IPA
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN

2004情財646号

情報セキュリティ対策事業

ストリーム暗号安全性検証用
グレブナー基底計算プログラムの開発

開発成果報告書

2005年2月

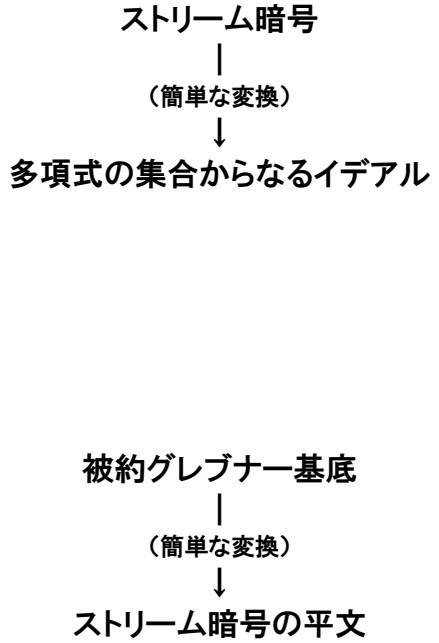
独立行政法人 情報処理推進機構

I. ソフトウェア概要編

1. 本ソフトウェアの目的と概観

本ソフトウェアは、ストリーム暗号のalgebraic attackに対する安全性評価に役立て、我が国のセキュリティ評価技術の向上に資することを目的として、下記を可能にするものである。

a. ストリーム暗号より変換された、$F_2[X]/(X^2+X)$上多項式の集合からなるイデアルの生成元に対する、被約グレブナー基底を算出すること。算出された被約グレブナー基底は、簡単な変換によりストリーム暗号の平文となる(下図)。



【図：赤矢印が、本ソフトウェアが可能にすること】

b. IPAのグリッドコンピュータ `murasaki.ipa.go.jp` にて協調動作をすること。

2. 本ソフトウェアを構成するプログラム一覧

本ソフトウェアを構成するプログラムの一覧を示す。なお、プログラム間の構成は発注仕様書の通りである。

a. 仕様上のプログラム(納品物件のプログラム)一覧

本ソフトウェアは、発注仕様書に記載のある各機能を実現するため、下記のプログラムを含む。

(1) 各アルゴリズム実行プログラム一覧

ア. F4アルゴリズム実行総括プログラム

本プログラムは、発注仕様書4.1の機能を実現する。

イ. F5アルゴリズム実行総括プログラム

本プログラムは、発注仕様書4.2の機能を実現する。

(2) サブプログラム一覧

ウ. 入力解析プログラム

本プログラムは、エ. 出力プログラムと合わせて、発注仕様書4.1(1)及び4.2(1)の機能を実現する。

エ. 出力プログラム

本プログラムは、ウ. 入力解析プログラムと合わせて、発注仕様書4.1(1)及び4.2(1)の機能を実現する。

オ. 各演算エンジン提供プログラム

本プログラムは、発注仕様書4.1(4)及び4.2(4)の機能を実現する。

カ. F4アルゴリズム実行スケジューラプログラム

本プログラムは、発注仕様書4.1(2)の機能を実現する。

キ. F4アルゴリズム実行プログラム

本プログラムは、発注仕様書4.1(3)の機能を実現する。

ク. F5アルゴリズム実行スケジューラプログラム

本プログラムは、発注仕様書4.2(2)の機能を実現する。

ケ. F5アルゴリズム実行プログラム

本プログラムは、発注仕様書4.2(3)の機能を実現する。

b. 自動単体試験を目的としたプログラム(参考資料となるプログラム)一覧

本ソフトウェアの仕様上の構成部品ではないが、a.(2)で示した各サブプログラムの単体試験を自動化することを目的とした下記に示すプログラムを作成した。

コ. 入力解析プログラム試験プログラム
本プログラムは、a.ウに示すプログラムの単体試験を行なう。

サ. 出力プログラム試験プログラム
本プログラムは、a.エに示すプログラムの単体試験を行なう。

シ. 各演算エンジン提供プログラム試験プログラム
本プログラムは、a.オに示すプログラムの単体試験を行なう。

ス. F4アルゴリズム実行スケジューラプログラム試験プログラム
本プログラムは、a.カに示すプログラムの単体試験を行なう。

セ. F4アルゴリズム実行プログラム試験プログラム
本プログラムは、a.キに示すプログラムの単体試験を行なう。

ソ. F5アルゴリズム実行スケジューラプログラム試験プログラム
本プログラムは、a.クに示すプログラムの単体試験を行なう。

タ. F5アルゴリズム実行プログラム試験プログラム
本プログラムは、a.ケに示すプログラムの単体試験を行なう。

II. 試験報告編

1. 結論

2005年2月24日をもって、本ソフトウェアの試験は合格とする。

2. 試験方法

a. 各サブプログラム(ソフトウェア概要編「2.ウーケ」)の試験(単体試験)

各サブプログラムの試験は、取扱説明編「3.試験方法」の手順に従って行なう。

b. 上位プログラム(ソフトウェア概要編 「2.アーイ」)の試験(結合試験)

上位プログラムの試験は、幾つかの問題を入力し、既存の同機能のソフトウェアの出力と比較することによって行なう。また、理論的に解の求まる入力に関しては出力結果を目視で確認する。

3. 試験結果

a. 各サブプログラムの試験

ソフトウェア概要編 「2.ウーケ」の各プログラムに対し、取扱説明編「3.試験方法」の手順に従って試験プログラムを実行し、結果が全て成功していることを確認した。

b. 上位プログラムの試験

ソフトウェア概要編「2.アーイ」の各プログラムに対し、P1 P2 P3 D10 D12 D13 D14 出力結果の比較を行ない、出力結果が正しいことを確認した。各入力ファイルは同梱されている。また、C8 C10 C12 C14 C16の各入力に関して出力結果が正しいことを確認した。これらの各入力ファイルを生成するプログラムが同梱されている。

III. 取扱説明編

0. 動作環境

本ソフトウェアは、以下の環境で動作する。

a. IPAのグリッドコンピュータ murasaki.ipa.go.jp上

b. Unix系OS(Linux, Darwinなど)の動作するPC(Pentium PCなど)

1. 設置方法

a. ソースファイルの展開及びコンパイル、及びバイナリプログラムの設置

(1) ソースファイルを展開する

同梱ファイル src.tar.gzを murasaki.ipa.go.jp上でtarプログラムなどを用いて展開する。コマンド入力例を下記に示す。なお、「%」はコマンドシェルのプロンプトであり、入力の必要はない。

```
% tar zxvf src.tar.gz
```

(2) プログラムをコンパイルする。

(1)で作成されるディレクトリ hotaru に移動し、app、para、para_hfeの各ディレクトリにてmakeを実行する。コマンド入力例を下記に示す。

```
% cd hotaru
% make -C app
% make -C para
% make -C para_hfe
```

※ (1)(2)の操作は、同梱の実行モジュールbin.tar.gzを展開することで変えることが出来る。

(3) 以上の操作で、下記の実行モジュールが生成される。

app/console	単体ノード用プログラム
para/console	並列動作用メインプログラム
para/child	並列動作用サブプログラム
para_hfe/console	並列動作用メインプログラム(HFE用協調方式)
para/child	並列動作用サブプログラム(HFE用)

(4) コンパイルオプション

以下は、app/console.cpp para/console.cpp para_hfe内に定義されるコンパイルオプションである。

USE_HFE (HFE暗号解読用の最適化を用いる、USE_F5と共存不可)
USE_F4 (F4アルゴリズムを用いる)
USE_F5 (F5アルゴリズムを用いる USE_HFEと共存不可)

b. 並列動作のための事前準備

para/console、para_hfe/consoleを使用する際には、使用するノード上でそれぞれ、para/child、para_hfe/childを起動しておく必要がある。詳細については「2. 起動方法」を参照。

2. 起動方法

a. 各プログラムの起動引数

(1) app/console

引数は指定しない。以下に起動例を示す。

```
% cd app
% ./console
```

(2) para/console

使用するPEの最初のID番号と、使用しないPEの最初のID番号を指定する。以下に起動例を示す。この例では、ID番号 0 1 2 3 4 5 6 7 8 9のchildサブプログラムが計算に使用される。

```
% cd para
% ./console 0 10
```

※ PE(processor element)のID番号は使用するグリッドマシン上のnode番号とは関係なく、0以上1023以下の値を指定することが可能である。

(3) para_hfe/console

使用するPEの数のレベルを1以上、10以下の値で指定する。使用されるPEの数は、2の(指定した数)乗となる。また、使用されるPEのIDは0から順に整数値が使用される。以下に起動例を示す。この例では0以上63以下のIDを持つchildサブプログラムが計算に使用される。

```
% cd para_hfe
% ./console 6
```

(4) para/child

PEのID番号を指定する。以下に起動例を示す。この例ではID番号0を持つchildサブプログラムとして起動される。

```
% cd para
% ./child 0
```

(5) para_hfe/child

PEのID番号を指定する。(4)と同様である。

b. 各プログラムの外部入出力

(1) app/console

入力： 標準入力に、イデアルの生成元となる多項式の集合を以下の規則に従って入力する。入力例は、「3. 試験方法」の入力ファイルを参照。

ア. 多項式は複数行に渡って入力することが出来る。

イ. 多項式の終了は「,」(カンマ)で示す。カンマから行末までは無視される。また、最後の多項式だけは終了の際に「,」(カンマ)が無くても良い。

ウ. 多項式集合の終了は、「.」(ピリオド)で始まる行で示す。ピリオドから行末まで

は無視される。
エ. 多項式集合が空集合となるとき、プログラムは終了する。
オ. 以上の規則を満さない入力は無視される。

出力： 下記の内容が、標準出力および標準エラー出力に出力される。
ア. プログラム実行中の中間のグレブナー基底の集合の大きさ、及び現在の計算対象ペアの残数が、標準出力に出力される。
イ. プログラム実行中に確保したメモリプール領域の大きさ、及び計算中の行列の大きさが、標準エラー出力に出力される。
ウ. プログラムの実行結果である、被約グレブナー基底の内容が標準出力に出力される。

(2) `para/console`
入力： `app/console`と同じ。

出力：下記の内容が、標準出力および標準エラー出力に出力される。
ア. プログラム実行中の中間のグレブナー基底の集合の大きさ、及び現在の計算対象ペアの残数が、標準出力に出力される。
イ. 各サブプログラムに、割り当てた計算対象のペアの数、及び各サブプログラムが計算した、新しい多項式の数、及び現在計算中のサブプログラムの数が標準出力に出力される。
ウ. プログラム実行中に確保したメモリプール領域の大きさ、及び最終的に被約グレブナー基底を計算する際の行列の大きさが、標準エラー出力に出力される。
エ. プログラムの実行結果である、被約グレブナー基底の内容が標準出力に出力される。

(3) `para_hfe/console`
入力： `app/console`と同じ。

出力：下記の内容が、標準出力および標準エラー出力に出力される。
ア. プログラム実行中の部分問題に対する被約グレブナー基底および、部分問題を結合した被約グレブナー基底が標準出力に出力される。
イ. 各サブプログラム割り当てた部分問題に対する計算された被約グレブナー基底が標準出力に出力される。
ウ. プログラム実行中に確保したメモリプール領域の大きさ、及び部分問題を結合する際に使用した行列の大きさが、標準エラー出力に出力される。
エ. プログラムの実行結果である、被約グレブナー基底の内容が標準出力に出力される。

(4) `para/child, para_hfe/child`
入力： なし。

出力：以下の内容が標準エラー出力に出力される。
ア．プログラム実行中に確保したメモリプール領域の大きさ、及び計算中の行列の大きさが、標準エラー出力に出力される。

c. 各プログラムの内部通信入出力

(1) app/console

なし

(2) para/console

入力：ID番号nを持つサブプログラムより下記の入力が起動ディレクトリより入力される。
ア．Fn：該当サブプログラムが計算した、新しい多項式の集合。

出力：ID番号nを持つサブプログラムに対する下記の出力が起動ディレクトリに出力される。
ア．Gn：現在計算中の中間のグレブナー基底。
イ．_Gn：ア．の出力途中の中間生成物である。Gnをアトミック(※)に出力するため、_Gnを生成した後Gnに名称変更する。
ウ．Pn：該当サブプログラムに割当てるペアの集合。
エ．_Pn：ウ．の出力途中の中間生成物である。イ.と同様である。

※ 存在しないか、あるいは存在すれば内容が完全であること。

(3) para_hfe/console

入力：ID番号nを持つサブプログラムより下記の入力が起動ディレクトリより入力される。
ア．Gn：該当サブプログラムが計算した、部分問題に対する被約グレブナー基底。

出力：ID番号nを持つサブプログラムに対する下記の出力が起動ディレクトリに出力される。
ア．Fn：ID番号nを持つサブプログラムに割当てる部分問題。
イ．_Fn：ア．の出力途中の中間生成物である。(2)イ.と同様である。

(4) para/child

入力：メインプログラムより下記の入力が起動ディレクトリより入力される。ただし、自身のID番号をnとする。
ア．Gn：現在計算中の中間のグレブナー基底。
イ．Pn：本プログラムに割当てられるペアの集合。

出力：メインプログラムへの下記の出力が起動ディレクトリに出力される。ただし、自身のID番号をnとする。

ア. Fn：本プログラムが計算した、新しい多項式の集合。
イ. _Fn：ア. の出力途中の中間生成物である。(2)イ.と同様である。

(5) para_hfe/child
入力：メインプログラムより下記の入力が起動ディレクトリより入力される。ただし、自身のID番号をnとする。
ア. Fn：本プログラムに割当てられた部分問題。

出力：メインプログラムへの下記の出力が起動ディレクトリに出力される。ただし、自身のID番号をnとする。
ア. Gn：本プログラムが計算した部分問題に対する被約グレブナー基底。
イ. _Gn：ア. の出力途中の中間生成物である。(2)イ.と同様である。

d. 並列処理のための起動の例
(1) para/console
para/consoleを起動するための、childプログラムを起動するスクリプトの例がpara/kick.shにある。また、これらchildプログラムを終了させるためのスクリプトの例が、para/killchild.shにある。これらを使用するためには、同梱のスクリプトlaunchをホームディレクトリに置くこと、また、実行環境を~/hotaru/src/以下に展開することが必要である。また、このスクリプトを利用した場合、node10よりnode49までの40個のノード上の80個のプロセッサに、0から79までのIDが割当てられる。また、このスクリプトを利用した場合、ファイル出力のためのディレクトリは、hotaru/src/hotaru/para/に設定される。

(2) para_hfe/console
para_hfe/consoleを起動するための、childプログラムを起動するスクリプトの例がpara/kick.shにある。また、これらchildプログラムを終了させるためのスクリプトの例が、para_hfe/killchild.shにある。これらを使用するためには、同梱のスクリプトlaunchをホームディレクトリに置くこと、また、実行環境を~/hotaru/src/以下に展開することが必要である。また、このスクリプトを利用した場合、node10よりnode41までの32個のノード上の64個のプロセッサに、0から63までのIDが割当てられる。また、このスクリプトを利用した場合、ファイル出力のためのディレクトリは、hotaru/src/hotaru/para_hfe/に設定される。

e. 並列処理時の注意事項
(1) 実行ディレクトリについて
起動前に実行ディレクトリを決定しておき、そのディレクトリからすべてのサブプログラムとメインプログラムを実行することが必要である。

(2) rshのタイムラグについて

rshプログラムにて各ノード上のサブプログラムを実行する場合は、rshが起動されてからchildプログラムが実行されるまでに時間がかかることがある。従ってchildプログラムを起動してから、実行ディレクトリ内に、para/childの場合はファイルFnが、para_hfe/childの場合はファイルGnが全てのnに関して作成されていることを確認した後、メインプログラムを起動する必要がある。

(3) 実行ディレクトリ内のファイルについて

childプログラム起動以前に、実行ディレクトリ内にFn, Gn, Pn及び、_Fn, _Gn, _Pnという名称のファイルが存在する場合(ただし、nは使用するすべてのID番号)、そのファイルの内容は破壊され、またプログラムが異常動作することがある。従って、起動ディレクトリ内には、F,G,P,_F,_G,_Pで始まる名称のファイルは置かないことが望ましい。

(4) para/console を複数回起動する場合について

para/consoleは、childプログラムの再起動なしに複数回起動することは可能であるが、2回目以降起動する場合は、1回目の起動時に消去されたファイルFnをtouchコマンドなどで作成しておく必要がある。para_hfe/consoleの複数回起動の場合はこのような制限はない。

3. 試験方法

試験方法について説明する。設置方法の(1)に従ってソースプログラムを展開してあるものと仮定する。

a. 各サブプログラムの試験方法

ア. 試験プログラムのコンパイル

testディレクトリ内でmakeコマンドを実行する。以下にコマンドの入力例である。

```
% cd test
% make
```

イ. 試験プログラムの実行

testディレクトリ内でmake testコマンドを実行することにより、すべてのサブプログラムの試験が実行される。以下にコマンドの入力例である。

```
% cd test
% make test
```

b. 上位プログラム試験のための入力ファイル

test/内にそれぞれ、P1 P2 P3 D10 D12 D13 D14 D16 という名称のファイルがある。

c. 入力ファイル作成プログラム

入力ファイルC10 C12 C14 C16を作成するプログラム make_cyclic.cppがtest/内にある。make -C test make_cyclicなどのコマンドによりこのプログラムがコンパイルされる。

以上