High-performance Elliptic Curve Cryptography by Using the CIOS Method for Modular Multiplication

A*mine Mrabet*, Nadia El-Mrabet, **Ronan Lashermes**, Jean-Baptiste Rigaud, Belgacem Bouallegue, Sihem Mesnager and Mohsen Machhout

September 2016











Introduction

- Public key cryptography is still costly (computing resources).
- Elliptic Curve Cryptography has a better cost/security trade-off w.r.t. RSA.
- We can still reduce the cost with better hardware architectures.





- ECC
- Montgomery Modular Multiplication

Our architecture

- Basics
- PEs
- Scheduling
- Resources

3 Results

- Results
- Conclusion



Our architecture

Results

Elliptic Curve Cryptography (ECC)

Why?

Elliptic curves allow to define groups with a hard Discrete Logarithm Problem. In the general case, cracking methods are far less efficient than for RSA.



Our architecture

Results

Elliptic Curve Cryptography (ECC)

Why?

Elliptic curves allow to define groups with a hard Discrete Logarithm Problem. In the general case, cracking methods are far less efficient than for RSA.

How? (simplified)

Let p > 3 a big prime, $E(\mathbb{F}_p)$ is the (short Weierstrass) elliptic curve

$$E(\mathbb{F}_p): y^2 = x^3 + ax + b,$$

where $x, y, a, b \in \mathbb{F}_p$ with $4a^3 + 27b^2 \neq 0$.



Arithmetic 0000000 ECC EC Group Our architecture

Results

The points (x, y) on the curve define an abelian group together with the **point at infinity** 0_{∞} , the neutral element for addition.



Mrabet et al.



Arithmetic 000000 ECC EC Group Our architecture

Results

The points (x, y) on the curve define an abelian group together with the **point at infinity** 0_{∞} , the neutral element for addition.



Jacobian coordinates

The triple (x : y : z) can be mapped to $(x/z^2, y/z^3)$ if $z \neq 0$. If z = 0 it is 0_{∞} . The curve becomes: $y^2 = x^3 + axz^4 + bz^6$.

Efficient MMM for ECC

Mrabet et al.

September 2016



Our architecture

Results

Operations in Jacobian coordinates (a = 0, points $\neq 0_{\infty}$)

Doubling (7S+5M+13A)

$$T(X_{T} : Y_{T} : Z_{T}) = 2 \cdot Q(X_{Q} : Y_{Q} : Z_{Q}).$$

$$X_{T} = 9X_{Q}^{4} - 8X_{Q}Y_{Q}^{2},$$

$$Y_{T} = 3X_{Q}^{2}(4X_{Q}Y_{Q} - X_{T}) - 8Y_{Q}^{4},$$

$$Z_{T} = 2Y_{Q}Z_{Q}.$$

Efficient MMM for ECC



Our architecture

Results

Operations in Jacobian coordinates (a = 0, points $\neq 0_{\infty}$)

Doubling (7S+5M+13A)

$$T(X_{T} : Y_{T} : Z_{T}) = 2 \cdot Q(X_{Q} : Y_{Q} : Z_{Q}).$$

$$X_{T} = 9X_{Q}^{4} - 8X_{Q}Y_{Q}^{2},$$

$$Y_{T} = 3X_{Q}^{2}(4X_{Q}Y_{Q} - X_{T}) - 8Y_{Q}^{4},$$

$$Z_{T} = 2Y_{Q}Z_{Q}.$$

Addition (4S + 14M + 6A)

$$\begin{aligned} R &= T + Q. \\ X_R &= (2Y_Q Z_T^3 - 2Y_T)^2 - 4(X_Q Z_T^2 - X_T)^3 - 8(X_Q Z_T^2 - X_T)^2 X_T, \\ Y_R &= \\ (2Y_Q Z_T^3 - 2Y_T)(4X_T (X_Q Z_T^2 - X_T) - X_R) - 8Y_T (X_Q Z_T^2 - X_T)^3, \\ Z_R &= 2Z_T (X_Q Z_T^2 - X_T). \end{aligned}$$

Efficient MMM for ECC



Results

Montgomery Modular Multiplication (MMM)

MMM

MMM provides an efficient way for modular multiplication mod p (noted \cdot): there is no division by p.





Results

Montgomery Modular Multiplication (MMM)

MMM

MMM provides an efficient way for modular multiplication mod p (noted \cdot): there is no division by p.

Residue

Let $a, b, R \in \mathbb{F}_p$ where R is Montgomery's residue. $a' = aR \mod p$ is said to be a in Montgomery's form. $a \cdot b = abR^{-1} \mod p$, as a consequence $a' \cdot b' = aRbRR^{-1} \mod p = abR \mod p = (ab)'$.



Montgomery Modular Multiplication (MMM)

MMM

MMM provides an efficient way for modular multiplication mod p (noted \cdot): there is no division by p.

Residue

Let $a, b, R \in \mathbb{F}_p$ where R is Montgomery's residue. $a' = aR \mod p$ is said to be a in Montgomery's form. $a \cdot b = abR^{-1} \mod p$, as a consequence $a' \cdot b' = aRbRR^{-1} \mod p = abR \mod p = (ab)'$.

Conversion

Field values are converted in Montgomery's form at the beginning of the computation and back to normal at the end.

Efficient MMM for ECC

Mrabet et al.



Our architecture

Results

How to compute MMM?

Koç's multiword CIOS algorithm

```
Algorithm 2: CIOS algorithm for Montgomery multiplication [11]
    Input: p < 2^K, p' = -p^{-1} \mod 2^w, w, s, K = s \cdot w: bit length, R = 2^K, a, b < p
    Output: a \cdot b \cdot R^{-1} \mod p
 1 T \leftarrow 0;
 2 for i \leftarrow 0 to s - 1 do
         C \leftarrow 0;
 3
         for i \leftarrow 0 to s - 1 do
 4
              (C,S) \leftarrow T[j] + a[i] \cdot b[j] + C
 5
          T[j] \leftarrow S
 6
 7
         (C, S) \leftarrow T[s] + C
         T[s] \leftarrow S
 8
         T[s+1] \leftarrow C
 9
         m \leftarrow T[0] \cdot p' \mod 2^w
10
         (C, S) \leftarrow T[0] + m \cdot p[0]
11
         for i \leftarrow 1 to s - 1 do
12
              (C,S) \leftarrow T[j] + m \cdot p[j] + C
13
          T[j-1] \leftarrow S
14
         (C, S) \leftarrow T[s] + C
15
         T[s-1] \leftarrow S
16
         T[s] \leftarrow T[s+1] + C
17
18 return T:
```



Our architecture

Results

CIOS details



18 return T;



Benefits

Our architecture

Results

- Low memory footprint,
- apart from some precomputations (p', R...), easy to change p and operand sizes,
- neat structure, without divisions, easy to implement in hardware.



Basics



Our architecture

Results

Here, each operation takes 1 unit of time. Let's compute $r = a \cdot b + b + c$.

Sequential

Time	•	+	Operations
1	х		$t1 = a \cdot b$
2		x	$t^2 = b + c$
3		х	r = t1 + t2

Efficient MMM for ECC



Basics



Our architecture

Results

Here, each operation takes 1 unit of time. Let's compute $r = a \cdot b + b + c$.

Sequential

Time	•	+	Operations
1	х		$t1 = a \cdot b$
2		x	$t^2 = b + c$
3		x	r = t1 + t2

Parallel

Time	•	+	Operations
1	x	х	$t1 = a \cdot b, \ t2 = b + c$
2		х	r = t1 + t2

Efficient MMM for ECC

Mrabet et al.

September 2016



Basics

Basics - 2

Here, each operation takes 1 unit of time. Let's compute $r = a \cdot b + b + c$.

Atomic				
Latency	Throughput	•	+	Operations
2	0.5	1	2	$r = a \cdot b + b + c$

The choice of operations and how they are chained together is called **scheduling**.

Efficient MMM for ECC



Basics

Basics - 2

 Results

Here, each operation takes 1 unit of time. Let's compute $r = a \cdot b + b + c$.

Atomic

Latency	Throughput	•	+	Operations
2	0.5	1	2	$r = a \cdot b + b + c$

Pipelined

Latency	Throughput	•	+	Operations
$2 + \epsilon$	0.5	1	1	$1: t1 = a \cdot b, t2 = b + c, 2: r = t1 + t2$
$2 + \epsilon$	1	1	2	$1: t1 = a \cdot b, t2 = b + c, 2: r = t1 + t2$

The choice of operations and how they are chained together is called **scheduling**.

Efficient MMM for ECC



Basics

Systolic arrays

Our architecture

Results

A systolic array is an architecture both parallel and pipelined. To create such an architecture, we have to identify small **Processing Elements (PEs)** (no control flow logic).



PEs

Our architecture

Results

Where is Waldo the PE?

Algorithm 2: CIOS algorithm for Montgomery multiplication [11]

```
Input: p < 2^{K}, p' = -p^{-1} \mod 2^{w}, w, s, K = s \cdot w: bit length, R = 2^{K}, a, b < p
     Output: a \cdot b \cdot R^{-1} \mod p
 1 T \leftarrow 0;
 2 for i \leftarrow 0 to s - 1 do
          C \leftarrow 0:
 3
          for i \leftarrow 0 to s - 1 do
 4
                (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C
 5
           T[j] \leftarrow S
 6
          (C,S) \leftarrow T[s] + C
 7
          T[s] \leftarrow S
 8
          T[s+1] \leftarrow C
 9
          m \leftarrow T[0] \cdot p' \mod 2^w
10
          (C,S) \leftarrow T[0] + m \cdot p[0]
11
12
          for i \leftarrow 1 to s - 1 do
               (C,S) \leftarrow T[j] + m \cdot p[j] + C
13
           T[j-1] \leftarrow S
14
          (C,S) \leftarrow T[s] + C
15
         T[s-1] \leftarrow S
16
          T[s] \leftarrow T[s+1] + C
17
```

18 return T;





```
Input: p < 2^{K}, p' = -p^{-1} \mod 2^{w}, w, s, K = s \cdot w: bit length, R = 2^{K}, a, b < p
     Output: a \cdot b \cdot R^{-1} \mod p
 1 T \leftarrow 0;
 2 for i \leftarrow 0 to s - 1 do
          C \leftarrow 0:
 3
          for i \leftarrow 0 to s - 1 do
 4
               (C,S) \leftarrow T[j] + a[i] \cdot b[j] + C
 5
                                                                α
             T[i] \leftarrow S
 6
 7
          (C, S) \leftarrow T[s] + C
 8
          T[s] \leftarrow S
          T[s+1] \leftarrow C
 9
          m \leftarrow T[0] \cdot p' \mod 2^w
10
          (C,S) \leftarrow T[0] + m \cdot p[0]
11
          for i \leftarrow 1 to s - 1 do
12
              (C,S) \leftarrow T[j] + m \cdot p[j] + C
13
           T[j-1] \leftarrow S
\mathbf{14}
          (C, S) \leftarrow T[s] + C
15
         T[s-1] \leftarrow S
16
          T[s] \leftarrow T[s+1] + C
17
18 return T;
```



PEs $lpha_{f}$

Our architecture

Results

```
Input: p < 2^{K}, p' = -p^{-1} \mod 2^{w}, w, s, K = s \cdot w: bit length, R = 2^{K}, a, b < p
     Output: a \cdot b \cdot R^{-1} \mod p
 1 T \leftarrow 0;
 2 for i \leftarrow 0 to s - 1 do
          C \leftarrow 0:
 3
          for i \leftarrow 0 to s - 1 do
 4
               (C,S) \leftarrow T[j] + a[i] \cdot b[j] + C
 5
                                                                 α
            T[i] \leftarrow S
 6
          (C, S) \leftarrow T[s] + C
 7
          T[s] \leftarrow S
                                                                 \alpha_{\rm f}
 8
          T[s+1] \leftarrow C
 9
          m \leftarrow T[0] \cdot p' \mod 2^w
10
          (C,S) \leftarrow T[0] + m \cdot p[0]
11
          for i \leftarrow 1 to s - 1 do
12
              (C,S) \leftarrow T[j] + m \cdot p[j] + C
13
           T[j-1] \leftarrow S
\mathbf{14}
          (C, S) \leftarrow T[s] + C
15
          T[s-1] \leftarrow S
16
          T[s] \leftarrow T[s+1] + C
17
18 return T;
```



 β

```
Input: p < 2^{K}, p' = -p^{-1} \mod 2^{w}, w, s, K = s \cdot w: bit length, R = 2^{K}, a, b < p
     Output: a \cdot b \cdot R^{-1} \mod p
 1 T \leftarrow 0;
 2 for i \leftarrow 0 to s - 1 do
          C \leftarrow 0:
 3
          for i \leftarrow 0 to s - 1 do
 4
                (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C
 5
                                                                  α
            T[i] \leftarrow S
 6
          (C, S) \leftarrow T[s] + C
 7
          T[s] \leftarrow S
                                                                 \alpha_{\rm f}
 8
          T[s+1] \leftarrow C
 9
          m \leftarrow T[0] \cdot p' \mod 2^w
10
                                                                  β
          (C,S) \leftarrow T[0] + m \cdot p[0]
11
          for i \leftarrow 1 to s - 1 do
12
               (C,S) \leftarrow T[j] + m \cdot p[j] + C
13
           T[j-1] \leftarrow S
\mathbf{14}
          (C, S) \leftarrow T[s] + C
15
          T[s-1] \leftarrow S
16
          T[s] \leftarrow T[s+1] + C
17
18 return T;
```



PEs γ

```
Input: p < 2^{K}, p' = -p^{-1} \mod 2^{w}, w, s, K = s \cdot w: bit length, R = 2^{K}, a, b < p
     Output: a \cdot b \cdot R^{-1} \mod p
 1 T \leftarrow 0;
 2 for i \leftarrow 0 to s - 1 do
          C \leftarrow 0:
 3
          for i \leftarrow 0 to s - 1 do
 4
                (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C
 5
                                                                  α
              T[i] \leftarrow S
 6
          (C, S) \leftarrow T[s] + C
 7
                                                                  \alpha_{\rm f}
 8
          T[s] \leftarrow S
          T[s+1] \leftarrow C
 9
          m \leftarrow T[0] \cdot p' \mod 2^w
10
                                                                   β
          (C,S) \leftarrow T[0] + m \cdot p[0]
11
          for i \leftarrow 1 to s - 1 do
12
                (C,S) \leftarrow T[j] + m \cdot p[j] + C
13
                                                                  γ
              T[i-1] \leftarrow S
\mathbf{14}
          (C, S) \leftarrow T[s] + C
15
          T[s-1] \leftarrow S
16
          T[s] \leftarrow T[s+1] + C
17
18 return T;
```



PEs

Algorithm 2: CIOS algorithm for Montgomery multiplication [11] **Input:** $p < 2^{K}$, $p' = -p^{-1} \mod 2^{w}$, w, s, $K = s \cdot w$: bit length, $R = 2^{K}$, a, b < p**Output:** $a \cdot b \cdot R^{-1} \mod p$

α

 α_{f}

β

γ

γ_f

1
$$T \leftarrow 0$$
;
2 for $i \leftarrow 0$ to $s - 1$ do

3 4

 $\mathbf{12}$ 13

 $\mathbf{14}$ 15

16

$$\begin{bmatrix} C \leftarrow 0; \\ \mathbf{for} \ i \leftarrow 0 \ to \ s - 1 \ \mathbf{do} \end{bmatrix}$$

$$\begin{array}{c} \mathsf{S} \\ \mathsf{G} \\ \mathsf$$

$$\begin{array}{c} (C,S) \leftarrow T[0] + m \cdot p[0] \\ \hline \mathbf{for} \ j \leftarrow 1 \ to \ s - 1 \ \mathbf{do} \\ (C,S) \leftarrow T[j] + m \cdot p[j] + C \\ T[j-1] \leftarrow S \\ \hline (C,S) \leftarrow T[s] + C \end{array}$$



Efficient MMM for ECC



Scheduling

S=8, Time=1



Results



Algorithm 2: CIOS algorithm fo	r Montgomery multiplication [11]
Input: $p < 2^{K}$, $p' = -p^{-1} \mod 2^{w}$, w	, s , $K = s \cdot w$: bit length, $R = 2^K, a, b < p$
Output: $a \cdot b \cdot R^{-1} \mod p$	
$1 T \leftarrow 0;$	
2 for $i \leftarrow 0$ to $s - 1$ do	
$3 = C \leftarrow 0;$	
4 for $j \leftarrow 0$ to $s - 1$ do	
5 $(C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$	<i>a</i>
6 $T[j] \leftarrow S$	
T $(C, S) \leftarrow T[s] + C$	
8 $T[s] \leftarrow S$	α _f
9 $T[s + 1] \leftarrow C$	
10 $m \leftarrow T[0]$, $n' \mod 2^{w}$	
11 $(C, S) \leftarrow T[0] + m \cdot n[0]$	β
12 for $i \in I$ to $s = I$ do	
13 $(U, S) \leftarrow T[y] + m \cdot p[y] + C$	Y
14 [10-16-2	
15 $(C, S) \leftarrow T[s] + C$	
16 $T[s-1] \leftarrow S$	Yr
17 $[T[s] \leftarrow T[s + 1] + C$	
18 return T:	

Efficient MMM for ECC

Mrabet et al.

September 2016



Scheduling

S=8, Time=2



Results



Algorithm 2: CIOS algorithm for Montgomery multiplication [11]		
Input: $p < 2^K$, $p' = -p^{-1} \mod 2^w$, w	, s , $K = s \cdot w$: bit length, $R = 2^{K}$, $a, b < p$	
Output: $a \cdot b \cdot R^{-1} \mod p$		
$1 T \leftarrow 0;$		
2 for $i \leftarrow 0$ to $s - 1$ do		
$3 = C \leftarrow 0;$		
4 for $j \leftarrow 0$ to $s - 1$ do		
5 $(C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$		
6 $T[j] \leftarrow S$	u	
T $(C, S) \leftarrow T[s] + C$		
8 $T[s] \leftarrow S$	α _f	
9 $T[s+1] \leftarrow C$		
10 m (T[0] of mud 2 ¹⁰		
10 $(C, S) \leftarrow T[0] + m \cdot n[0]$	β	
(c) b) c a [o] (m p[o]		
12 for $j \leftarrow 1$ to $s - 1$ do		
13 $(C, S) \leftarrow T[j] + m \cdot p[j] + C$	v	
14 $T[j-1] \leftarrow S$		
15 $(C, S) \leftarrow T[s] + C$		
16 $T[s-1] \leftarrow S$	Yr	
17 $T[s] \leftarrow T[s+1] + C$		
18 return T:		

Efficient MMM for ECC

Mrabet et al.

September 2016



Scheduling

S=8, Time=3



Results



Algorithm 2: CIOS algorithm for	Montgomery multiplication [11]
Input: $p < 2^{K}$, $p' = -p^{-1} \mod 2^{w}$, w	s , $K = s \cdot w$: bit length, $R = 2^{K}$, $a, b < p$
Output: $a \cdot b \cdot R^{-1} \mod p$	
$1 T \leftarrow 0;$	
2 for $i \leftarrow 0$ to $s - 1$ do	
$3 \mid C \leftarrow 0;$	
4 for $j \leftarrow 0$ to $s - 1$ do	
5 $(C, S) \leftarrow T[i] + a[i] \cdot b[i] + C$	
$T[d] \leftarrow S$	α
T $(U, S) \leftarrow T[S] + U$	~
8 $T[s] \leftarrow S$	ur .
$y = 1 [s + 1] \leftarrow C$	
10 m T[0], n' mod 2"	
11 $(C, S) \leftarrow T[0] + m \cdot n[0]$	β
10 Family 1 fame 1 day	
14 IOF J (= 1 10 8 - 1 do	
13 $(C, S) \leftarrow T[j] + m \cdot p[j] + C$	v
14 $T[j-1] \leftarrow S$	the second se
15 $(C, S) \leftarrow T[s] + C$	
16 $T[s-1] \leftarrow S$	Yr
17 $T[s] \leftarrow T[s+1] + C$	
18 return T	

Efficient MMM for ECC

Mrabet et al.

September 2016



Scheduling

S=8, Time=4



Results



Algorithm 2: CIOS algorithm for Montgomery multiplication [11]				
Input: $p < 2^K$, $p' = -p^{-1} \mod 2^w$, w	, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$			
Output: $a \cdot b \cdot R^{-1} \mod p$	Output: $a \cdot b \cdot R^{-1} \mod p$			
$1 T \leftarrow 0;$				
2 for $i \leftarrow 0$ to $s - 1$ do				
$3 = C \leftarrow 0;$				
4 for $j \leftarrow 0$ to $s - 1$ do				
5 $(C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$	<i>a</i>			
6 $T[j] \leftarrow S$	u			
7 $(C, S) \leftarrow T[s] + C$	the second se			
$T[s] \leftarrow S$	α,			
9 $T[s+1] \leftarrow C$				
a fa a star a				
10 $m \leftarrow T[0] \cdot p' \mod 2^w$	ß			
11 $(C, S) \leftarrow T[0] + m \cdot p[0]$	P			
12 for $j \leftarrow 1$ to $s - 1$ do				
13 $(C, S) \leftarrow T[i] + m \cdot p[i] + C$				
14 $T[i-1] \leftarrow S$	Y			
15 $(U, S) \leftarrow I[s] + U$	N.			
16 $1 s-1 \leftarrow 3$				
11 $[1 s] \leftarrow 1 s+1 +c$				
18 return T-				

Efficient MMM for ECC

Mrabet et al.

September 2016



Scheduling

S=8, Time=10

Our architecture

Results



Algorithm 2: CIOS algorithm for Montgomery multiplication [11]		
Input: $p < 2^K$, $p' = -p^{-1} \mod 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$		
Output: $a \cdot b \cdot R^{-1} \mod p$		
$1 T \leftarrow 0;$		
2 for $i \leftarrow 0$ to $s - 1$ do		
$3 \mid C \leftarrow 0;$		
4 for $j \leftarrow 0$ to $s - 1$ do		
$(C, S) \leftarrow T[i] + a[i] \cdot b[i] + C$		
TIG + S	α	
1 1 D1 (- D	1	
T $(C, S) \leftarrow T[s] + C$		
8 $T[s] \leftarrow S$	α _f	
9 $T[s + 1] \leftarrow C$		
and all a state		
10 $m \leftarrow T[0] \cdot p \mod 2^n$	ß	
11 $(C, S) \leftarrow T[0] + m \cdot p[0]$		
12 for $j \leftarrow 1$ to $s - 1$ do		
18 $(C, S) \leftarrow T[i] \pm m \cdot n[i] \pm C$		
$T_{ij} = 1 \leftarrow S$	γ	
14 [1] y = 1] (= 3		
15 $(C, S) \leftarrow T[s] + C$		
16 $T[s-1] \leftarrow S$	Yr	
17 $T[s] \leftarrow T[s + 1] + C$		
18 return T:		

Efficient MMM for ECC

Mrabet et al.

September 2016



Scheduling

S=8, Time=10

Our architecture

Results



Algorithm 2: CIOS algorithm	Algorithm 2: CIOS algorithm for Montgomery multiplication [11]		
Input: $p < 2^{K}$, $p' = -p^{-1} \mod 2^{w}$, w, s , $K = s \cdot w$: bit length, $R = 2^{K}$, $a, b < p$			
Output: $a \cdot b \cdot R^{-1} \mod p$			
$1 T \leftarrow 0;$			
2 for $i \leftarrow 0$ to $s - 1$ do			
$C \leftarrow 0;$			
4 for $j \leftarrow 0$ to $s - 1$ do			
5 $(C, S) \leftarrow T[j] + a[i] \cdot b[j] +$	C		
6 $T[j] \leftarrow S$	u u		
7 $(C, S) \leftarrow T[s] + C$			
8 $T[s] \leftarrow S$	α _f		
9 $T[s + 1] \leftarrow C$			
10 $m \leftarrow T[0] \cdot n' \mod 2^{w}$	0		
11 $(C, S) \leftarrow T[0] + m \cdot p[0]$	P		
12 for $i \leftarrow 1$ to $s - 1$ do			
13 $(C, S) \leftarrow T[i] + m \cdot p[i] + 0$	g		
14 $T[j-1] \leftarrow S$	Y		
15 $(C, S) \leftarrow T[s] + C$			
16 $T[s-1] \leftarrow S$	Vr.		
17 $T[s] \leftarrow T[s + 1] + C$			
18 wetness T.			

Efficient MMM for ECC



Scheduling

S=8, Time=13

Our architecture

Results



Algorithm 2: CIOS algorithm for Montgomery multiplication [11]					
Input: $p < 2^{K}$, $p' = -p^{-1} \mod 2^{w}$, w ,	s , $K = s \cdot w$: bit length, $R = 2^{K}$, $a, b < p$				
Output: $a \cdot b \cdot R^{-1} \mod p$					
$1 T \leftarrow 0;$					
2 for $i \leftarrow 0$ to $s - 1$ do					
$C \leftarrow 0;$					
4 for $j \leftarrow 0$ to $s - 1$ do					
5 $(C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$					
6 $T[j] \leftarrow S$	u				
7 $(C, S) \leftarrow T[s] + C$					
8 $T[s] \leftarrow S$	α _f				
9 $T[s+1] \leftarrow C$					
10 $m \leftarrow T[0]$, $n' \mod 2^{w}$					
11 $(C, S) \leftarrow T[0] + m \cdot p[0]$	þ				
12 for $j \leftarrow 1$ to $s - 1$ do					
13 $(C, S) \leftarrow T[j] + m \cdot p[j] + C$	X				
14 $T[j-1] \leftarrow S$	Y				
15 $(C, S) \leftarrow T[s] + C$					
16 $T[s-1] \leftarrow S$	Yr				
17 $T[s] \leftarrow T[s+1] + C$					
18 return T:					

Efficient MMM for ECC

September 2016



Scheduling

S=8, All

Our architecture

Results



Efficient MMM for ECC

Mrabet et al.

September 2016



Resources



Our architecture

Results



Efficient MMM for ECC

Mrabet et al.



Resources



Our architecture

Results



Efficient MMM for ECC

Mrabet et al.

September 2016



Resources



Our architecture

Results

Our architecture requires:

- 3 α,
- 3 γ,
- 1 β,
- 1 α_f ,
- 1 γ_f .



Resources

Regrouping

Our architecture

Results



Efficient MMM for ECC

Mrabet et al.

September 2016



Resources

Block diagram

Our architecture

Results





Our architecture



MMM architecture variants

CIOS (bits per word)	s=8	s=16	s=32	s=64	
K=256	32	16	8	4	
K=512	64	32	16	8	
K=1024	128	64	32	16	
K=2048	256	256 128		32	
$Clock cycles = 3 \times (s + nb)$	33	66	132	264	
Number of cells	6 +3	12 +3	24 +3	48 +3	

Efficient MMM for ECC



Results

Results 0●000

ECC results (Artix-7)

	Slice	DSPs	BRAM	Freq	Slice FF	Slice LUT
NW-8 (256)	3745	33	12	98	8281	9722
NW-16 (256)	3770	34	12	130	8313	9255
NW-8 (512)	7066	92	23	59	16500	20394
NW-16 (512)	7116	60	23	74	16501	19199



Conclusion

Conclusion

- Very efficient Montgomery Modular Multiplication with low latency.
- Give mixed results for a straightforward ECC implementation.
- Yet improvements are still possible: we should not wait the complete ending of an MMM to start the next.
- Should be particularly interesting for latency and throughput.



Conclusion

Thank you!

Our architecture







Efficient MMM for ECC

Mrabet et al.

September 2016



Conclusion

ECC results

et al.	Curve	Device	Lut	Reg	Size (DSP)	Freq.
Bajard 2014	256 any	Kintex-7	4250	3532	1630 slices (46)	281
Bajard 2014	521 any	Kintex-7	7067	5882	2565 slices (91)	266
Bajard classic	256 any	_	7482	4605	– slices (46)	-
Guillermin	256 any	Stratix-2	_	-	9177 ALM (96)	157
Guillermin	512 any	Stratix-2	_	-	17017 ALM (244)	145
Güneysu	256 NIST	Virtex-4	-	-	1715 slices (32)	490
Yuan Ma	256 any	Virtex-4	5740	4876	4655 slices (37)	250
Yuan Ma	256 any	Virtex-5	4177	4792	1725 slices (37)	291
McIvor	256 any	Virtex-II	_	_	15755 slice	39
Us NW-8	256 any	Artix-7	9722	8281	3745 slices (33)	98
Us NW-8	512 any	Artix-7	20394	16500	7066 slices (92)	59
Us NW-16	256 any	Artix-7	9255	8313	3770 slices (34)	130
Us NW-16	512 any	Artix-7	19199	16501	7116 slices (60)	74

Efficient MMM for ECC

Mrabet et al.

September 2016

