2006－CSEC－35 (5)
2006／12／8

携帯電話における RSA ソフトウェア実装の高速化

駒場素生 宇田隆哉
東京工科大学コンピュータサイエンス学部

本研究は，Java で作成された i アプリを使用可能な携帯電話端末において，ソフトウェアを用いて RSA 暗号の計算を実現することを目的としている．最新の携帯電話端末には PKI に基づく認証機能がハードウェア実装されているが，この認証は通信キャリアが提供している基盤に制約されている．そこで，本研究では bouncy castle が提供している Java の暗号ライブラリを DoJa の仕様に合わせて改変することで，携帯電話で動作可能となるように実装を行い，携帯電話上のソフトウェアで公開鍵暗号演算が行えるようにした．さらに，暗号の処理過程を解析し，ボトルネック部分に改良を加えることで演算の高速化を実現した．

A Method of Accelerating Operations of RSA within Software on a Cellular Phone

Motoki Komaba, Ryuya Uda
Tokyo University of Technology School of Computer Science

The aim of the study is realizing calculation of RSA cryptosystem within software on a cellular phone using i-appli which is created with Java program. Authentication mechanism which is based on PKI is implemented within hardware in a newest cellular phone device. However, the authentication is limited to the infrastructure of the communication carrier. Therefore, in this study, we realize a calculation of public-key cryptosystem within software on a cellular phone, applying Java cryptosystem library provided by bouncy castle to DoJa specification. Moreover, analysis of calculation process of the cryptosystem and improvement of bottleneck lead to the realization of the accelerating operations.

1. はじめに

1.1 背景

近年，携帯電話はほぼ一人一台の時代であり，携帯電話が財布や切符の代わりやロッカーの鍵になったり，携帯電話でネットショッピングが行えたりするなど，携帯型端末を用いて商品購入や行政手続きを安全に行えるサービスが提供され，生活に密着したアイテムとなっている．最近では携帯電話自体の機能や演算能力も向上し，携帯電話上で動作するブラウザやアプリなどと連携して，様々なサービスも展開されている．

その流れを受け，携帯電話を用いてセキュアな認証を切望する声もある．ネイティブの機能としては SSL が存在するが，SSL は通信の暗号化とサイトの認証を行うものの，ユーザ認証を含めた通信データの認証は別であるため，現状では多くの場合 ID&パスワードを使用した認証方法が採用されている．

しかし，ID&パスワード方式による認証は盗聴が容易である上に，パスワード忘却時の対処，ID を持つデバイスの不正複製，パスワード入力の利便性の悪さなど，問題が多いため最善の認証方式と呼ぶにはほど遠い．

そこで，現在，NTT DoCoMo が FirstPass[1]という電子認証サービスを提供している．これは，NTTDoCoMo が発行するユーザ証明書を使用したサービスである．また，KDDI からは Security Pass[2]と呼ばれる証明書を発行するサービスが提供されており，au 証明書と呼ばれる電子証明書を KDDI が発行する電子認証サービスである．

しかし，これらはいずれも携帯電話会社に依存しており，サービスを展開する側はキャリア毎に携帯電話会社との契約が必要になり，暗号化方式も携帯電話会社に提供されているものしか使用出来ないと言った欠点がある．

このような PKI 基盤を携帯電話会社に束縛されずに独自に実現するには，携帯電話上で動作するアプリを作成する方法がある．しかし，現状では認証などを行うための暗号ライ

ブラリを携帯電話で動かそうとしても，NTT DoCoMo の i モード端末仕様の Java である DoJa では基盤となる Java アーキテクチャが縮小されているので単純な移植は難しく，非常に手間がかかる．

そこで本研究では，既存の携帯電話を用いて，ソフトウェアによる公開鍵暗号の実装を行うことで，特別なハードウェアを加えることなく，端末及びサービス提供者を相互に認証可能な仕組みを提案する．これにより，アプリケーションごとに認証における独立性を保てるため，特定の認証機関やハードウェアに依存することなくサービスを提供できる．

2. 関連研究

本章では，携帯電話端末に公開鍵暗号をソフトウェア実装した研究について解説を行う．既存の研究としては，公開鍵暗号による携帯電話を用いた相互認証システム[3]がある．この研究では，RSA-PSS と ECDSA を携帯電話上で動作するようにソフトウェア実装させた研究である．

しかし，この研究では，Bouncy Castle[4]を単に NTT DoCoMo の i モード端末 DoJa に移植したのに過ぎないので，処理速度に時間が掛かる．

そこで，本研究では，Bouncy Castle の移植時にボトルネックとなる箇所を調査・分析し，改変を行うことで演算速度の向上を図る．

3. 提案方式

現在，携帯電話は電話機自体の機能や演算能力が向上してきているため，携帯電話上で動作するブラウザやアプリが多数開発され，様々なサービスとして展開されている．それらを利用するに当たり，ユーザはセキュアな認証を行いたい，あるいは必要であると思う機会も増えてきていると考えられる．そこで，本研究では携帯電話上でセキュアな認証を行うための，公開鍵ペアや署名の生成を可能とするために，暗号ライブラリの携帯電話への移植を行う．

しかし，ここで問題となってくるのは携帯電話の演算能力である．現在の携帯電話の演算能力は向上してきてはいるものの，認証に必要な鍵ペアや署名の生成を容易に行えるほどの演算能力を，まだ持ち合わせてはいない．携帯電話に対する暗号ライブラリの移植については，過去すでに行われた例[3]があるが，それは単にライブラリの移植を行っただけであるため，処理にやや時間がかかってしまうという難点がある．そこで，携帯電話上での認証のユーザビリティを向上させるために，処理を高速化し，かつ現在の携帯電話での実用に耐えうる暗号化ライブラリを作成する．

本研究では，まず，Java 言語で提供されているオープンソースの暗号ライブラリである，Bouncy Castle Cryptography Library[4]を基盤にし，NTT DoCoMo の定める i アプリ用のプロファイル DoJa で使用できるように移植し，i アプリ上で動作可能にした．さらに，ボトルネックを分析し，Bouncy Castle に改変を加えることで，高速な演算処理を実現した．

前述のとおり，この暗号ライブラリの携帯電話への移植はすでに行われているが，機種によっては処理に時間がかかり，またライブラリで i アプリの容量を多く占有してしまうと，肝心の本体のアプリにコードを裂けなくなるという問題点も出てくる．そこで，不要な部分を削り，ライブラリの軽量化も目指した．

具体的な高速化の案として，まずライブラリ内のボトルネックであると考えられる java.math.BigInteger クラスの改良を考えた．これは，任意精度の整数を扱い，通常の整数型では表現できない巨大な数字での演算を可能としているクラスである．このクラスは，巨大な整数での演算が多く必要な暗号ライブラリ内で何度も使用されているため，この部分の高速化に成功すれば，結果的に暗号ライブラリの処理にかかる処理の時間も短くなると考えられる．Bouncy Castle の J2ME 向けのリリースには BigInteger クラスが含まれており，前述の既存研究ではこちらを利用しているが，今回はこれを J2SE 内に含まれる同クラスと入れ替えて速度差を測定する．また，暗号化処理を行ううえで重要な SecureRandom クラスというものがある．このクラスはその名のとおり安全な乱数，つまりなるべく偏りのない乱数を生成するためのものであるが，このク

ラスもBouncy Castle内のものが使用されているため，入れ替えを試みる．

4. 実装

本節では，実装について述べる．実装は，既存のライブラリの移植，計測用アプリの作成，ボトルネック解析，クラスの入れ替えという順序で行った．以下に，各項目について述べていく．

4.1 既存ライブラリの移植

携帯電話上で動作する言語として，Java を使用する．既に関連研究[3]で取られている手法を基に，Java 言語で提供されているフリーの暗号化ライブラリ Bouncy Castle[4]を携帯電話端末上に移植する．携帯電話アプリは基盤となるアーキテクチャが縮小されているので，不要な部分を削除し，必要な部分のみ実装を行う．本項目では，移植にのみこだわり最終的に組み込み機器向けの Java 言語仕様である J2ME の一部として定義されている想定実行環境の CLDC 上で動作するものとする．これは携帯電話やPDAなどの小型端末を対象としたものである．i アプリはこの CLDC の上に独自の i アプリ API が提供された環境で動作している．実装内容は，暗号化ライブラリの移植と，ECDSA 及び RSA-PSS での任意条件でのデジタル署名（鍵生成，署名生成，署名検証）を行えるアプリケーションである．

4.2 計測用アプリの作成

4.1 節で作成したアプリケーションに対して，ECDSA と RSA-PSS での任意条件でのデジタル署名の各種項目の計測を取るための i アプリを作成する．この作成された i アプリを基にボトルネックの解析を行い高速化する．評価は，単に移植した段階での計測結果とボトルネックの解析を行い，改良を加えたものを比較している．

4.3 ボトルネックの解析

既存研究である[3]で，既に暗号化処理の際にボトルネックと考えられる箇所として，BigInteger クラスというものが挙げられている．今回は，BigInteger クラスを中心に解析を行った．BigInteger クラスは，ネイティブ型では扱えない巨大な整数を扱う為のクラスである．暗号化処理では巨大な整数を扱う必要がある為，ライブラリ内ではこのクラスを多用する．その為，このクラスの高速化が図られればライブラリ全体の速度も向上すると考えられる．Bouncy Castle[4]の J2ME 向けのリリースには BigInteger クラスが含まれており，前述の既存研究[3]ではこちらを使用しているが，これを J2SE 内に含まれる同クラスと入れ替えを試みる対象とした．

また，暗号化処理を行う上で重要な SecureRandom クラスというものがある．その名の通り安全な乱数，つまりなるべく偏りのない乱数を生成するためのもので，このクラスも Bouncy Castle[4]内のものが使用されている為，入れ替えを試みる対象とした．

4.4 クラスの入れ替え

前節で述べたように，BigInteger クラスと SecureRandom クラスを既存研究[3]では，Bouncy Castle[4]のクラスが使われていたが，J2ME から内から BigInteger クラスと SecureRandom クラスの入れ替えを行った．クラス入れ替え後，比較用に計測アプリを作成した．

5. 評価

前節までの提案方式を実装することにより，どの程度の高速化が見込めるかを評価した．計測には，本計測用に作製した計測用アプリを用いている．実験協力者の協力を仰ぎ，現在までに下記に述べる機種を用いて実際に計測を行っている．機種毎に移植前・移植後において計測を行い，その平均値を割り出した値として，移植後を表 1，移植前を表 2 に示す．本実験で計測に使用した機種は 2005 年冬季に NTT DoCoMo より発売された 902i シリーズ及び，902iS のうち 3 機種である．携帯機端末としては比較的新しい部類に入る 902i シリーズは，今後様々なシーンで携帯電話が活用されることを想定しても，充分な処理能力を持っており，本実験においても支障なく実験を行うことができている．計測した評価対象を以下に挙げる．本研究で利用した暗号化方式は，

ECDSA 楕円曲線暗号と，RSA 暗号である．RSA においては，通常利用するにあたって充分な安全性を持つ鍵長 1024bit の署名に加え，さらに安全性に特化した鍵長 1536bit の 2 種類を検証した．ECDSA においては，さらに 2 種類の規格[5]に分けて検証を行った．ランダム曲線の他に Koblitz 曲線を推奨した SEC1（表では k1 と記）と，サンプルとしてランダム曲線と Weil 法によって生成された曲線を利用した ANSI（表では r1 と記）を利用している．それぞれを RSA1024bit とほぼ同等の安全性を有していると評価される，鍵長 160bit, 192bit, 256bit の計 6 種類を検証した．機種メーカーによって若干の誤差があるものの，鍵生成・署名生成・署名検証のすべてを 6 種類において，移植前と比べ数秒～数十秒の単位での速度の向上が観測された．一部機種では速度の減少が現れた部分もあるが，これに関しては今後検証回数をさらに増やし，その他機種を利用しての検証も行った上で，原因を突き止めていく．ECDSA に関しては，後に新たな曲線を推奨した SEC1 の方式が若干ではあるが速度が速く，移植後もその差はほぼ同等であることが分かる．規格現段階では，各計測結果でかなりのバラつきがあるため，平均値を利用しているが，このバラつきの原因に関しても今後検討していく予定である．

表 1　計測結果 (平均) － 移植前

機種	secp160k1			secp160r1			secp192r1			secp192k1		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	5690	4557	9404	4829	4844	10054	7355	7411	15409	7203	7268	14960
D902i	7225	7005	14031	6941	7246	14616	10692	10548	21312	10868	10461	22210
F902iS	7520	6839	14229	6885	7140	14189	10406	10480	21646	10397	10629	21455

機種	secp256r1			secp256k1			rsa1024			rsa1536		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	13972	13934	28980	14290	14136	29384	86657	974	57	491693	3203	106
D902i	19776	20179	40615	19811	20307	40984	82043	1163	73	146808	3885	134
F902iS	19675	19756	40421	19650	19964	40835	112105	1212	73	457173	3873	133

単位（ms）

表 2　計測結果 (平均) － 移植後

機種	secp160k1			secp160r1			secp192r1			secp192k1		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	832	713	1399	707	723	1567	1380	1451	2583	1290	1295	2662
D902i	992	985	2004	1285	1223	2486	1703	1822	3666	1832	1688	3502
F902iS	1017	1039	2016	1267	1244	2450	1725	1786	3677	1782	1685	3472

機種	secp256r1			secp256k1			rsa1024			rsa1536		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	2385	2416	4892	2218	2308	4722	25911	532	21	83401	1602	37
D902i	3055	3190	6174	3029	3179	6143	31412	704	31	216935	2075	48
F902iS	3129	3288	6420	3121	3225	6213	36302	696	30	111127	2120	47

単位（ms）

表３　計測結果 (分散)　－　移植前

機種	secp160k1			secp160r1			secp192r1			secp192k1		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	110327	17008	86052	20650	16458	116456	42893	48815	106457	132904	10690	63757
D902i	10712	1560	2304	62750	900	75350	141000	462	147072	73170	216225	905352
F902iS	89415	46561	196732	133762	99715	339660	132735	130154	661536	154855	138849	966814

機種	secp256r1			secp256k1			rsa1024			rsa1536		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	53882	110780	234188	136150	119108	301760	452793976	414	2	15226797134	3351	3
D902i	6	93330	71022	287832	563250	1630729	1690566572	100	1	21316	4160	2
F902iS	500949	634153	1546725	538285	255992	2605273	3460605853	1469	13	71396137173	3489	24

単位（ms）

表４　計測結果 (分散)　－　移植後

機種	secp160k1			secp160r1			secp192r1			secp192k1		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	501	80	2606	131	40	1060	2411	934	6171	1338	1561	11260
D902i	2070	100	9	30	182	4900	30	1225	64	961	7744	16900
F902iS	913	3841	4985	2131	2090	6901	1788	2742	8941	3266	3830	8915

機種	secp256r1			secp256k1			rsa1024			rsa1536		
	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証	鍵生成	署名生成	署名検証
P902i	1387	610	4214	3508	7926	15308	248823698	160	35	92356256	460	0
D902i	1980	4225	30	6241	7225	38612	138615302	36	0	7956729200	552	0
F902iS	5310	8153	46571	13691	12322	23864	271526282	262	2	4776120160	2135	1

単位（ms）

6.　考察

現在までの評価として，実装した 8 種類の規格全てにおいて数秒～数十秒単位での速度の向上が観測された．この結果から，今後，携帯電話を用いたスピード認証が必要とされるシーンの中で必ず本研究の有用性が見出されると予想できる．絶対的な数字の変化は確かに小さいが，携帯電話を利用した個人認証の場である，ATM の個人認証や，定期・切符代わりに利用することなどを想定すると，数十秒の高速化というものは非常に大きく，常に利便性を求める日本のニーズに答えるものになりうると考えられる．現在は一部の携帯電話利用システムにおいて，スピード・効率を求めるがために，本人以外が利用できてしまうなどのセキュリティ的な脆弱性が存在する．本研究では，このような問題点を改善するためのアプリケーションの土台になるものとして広く利用されることを視野に入れ，さらなる高速化とベンダーや機種に依存しない汎用性の高い実装を目指す．

7.　おわりに

近年，携帯電話はほぼ一人一台の時代であり，生活に密着したアイテムとなっている．また，携帯電話自体の機能や演算能力も向上し，携帯電話上で動作するブラウザやアプリなどと連携して，様々なサービスが展開されている．

携帯電話で電子認証サービスを行う機構を携帯電話会社にとらわれることなく独自に実現するために，携帯電話上で動作するアプリを作成することは非常に重要な技術の一つとなることが予想される．

本システムでは，既存の携帯電話を用いて，ソフトウェアによる公開鍵暗号の実装を行うことで，特別なハードウェアを加えることなく，端末及びサービス提供者を高速に相互認証可能な仕組みを提供することに成功した．これにより，アプリケーションごとに認証基盤の独立性を保てるため，特定の認証機関やハードウェアに依存することなく高速な認証サービスを提供できるようになると考えられる．

参考文献

1 ） NTT DoCoMo FirstPass
http://www.nttdocomo.co.jp/service/other/firstpass/index.html
2 ） KDDI au SecurityPass
http://www.au.kddi.com/notice/securitypass/index.html
3 ） 尾崎啓，宇田隆哉，棟上昭男：公開鍵暗号による携帯電話を用いた相互認証システム，情報処理学会コンピュータセキュリティシンポジュウム 2005，pp.535-540 (2005)
4 ） Bouncy Castle
http://www.bouncycastle.org/
5 ） 新保淳,"ECDSA評価の現状報告（詳細評価 ）",
http://www.ipa.go.jp/security/enc/CRYPTREC/fy13/doc/18_shinbo.pdf,（2006-11-9 参照）