F_{2^m}上の楕円曲線ペアリングのJava実装

筑波大学 システム情報工学研究科
リスク工学専攻 暗号・情報セキュリティ研究室
張一凡、金山直樹、岡本栄司

発表内容

- Pairingとは
- 研究背景
- 研究目的
- 楕円曲線
- 研究内容
  - 研究目標
  - 実装内容
- パフォーマンス評価
  - Pairing
  - 有限体演算
- 考察
- まとめ
- 今後の課題

Pairingとは？

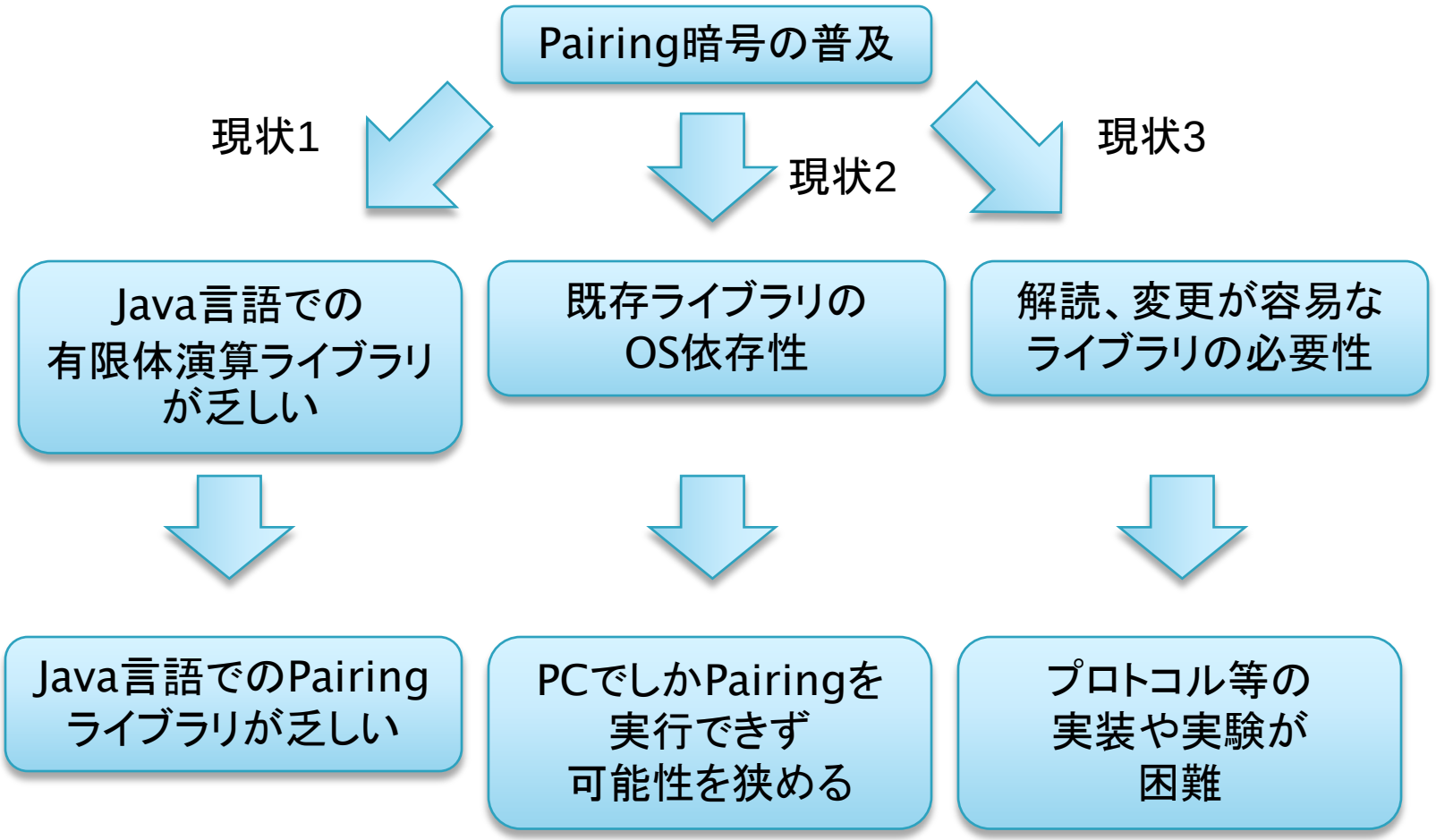
- Pairingとは？
  - 2入力1出力の一方向性関数

    $e(P,Q) \to \mu$ P,Q楕円曲線の点、μ有限体の元
  - 双線形性

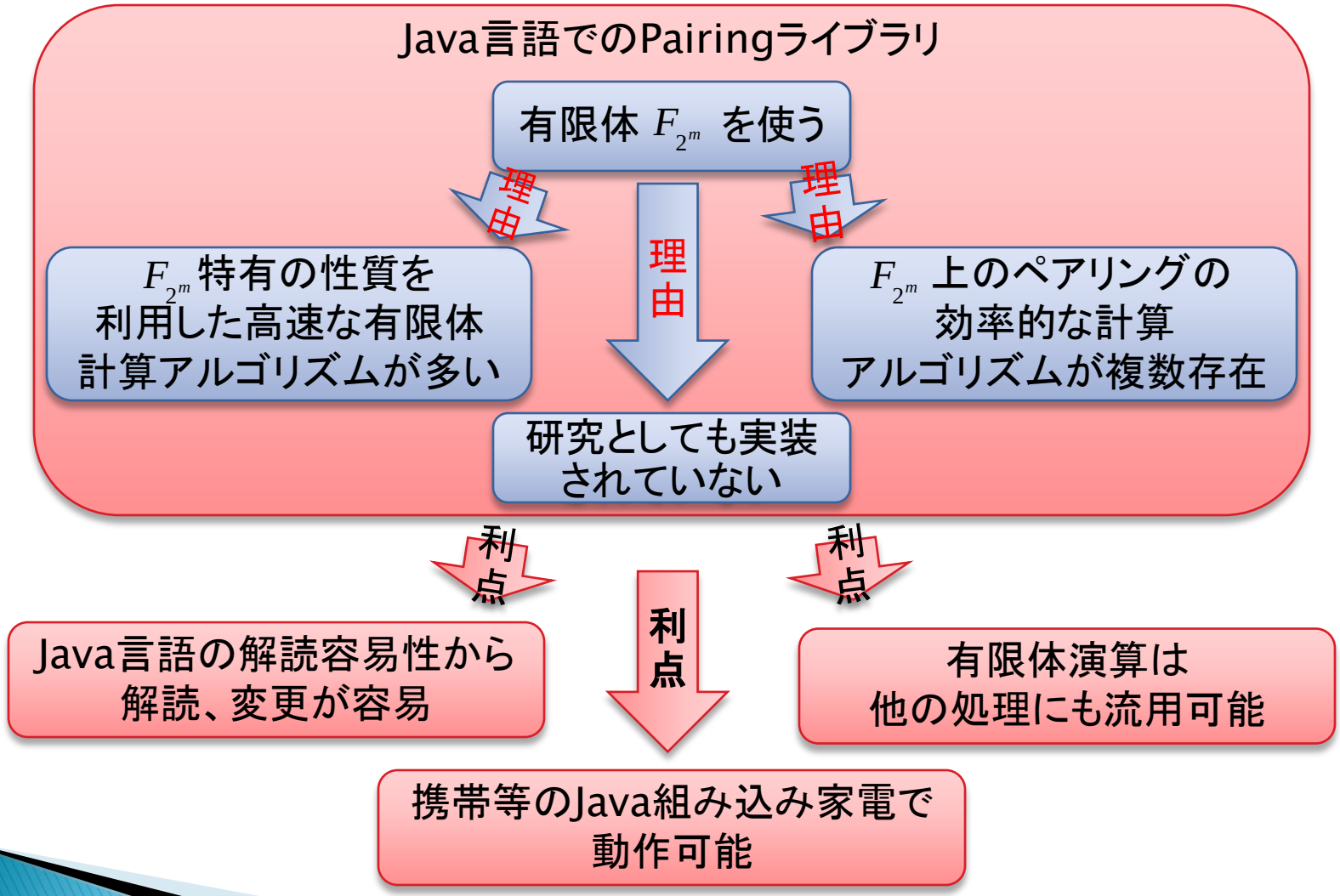
    $$e(aP,bQ) = e(aP,Q)^b = e(P,bQ)^a = e(P,Q)^{ab}$$

- Pairingの利点・需要
  - Jouxの「三者間鍵共有」(2000)
  - Sakaiらの「IDベース暗号」(2000)
  - Bonehらの「ショート署名方式」(2001)

研究背景


Pairing暗号の普及
現状1
現状2
現状3
Java言語での
有限体演算ライブラリ
が乏しい
既存ライブラリの
OS依存性
解読、変更が容易な
ライブラリの必要性
Java言語でのPairing
ライブラリが乏しい
PCでしかPairingを
実行できず
可能性を狭める
プロトコル等の
実装や実験が
困難

研究目的


Java言語でのPairingライブラリ
有限体 F_{2^m} を使う
理由
理由
理由
F_{2^m} 特有の性質を
利用した高速な有限体
計算アルゴリズムが多い
F_{2^m} 上のペアリングの
効率的な計算
アルゴリズムが複数存在
研究としても実装
されていない
利点
利点
利点
Java言語の解読容易性から
解読、変更が容易
有限体演算は
他の処理にも流用可能
携帯等のJava組み込み家電で
動作可能

既存研究

- 標数3のJava実装(2007)
  - 川原、高木ら
  - 携帯向けJavaPairing実装
  - PentiumM　1.73GHz with J2SE　10.15msec
  - FOMA SH901iS J2ME 509.22msec

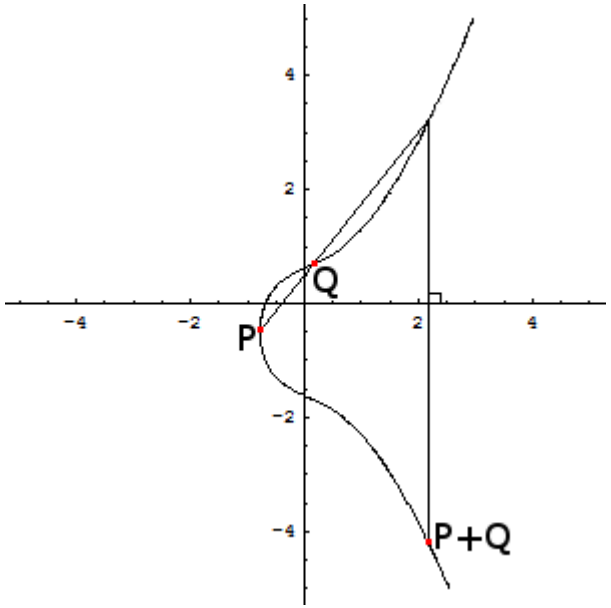
- η_TPairingアルゴリズム(2005)
  - Barreto,Galbraithら
  - 956bit安全性で1.70msec(標数2楕円曲線)
    - Pentium4　3GHz　with SSE2多倍長演算

楕円曲線

- 以下の式が表わす曲線

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad a_i \in F_q$$

- 曲線上の点に対して
  加法、スカラー倍が定義できる
- 本研究では特に超特異曲線
  $y^2 + y = x^3 + x + b \quad b \in \{0,1\}$
  を用いる



研究目標

- Pairingのライブラリを作成
  - 時間のかかるPairing計算を高速化する
  - Pairingの高速化のため有限体演算も最適化する
  - ライブラリに拡張性を持たせる

- 有限体F_{2^m}を用いたPairingの性能評価
  - 高速なPairingアルゴリズムとして知られるη_T Pairingを評価
  - 有限体の演算を評価

実装内容

- F_{2^m}の有限体演算を実装
  - ${}^{2}F_{2^m}$ の元は0，1で表現できる

$$F_{2^m} \leftrightarrow \{a_{m-1}a_{m-2}\cdots a_1a_0 : a_i \in \{0,1\}\}$$

同一視

計算機のビット表現に対応

  - 有限体 $F_{2^m}, F_{2^{mk}}$ の元の表現
    - mビットの元の格納にJavaの標準クラス「BigInteger」でテスト
    - int配列によるmビットのF_{2^m} 元を格納
  - 有限体演算
    - 加算、乗算、逆元計算、平方計算、平方根計算
    - 拡大体演算の実装
- Pairing実装
  - 楕円曲線において η_T Pairingを実装

Pairing計算(MillerLoop)

Algorithm 1 supersingular 楕円曲線における Tate ペアリング.

Input: $P, Q \in E(\mathbb{F}_{2^m})$
Output: $e(P,Q) \in \mathbb{F}_{2^{4m}}$

$r = (r_{t-1}, \ldots, r_0)$, $f = 1$, $V = P$
for $i = t-1$ to 0 **do**
  $f \leftarrow f^2 \cdot l_{V,V}(\phi(Q))$
  $V \leftarrow 2V$
  if $r_i = 1$ **then**
    $f \leftarrow f^2 \cdot l_{V,P}(\phi(Q))$
    $V \leftarrow V + P$
  end if
end for
return $f^{\frac{q^k-1}{r}}$

- r:Pの位数のビット列
- f: 下記のPairingの出力の $f_P(Q)$ 部分
- $l_{a,b}(Q)$:楕円の二点a,bを通る直線に$\psi(Q)$を代入して得られる答え

$$e(P,Q) = f_P(Q)^{\frac{q^k-1}{r}}$$

Pairing計算(etaT Pairing)

Algorithm 1 楕円曲線上の η_T ペアリング.

Input: $P, Q \in E(\mathbb{F}_{2^m})$

Output: $e(P,Q) \in \mathbb{F}_{2^{mk}}$

$P = (x_P, y_P)$, $Q = (x_Q, y_Q)$

$u \leftarrow \lambda$

$f \leftarrow y_Q + u(x_Q + x_P + 1) + y_P + \epsilon + s(x_Q + u) + t$

for $i = 0$ to $(m-1)/2$ **do**

$u \leftarrow x_P + v_1$

$x_P \leftarrow \sqrt{x_P}$, $y_P \leftarrow \sqrt{y_P}$

$g \leftarrow u(x_P + x_Q + v_1) + y_P + y_Q + (1 - v_1)x_P + v_2 + s(u + x_Q) + t$

$f \leftarrow f \times g$

$x_Q \leftarrow x_Q^2$, $y_Q \leftarrow y_Q^2$

end for

return $f^{(2^{2m}-1)(2^m \mp 2^{(m+1)/2}+1)}$

- f: 下記のPairingの出力の $f_P(Q)$ 部分
- $v_1, \lambda, \varepsilon$ はmによって決まる定数

$$< P, Q >_N^{(q^k-1)/N} = f_P(Q)^{\frac{q^k-1}{N}}$$

パフォーマンス(有限体)

- 本研究でのライブラリによる有限体計算時間[nsec]

F2m	m=79	時間比	m=241	時間比
addition	30	1.000	48	1.000
multiplication	2268	75.60	5719	119.145
square	137	4.566	252	5.250
square root	2141	71.366	5683	118.395
inversion	17083	569.433	71388	1487.250
modulus	96	3.200	192	4.000

時間比:additionを1とした時の値

CPU	Core2Duo T7400
memory	DDR2-SDRAM 2GB
OS	Windows Vista
Language	Java1.6.0_03

パフォーマンス(Pairing)

- 本研究でのライブラリによるPairing計算時間[msec]

main loop	8.51	94%
final exponentiation	0.47	5%
all pairing	9.07	100%

安全性　964bit

- 振り分け
  - main loop
  - final exponentiation

Pairing計算式 $e(P,Q) = f_p(Q)^{\frac{q^k-1}{r}}$

CPU	Core2Duo T7400
memory	DDR2-SDRAM 2GB
OS	Windows Vista
Language	Java1.6.0_03

考察

- BigInteger依存の実装結果は期待していた実行速度[約10msec]より十数倍遅い
- int実装によって高速化できたが有限体演算に依存して時間がかかる。現状ではまだ最適化の余地がある
- 最終べきを高速化すればパフォーマンスが向上する可能性がある

まとめ

- 本研究では汎用PCで10msecを切るスピードでpairing計算を行うJavaライブラリを実装できた
- 外部ライブラリに依存しないため、有限体、Pairingともに新しいアルゴリズムの性能評価を簡単に行える
- Pairing演算と有限体演算を分別化し、アセンブリ編集を行わなかったためにアルゴリズムの改良に伴い簡単にライブラリの再構築を行える

今後の課題

- Pairingライブラリの性能向上
  - int構成による有限体演算の高速化手法を検討
- Pairingのアルゴリズムの高速化
  - η_T Pairingを実装に適したアルゴリズム的な高速化を行い、ライブラリに組み込む
  - 最終べきのさらなる高速化手法を検討
- ライブラリの充実化
  - ライブラリ上で未実装のPairingアルゴリズムをライブラリ組み込む

ご清聴ありがとうございました