. 7) - novel approach

- Traditional optimization method, the Frobenius mapping is not applicable to supersingular curves. $F^{(p-1)}$ and $F^{-1}$ are equally difficult!
- The fixed exponent $e = \frac{p^2 - 1}{r}$, after Booth recoding can be represented as five term Solinas prime (e.g.: 128-bit: $2^{2716} + 2^{2461} - 2^{2449} + 2^{2448} - 2^{1225}$)
- The computation of the Final exponentiation
$\left( F^e = F^{2^{a_1} + 2^{a_2} - 2^{a_3} + 2^{a_4} - 2^{a_5}} = \frac{F^{2^{a_1}} F^{2^{a_2}} F^{2^{a_4}}}{F^{2^{a_3}} F^{2^{a_5}}}) \right)$: $a_1$ complex squarings and the three complex multiplications of five intermediate values
- Two modular multiplications (complex squaring) can be computed using 4 multipliers working in the double speed mode

Final result reconstruction:

$$F \leftarrow \frac{R_u}{R_d} \leftarrow \frac{x'i + y}{i + z} \quad (vi + z)^{-1} = (v'i + z') = \begin{cases} v' = \frac{-v}{z^2 + v^2} \\ z' = \frac{z}{z^2 + v^2} \end{cases}$$
Motivation and introduction
Efficient arithmetic in FPGAs
Pairing on Edwards curves
Tate pairing coprocessor
Results and conclusions

The overview of novel coprocessor block diagram

The overview of novel coprocessor block diagram

Number of 17-bit words:
N = 31 (80-bit), 75 (120-bit), 88 (128-bit)
A = \{a(N−1), ..., a(0)\}
B = \{b(N−1), ..., b(0)\}
R = \{r(N−1), ..., r(0)\}
FPGA-based hardware architectures - preliminary results Stratix V

- 80-bit security coprocessor: logic - 41471 ALM, memory - 552k, 120 DSPs, 263 MHz, latency: \(133\mu s\)
- 120-bit security coprocessor: logic - 120628 ALM, memory - 1327k, 288 DSPs, 257 MHz, latency: \(541\mu s\)
- 128-bit security coprocessor: logic - 137484 ALM, memory - 1432k, 336 DSPs, 242 MHz, latency: \(697\mu s\)

Reference software implementation results:

- GNU Multiple Precision Arithmetic Library
- Testing platform: Mac OS X 10.6.8, CPU: Intel Core i7 2.8GHz, 8GB 1067 MHz DDR3
- **80-bit**: 5.09ms, **120-bit**: 29.41ms, **128-bit**: 37.11ms
## Speed records for the range of 120-128-bits security for the pairing transformations over prime fields

<table>
<thead>
<tr>
<th>Publication</th>
<th>Curve Type</th>
<th>Security</th>
<th>Type</th>
<th>Platform</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>twisted supersingular Edwards</td>
<td>120-bit</td>
<td>Tate</td>
<td>Stratix V</td>
<td>0.54ms</td>
</tr>
<tr>
<td>Cheung et al. [CHES’11]</td>
<td>Barreto-Naehring</td>
<td>126-bit</td>
<td>Opt.-Ate</td>
<td>Virtex-6</td>
<td>0.57ms</td>
</tr>
<tr>
<td>This work</td>
<td>twisted supersingular Edwards</td>
<td>128-bit</td>
<td>Tate</td>
<td>Stratix V</td>
<td>0.70ms</td>
</tr>
<tr>
<td>This work</td>
<td>twisted supersingular Edwards</td>
<td>120-bit</td>
<td>Tate</td>
<td>Stratix IV</td>
<td>0.70ms</td>
</tr>
<tr>
<td>Beuchat et al. [Pairing’10]</td>
<td>Barreto-Naehring</td>
<td>126-bit</td>
<td>Opt.-Ate</td>
<td>Core i7 2.8</td>
<td>0.83ms</td>
</tr>
<tr>
<td>This work</td>
<td>twisted supersingular Edwards</td>
<td>128-bit</td>
<td>Tate</td>
<td>Stratix IV</td>
<td>0.88ms</td>
</tr>
<tr>
<td>This work</td>
<td>twisted supersingular Edwards</td>
<td>120-bit</td>
<td>Tate</td>
<td>Virtex-6</td>
<td>1.05ms</td>
</tr>
<tr>
<td>Cheung et al. [CHES’11]</td>
<td>Barreto-Naehrig</td>
<td>126-bit</td>
<td>Opt.-Ate</td>
<td>Stratix III</td>
<td>1.07ms</td>
</tr>
<tr>
<td>Fan et al. [Computers’11]</td>
<td>Barreto-Naehrig</td>
<td>128-bit</td>
<td>Opt.-Ate</td>
<td>Virtex-6</td>
<td>1.36ms</td>
</tr>
<tr>
<td>This work</td>
<td>twisted supersingular Edwards</td>
<td>128-bit</td>
<td>Tate</td>
<td>Virtex-6</td>
<td>1.05ms</td>
</tr>
<tr>
<td>Fan et al. [Computers’11]</td>
<td>Barreto-Naehrig</td>
<td>128-bit</td>
<td>Ate</td>
<td>Virtex-6</td>
<td>1.60ms</td>
</tr>
<tr>
<td>Cheung et al. [CHES’11]</td>
<td>Barreto-Naehrig</td>
<td>126-bit</td>
<td>Opt.-Ate</td>
<td>Cyclone II</td>
<td>1.93ms</td>
</tr>
</tbody>
</table>

**Comment:**

The fastest reported pairing coprocessor over prime fields for security level above 120 bits!
Major Contributions

- Novel, low latency, generic, optimized for fast carry-chains (FPGA), hybrid adder for big numbers (thousand of bits and more)
- Solinas primes-based, DSP-oriented, modular arithmetic architectures for addition, subtraction and multiplication
- First hardware architectures for 80, 120 and 128-bit pairing on Edwards curves
- Our coprocessor (on Stratix V) computes 120 and 128-bit secure pairing over prime field in less than 0.54 and 0.70 ms, respectively. It is the fastest pairing implementation over prime fields in this security range