

# Kummer strikes back: new DH speed records

Chitchanok Chuengsatiansup

9<sup>th</sup> December 2014

Joint work with Daniel J. Bernstein, Tanja Lange, and Peter Schwabe

# Diffie–Hellman Key Exchange

*a*

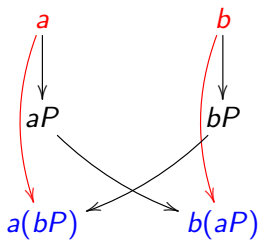
*b*

# Diffie–Hellman Key Exchange

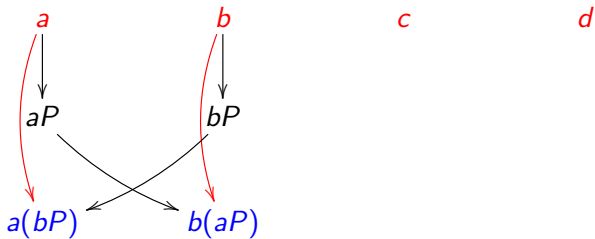
$a$   
↓  
 $aP$

$b$   
↓  
 $bP$

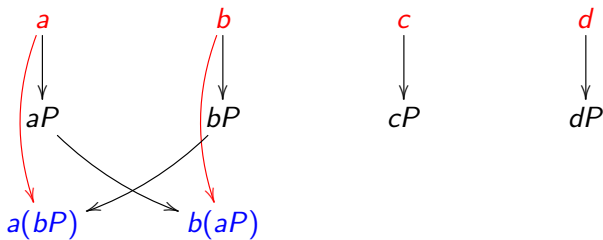
# Diffie–Hellman Key Exchange



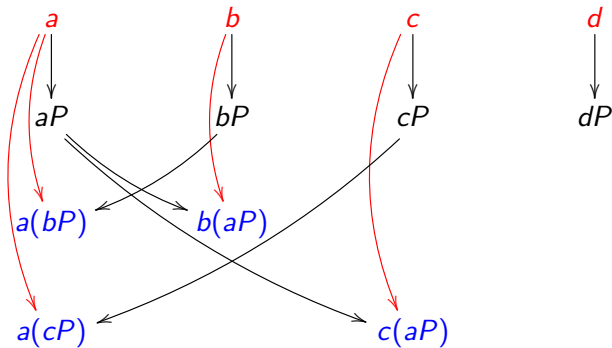
# Diffie–Hellman Key Exchange



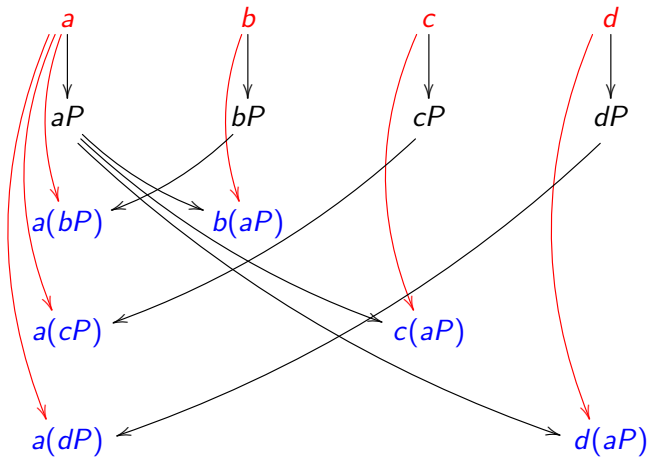
# Diffie–Hellman Key Exchange



# Diffie–Hellman Key Exchange

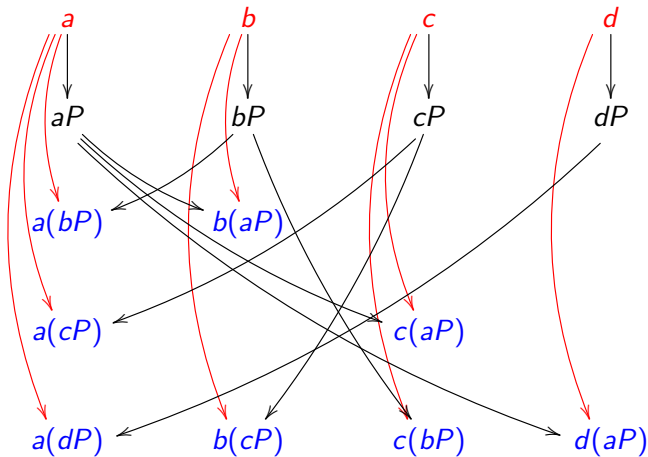


# Diffie–Hellman Key Exchange

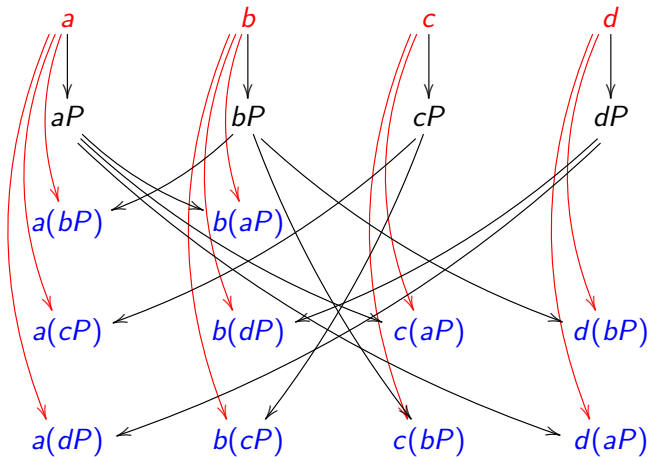




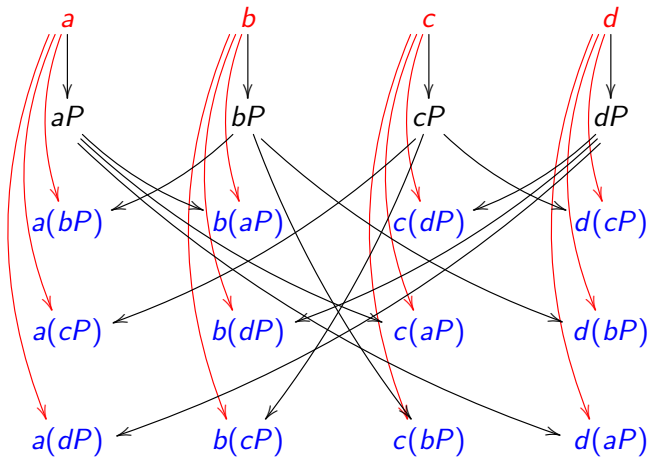
# Diffie–Hellman Key Exchange



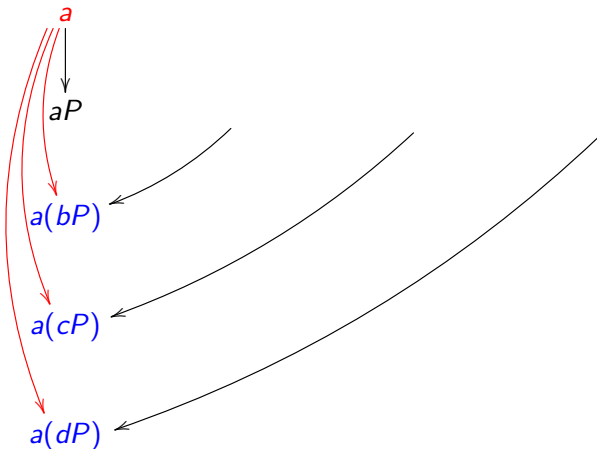
# Diffie–Hellman Key Exchange



# Diffie–Hellman Key Exchange

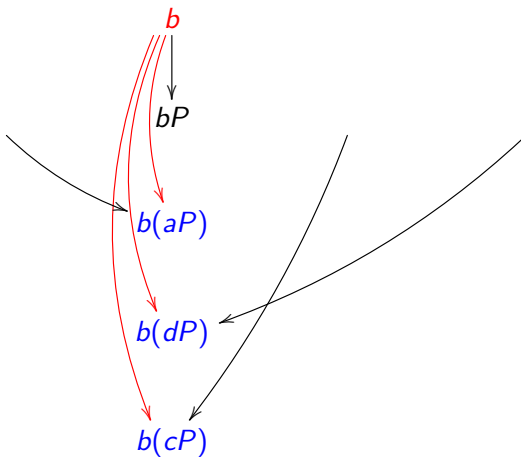


# Diffie–Hellman Key Exchange



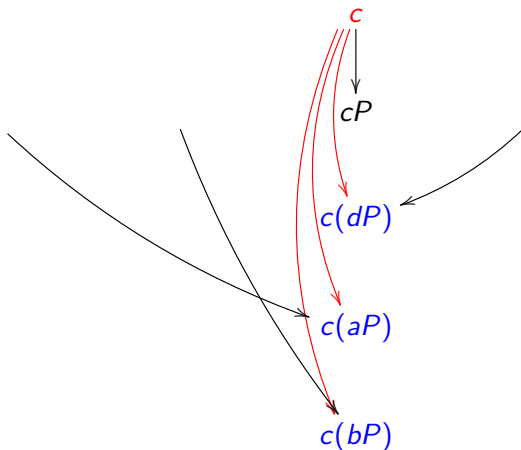
main DH challenge: make **variable-base** scalar mult as fast as possible

# Diffie–Hellman Key Exchange



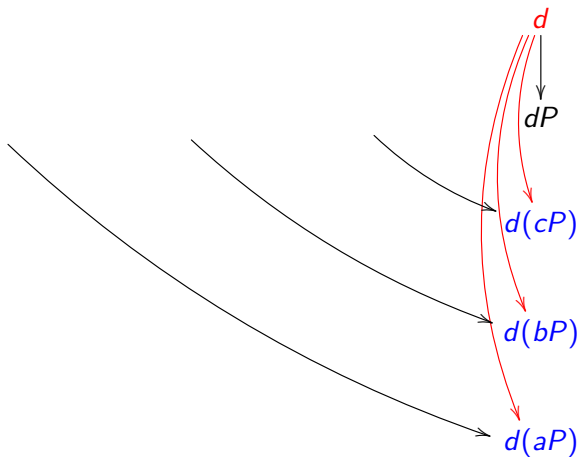
main DH challenge: make **variable-base** scalar mult as fast as possible

# 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

**Input:**  $Q, n = (n_{i-1}, \dots, n_0)_2$

**Output:**  $R_0 = nQ$

---

$R_0 \leftarrow 0Q; \quad R_1 \leftarrow 1Q$

**if**  $n_i = 0$  **then**

$R_0 \leftarrow 2R_0; \quad R_1 \leftarrow R_0 + R_1$

**else**

$R_0 \leftarrow R_0 + R_1; \quad R_1 \leftarrow 2R_1$

**Return**  $R_0$

**Example:** Compute  $9Q$

$9 = 1001_2$

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

$n_3 = 1 : (1Q, 2Q)$

$n_2 = 0 : (2Q, 3Q)$

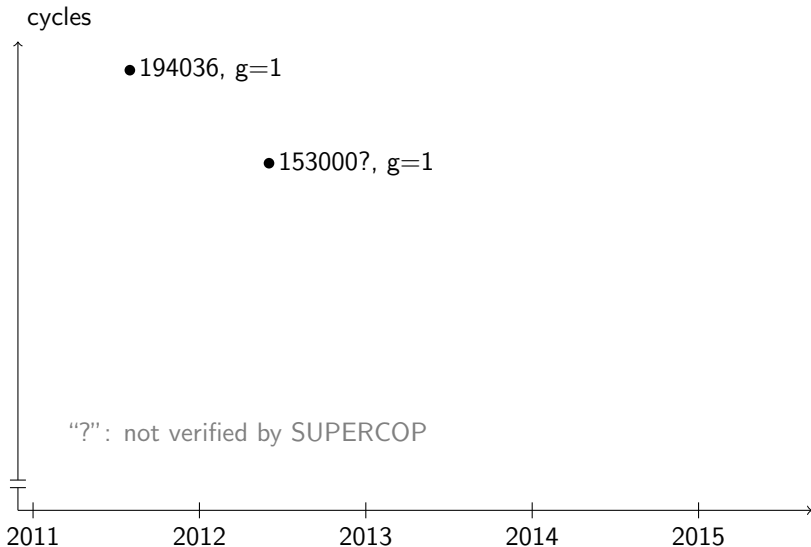
$n_1 = 0 : (4Q, 5Q)$

$n_0 = 1 : (9Q, 10Q)$



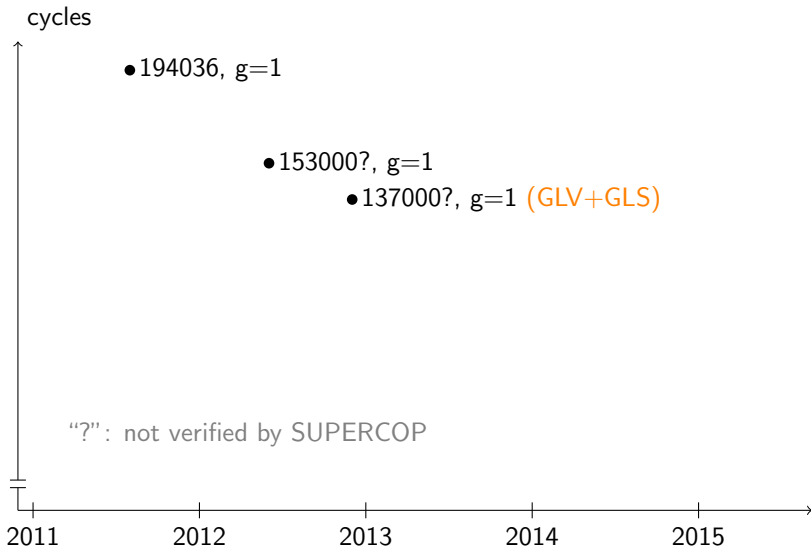
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



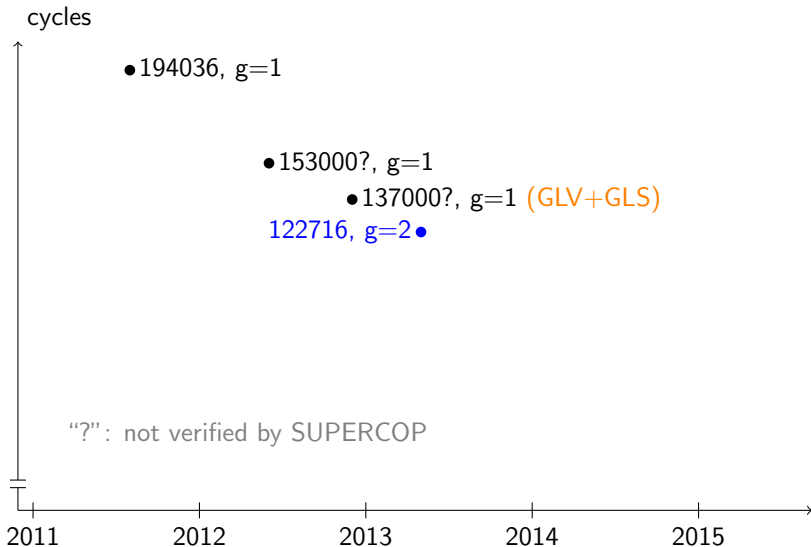
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



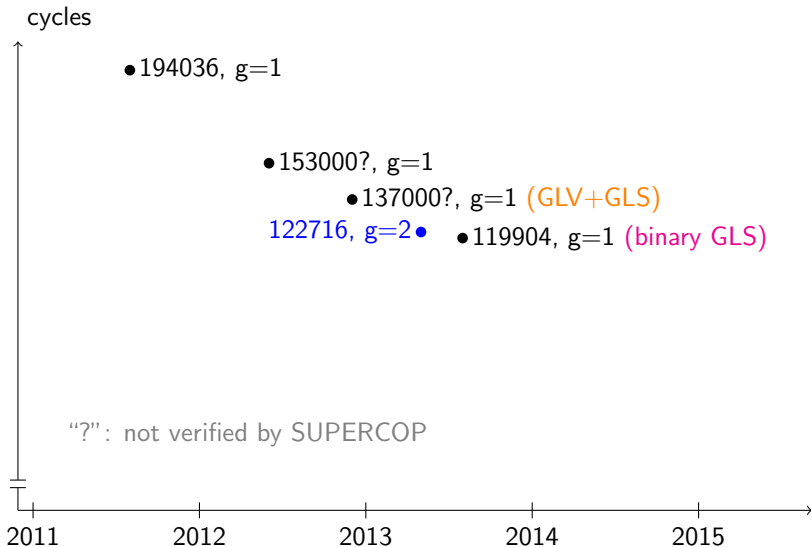
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



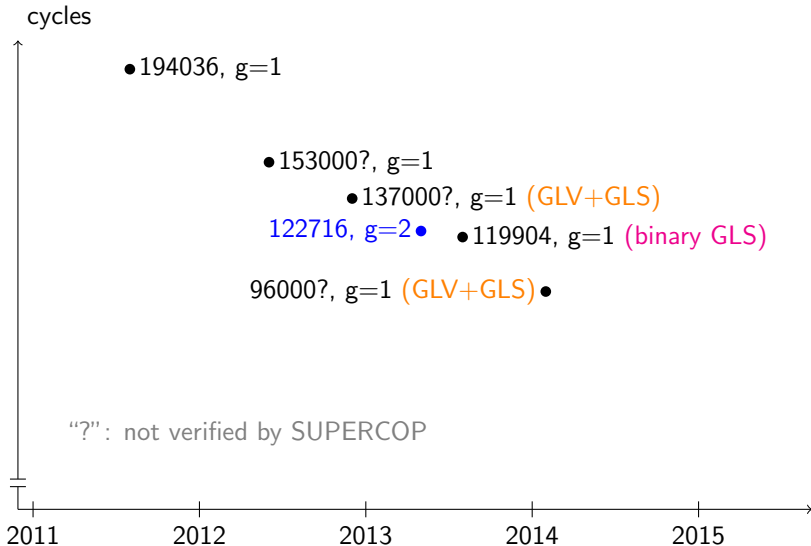
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



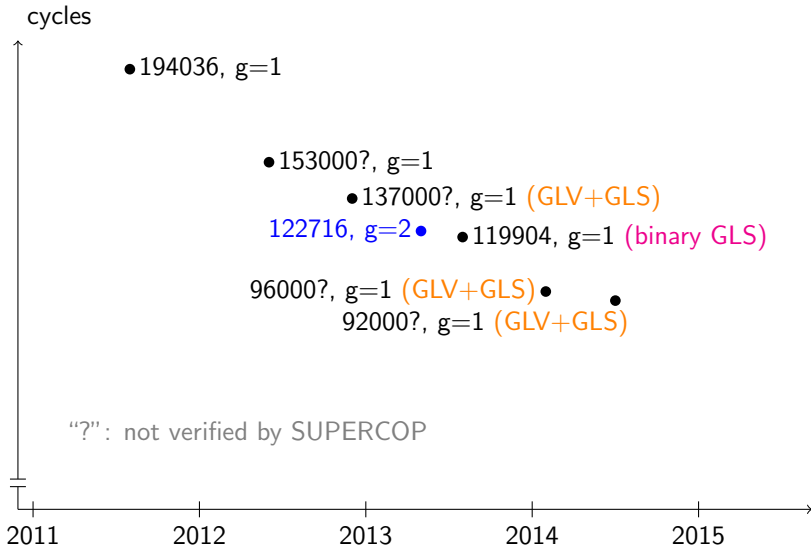
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



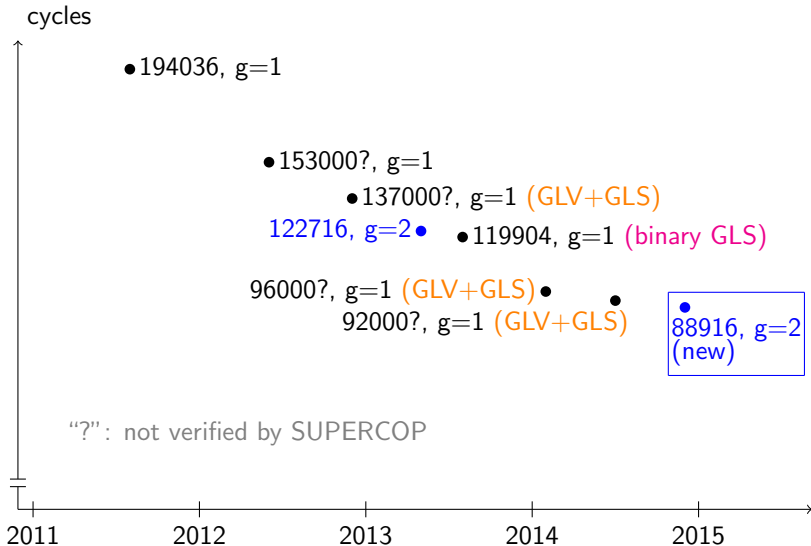
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



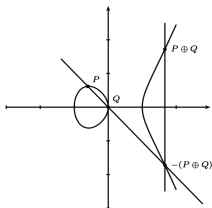
# Elliptic VS Hyperelliptic

Sandy Bridge at high security level and (claimed) constant time



# Elliptic-Hyperelliptic Analogy

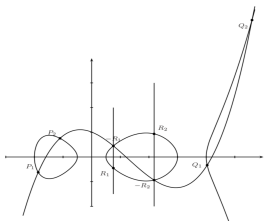
ECC



x-line  
represented as  
(X : Z)

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

HECC



Kummer surface  
represented as  
(X : Y : Z : T)

$$v^2 = u^5 + f_4u^4 + f_3u^3 + f_2u^2 + f_1u^1 + f_0$$



# Hyperelliptic Advantages

- smaller field size

# Hyperelliptic Advantages

- smaller field size
  - need field size  $\approx 2^{128}$  for group size  $\approx 2^{256}$   
(ECC would need field size  $\approx 2^{256}$ )
  - field arithmetic 2–4 times faster

# Hyperelliptic Advantages

- smaller field size
  - need field size  $\approx 2^{128}$  for group size  $\approx 2^{256}$   
(ECC would need field size  $\approx 2^{256}$ )
  - field arithmetic 2–4 times faster
- improvement to HECC formulas

# Hyperelliptic Advantages

- smaller field size
  - need field size  $\approx 2^{128}$  for group size  $\approx 2^{256}$   
(ECC would need field size  $\approx 2^{256}$ )
  - field arithmetic 2–4 times faster
- improvement to HECC formulas
  - only 25 multiplications with Gaudry ladder for Kummer surface

# Hyperelliptic Advantages

- smaller field size
  - need field size  $\approx 2^{128}$  for group size  $\approx 2^{256}$   
(ECC would need field size  $\approx 2^{256}$ )
  - field arithmetic 2–4 times faster
- improvement to HECC formulas
  - only 25 multiplications with Gaudry ladder for Kummer surface
- discovery of *small* constants for Kummer surface

# Hyperelliptic Advantages

- smaller field size
  - need field size  $\approx 2^{128}$  for group size  $\approx 2^{256}$   
(ECC would need field size  $\approx 2^{256}$ )
  - field arithmetic 2–4 times faster
- improvement to HECC formulas
  - only 25 multiplications with Gaudry ladder for Kummer surface
- discovery of *small* constants for Kummer surface
  - 6 of 25 multiplications are by *small* constant

# Hyperelliptic Advantages

- smaller field size
  - need field size  $\approx 2^{128}$  for group size  $\approx 2^{256}$   
(ECC would need field size  $\approx 2^{256}$ )
  - field arithmetic 2–4 times faster
- improvement to HECC formulas
  - only 25 multiplications with Gaudry ladder for Kummer surface
- discovery of *small* constants for Kummer surface
  - 6 of 25 multiplications are by *small* constant
- new: formulas well suited for vectorization

without vector

$$\boxed{a}$$

+

$$\boxed{b}$$

=

$$\boxed{a + b}$$



# Vectorization

without vector

$$\begin{array}{c} \boxed{a} \\ + \\ \boxed{b} \\ = \\ \boxed{a + b} \end{array}$$

with vector

$$\begin{array}{cccc} \boxed{a_0} & \boxed{a_1} & \boxed{a_2} & \boxed{a_3} \\ + & + & + & + \\ \boxed{b_0} & \boxed{b_1} & \boxed{b_2} & \boxed{b_3} \\ = & = & = & = \\ \boxed{a_0 + b_0} & \boxed{a_1 + b_1} & \boxed{a_2 + b_2} & \boxed{a_3 + b_3} \end{array}$$

# Vectorization

without vector

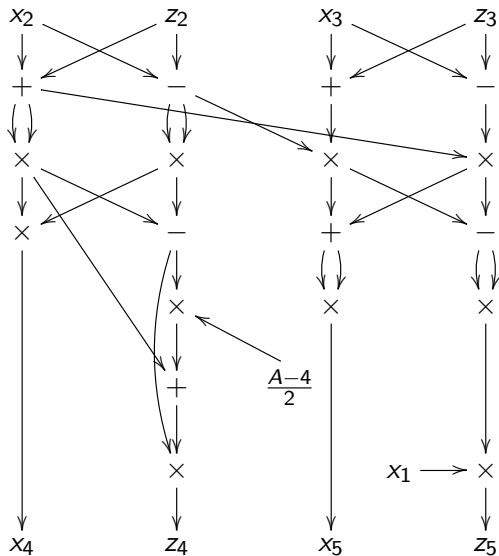
$$\begin{array}{c} \boxed{a} \\ + \\ \boxed{b} \\ = \\ \boxed{a + b} \end{array}$$

with vector

$$\begin{array}{cccc} \boxed{a_0} & \boxed{a_1} & \boxed{a_2} & \boxed{a_3} \\ + & + & + & + \\ \boxed{b_0} & \boxed{b_1} & \boxed{b_2} & \boxed{b_3} \\ = & = & = & = \\ \boxed{a_0 + b_0} & \boxed{a_1 + b_1} & \boxed{a_2 + b_2} & \boxed{a_3 + b_3} \end{array}$$

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

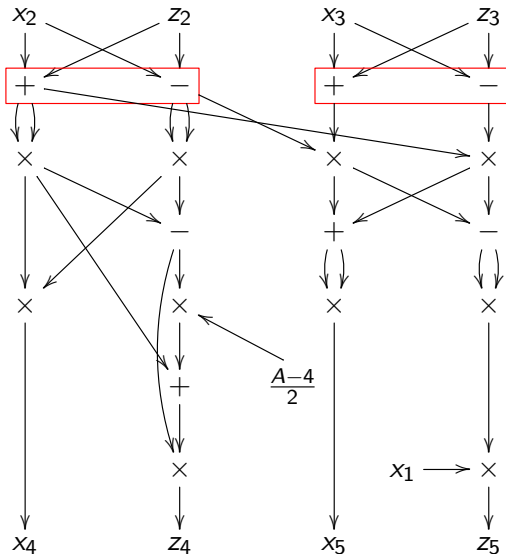
# ECC Montgomery Ladder (original)





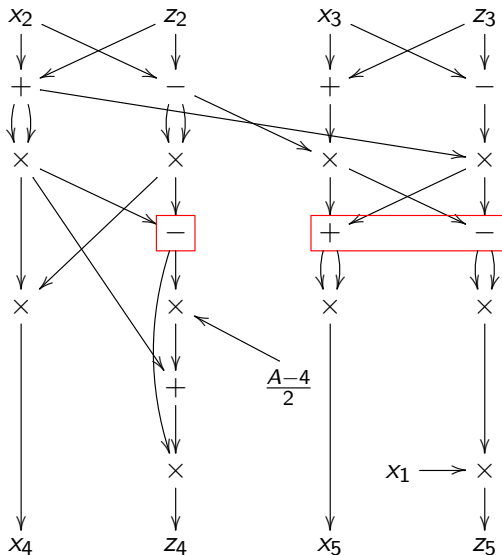
# ECC Montgomery Ladder (2-way vectorization)

- require permutation
- move  $\times$  down



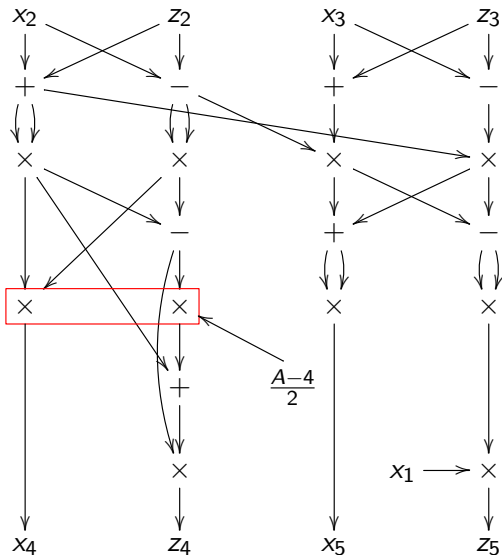
# ECC Montgomery Ladder (2-way vectorization)

- require permutation
- move  $\times$  down
- require permutation and leave  $+$  idle



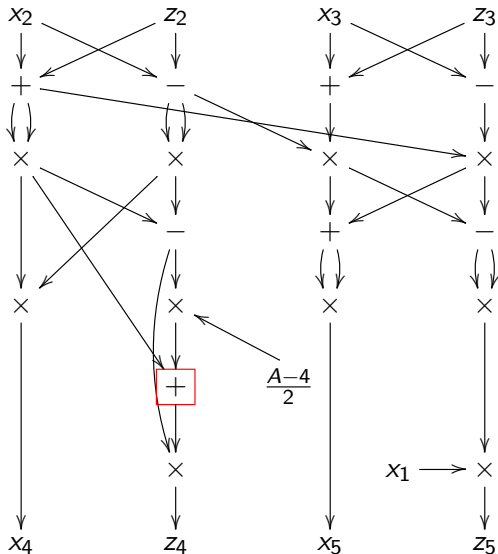
# ECC Montgomery Ladder (2-way vectorization)

- require permutation
- move  $\times$  down
- require permutation and leave  $+$  idle
- slow down mult. by constant



# ECC Montgomery Ladder (2-way vectorization)

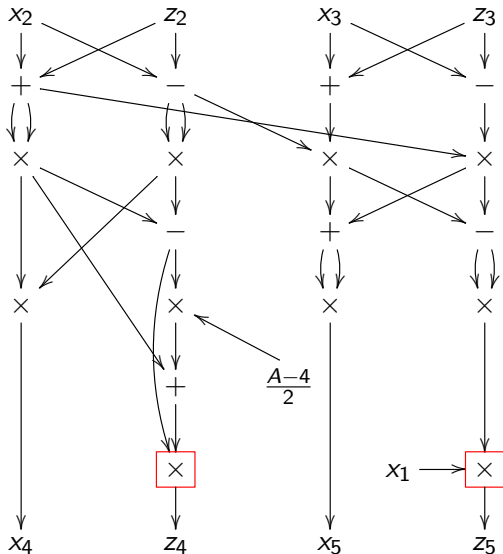
- require permutation
- move  $\times$  down
- require permutation and leave  $+$  idle
- slow down mult. by constant
- nothing to match  $+$





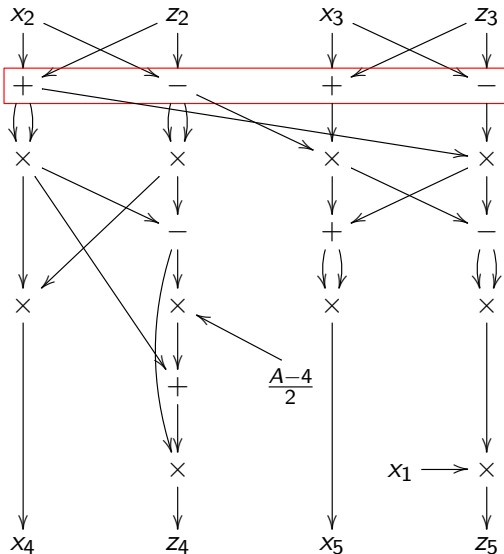
# ECC Montgomery Ladder (2-way vectorization)

- require permutation
- move  $\times$  down
- require permutation and leave  $+$  idle
- slow down mult. by constant
- nothing to match  $+$
- require permutation



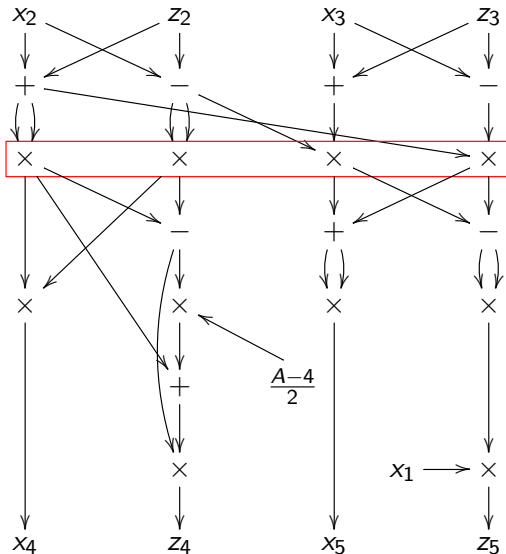
# ECC Montgomery Ladder (4-way vectorization)

- match + with -



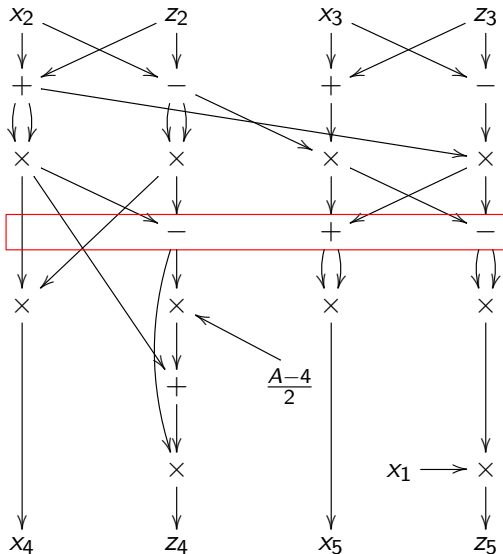
# ECC Montgomery Ladder (4-way vectorization)

- match + with -
- slow down squaring



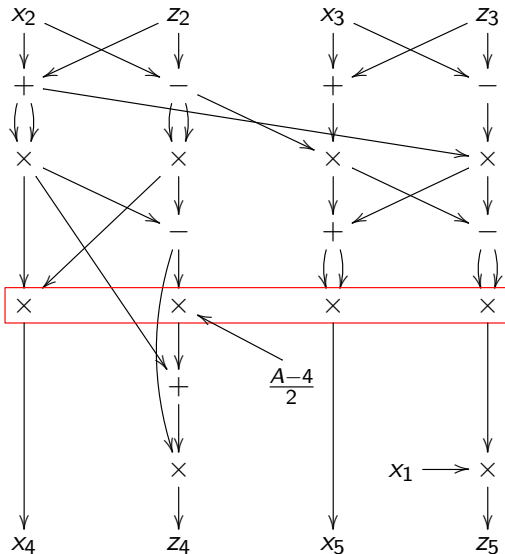
# ECC Montgomery Ladder (4-way vectorization)

- match + with -
- slow down squaring
- nothing to match and match + and -



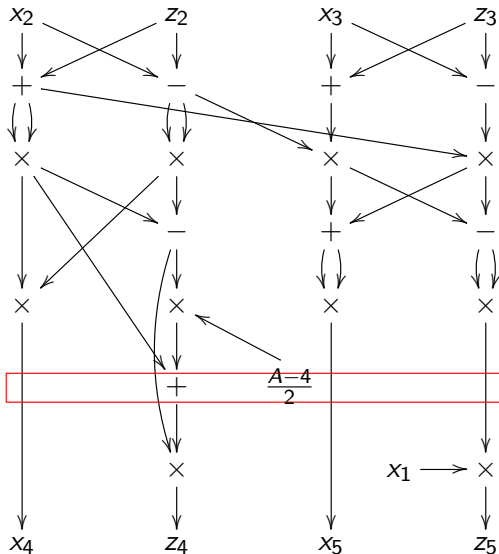
# ECC Montgomery Ladder (4-way vectorization)

- match + with -
- slow down squaring
- nothing to match and match + and -
- slow down mult. by constant and squaring



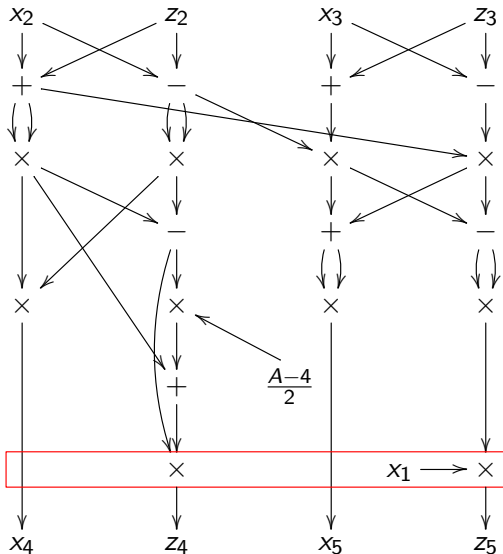
# ECC Montgomery Ladder (4-way vectorization)

- match + with -
- slow down squaring
- nothing to match and match + and -
- slow down mult. by constant and squaring
- nothing to match +

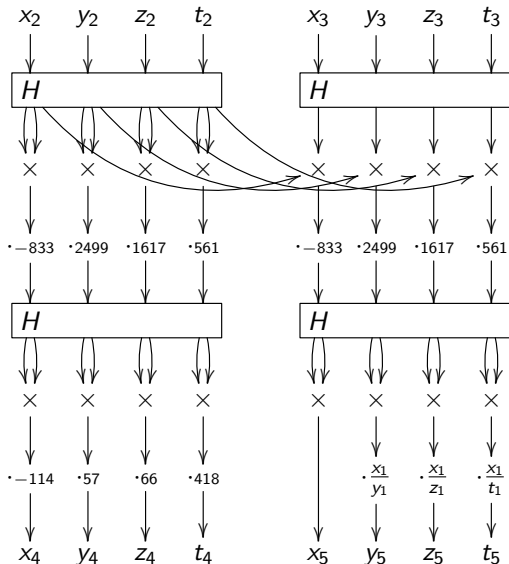


# ECC Montgomery Ladder (4-way vectorization)

- match + with -
- slow down squaring
- nothing to match and match + and -
- slow down mult. by constant and squaring
- nothing to match +
- nothing to match  $\times$



# Squared Kummer Surface Ladder





# Making It Run Really Fast

- maximize usage of available vector multipliers
- minimize cost from carries
  - use redundant representation
  - use non-integer radix, e.g.,  $2^{25.4}$  on Cortex-A8
  - do not perform full carry
  - do parallel carry chain
- eliminate redundancy inside field operations
  - precompute to reuse values  $2f_1, 2f_2, 2f_3, 2f_4$
- minimize overhead from permutations
  - organize data to fit instruction format
- minimize permutations in  $H$ 
  - use different order of input/output and change sign
  - e.g. we reduced 144 Sandy Bridge permutations to 36
- schedule instructions to keep CPU as busy as possible
- see paper for details

# Fastest Diffie–Hellman Ever!!!

Arch	Cycles	$g$	Field	Source of software
A8-slow	497389	1	$2^{255} - 19$	BeSc CHES 2012
A8-slow	305395	2	$2^{127} - 1$	<b>new (this paper)</b>
A8-fast	460200	1	$2^{255} - 19$	BeSc CHES 2012
A8-fast	273349	2	$2^{127} - 1$	<b>new (this paper)</b>
Sandy	194036	1	$2^{255} - 19$	BeDuLaScYa CHES 2011
Sandy	153000?	1	$2^{252} - 2^{232} - 1$	Hamburg
Sandy	137000?	1	$(2^{127} - 5997)^2$	LoSi Asiacrypt 2012
Sandy	122716	2	$2^{127} - 1$	BoCoHiLa Eurocrypt 2013
Sandy	119904	1	$2^{254}$	OILóArRo 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	$2^{127} - 1$	<b>new (this paper)</b>
Haswell	161648	1	$2^{255} - 19$	BeDuLaScYa CHES 2011
Haswell	110740	2	$2^{127} - 1$	BoCoHiLa Eurocrypt 2013
Haswell	61712	1	$2^{254}$	OILóArRo CHES 2013
Haswell	60556	2	$2^{127} - 1$	<b>new (this paper)</b>