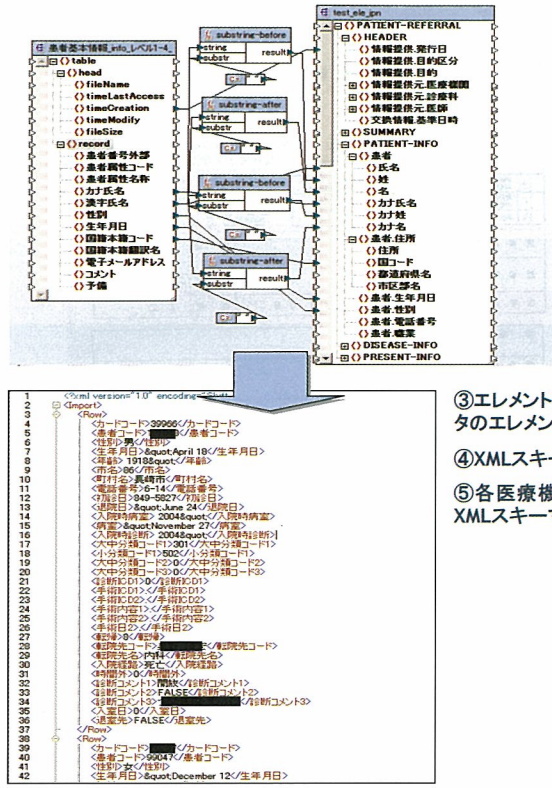
図3-2(2)　各医療機関診療データをXMLスキーマで処理する手順(1／2)



③エレメント変換ソフトウェアで各医療機関医療データのエレメントをJ-MIX選定エレメントにマッピング

④XMLスキーマを作成(原型スタートXMLスキーマ)

⑤各医療機関の医療情報を作成した原型スタートXMLスキーマで変換

図3-3　暗号化プログラムの作成フロー

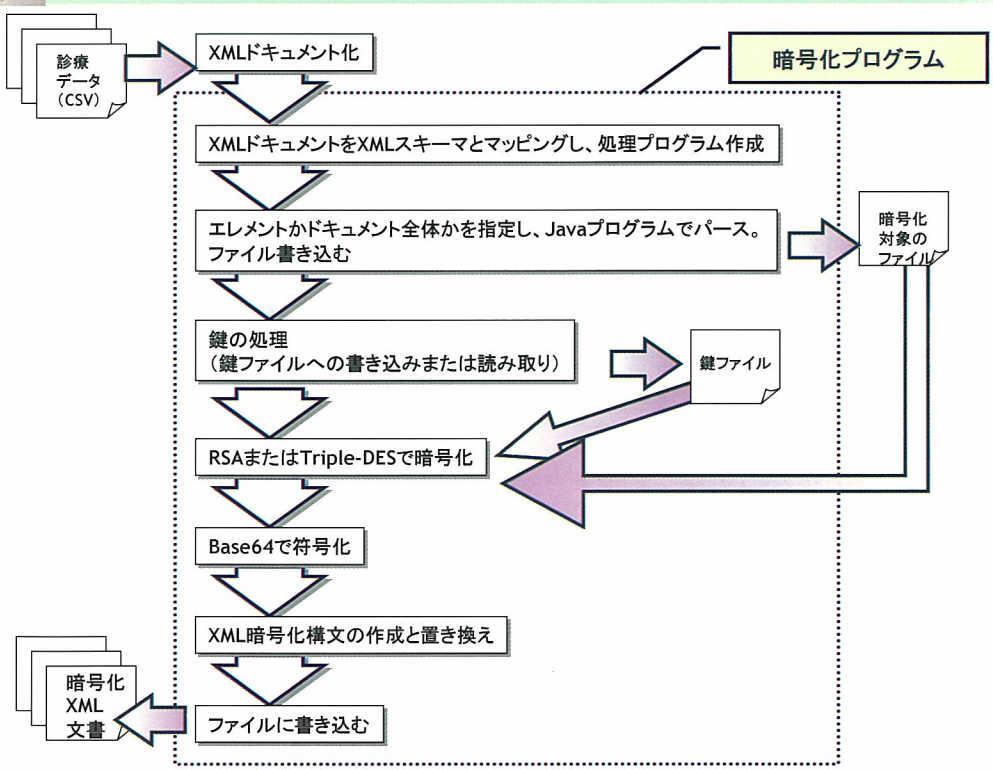


図3-4 診療情報提供書のXMLファイルフォーマット

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!-- 長崎大学医学部・歯学部附属病院　診療情報提供書 XML ドキュメント例(J-MIX参考) 2006/07/26 -->
<PATIENT-REFERRAL>
    <HEADER>
        <情報提供.発効日>
            <年号>平成</年号>
            <年>18</年>
            <月>7</月>
            <日>26</日>
        </情報提供.発効日>
        <情報提供.目的>一般診療依頼</情報提供.目的>
        <情報提供先.医療機関.名称>1 病院</情報提供先.医療機関.名称>
        <情報提供先.医師姓>山田</情報提供先.医師姓>
        <情報提供先.医師名>二郎</情報提供先.医師名>
        <情報提供先.医師カナ姓></情報提供先.医師カナ姓>
        <情報提供先.医師カナ名></情報提供先.医師カナ名>
        <情報提供先.医療機関.郵便番号>852-8501</情報提供先.医療機関.郵便番号>
        <情報提供先.医療機関.住所>長崎県長崎市坂本１丁目７番１号</情報提供先.医療機関.住所>
        <情報提供先.医療機関.名称>長崎大学医学部・歯学部附属病院</情報提供先.医療機関.名称>
        <情報提供先.医療機関.電話番号>095(849)7200</情報提供先.医療機関.電話番号>
        <情報提供先.医療機関.電話番号>095(849)</情報提供先.医療機関.電話番号>
        <情報提供先.医療機関.FAX 番号>095(849)</情報提供先.医療機関.FAX 番号>
        <情報提供先.診療科.名称>第一内科</情報提供先.診療科.名称>
        <情報提供先.医師姓>長崎</情報提供先.医師姓>
        <情報提供先.医師名>太郎</情報提供先.医師名>
        <情報提供先.医師カナ姓></情報提供先.医師カナ姓>
        <情報提供先.医師カナ名></情報提供先.医師カナ名>
    </HEADER>
    <PATIENT-INFO>
        <患者.姓>患者</患者.姓>
        <患者.名>太郎</患者.名>
        <患者.カナ姓>カンジャ</患者.カナ姓>
        <患者.カナ名>タロウ</患者.カナ名>
        <患者.性別>M</患者.性別>
        <患者.郵便番号>000-0000</患者.郵便番号>
        <患者.住所>長崎県長崎市１町</患者.住所>
        <患者.電話番号>000-000-0000</患者.電話番号>
        <患者.生年月日>
            <年号>昭</年号>
            <年>52</年>
            <月>3</月>
            <日>9</日>
        </患者.生年月日>
        <患者.職業>大工</患者.職業>
    </PATIENT-INFO>
    <DISEASE-INFO>
        <現疾患.診断.名称>慢性胃炎</現疾患.診断.名称>
        <既往歴>特記すべきことなし</既往歴>
        <家族歴></家族歴>
        <現病歴>症状経過は別紙参照。</現病歴>
    </DISEASE-INFO>
    <PRESENT-INFO>
        <現検査所見要約.自由記載>2006.7.1 GOT=23 GPT=16 γ-GTP=20</現検査所見要約.自由記載>
        <現投与.薬剤商品名>ロペミンカプセル</現投与.薬剤商品名>
        <現投与.薬剤一般名></現投与.薬剤一般名>
        <現投与.薬剤用量>1mg 2 カプセル</現投与.薬剤用量>
        <現投与.薬剤用量基準時間></現投与.薬剤用量基準時間>
        <現投与.薬剤単位></現投与.薬剤単位>
        <現投与.薬剤用法>1 日 2 回朝夕食後に</現投与.薬剤用法>
        <現投与.薬剤適用備考></現投与.薬剤適用備考>
    </PRESENT-INFO>
    <SUMMARY>
        <診療要約>下記処方は薬空のものです。</診療要約>
    </SUMMARY>
    <備考>特記すべき事はありません。</備考>
</PATIENT-REFERRAL>
```

表3-1 診療情報提供書の項目とJ-Mix、セクションの関係

●診療情報提供書の項目と J-MIX、セクションの関係

診療情報提供書の項目	J-MIX 大分類	セクションのエレメント名
診療情報提供書の発行元、発行先等の機関や医師などの情報	診療情報交換情報	HEADER
患者の氏名、性別、生年月日などの患者基本情報	患者基本情報	PATIENT-INFO
現疾患、病歴、家族歴など	診療要約情報、医学的背景情報	DISEASE-INFO
診療の要約	診療要約情報	SUMMARY
現在の所見や処方	診療要約情報	PRESENT-INFO
検査データ、処方箋、画像データ等 （診療情報提供書に該当項目無し）	指示実施記録情報	PATIENT-DATA
何らかの依頼としての診療情報提供の場合の依頼の詳細 （診療情報提供書に該当項目無し）	指示実施記録情報	ORDER-INFO

図3-5　原型スタートXMLスキーマとXMLﾄﾞｷｭﾒﾝﾄ(診療情報提供書)をﾏｯﾋﾟﾝｸﾞ

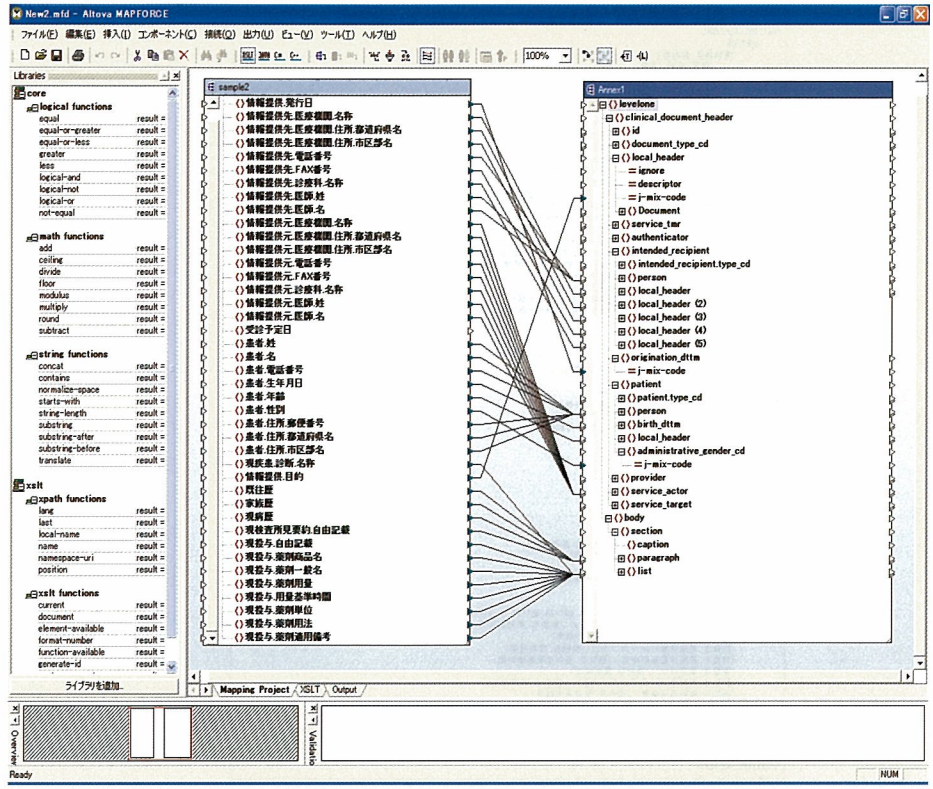


図3-6　J-MIXを利用して作成した原型スタートXMLスキーマ（抜粋）

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!--W3C Schema は xmlspy v2004 rel. 4 U (http://www.xmlspy.com) で生成されています-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="PATIENT-REFERRAL">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="HEADER">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="情報提供.発行日" type="xs:string"/>
                            <xs:element name="情報提供.目的区分" type="xs:string"/>
                            <xs:element name="情報提供.目的" type="xs:string"/>
                            <xs:element name="情報提供元.医療機関">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element ref="名称"/>
                                        <xs:element ref="コード"/>
                                        <xs:element ref="コード体系コード"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="情報提供元.診療科">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element ref="名称"/>
                                        <xs:element ref="コード"/>
                                        <xs:element ref="コード体系コード"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="情報提供元.医師">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element ref="氏名"/>
                                        <xs:element ref="姓"/>
                                        <xs:element ref="名"/>
                                        <xs:element ref="カナ氏名"/>
                                        <xs:element ref="カナ姓"/>
                                        <xs:element ref="カナ名"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="交換情報.基準日時" type="xs:string"/>
                            <xs:element name="提供情報説明" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="SUMMARY">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="入院時説明.検査.計画名" type="xs:string"/>
                            <xs:element name="入院時説明.検査.計画内容" type="xs:string"/>
                            <xs:element name="入院時説明.検査.予定期間.開始日時"
type="xs:string"/>
                            <xs:element name="入院時説明.検査.予定期間.終了日時"
type="xs:string"/>
                            <xs:element name="入院時説明.検査.予定者名" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
```

図3-7 Altova Mapforce出力コードの変更(C:¥Com¥mapforce¥MappingConsole.java)

C:¥Com¥mapforce¥MappingConsole.java 2007/01/17 11:18

```
/**
 * MappingConsole.java
 *
 * This file was generated by MAPFORCE 2004.
 *
 * YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
 * OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.
 *
 * Refer to the MAPFORCE Documentation for further details.
 * http://www.altova.com/mapforce
 */


package com.mapforce;

import enc.DOMCipher;

/**
 * Libraries:
 *   - Annex1 (XML Library)
 *   - sample2 (XML Library)
 */
public class MappingConsole {

  public static void main(String[] args) {
    try {
      System.out.println("Mapping Application");
      TraceTargetConsole ttc = new TraceTargetConsole();

      String strTargetXml = args[0]; // 変換元XMLファイル
      String strTargetSchema = "temp.xml"; // 変換後XMLファイル

      MappingMain1 MappingMain1Object = new MappingMain1();
      MappingMain1Object.registerTraceTarget(ttc);
      MappingMain1Object.run(strTargetXml, strTargetSchema);

      DOMCipher objDomCipher =  new DOMCipher();
      int count;
      int eleCount;
      final int BASE_ARGS_COUNT = 2;
      final int BASE_ITEM_COUNT = 5;

      eleCount = args.length - BASE_ARGS_COUNT;
      count = eleCount * 2;
      count = count + BASE_ITEM_COUNT;

      String[] altem = new String[count];

      altem[0] = "-e";
      altem[1] = "-v";
      altem[2] = "keyinfo1.xml";
      altem[3] = "temp.xml";
      altem[4] = args[1];

      count = 0;

      for (int i=0;i<eleCount;i++) {
          altem[BASE_ITEM_COUNT + count] = "//*[name()='" + args[BASE_ARGS_COUNT + i] + "']";
          altem[BASE_ITEM_COUNT + count + 1] = "template1.xml";
          count = count + 2;
      }

      objDomCipher.StartAngou(altem);

      System.out.println("Output " + args[1]);
      System.out.println();

      System.out.println("Finished");
    } catch (Exception e) {
      System.out.print("ERROR:");
      System.out.println( e.getMessage() );
      e.printStackTrace();
      System.exit(1);
    }
  }
}


class TraceTargetConsole implements com.altova.TraceTarget {
  public void writeTrace(String info) {
    System.out.println(info);
  }
}
```

図3-8 XML Security Suiteのコード変更(C:¥enc¥DOMCipher.java)

C:¥enc¥DOMCipher.java 2007/01/17 11:23

```
/*
 * (C) Copyright IBM Corp. 1999-2001  All rights reserved.
 *
 * US Government Users Restricted Rights Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
 *
 * The program is provided "as is" without any warranty express or
 * implied, including the warranty of non-infringement and the implied
 * warranties of merchantibility and fitness for a particular purpose.
 * IBM will not be liable for any damages suffered by you as a result
 * of using the Program. In no event will IBM be liable for any
 * special, indirect or consequential damages or lost profits even if
 * IBM has been advised of the possibility of their occurrence. IBM
 * will not be liable for any third party claims against you.
 */

package enc;

public class DOMCipher {
    //
    // Methods
    //

  private String mOutputFile = new String();


// args[0] = 暗号化:e、復号化:d
// args[1] = プロバイダ
// args[2] = 鍵情報ファイル
// args[3] = 暗号化&復号化対象xmlファイル
// args[4] = 変換後のXMLドキュメント
// args[5] = 暗号化エレメント
// args[6] = テンプレートxmlファイル
//                .
//                .
//    public static void main(String[] args) {
  public void StartAngou(String[] args) {

    if (args.length < 1) {
        printUsage();
        System.exit(1);
    }
```

図3-9　XML変換プログラムへの暗号化処理の組み込み

Step1　「Altova Mapforce」（Altova社）にてXMLﾄﾞｷｭﾒﾝﾄとXMLｽｷｰﾏをﾏｯﾋﾟﾝｸﾞ後、変換ﾌﾟﾛｸﾞﾗﾑを生成（Java言語）

Step2　「IBM XML Security Suite」の暗号化ﾌﾟﾛｸﾞﾗﾑを「Step1」で生成した変換ﾌﾟﾛｸﾞﾗﾑへ組み込む。（注）暗号化ﾌﾟﾛｸﾞﾗﾑを一部カスタマイズ

Step3　Step2で作成したﾌﾟﾛｸﾞﾗﾑ（MappingConsole）をｺﾏﾝﾄﾞﾌﾟﾛﾝﾌﾟﾄより実行

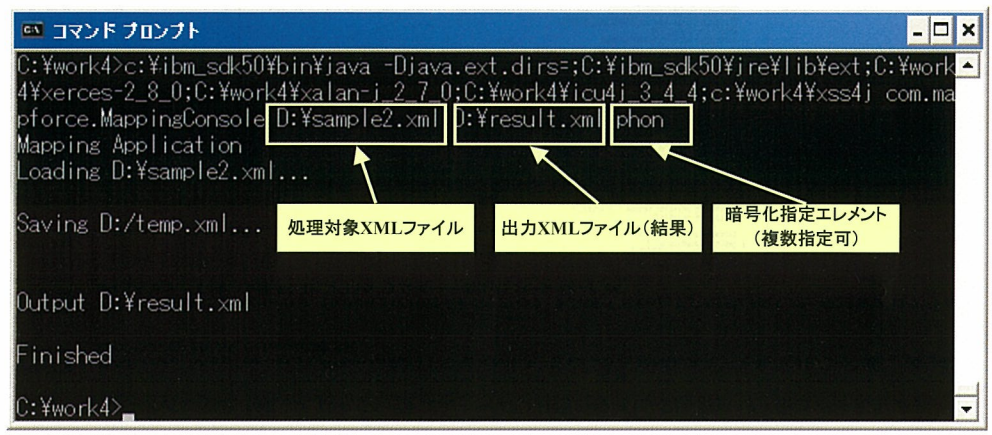


図3-10　暗号化XMLプログラムによる実行結果

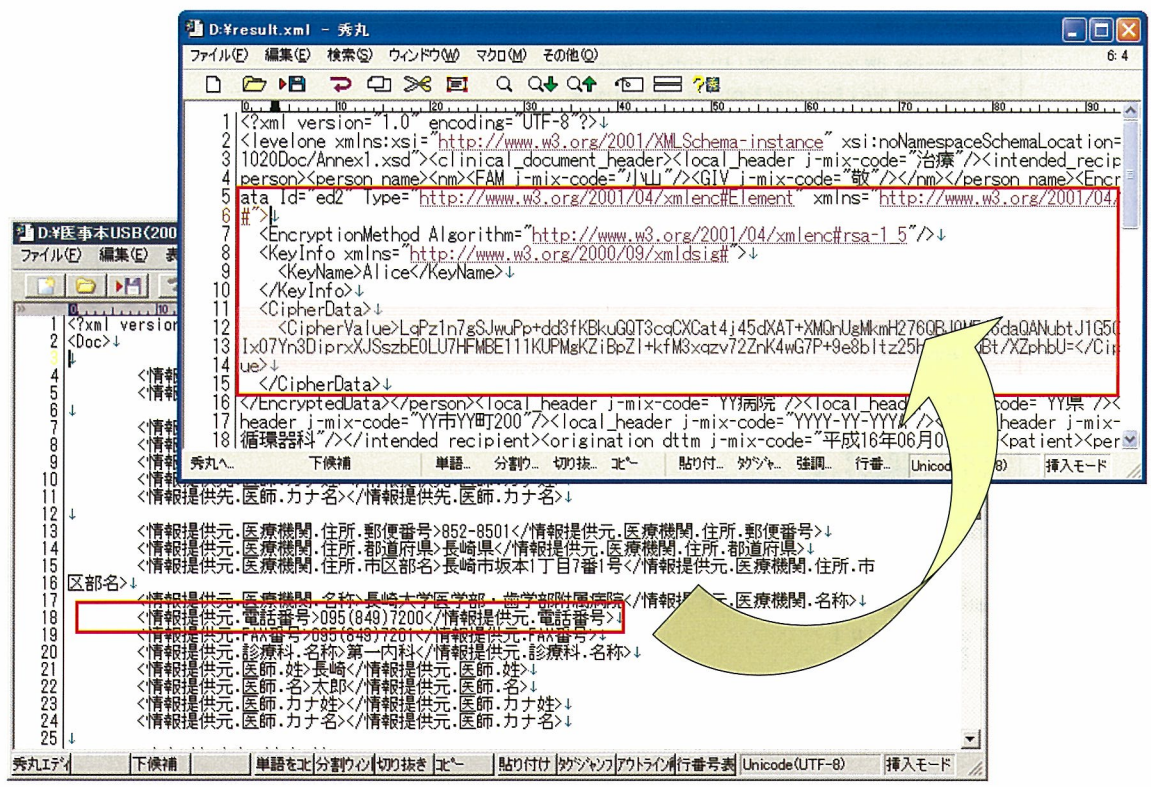


表3-2　エレメント暗号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

表　エレメント暗号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

●RSA 512bit

エレメント数		1	2	3	4	5	全て(40)
処理時間(msec)		2600	2800	2800	2800	2900	3700
CPU負荷(%)	実施前	0	0	0	0	0	0
	実施後	78	81	75	84	79	85
メモリ使用状況(MB)	実施前	521	521	521	521	521	521
	実施後	538	542	544	544	544	544

●RSA 2048bit

エレメント数		1	2	3	4	5	全て(40)
処理時間(msec)		3100	3100	3100	3200	3200	3900
CPU負荷(%)	実施前	0	0	0	0	0	0
	実施後	78	79	79	79	79	85
メモリ使用状況(MB)	実施前	521	521	521	521	521	521
	実施後	542	543	543	544	544	545

表3-3　エレメント暗号化と処理時間、CPU負荷、メモリ使用状況の関係（Triple-DES）

表　エレメント暗号化と処理時間、CPU負荷、メモリ使用状況の関係（Triple-DES）

●TripleDES 112bit

エレメント数		1	2	3	4	5	全て(40)
処理時間(msec)		3200	3400	3600	3600	3700	5400
CPU負荷(%)	実施前	0	0	0	0	0	0
	実施後	78	78	78	83	83	97
メモリ使用状況(MB)	実施前	442	442	442	442	442	442
	実施後	465	465	467	467	467	467

●TripleDES 168bit

エレメント数		1	2	3	4	5	全て(40)
処理時間(msec)		3400	3400	3500	3600	3600	6000
CPU負荷(%)	実施前	0	0	0	0	0	0
	実施後	74	77	83	83	83	98
メモリ使用状況(MB)	実施前	442	442	442	442	442	442
	実施後	465	465	465	465	465	465

図3-11　暗号化エレメント数と処理時間の関係

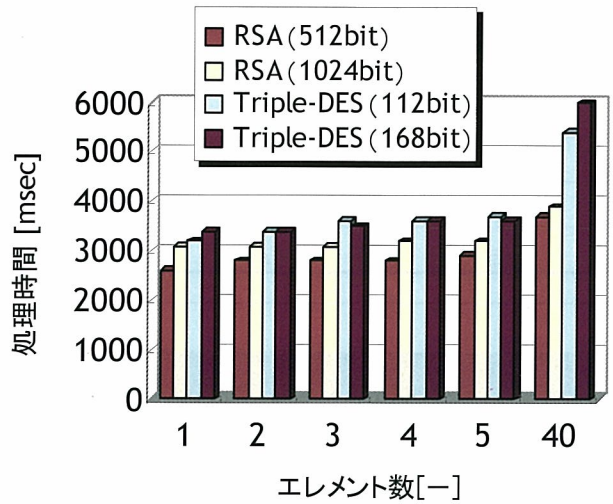


図　暗号化エレメント数と処理時間の関係

表3-4　診療情報提供書1枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係(RSA512bit)

表　診療情報提供書1枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係(RSA)

●RSA 512bit -暗号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		3600	3900	4300	4700	5600	5800	6900	7500	7600	8700	9100	10200	13600
CPU負荷(%)	最大	85	96	87	92	98	98	98	99	100	100	100	100	100
メモリ(MB)	増加量	23	23	24	26	29	32	33	24	21	31	30	27	37

●RSA 512bit -復号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		3500	3900	4100	4800	5100	5200	5200	5500	5700	6900	7100	7600	8700
CPU負荷(%)	最大	84	97	97	89	100	99	94	93	90	94	99	90	98
メモリ(MB)	増加量	21	23	25	24	28	33	27	27	26	28	29	30	36

表3-5　診療情報提供書2枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA512bit）

表　診療情報提供書2枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

●RSA 512bit -暗号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		6400	7800	8500	9700	10900	12400	13300	14300	15200	16000	16100	19300	25800
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	45	46	48	50	50	50	53	53	55	55	54	58	66

●RSA 512bit -復号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		6500	7200	7400	9300	9800	10200	10300	11000	11200	12000	12600	13500	15900
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	44	52	50	60	53	56	55	56	57	57	59	63	68

表3-6　診療情報提供書3枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA512bit）

表　診療情報提供書3枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

●RSA 512bit -暗号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		8700	12000	12100	14200	17000	18200	18900	20900	23500	24000	26600	31600	37600
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	70	73	74	74	75	78	76	75	80	77	78	91	125

●RSA 512bit -復号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		9000	11200	12100	13000	13200	14200	15300	15900	17100	17400	18000	19300	23500
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	69	67	75	74	66	84	84	82	83	81	90	98	103

表3-7　診療情報提供書1枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA2048bit）

表　診療情報提供書1枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

●RSA 2048bit -暗号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		3700	4200	5200	5700	6000	6500	7500	7800	8900	9700	9800	11000	14400
CPU負荷(%)	最大	91	91	98	100	99	100	99	100	100	100	99	100	100
メモリ(MB)	増加量	22	25	24	26	26	25	24	24	26	27	28	31	35

●RSA 2048bit -復号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		3800	5900	7800	9000	10800	12400	14000	15700	18200	19900	21800	25300	33100
CPU負荷(%)	最大	99	90	98	95	98	99	98	98	97	98	96	98	98
メモリ(MB)	増加量	22	24	26	25	25	27	27	28	28	31	31	31	30

表3-8　診療情報提供書2枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA2048bit）

表　診療情報提供書2枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

●RSA 2048bit -暗号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		6200	7600	9000	9900	11900	12800	13800	15500	16000	16800	19000	20500	28600
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	38	48	47	48	50	54	49	54	54	53	54	59	87

●RSA 2048bit -復号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		7000	10300	13600	16600	20200	23300	25600	28700	31700	33900	36400	42400	54100
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	45	52	55	58	55	56	57	54	52	59	57	60	62

表3-9　診療情報提供書3枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA2048bit）

表　診療情報提供書3枚でのエレメント暗号化・復号化と処理時間、CPU負荷、メモリ使用状況の関係（RSA）

●RSA 2048bit -暗号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		8800	11200	13300	14100	17500	19200	19700	21600	24900	25300	27200	31200	39700
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	60	55	57	57	78	64	74	75	82	78	84	80	90

●RSA 2048bit -復号化-

エレメント数		1	40	80	120	160	200	240	280	320	360	400	480	640
処理時間(msec)		10300	15800	20300	24400	30000	34800	38500	42700	46600	50600	53900	64600	81300
CPU負荷(%)	最大	100	100	100	100	100	100	100	100	100	100	100	100	100
メモリ(MB)	増加量	68	76	77	81	83	85	83	85	90	90	94	96	102

図3-12　暗号化エレメント数と処理時間の関係(RSA512bit)

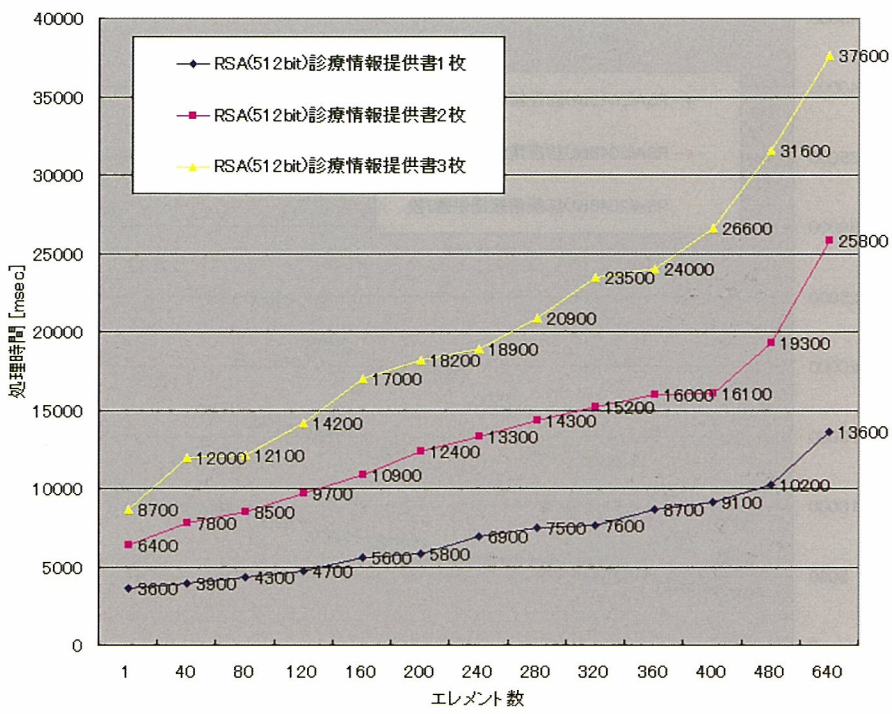


図　暗号化エレメント数と処理時間の関係（RSA512bit）

図3-13　復号化エレメント数と処理時間の関係(RSA512bit)

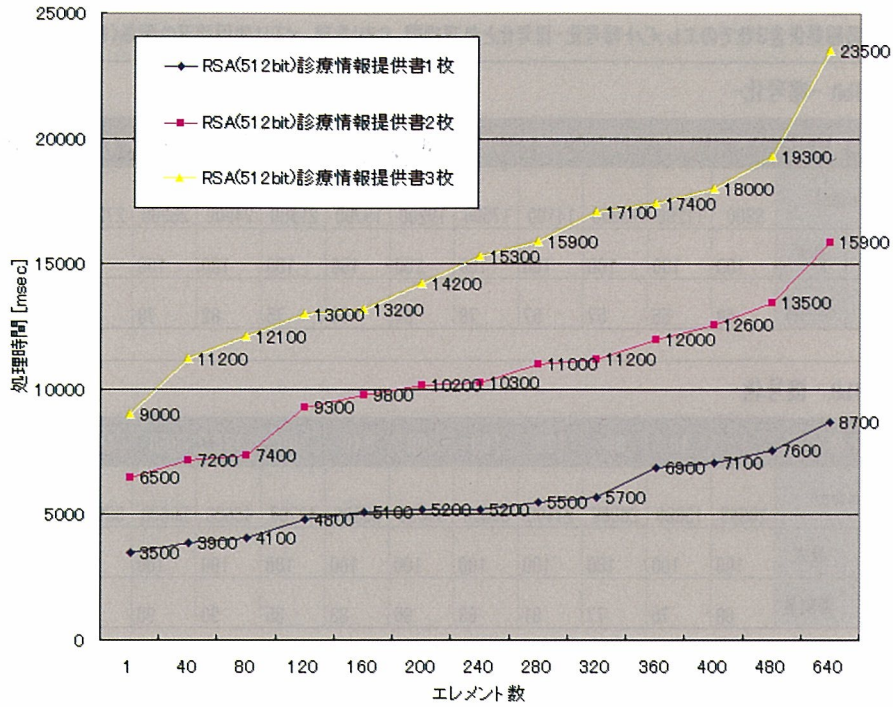


図　復号化エレメント数と処理時間の関係（RSA512bit）

図3-14　暗号化エレメント数と処理時間の関係(RSA2048bit)

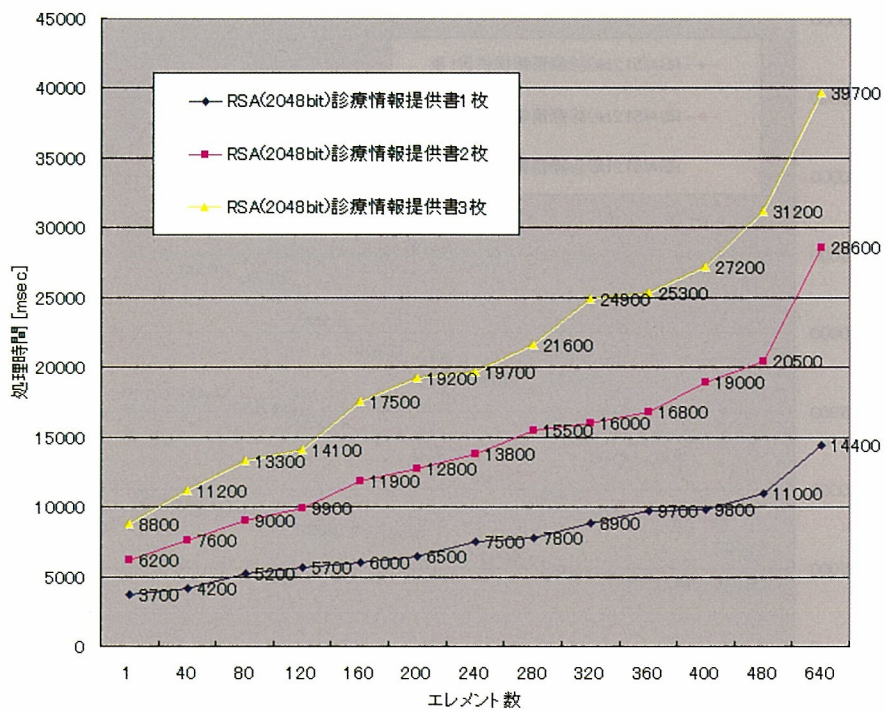


図　暗号化エレメント数と処理時間の関係（RSA2048bit）

図3-15　復号化エレメント数と処理時間の関係(RSA2048bit)

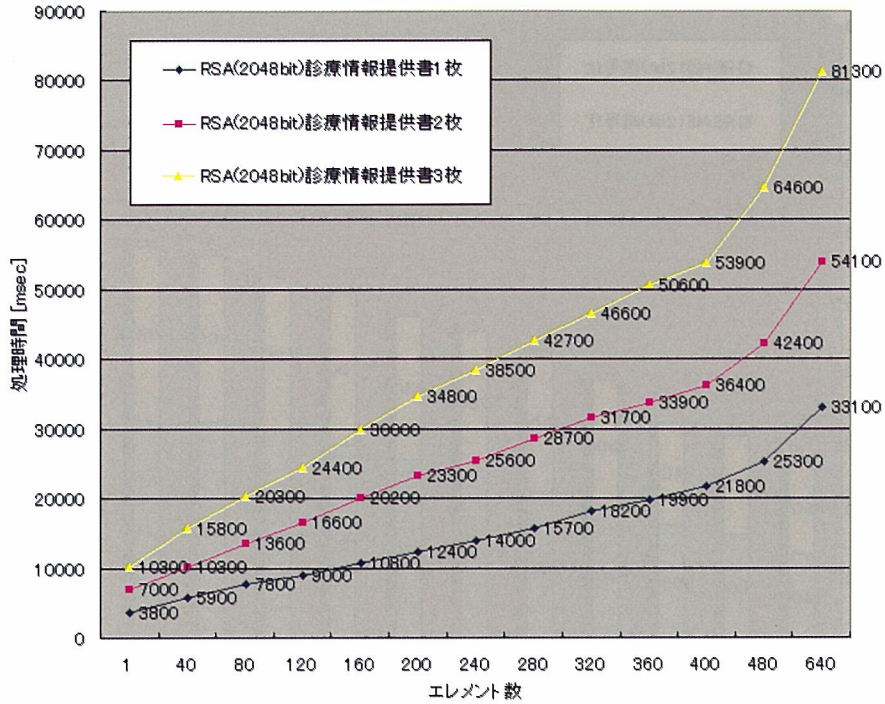


図　復号化エレメント数と処理時間の関係（RSA2048bit）

図3-16　診療情報提供書1枚での暗号化・復号化エレメント数と処理時間の関係(RSA512bit)

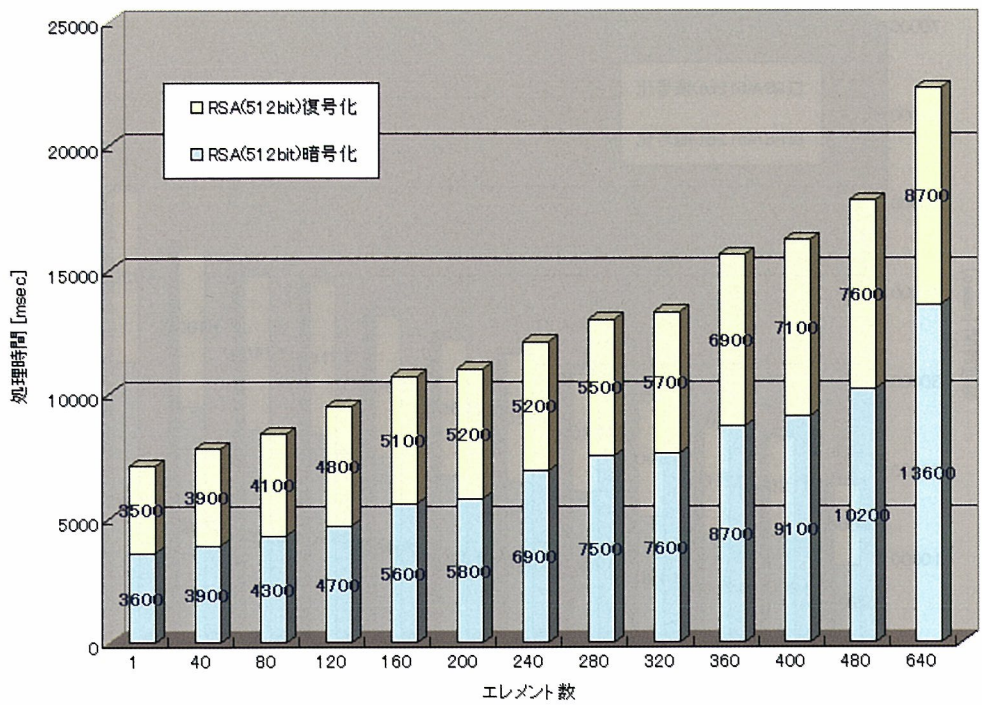


図　診療情報提供書1枚での暗号化・復号化エレメント数と処理時間の関係（RSA512bit）

図3-17　診療情報提供書2枚での暗号化・復号化エレメント数と処理時間の関係(RSA512bit)

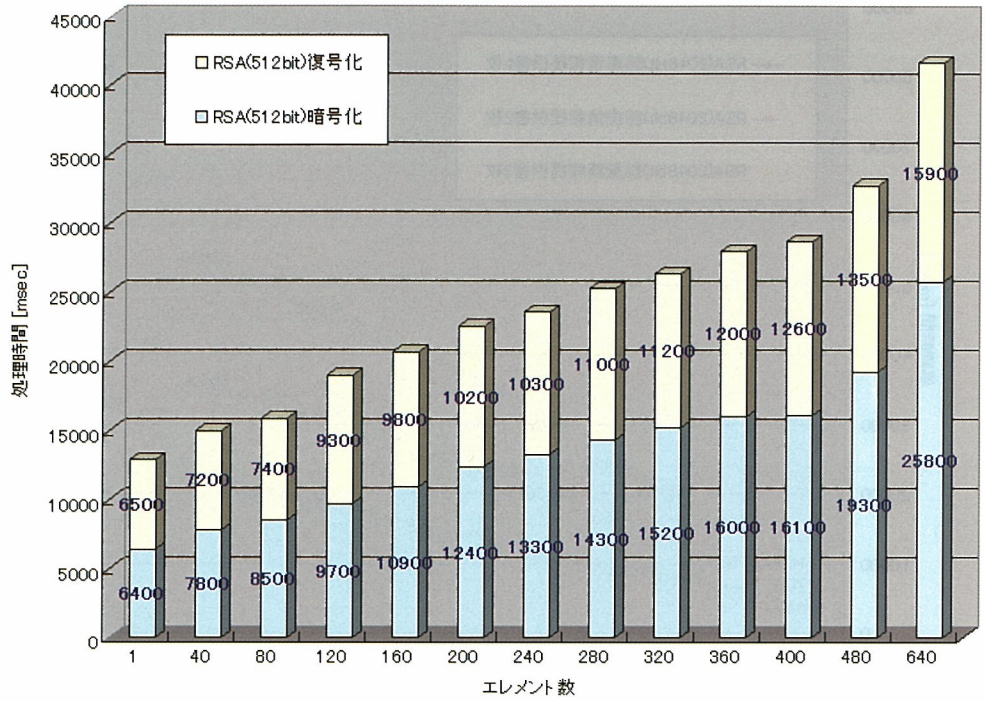


図　診療情報提供書2枚での暗号化・復号化エレメント数と処理時間の関係（RSA512bit）

図3-18　診療情報提供書3枚での暗号化・復号化エレメント数と処理時間の関係(RSA512bit)

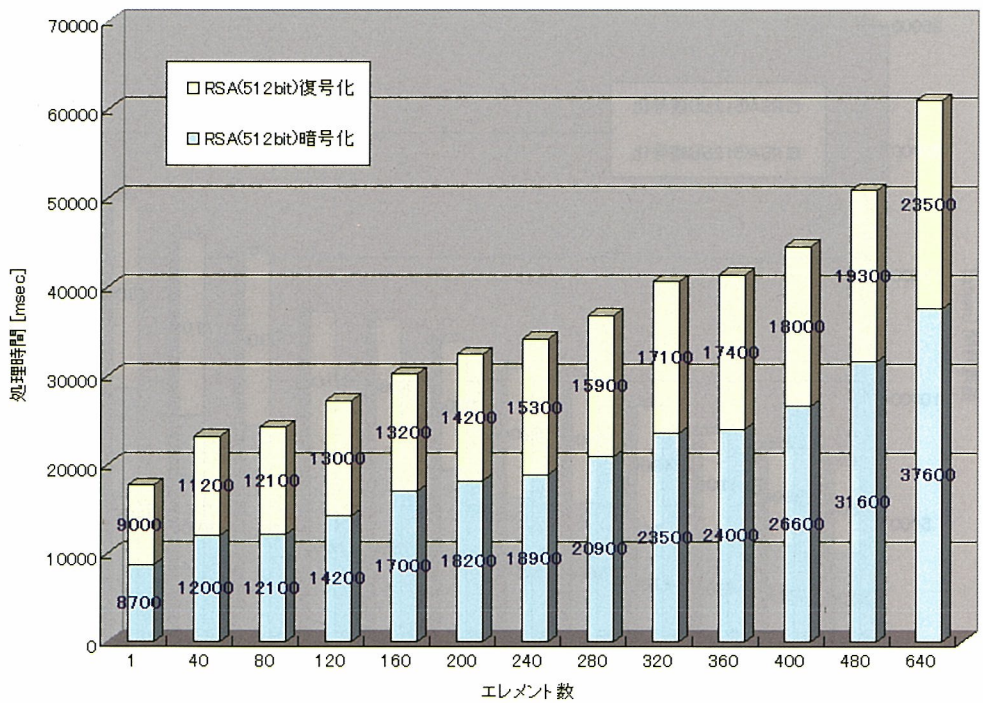


図　診療情報提供書3枚での暗号化・復号化エレメント数と処理時間の関係（RSA512bit）

図3-19　診療情報提供書1枚での暗号化・復号化エレメント数と処理時間の関係(RSA2048bit)

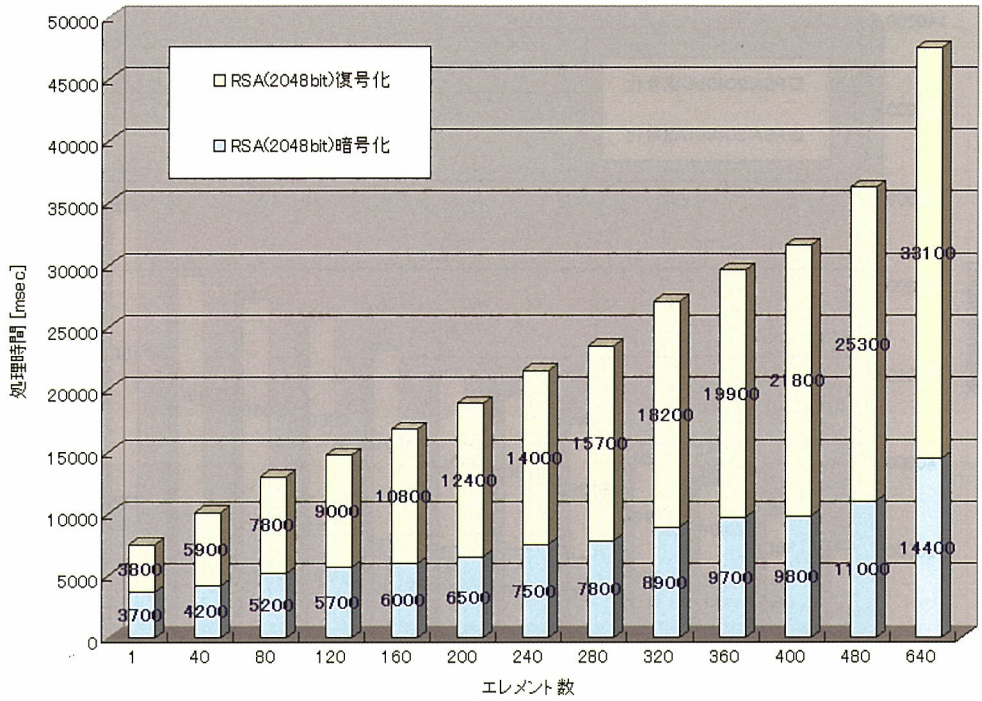


図　診療情報提供書1枚での暗号化・復号化エレメント数と処理時間の関係（RSA2048bit）

図3-20　診療情報提供書2枚での暗号化・復号化エレメント数と処理時間の関係(RSA2048bit)

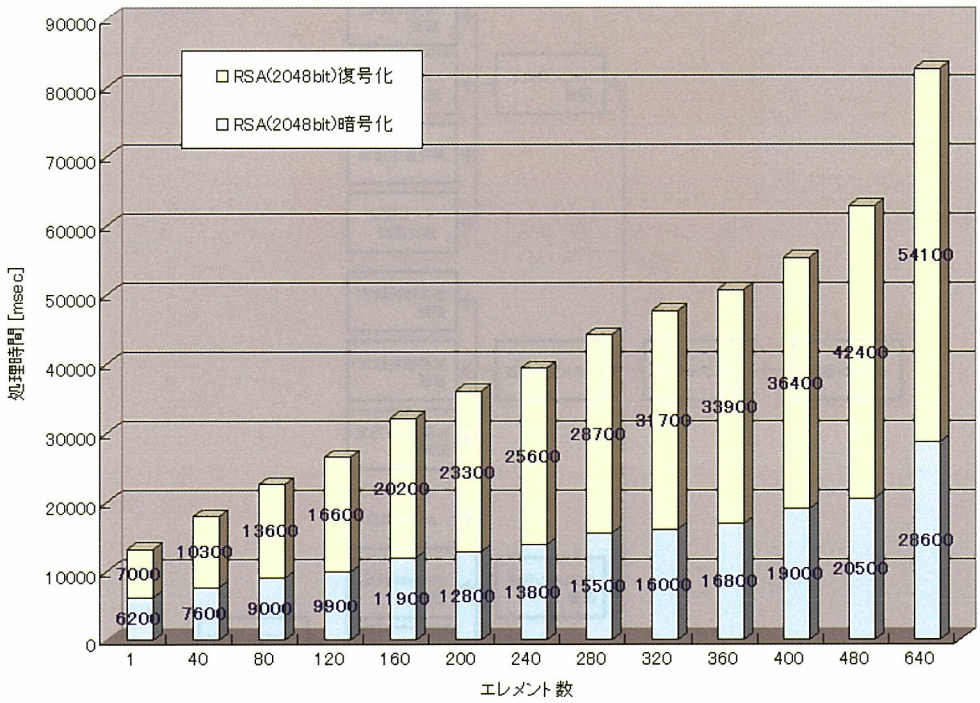


図　診療情報提供書2枚での暗号化・復号化エレメント数と処理時間の関係（RSA2048bit）

図3-21　診療情報提供書3枚での暗号化・復号化エレメント数と処理時間の関係(RSA2048bit)

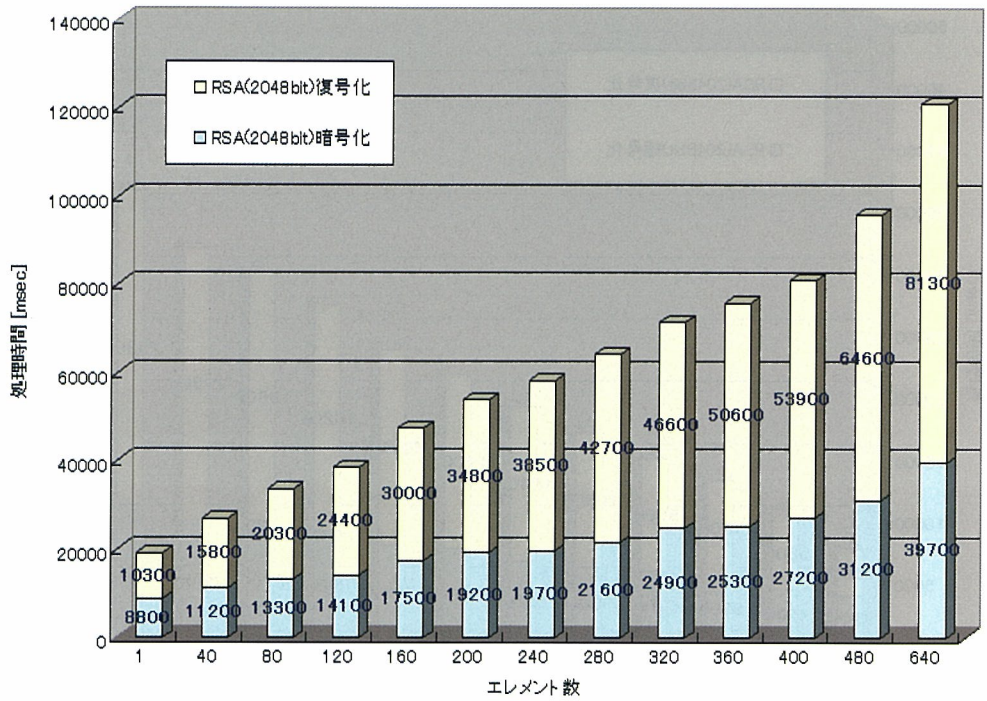


図　診療情報提供書3枚での暗号化・復号化エレメント数と処理時間の関係（RSA2048bit）

図3-22　医療機関文書暗号化システムの画面遷移

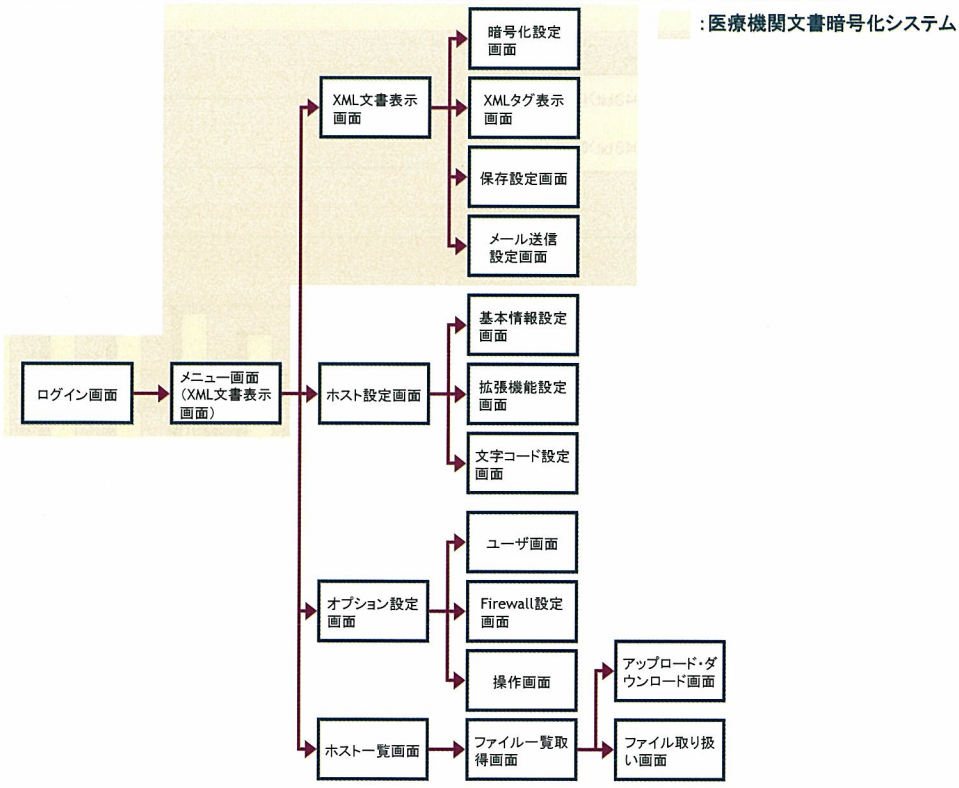


図3-23　医療機関文書暗号化システム／ログイン

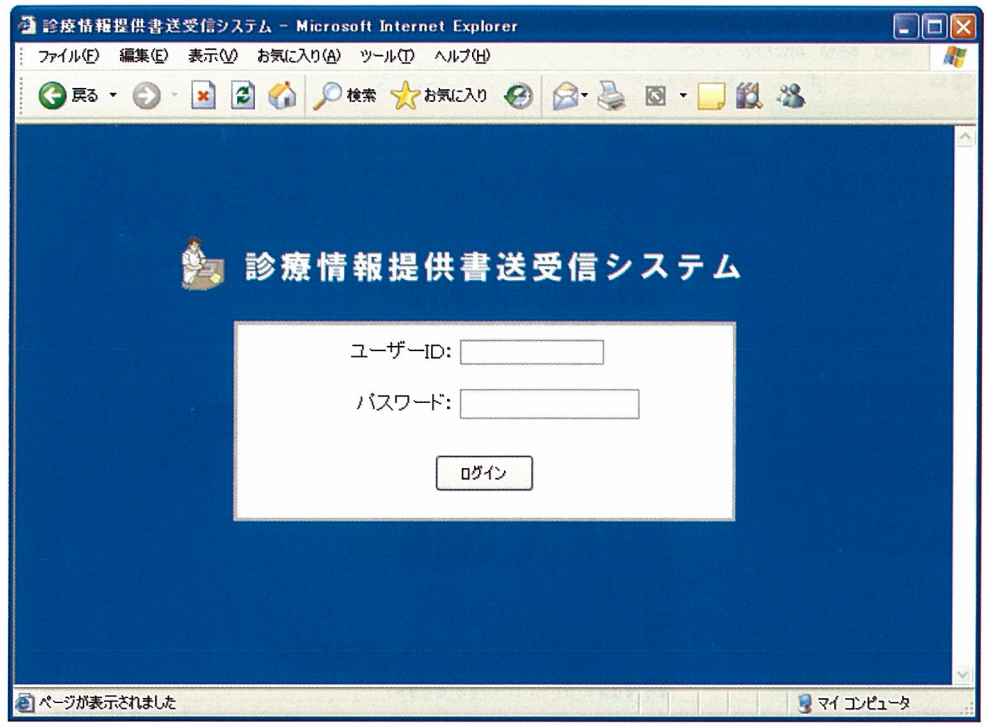


図3-24　医療機関文書暗号化システム／XML文書表示

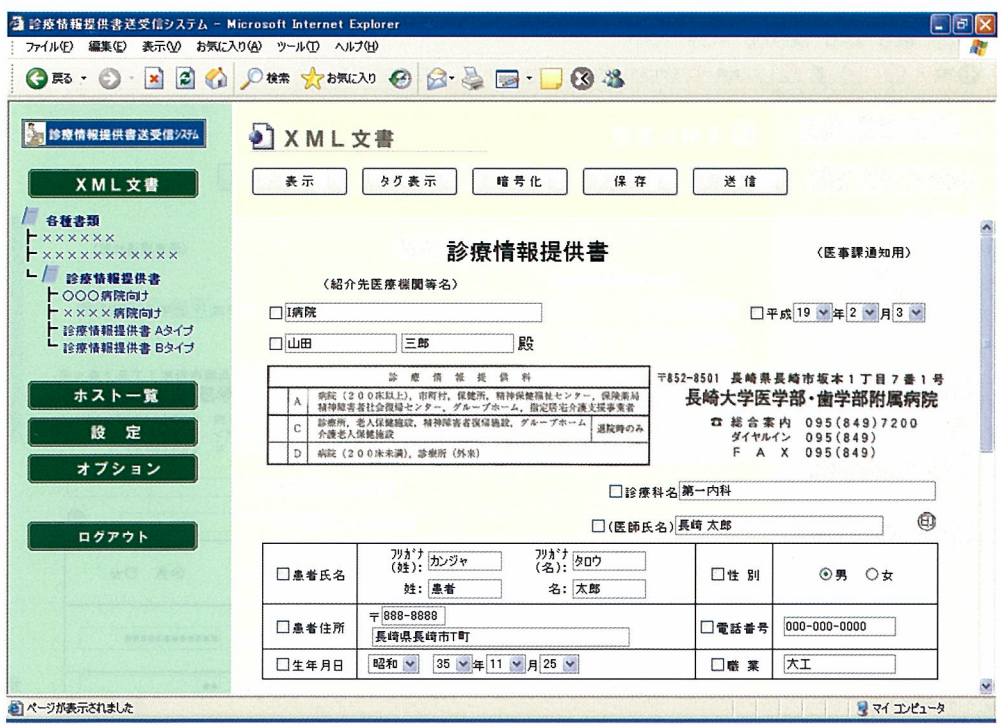


図3-25　医療機関文書暗号化システム／XMLタグ表示

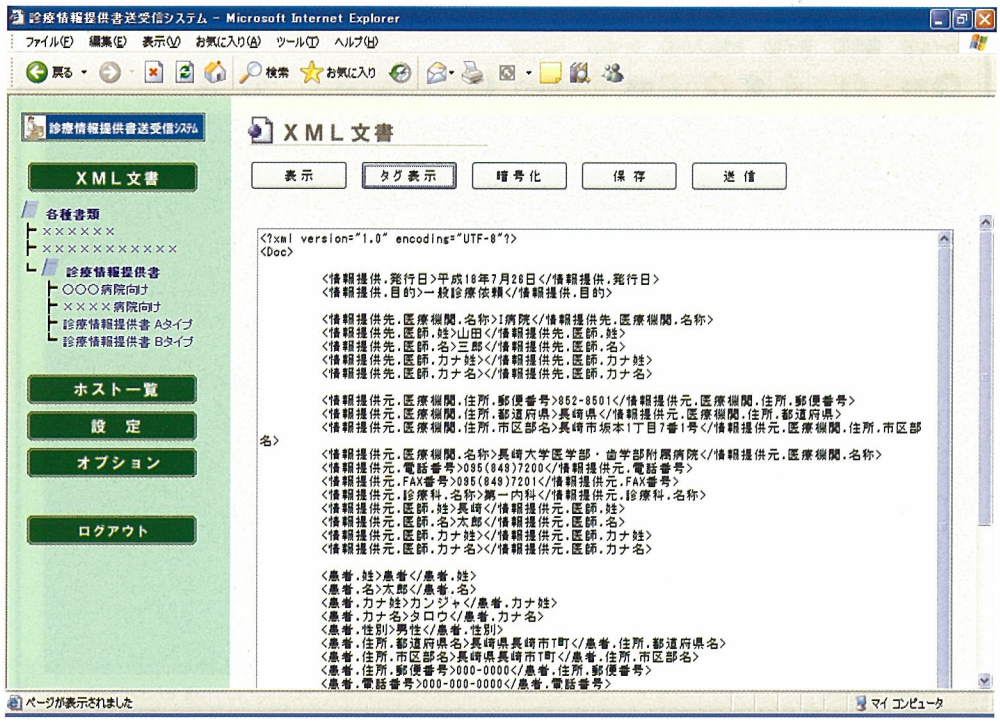


図3-26　医療機関文書暗号化システム／XMLエレメント暗号化

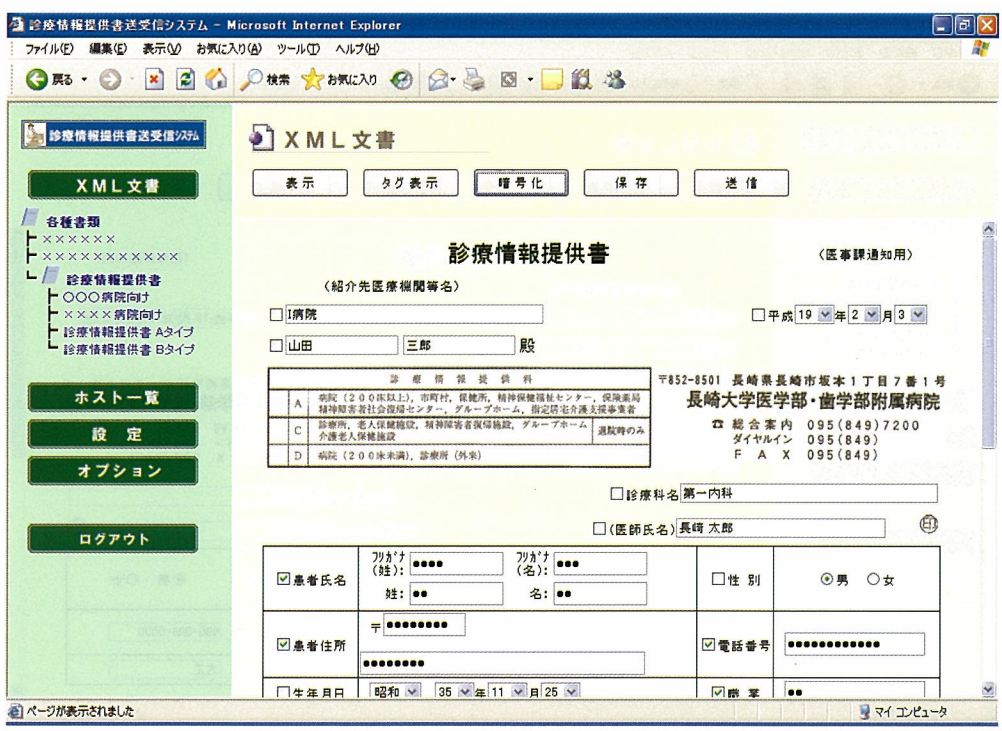
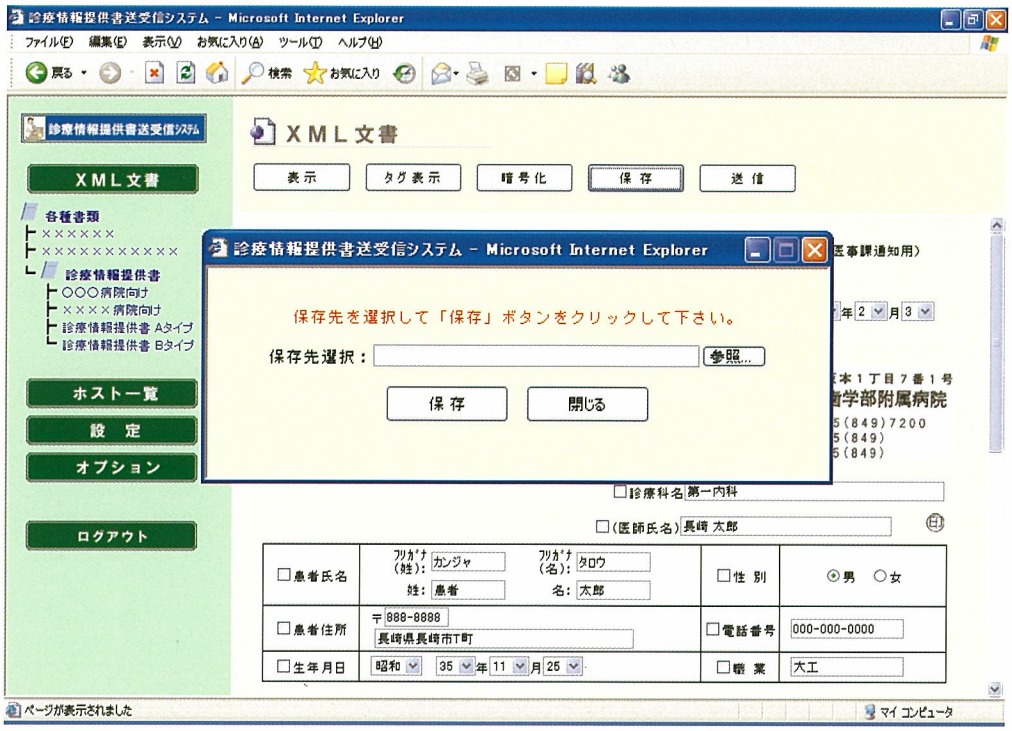


図3-27 医療機関文書暗号化システム／XML文書保存



参考文献

1. 横森 哲也. XML文書暗号化規格の実装実験. http://www.nol.info.kanagawa-u.ac.jp/research/2004/yokomori.pdf

第４章　医療機関間暗号化XML文書情報連携システムの研究開発

本多 正幸・中山 良幸・梁瀬 和夫

第４章　医療機関間暗号化 XML 文書情報連携システムの研究開発（担当：日立）

研究要旨

様々な情報システムで XML の導入が本格化してきており、XML データベースの利用も増加する傾向にある。また文書管理フォーマットとしての XML の役割も大きくなってきており、XML や XML データベースに対するセキュリティ技術にも関心が高まっている。

XML データベースと PKI を組み合わせることにより、新たなセキュリティ機能を XML データベースに付与できると考えられるが、その具体的な方法や有効性についての検討はまだ十分ではない。そこで本研究では、PKI を XML データベースに適用しセキュアな暗号化 XML スキーマを利用した医療機関間暗号化 XML 文書情報連携システムのプロトタイプの設計・作成を実施し、その有効性の評価を試みた。

A. 研究目的

コンピュータネットワークの分野では、SSL/TLS、VPN、S/MIME など多くのプロトコルやデバイスで、PKI の技術が広く用いられているのに対して、データベースへの応用例は多くない。しかしながら医療機関間の診療データの連携には、XML データベースへの PKI 技術を始めとしたセキュリティ技術の導入は、必要不可欠であると考えられる。

XML セキュリティは、XML 文書に対し署名や暗号化といった機能を提供する技術である。XML に関する技術の多くは、World Wide Web Consortium（以下 W3C）や OASIS などの団体から仕様が公開されており、この XML セキュリティも W3C にて策定されている。本研究では、XML セキュリティの 2 大要素である XML 署名と XML 暗号化の医療期間間医療情報連携システムの適用について検討した。

XML 署名は、XML 文書に対し本人証明・完全性・否認防止といった機能を提供する技術である。W3C からは” XML-Signature Syntax and Processing” と” Canonical XML” の二つを用いることが勧告されている。また XML 署名には、その形式により Detached 署名（署名対象となる XML 文書とが別に署名要素を構築）、Enveloped 署名（署名対象となる XML 文書内に署名要素を含む）Enveloping 署名（署名要素内部に署名対象となる XML 文書を埋め込む）があり、アプリケーションの機能や用途により、それぞれ使い分けが行われている。

一方 XML 暗号化は、XML 文書に対して秘匿性の機能を提供する技術である。暗号化方式には秘密鍵暗号方式や公開鍵暗号方式が使用可能である。XML 暗号化に関しても、W3C から” XML Encryption Syntax and Processing” の仕様が公開されている。XML 暗号化には、XML 文書中の指定した要素以下を暗号化するエレメント暗号と、指定した要素のコンテンツ以下を暗号化するコンテンツ暗号の 2 つのタイプが依存する。その他、XML 文書