BN曲線上のOptimal Ateペアリングのソフトウェア実装

光成滋生（サイボウズ・ラボ）

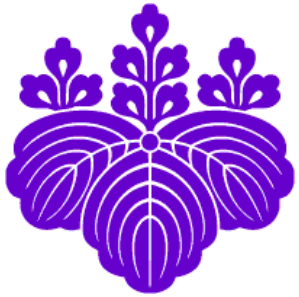
共同研究者

Jean-Luc Beuchat

照屋 唯紀

岡本 栄司

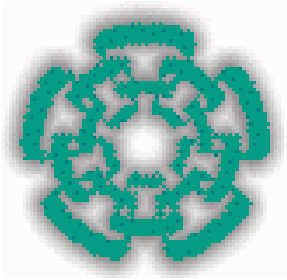
- 以上，筑波大学



Jorge Enrique Gonzalez Diaz

Francisco Rodriguez-Henriquez

- 以上，Instituto Politecnico Nacional



暗号で使われるペアリング

非退化双線形写像 $f: G_1 \times G_2 \rightarrow G_3$

- G_1, G_2：楕円曲線上のn等分点のなす群
- G_3： 1の巾乗根のなす群への写像

Weilペアリング以降，扱いやすさ，効率のよさを求めて多くの定義が派生した

安全性

各暗号の安全性とサイズ(bit)の比較

共通鍵暗号	RSA	楕円素体
72	1024	138
108	2048	206
128	2832	245

- 楕円曲線とRSA暗号の安全性比較(富士通研究所：2010/8/20)より

以前は楕円曲線暗号は160bit程度が主流だったが，最近はより高い安全性を求める傾向にある

既存の成果（128bit安全）

F_3^{509}上のetaTペアリング（我々）

- Core i7で15.01Mclk
  - 806bitで共通鍵暗号128bit相当の安全性（鍵サイズが非効率）
  - SSSE3を使い，Intel CPU専用

R-ateペアリング（Hankerson）

- Core 2 Duoで10Mclk

BN曲線上のoptimal ateペアリング（Naehrig）

- Core 2 Quadで4.47Mclk
  - SSE2を使う，AMD系でも動くが遅い（11.27Mclk）

今回の成果

126bitBN曲線上のoptimal Ateペアリング

- ターゲットはx64-86アーキテクチャ
- Core i7 2.8GHz, Athlon 64 X2 6000+などのCPU上で1msec以下の速度を実現
  - i7で2.33M, Opteronで2.40M, Core2で2.95M

128bit相当の安全性を持つペアリング演算で1msec以下の速度を実現したのはハードウェア，ソフトウェアに関わらず初めて

BN曲線

128bit安全性に適した曲線

- $E: y^2 = x^3 + b, b \in F_p$
- $p := p(t) = 36t^4 + 36t^3 + 24t^2 + 6t + 1$
- $r := r(t) = 36t^4 + 36t^3 + 18t^2 + 6t + 1$
- 埋め込み（拡大）次数 $k = 12$

記号

- $\pi: (x, y) \to (x^p, y^p)$: Frobenius写像
- $f_{s,Q}: \left(f_{s,Q}\right) = s(Q) - ([s]Q) - (s-1)(\mathcal{O})$となる関数
- l_{Q_1,Q_2} : Q_1とQ_2の加算に対応する直線の式

optimal Ateペアリング

$$G_1 = E[r] \cap \ker(\pi_p - [1]) = E(F_p)[r]$$

$$G_2 = E[r] \cap \ker(\pi_p - [p]) \subseteq E(F_p{}^{12})[r]$$

$$G_3 = \mu_r \subset (F_p{}^{12})^*$$

$$e : G_2 \times G_1 \ni (Q, P) \longmapsto m(Q, P)^{\frac{p^{12}-1}{r}} \in G_3$$

- $m(Q, P) := f_{6t+2,Q}(P) \cdot g_Q(P)$
- $g_Q(P) := l_{[6t+2]Q,\pi_p(Q)}(P) \cdot l_{[6t+2]Q+\pi_p(Q),-\pi_p^2(Q)}(P)$

体と曲線の構成

$t = 2^{63} - 2^{54} + 2^{44}$を採用

- このときpとrは254bit素数（126bit安全性）

拡大体の既約多項式

- $F_p{}^2 = F_p[u]/\,(u^2 - \beta), \beta = -5$
- $F_p{}^6 = F_p{}^2[v]\,/\,(v^3 - \xi), \xi = u$（予稿集から改善）
- $F_p{}^{12} = F_p{}^6[w]/\,(w^2 - v)$

楕円曲線とその6次twist

- $E : Y^2 = X^3 + 5$
- $E' : Y^2 = X^3 + 5\,/\,\xi$

ペアリングのアルゴリズム

1）$[6t+2]Q$ と $f_{6t+2,Q}(P)$ を算出（Millerループ）

2）$m(Q,P) = f_{6t+2,Q}(P) \cdot g_Q(P)$を算出

3）$\frac{p^{12}-1}{r}$乗する（最終巾）

最終巾では$F_p{}^{12}$の部分群

$$G_{\Phi_6}(F_p{}^2) = \{a^{p^4-p^2+1} = 1\}$$を利用

- 逆元が共役操作と同値

巾乗は符号つき2進数展開を利用

- $t, 6t+2$のハミング重みは3と7

F_p^2上の演算

基本戦略は乗算回数を減らすこと

乗算

- $(a+bu)(c+du) =$
  $(ac-5bd) + \big((a+b)(c+d) - ac - bd\big)u$

平方

- $(a+bu)^2 = \big((a-b)(a+5b) - 4ab\big) + (2ab)u$

逆元

- Itoh-Tsujiiアルゴリズム

$F_p{}^6, F_p{}^{12}$上の演算

$F_p{}^6$

- 乗算：Karatsuba法
- 平方：SQR2法（Chung, Hasan@ARITH2007）
- 逆元：Scott（Pairing2007）による方法

$F_p{}^{12}$

- 乗算，平方，逆元は$F_p{}^2$と同じ
- 最終巾では$(p^6-1)(p^2+1)$乗して$G_{\Phi_6}(F_p{}^2)$の元に移す

$G_{\Phi_6}(F_p{}^2)$

- 乗算：$F_p{}^{12}$と同じ
- 平方: Granger, Scott（ePrint 2009/565）による方法
- 逆元: 共役操作に等しい

演算回数比較

		F_p^2 の乗算	F_p^2 の平方	F_p^2 の加算	ξ との乗算
Hankerson R-ate	Miller loop	2277	356	6712	412
	final exp	1616	1197	8977	1062
Naehrig optimal ate	Miller loop	2022	590	7140	410
	final exp	678	1719	7921	988
我々の成果 optimal ate	Miller loop	1954	568	6912	400
	final exp	429	1719	7021	898

乗算回数を13.3%, 加算回数を8%削減

既存の基礎体の乗算方法

Fanら

- Hybrid Modular乗算(HMMB)
- pがtの多項式$p(t)$で, $p^{-1}(t) = 1 \bmod t$を利用

	32x32	32x64	64x64	10x10程度
Montgomery			36	
HMMB	1	8	16	13

Naehrigら

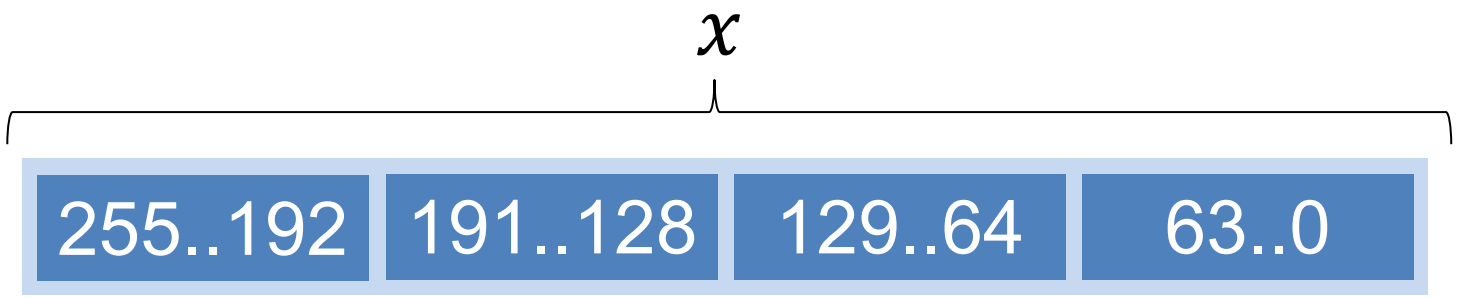
- 多項式の係数を6個のSSEレジスタで表現
  - Bernsteinの浮動小数点数を使う方法
- Karatsuba法ベース

我々の方針

高速な64bit x2→128bit乗算命令（mul）に着目

- 加算に対する乗算の相対コストは低い
- シンプルなアルゴリズム，pのビット長のみに依存

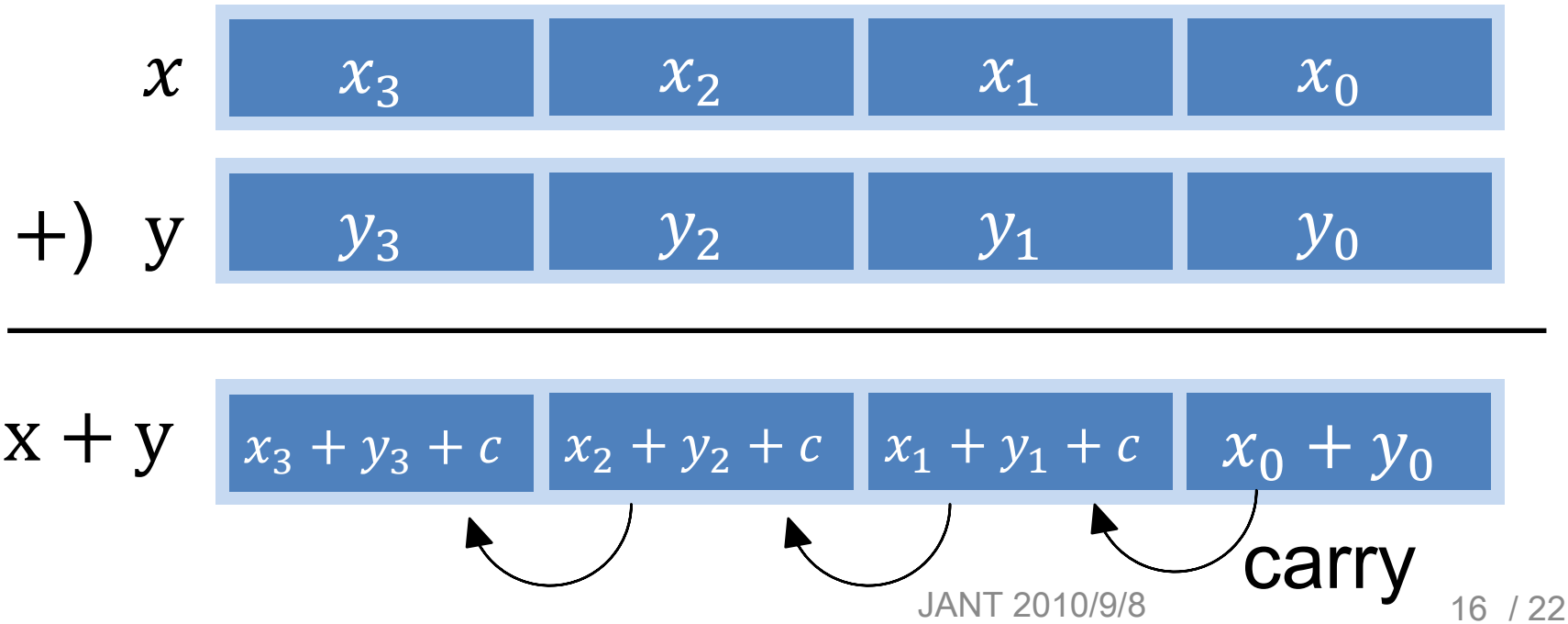
$p \fallingdotseq$ 256bitなので
F_pの元xを4個の64bit整数に分割して表現



F_pの加減算

加算(減算)

- 256bit進数として繰り上がり加算(減算)
- mod$p$:結果がp以上ならpを引く(負ならpを足す)

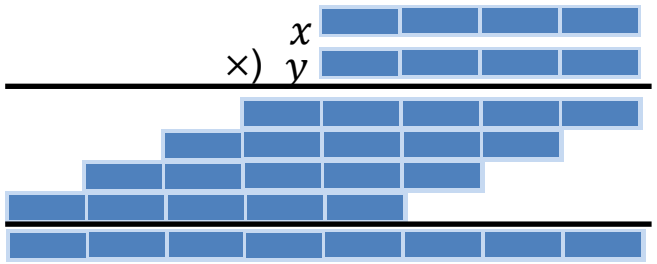


F_pの乗算

2^{64}のMontgomery乗算

- 但し後述の理由により乗算を二つの関数に分離
  - mul256（256×256→512bit整数乗算）: 60clk
  - mod512（512→256bit reduction）: 100clk
    - $\mathrm{mont}(x, y) = \mathrm{mod512}(\mathrm{mul256}(x, y))$

mul256は筆算方式を採用



- mul16回
- Karatsuba法（mul 9回）は遅かった
  - 加算回数が増えるためトータルで不利

$F_p{}^2$上の乗算の改良

F_p上の乗算

- $x \leftarrow ac,\ y \leftarrow bd,\ z \leftarrow (a+b)(c+d)$の3回
- $s \leftarrow x - 5y,\ t \leftarrow z - x - y$

mod512の回数を減らす（3回→2回）

- $ac,\ bd,\ (a+b)(c+d)$の乗算をmul256で止める
- 512bit整数のまま$s,\ t$相当のものを求める
- 最後にmod512を呼ぶ

$F_p{}^2$上の平方の改良

$F_p{}^2$上の平方

- $x \leftarrow \underbrace{(a-b)}(a+5b)\underbrace{-4ab}, y \leftarrow 2ab$

mod$p$が必要

mod512の定義域は$[0, pN), N := 2^{256}$

- （今回の）pに対して$7p < N$が成立
  - 7倍までの値はmod$p$しなくてもmod512を適用可能
- $0 \leq (a+p-b)(a+5b) - 4ab < pN$
- $a+p-b, a+5b, -4ab$の計算にmod$p$は不要

（小さな値の）mod$p$の高速化

乗算,平方の計算には小さい値のmod$p$が頻出

- 分岐のない高速なものがほしい

ip for $i \in [0, 8]$を観察

- $[ip \,/\, 2^{253}] = i$が成立

テーブルを用意

- $\mathrm{Tbl}[i] = ip$ for $0 \leq i \leq 8$
- $|x - \mathrm{Tbl}[x \gg 253]| < p$ for $x \in [0, 8p]$
- シフト/テーブル引き/減算で$(-p, p)$の範囲に入る

ベンチマーク

	i7	Opteron	Core 2 Duo	Athlon 64
$F_p{}^2$上の乗算		695	693	1714
	435	443	558	473
$F_p{}^2$上の二乗		614	558	1207
	342	355	445	376
Millerループ		2.48M	2.26M	5.76M
	1.33M	1.36M	1.68M	1.48M
最終巾		2.52M	2.21M	5.51M
	1.00M	1.04M	1.27M	1.15M
ペアリング	4.00M	5.00M	4.47M	11.27M
	2.33M	2.40M	2.95M	2.63M

上段はNaehrigらの結果,下段は我々の結果

処理clk数（小さいほど速い）

参考文献

High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves

- http://eprint.iacr.org/2010/354
- Pairing 2010で発表予定

ソースコード

- http://homepage1.nifty.com/herumi/crypt/ate-pairing.html