Elliptic and Hyperelliptic Curves: a Practical Security Comparison

Joppe W. Bos (Microsoft Research), Craig Costello (Microsoft Research), Andrea Miele (EPFL)

Motivation and Goal(s)

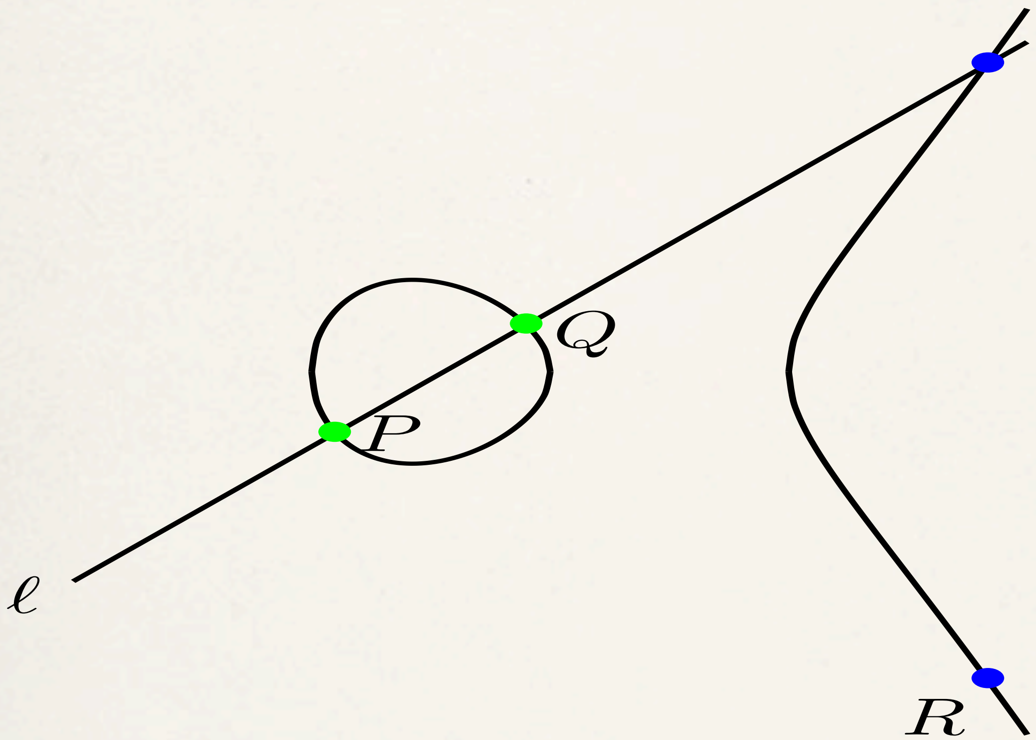
- Elliptic curves (standard) and genus 2 hyper-elliptic curves (object of research) over prime fields: similar performance [Gaudry07] [BCHL13]
- Security: Pollard rho $O(\sqrt{|G|})$ Using automorphisms $\approx \sqrt{\frac{\pi |G|}{2(\# Aut)}}$

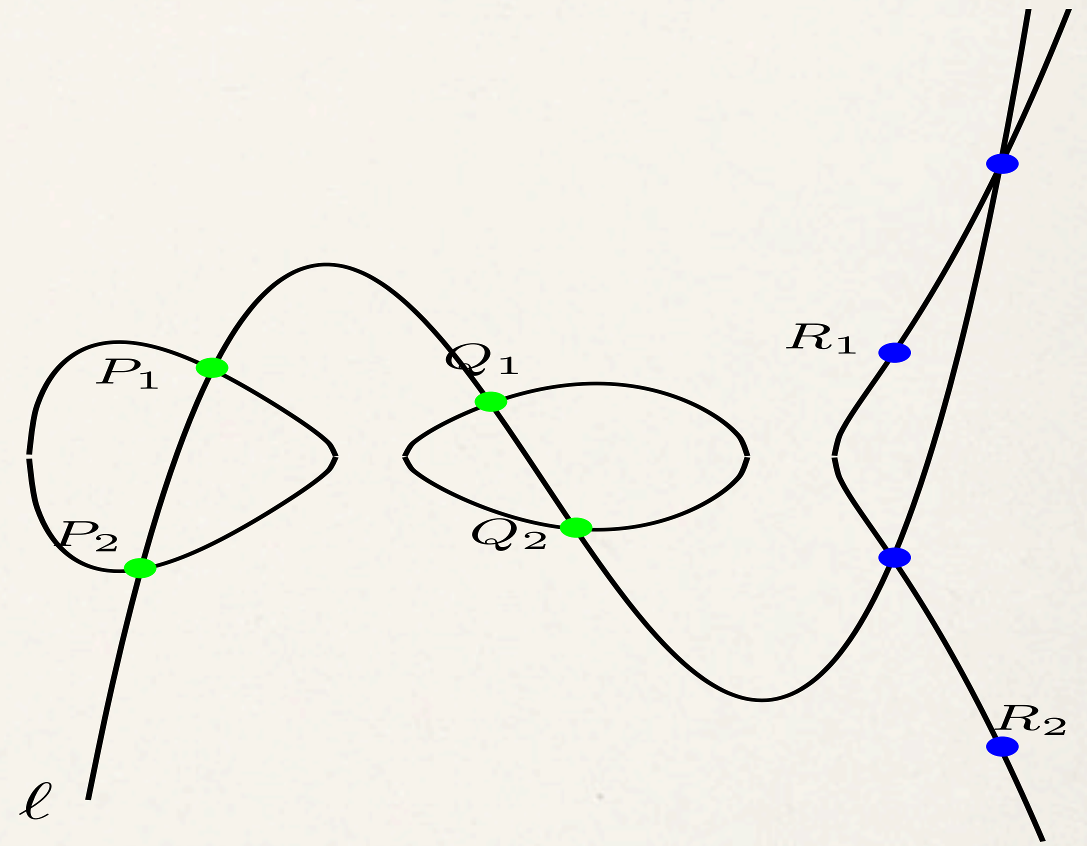
1. Estimate practical speed-up using automorphisms in genus 1 and genus 2
Tradeoff: reduced search space vs. more costly iteration
2. Estimate complexity of the attack on 4 curves (128-bit security)
3. Implement Pollard rho for genus 1 and genus 2 curves (x86 64-bit)

Curves used

NISTp-256 **Genus: 1** **Field size: 256 bits** **# Aut: 2** **Theoretical security: 127.8 bits**	**BN254 (pairing friendly)** **Genus: 1** **Field size: 254 bits** **# Aut: 6** **Theoretical security: 126.4 bits**
Generic-1271 **Genus: 2** **Field size: 127 bits** **# Aut: 2** **Theoretical security: 126.8 bits**	**GLV4-BK** **Genus: 2** **Field size: 127 bits** **# Aut: 10** **Theoretical security: 125.7 bits**

Elliptic and genus 2 hyperelliptic curves in one slide…





$y^2=x^3+a_1x+a_0$
$\#E(F_p) \approx \mathbf{p}$
Weierstrass coordinates: (x,y)
Affine addition: 2**m**+1**s**+6**a**+1**i**
Affine doubling: 2**m**+2**s**+7**a**+1**i**

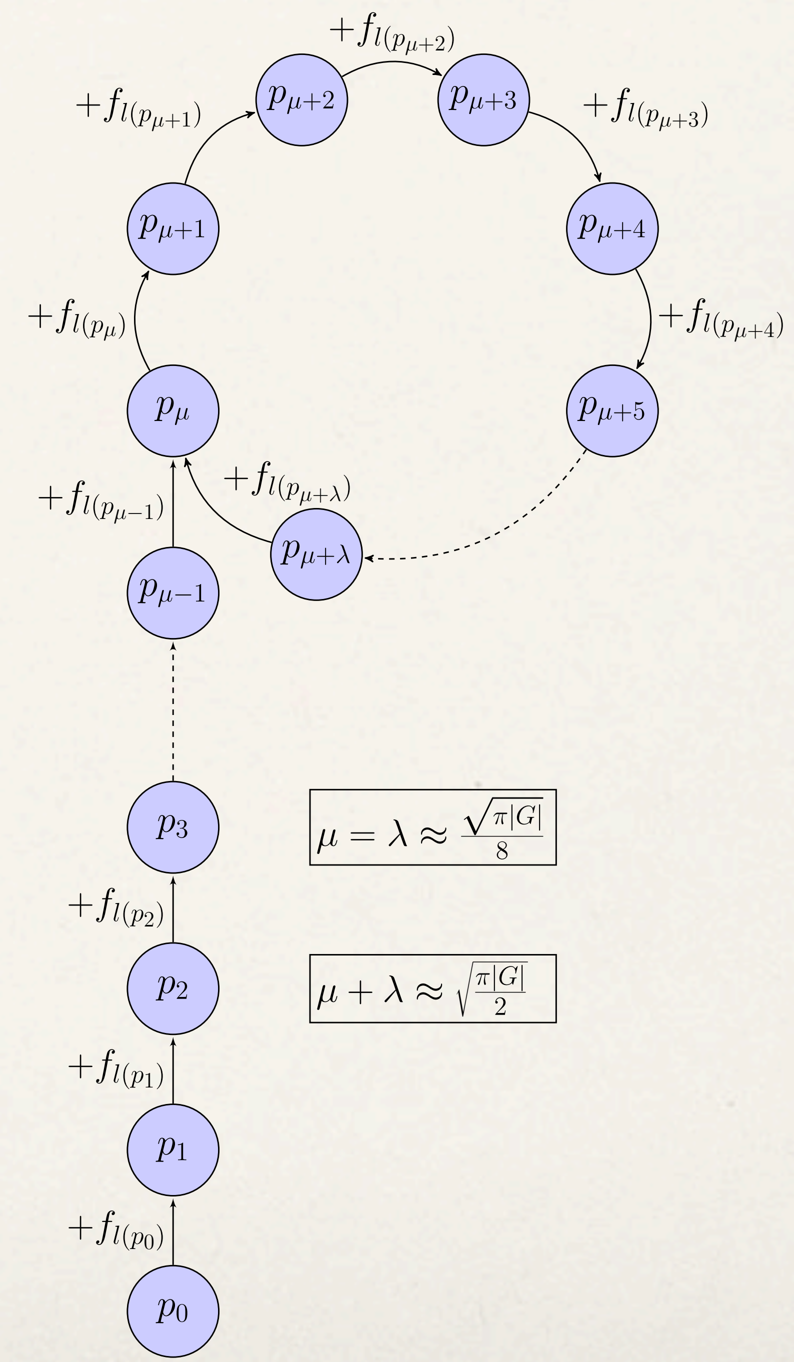
$y^2=x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0$
$\#Jac(C(F_p)) \approx \mathbf{p^2}$
Mumford coordinates: (u_1,u_0,v_1,v_0)
Affine addition: 17**m**+4**s**+48**a**+1**i**
Affine doubling: 19**m**+6**s**+52**a**+1**i**

Pollard's rho algorithm [P78]

- Discrete log: given **h** in <g> = G find integer k such that **h**=k**g**.

- **Ideal rho**, random walk:
  $\mathbf{p_i}=a_i\mathbf{g}+b_i\mathbf{h}$ for i=0,1,2,…
  Expect collision $\mathbf{p_i}=\mathbf{p_j}$ (j<i) in $\sqrt{\frac{\pi|G|}{2}}$ steps, $k = (a_i-a_j)/(b_j-b_i)$.

- **r-adding walk**: table of random $\mathbf{f_k}=a_k\mathbf{g}+b_k\mathbf{h}$, $0 \le k \le r-1$.
  $\mathbf{p_0}=a_0\mathbf{g}$, $\mathbf{p_i}=\mathbf{p_{i-1}}+\mathbf{f_{l(p_{i-1})}}$ for i=1,2,…
  with $0 \le \mathbf{l(p_i)} \le r-1$ ($\mathbf{p_i}$ has index $\mathbf{l(p_i)}$).

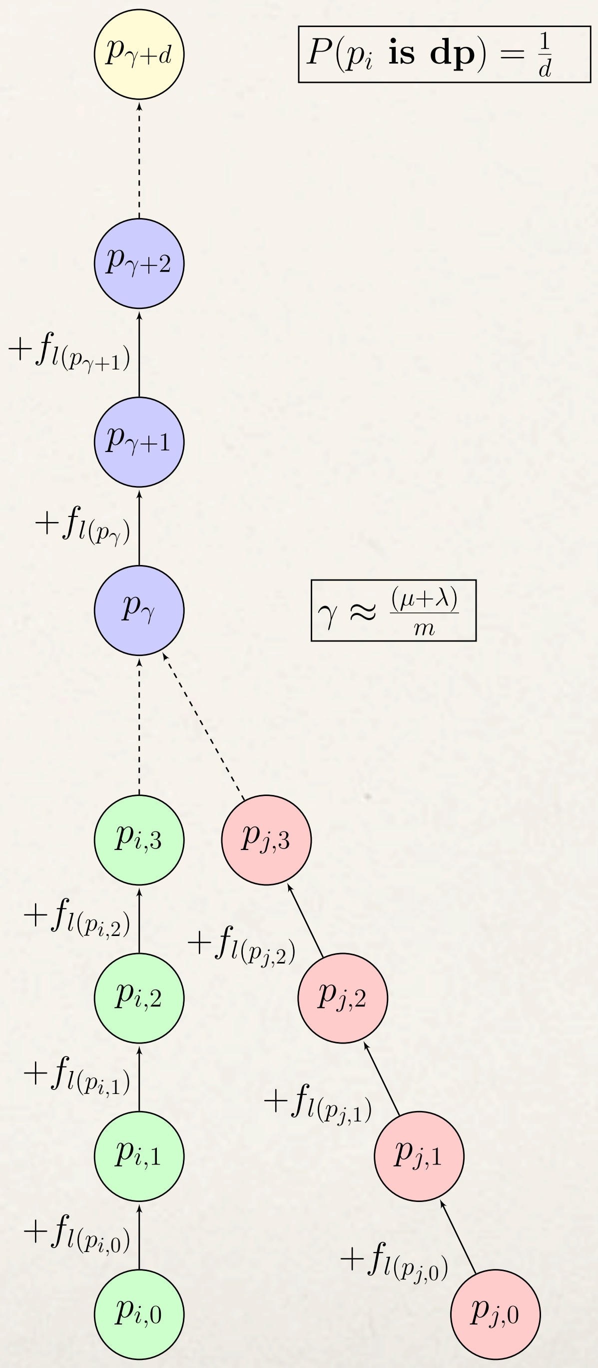


Parallelizable Pollard's rho [VOW97]

- Run **m** independent adding walks using the same table.
  Define set of **distinguished points** (easy to check property).

- Each node reports **dp's** to central node that checks for **dp** collision (**m**-fold speed-up if run on **m** nodes).

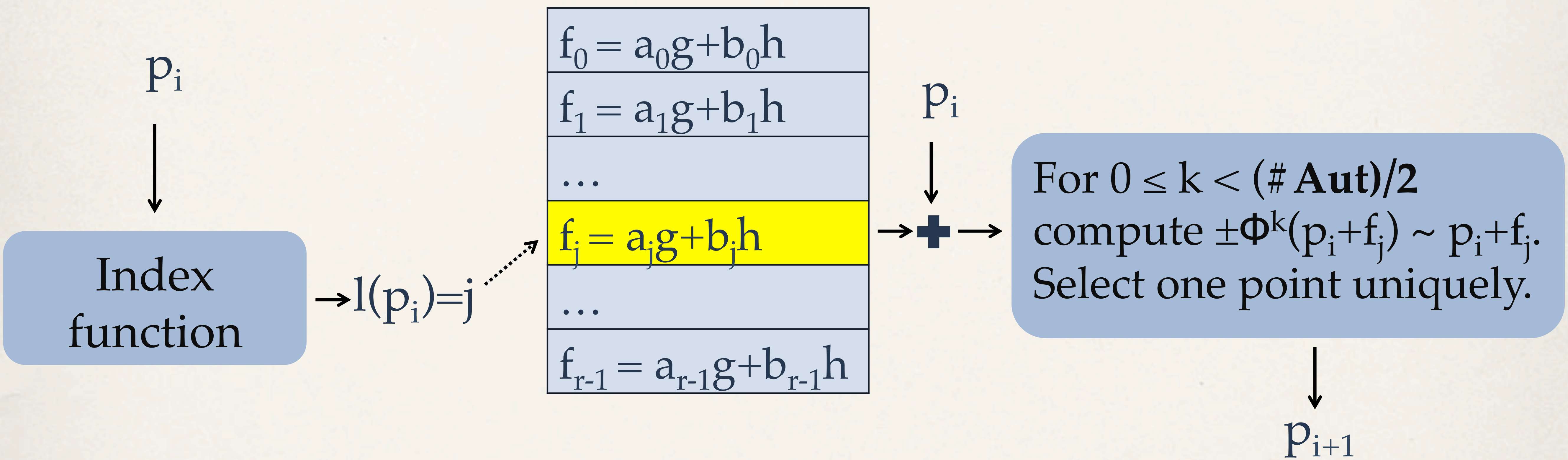
- Simultaneous inversion trick [M87]: (m)**inv**=3(m-1)**mul**+1**inv**.
  Extra steps due to dp's: ≈ **dm**.



Using automorphisms [WZ99],[DGM99]

- The group of curve automorphisms define equivalence classes of points. The size of an equivalence class is the size of the **Aut** group
- Idea: search for collision of equivalence classes of size # **Aut**
- If # **Aut** = **c** the search space is reduce by a factor **c** ($\sqrt{c}$ speed-up)
- Ex., negation map: p ~ -p, search for collision of ±p ($\sqrt{2}$ speed-up)
- # Aut for cryptographically interesting curves over prime fields
  Elliptic curves: **min=2, max=6**
  Genus 2 Hyperelliptic curves: **min=2, max=10**

Adding walk with automorphisms



Selection (remark: **-**(x,y)=(x,-y) on **E**, $-(u_1,u_0,v_1,v_0)=(u_1,u_0,-v_1,-v_0)$ on **Jac(C)**)

1. \# **Aut = 2:** choose point with odd value in **y** ($\mathbf{v_1}$) coord.
2. \# **Aut > 2:** choose $\pm\Phi^k(p_i+f_j)$ with least value in **x** ($\mathbf{u_1}$) and odd value in **y** ($\mathbf{v_1}$).

Selected curves: iteration cost

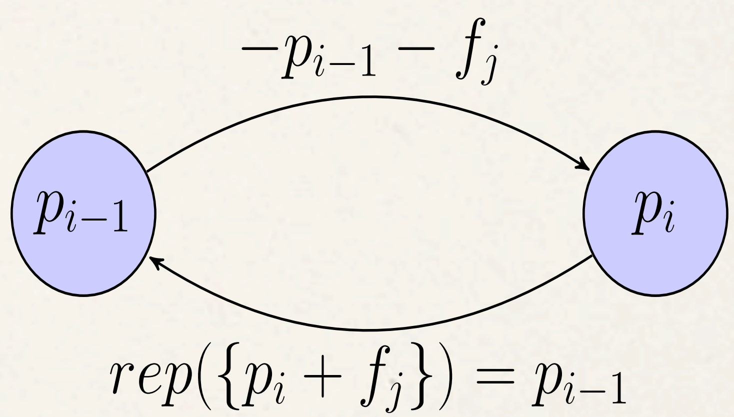
NISTp-256 $\sqrt{2}$ **- (neg)**: **(x,y) -> (x,-y)** Aut: **{id,-}** Regular iteration: **6m** Aut overhead: **negligible** Slowdown factor: **1**	**BN254** $\sqrt{6}$ $\pm\phi^i$: **(x,y) -> (ξ^ix, $\pm$y), ξ^3=1 mod p** Aut: **{id, -, -ϕ, ϕ, -ϕ², ϕ²}** Regular iteration: **6m** Aut overhead: **1m** Slowdown factor: **0.857**
Generic-1271 $\sqrt{2}$ **- (neg)**: **$(u_1,u_0,v_1,v_0)\to(u_1,u_0,-v_1,-v_0)$** Aut: **{id,-}** Regular iteration: **24m** Aut overhead: **negligible** Slowdown factor: **1**	**GLV4-BK** $\sqrt{10}$ $\pm\phi^i$: **(u_1,u_0,v_1,v_0) -> $(\xi^i u_1, \xi^{2i}u_0, \pm\xi^{4i}v_1, \pm v_0)$, ξ^5=1 mod p** Aut: **{id, -, -ϕ, ϕ, …, -ϕ⁴, ϕ⁴}** Regular iteration: **24m** Aut overhead: **6m + (1/5)m** Slowdown factor: **0.795**

Fruitless cycles

- Adding walk with automorphisms: **fruitless cycles**
- Fruitless cycle sizes: all multiples of primes dividing **c=# Aut**
- **The shorter the more likely…** Most frequent: 2-cycles, **P=1/(cr)**
- The larger r, the less likely are the cycles, but will eventually occur…

2-cycle example

After computing $l(p_{i-1}) = j$ **and** $p_{i-1}+f_j$ **assume (1):** $rep\{p_{i-1} + f_j\} = -p_{i-1} - f_j$



If (2): $l(p_i) = j$ **then (3):** $p_{i+1} = p_{i-1}$

$P(\textbf{(1)}) = 1/c$ **and** $P(\textbf{(2)}) = 1/r$ **so**
$P(\textbf{(3)}) = P(\textbf{(1)}) \cdot P(\textbf{(2)}) = 1/(cr)$

Cycle reduction, detection and escape

- **Detection and escape by doubling a point in the cycle**
  (lcm): After α iterations record point **p**. After β more iterations check if current point is equal to **p**. Detects cycles of length divisible by β

  (trail): After α iterations record **trail** of β points. Look for collision. Detects cycles of length divisible by 2 up to β.

- **Reduction**
  No: just detect and escape more often. Good for SIMD archs [BLS11].

  Extra table: f'_i for $0 \leq i < r$. If $l(p_i) = l(p_{i+1}) = k$, set $p_{i+1} = p_i + f'_k$. **$P = 1/(cr^3)$.**

- **Best combination depends on architecture used…**
  Analysis of overhead given memory constraints + tests

Performance using automorphisms

Automorphisms	r	#walks
Without	32	2048
With	1024	2048

Curve	Ideal speed-up	Updated speed-up	Measured speed-up[1]	Core-years[1]	Relative security
NIST CurveP-256	$\sqrt{2}$	$\sqrt{2}$	$0.947\ \sqrt{2}$	3.946×10^{24}	**128.0**
BN254	$\sqrt{6}$	$0.857\sqrt{6}$	$0.790\ \sqrt{6}$	9.486×10^{23}	125.9
Generic 1271	$\sqrt{2}$	$\sqrt{2}$	$0.940\ \sqrt{2}$	1.736×10^{24}	126.8
4GLV127-BK	$\sqrt{10}$	$0.795\sqrt{10}$	$0.784\sqrt{10}$	1.309×10^{24}	126.4

[1]Intel Core i7-3520M (Ivy Bridge), 2893.484 MHz

Conclusions

- In all cases automorphisms can be profitably used in practice, but the ideal speed-up is not achieved due to increased iteration complexity.
- Better understanding of the practical trade-off in the case of genus 2 hyperelliptic curves and elliptic curves with $\#\mathrm{Aut} > 2$, like BN254.
- Useful analysis when constant factors matter, e.g., solving ECDLP challenges.