Fast and secure DH implementation

Chitchanok Chuengsatiansup

Technische Universiteit Eindhoven

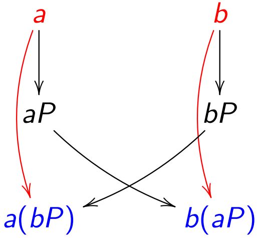
July 22nd, 2016

Diffie–Hellman key exchange

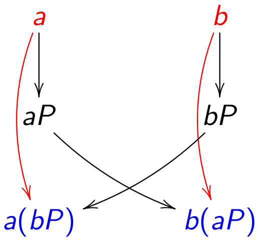
a b

Diffie–Hellman key exchange

Diffie–Hellman key exchange

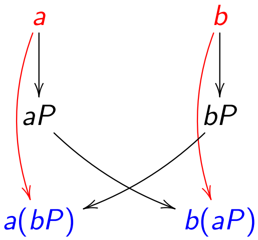


Diffie–Hellman key exchange

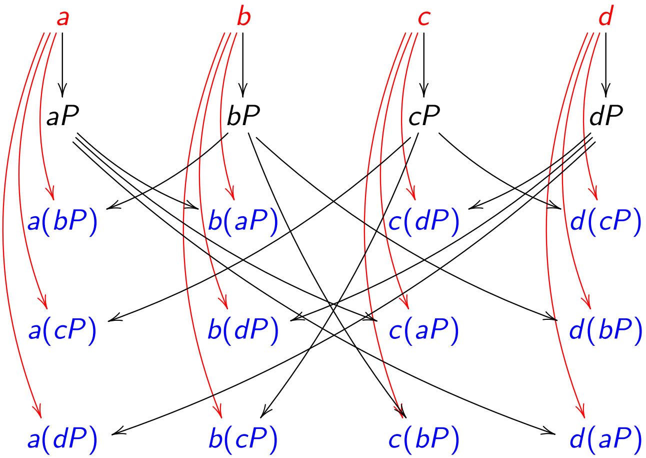


c d

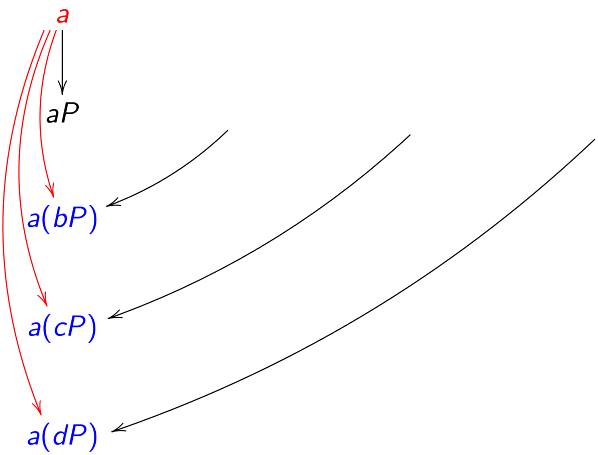
Diffie–Hellman key exchange



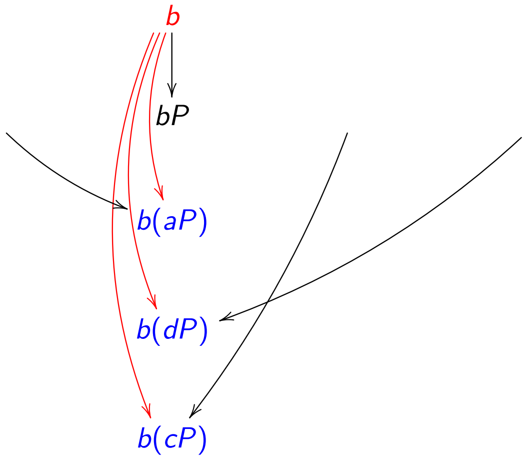
Diffie–Hellman key exchange



Diffie–Hellman key exchange

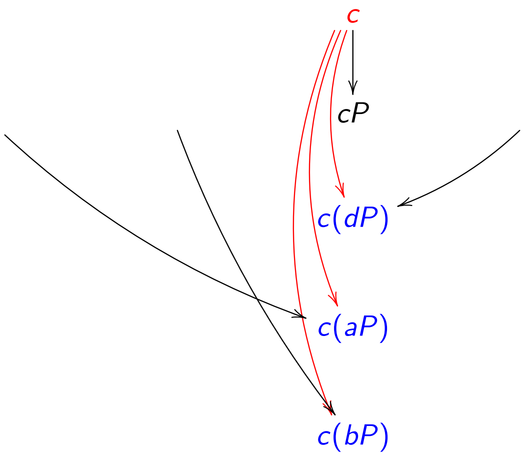


main DH challenge: make variable-base scalar mult as fast as possible



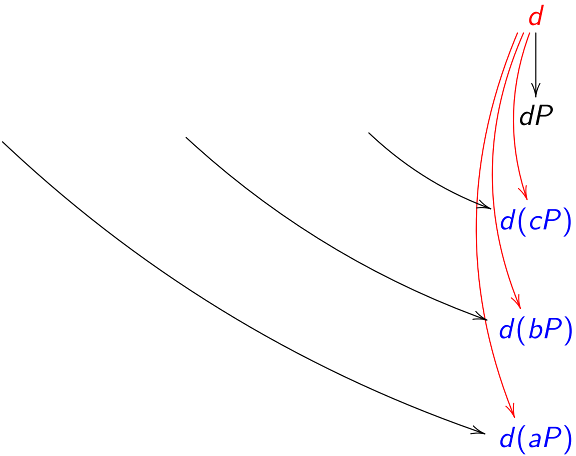
main DH challenge: make variable-base scalar mult as fast as possible

Diffie–Hellman key exchange



main DH challenge: make variable-base scalar mult as fast as possible

Diffie–Hellman key exchange



main DH challenge: make variable-base scalar mult as fast as possible

Scalar multiplication

- Given scalar n and point P
- Compute $nP = \underbrace{P + P + \ldots + P}_{n}$

Scalar multiplication

- Given scalar n and point P
- Compute $nP = \underbrace{P + P + \ldots + P}_{n}$
- Algorithms:
  - Double-and-add
  - Ladder

Double-and-add

Input: $P, n = (n_{i-1}, \ldots, n_0)_2$
Output: $R = nP$

$R \leftarrow 1P$

for $c = n_{i-2}$ **to** n_0
  $R \leftarrow 2R$
  if $c = 1$ **then**
    $R \leftarrow R + P$

Return R

Example: Compute $9P$

$9 = 1001_2$

$R = \quad 1P$

$n_2 = 0 : \quad 2P$

$n_1 = 0 : \quad 4P$

$n_0 = 1 : \quad 8P + P = 9P$

Windowing method

Input: $P, n = (n_{i-1}, \ldots, n_0)_2$
Output: $R = nP$

---

convert n to radix 2^ω:
$n = (c_j, \ldots, c_0)_{2^\omega}$
$R \leftarrow c_j P$

for $k = j - 1$ **down to** 0
$R \leftarrow 2^\omega R + c_k P$

Return R

Note: ω = window width

Example: Compute $2345P$

$2345 = \underline{10}\ \underline{01001}\ \underline{01001}\ _2$

$= 299_{32}$

$R = 10_2 = 2P$

$01001_2 = 9$:

$2^5(2P) + 9P = 73P$

$01001_2 = 9$:

$2^5(72P) + 9P = 2345P$

Ladder

Input: $P, n = (n_{i-1}, \ldots, n_0)_2$
Output: $R_0 = nP$

$R_0 \leftarrow 0P; \quad R_1 \leftarrow 1P$
if $n_i = 0$ **then**
$\quad R_1 \leftarrow R_0 + R_1; \quad R_0 \leftarrow 2R_0$
else
$\quad R_0 \leftarrow R_0 + R_1; \quad R_1 \leftarrow 2R_1$

Return R_0

Example: Compute $9P$

$9 = 1001_2$

$(R_0, R_1) = (0P, 1P)$

$n_3 = 1: \quad (1P, 2P)$

$n_2 = 0: \quad (2P, 3P)$

$n_1 = 0: \quad (4P, 5P)$

$n_0 = 1: \quad (9P, 10P)$

Side-channel attacks

- Try to discover secret information via physical measurements such as
  - electromagnetic radiation
  - power consumption
  - run time
  - noise

Side-channel attacks

- Try to discover secret information via physical measurements such as
  - electromagnetic radiation
  - power consumption
  - run time
  - noise
- Examples of side-channel attacks are:
  - timing attack
  - power attack
  - fault attack

Secure the implementation

- Prevent software side-channel attacks:
  - constant-time
  - no input-dependent branch
  - no input-dependent array index

Secure the implementation

- Prevent software side-channel attacks:
  - constant-time
  - no input-dependent branch
  - no input-dependent array index
- Constant-time table-lookup:
  - read entire table
  - select via arithmetic
    if c is 1, select tbl[i]
    if c is 0, ignore tbl[i]

$$t = (t \cdot (1-c)) + (\text{tbl[i]} \cdot (c))$$

$$t = (t \ \texttt{and} \ (c-1)) \ \texttt{xor} \ (\text{tbl[i]} \ \texttt{and} \ (-c))$$

Vectorization speedups

without vector

a

$+$

b

$=$

$a + b$

Vectorization speedups

without vector

a
$+$
b
$=$
$a + b$

with vector

a_0	a_1	a_2	a_3
$+$	$+$	$+$	$+$
b_0	b_1	b_2	b_3
$=$	$=$	$=$	$=$
$a_0 + b_0$	$a_1 + b_1$	$a_2 + b_2$	$a_3 + b_3$

Vectorization speedups

without vector	with vector			
a	a_0	a_1	a_2	a_3
$+$	$+$	$+$	$+$	$+$
b	b_0	b_1	b_2	b_3
$=$	$=$	$=$	$=$	$=$
$a + b$	$a_0 + b_0$	$a_1 + b_1$	$a_2 + b_2$	$a_3 + b_3$

- **single** instruction performing n **independent** operations on **aligned** inputs

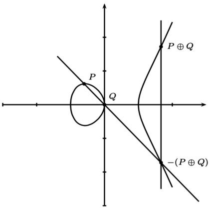
Based on

Kummer strikes back: new DH speed records
Joint work with Daniel J. Bernstein, Tanja Lange & Peter Schwabe

---

Security level is $\approx \sqrt{p^g}$ where p is the field size and g is the genus.
We work on Kummer surface of genus-2 hyperelliptic curve over $\mathbf{F}_p$ where $p = 2^{127} - 1$, i.e., $\approx 2^{127}$ security.
Scalar multiplication is computed using ladder.

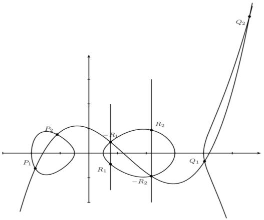
ECC



$$y^2 = x^3 + ax + b$$

$\longrightarrow$ x-line represented as $(X : Z)$

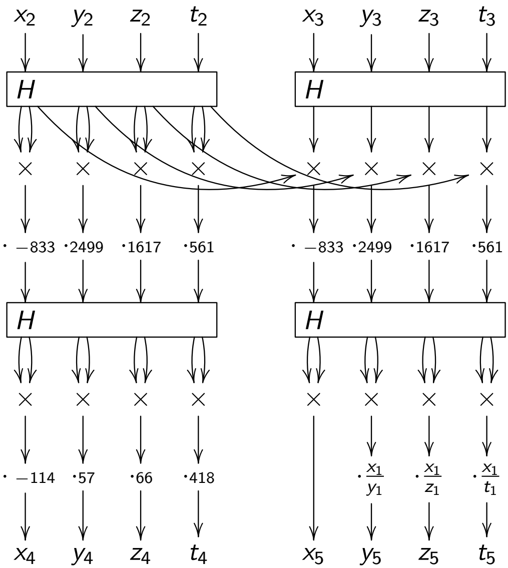
HECC



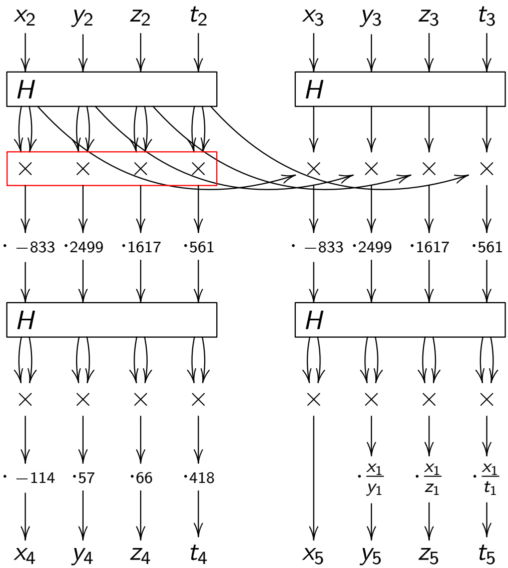
$$v^2 = u^5 + f_4 u^4 + f_3 u^3 + f_2 u^2 + f_1 u^1 + f_0$$

$\longrightarrow$ Kummer surface represented as $(X : Y : Z : T)$

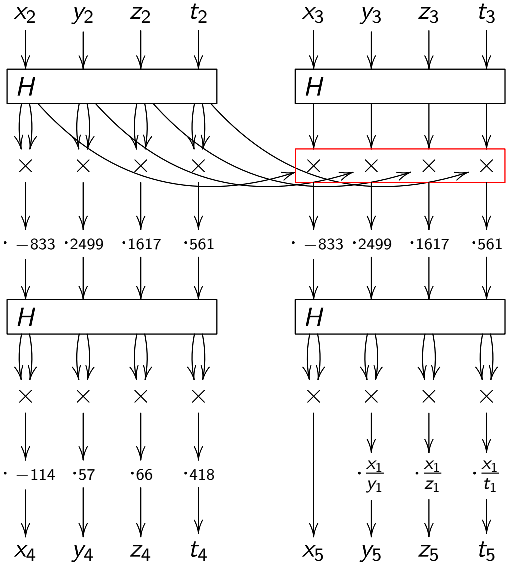
Squared Kummer surface ladder



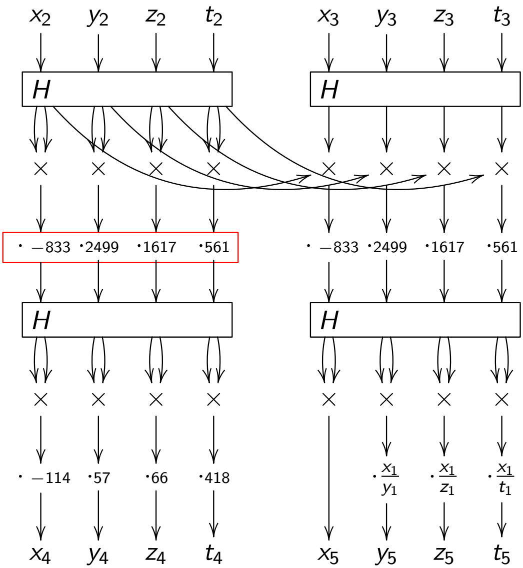
Squared Kummer surface ladder



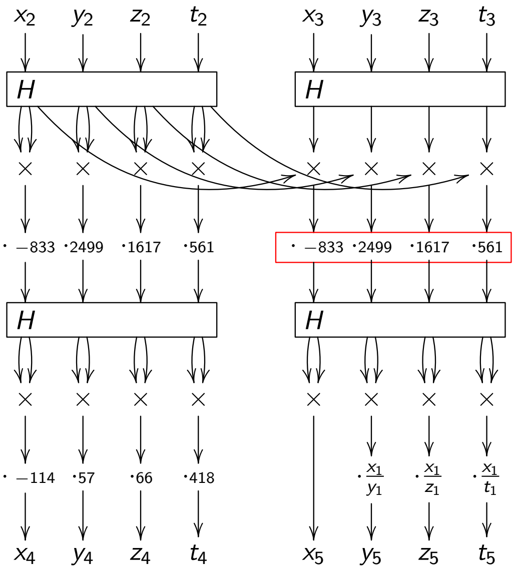
Squared Kummer surface ladder



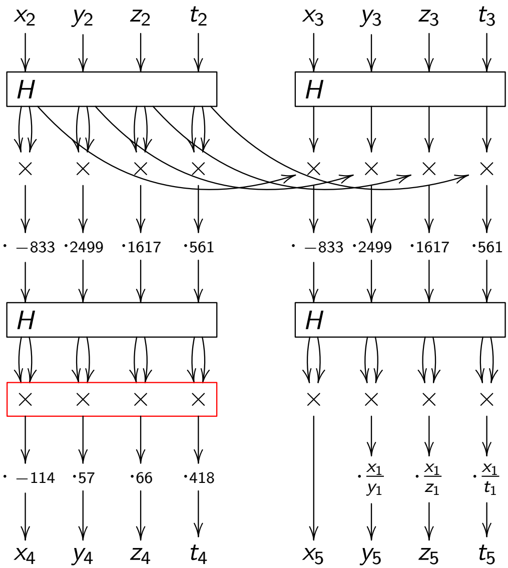
Squared Kummer surface ladder



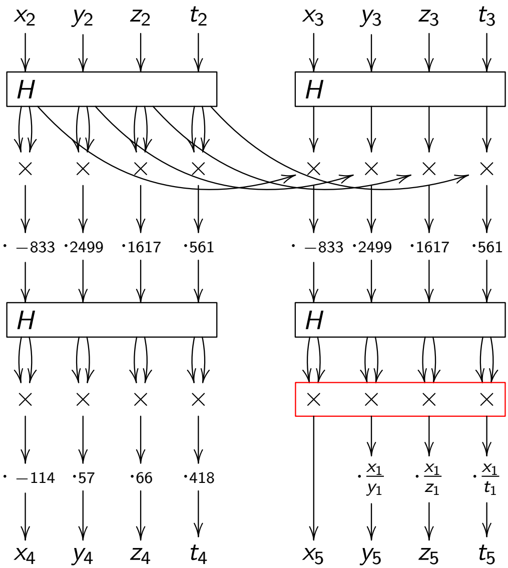
Squared Kummer surface ladder



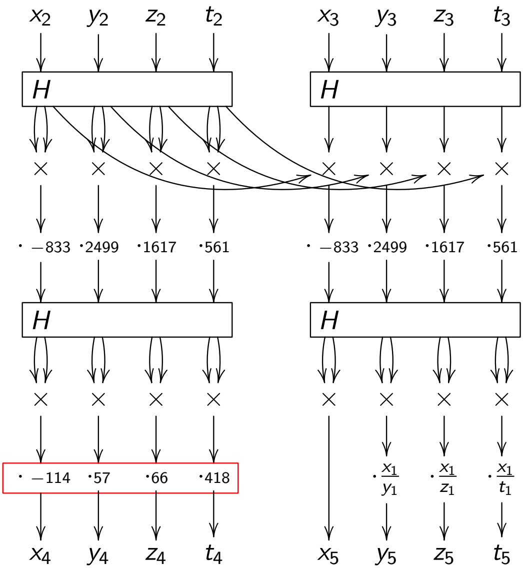
Squared Kummer surface ladder



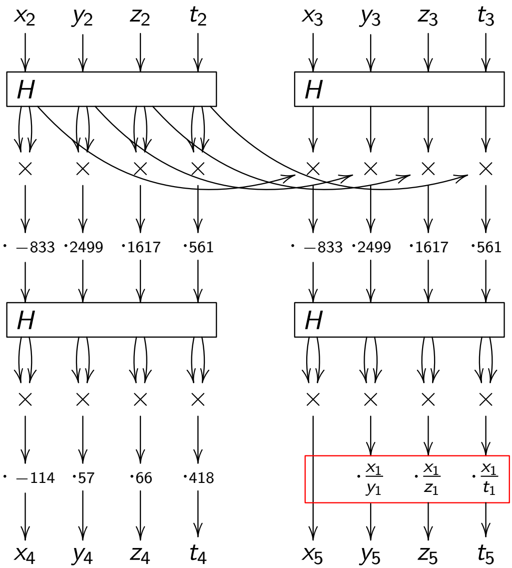
Squared Kummer surface ladder



Squared Kummer surface ladder



Squared Kummer surface ladder



Performance comparison (1/2)

Arch	Cycles	g	Field	Source of software
TI Sitara	497389	1	$2^{255}-19$	BeSc CHES 2012
TI Sitara	305395	2	$2^{127}-1$	**BeChLaSc** Asiacrypt 2014
i.MX515	460200	1	$2^{255}-19$	BeSc CHES 2012
i.MX515	273349	2	$2^{127}-1$	**BeChLaSc** Asiacrypt 2014

picture credit:

http://www.pngall.com/wp-content/uploads/2016/03/Smartphone-PNG-Pic.png

Performance comparison (2/2)

Arch	Cycles	g	Field	Source of software	
Sandy	122716	2	$2^{127}-1$	BoCoHiLa	Eurocrypt 2013
Sandy	119904	1	2^{254}	OlLóArRo	CHES 2013
Sandy	96000?	1	$(2^{127}-5997)^2$	FaLoSá	CT-RSA 2014
Sandy	92000?	1	$(2^{127}-5997)^2$	FaLoSá	July 2014
Sandy	88916	2	$2^{127}-1$	**BeChLaSc**	Asiacrypt 2014
Haswell	161648	1	$2^{255}-19$	BeDuLaScYa	CHES 2011
Haswell	110740	2	$2^{127}-1$	BoCoHiLa	Eurocrypt 2013
Haswell	61712	1	2^{254}	OlLóArRo	CHES 2013
Haswell	60556	2	$2^{127}-1$	**BeChLaSc**	Asiacrypt 2014

pictures credit:

http://mitechnews.com/tag/personal-computer/

http://blogs.which.co.uk/technology/wp-content/uploads/2012/04/pc-versus-mac.jpg

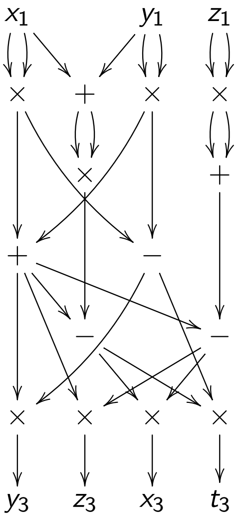
Based on

Curve41417: Karatsuba revisited
Joint work with Daniel J. Bernstein & Tanja Lange

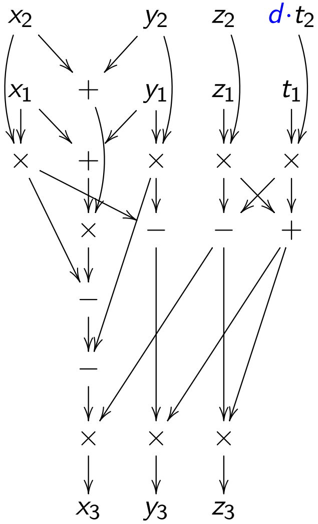
Curve41417 is a high-security elliptic curve
in Edwards form $x^2 + y^2 = 1 + 3617x^2y^2$
which is defined over prime field $\mathbf{F}_p$ where $p = 2^{414} - 17$.
Scalar multiplication is computed using signed fixed window
with width $\omega = 5$.

Point operations

Point doubling



Point addition



Polynomial multiplication

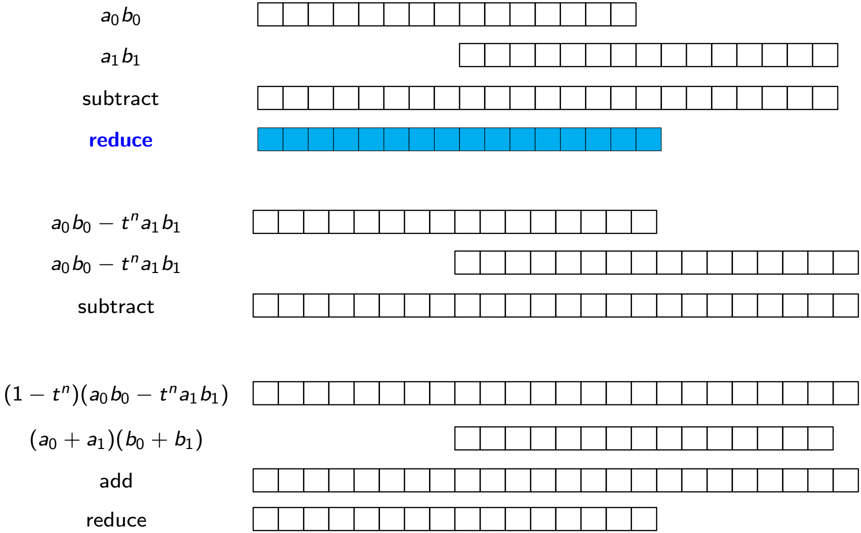
- Goal: Compute $P = AB$
  given $A = a_0 + a_1 t^n$ and $B = b_0 + b_1 t^n$
- Method 1: schoolbook
  $P = a_0 b_0 + (a_0 b_1 + a_1 b_0) t^n + a_1 b_1 t^{2n}$
- Method 2: Karatsuba ($8n-4$ additions)
  $P = a_0 b_0 + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1) t^n + a_1 b_1 t^{2n}$
- Method 3: refined Karatsuba ($7n-3$ additions)
  $P = (a_0 b_0 - a_1 b_1 t^n)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n$

Polynomial multiplication $\bmod Q$

- Goal: Compute $P = AB \bmod Q$
  given $A = a_0 + a_1 t^n$ and $B = b_0 + b_1 t^n$
- Method 1: schoolbook
  $P = a_0 b_0 + (a_0 b_1 + a_1 b_0) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 2: Karatsuba ($8n-4$ additions)
  $P = a_0 b_0 + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 3: refined Karatsuba ($7n-3$ additions)
  $P = (a_0 b_0 - a_1 b_1 t^n)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n \bmod Q$

Polynomial multiplication $\bmod Q$

- Goal: Compute $P = AB \bmod Q$
  given $A = a_0 + a_1 t^n$ and $B = b_0 + b_1 t^n$
- Method 1: schoolbook
  $P = a_0 b_0 + (a_0 b_1 + a_1 b_0) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 2: Karatsuba ($8n-4$ additions)
  $P = a_0 b_0 + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1) t^n + a_1 b_1 t^{2n} \bmod Q$
- Method 3: refined Karatsuba ($7n-3$ additions)
  $P = (a_0 b_0 - a_1 b_1 t^n)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n \bmod Q$
- Method 4: reduced refined Karatsuba ($6n-2$ additions) (new)
  $P = (a_0 b_0 - a_1 b_1 t^n \bmod Q)(1 - t^n) + (a_0 + a_1)(b_0 + b_1) t^n \bmod Q$



$a_0 b_0$
$a_1 b_1$
subtract
reduce
$a_0 b_0 - t^n a_1 b_1$
$a_0 b_0 - t^n a_1 b_1$
subtract
$(1 - t^n)(a_0 b_0 - t^n a_1 b_1)$
$(a_0 + a_1)(b_0 + b_1)$
add
reduce

Performance comparison

- OpenSSL (on Cortex-A8)

curve	# cycle on i.MX515	# cycle on Sitara
`secp160r1`	$\approx$ 2.1 million	$\approx$ 2.1 million
`nistp192`	$\approx$ 2.9 million	$\approx$ 2.8 million
`nistp224`	$\approx$ 4.0 million	$\approx$ 3.9 million
`nistp256`	$\approx$ 4.0 million	$\approx$ 3.9 million
`nistp384`	$\approx$ 13.3 million	$\approx$ 13.2 million
`nistp521`	$\approx$ 29.7 million	$\approx$ 29.7 million

- Curve41417 (security level above 2^{200})
  - $\approx$ 1.6 million cycles on i.MX515
  - $\approx$ 1.8 million cycles on TI Sitara

Making it run really fast

- Maximize usage of available vector multipliers
- Minimize cost from carries
  - use redundant representation
  - use non-integer radix, e.g., $2^{25.4}$ for Kummer on Cortex-A8
  - do not perform full carry
  - do parallel carry chain
- Eliminate redundancy inside field operations
  - precompute to reuse values
- Minimize overhead from permutations
  - organize data to fit instruction format
- Schedule instructions to keep CPU as busy as possible
- See papers for details
  - https://cr.yp.to/hecdh/kummer-20140218.pdf
  - https://cr.yp.to/ecdh/curve41417-20140706.pdf