

匿名性を保証するリング署名を可能にする XML 署名ツール“RIN”の開発

上山 真梨 菊池 浩明

東海大学電子情報学部情報メディア学科
259-1292 平塚市北金目 1117

ueyama@cs.dm.u-tokai.ac.jp, kikn@tokai.ac.jp

あらまし 匿名性を保証するリング署名を Java を用いて実装した. 本ツールでは, RSA と Schnorr を混在したリング署名を用いている. 開発したツールはリング署名を様々なオブジェクトに対して適用するため, XML 署名形式を用いている. 本稿では本ツールの性能を示し, リング署名の実用性を検証する.

RIN-XML Signature Tool which Enables Ring Signature to Guarantee Anonymity

Mari Ueyama Hiroaki Kikuchi

Dept. of Info. Media Technology, School of Info. Technology and Electronics, Tokai University
1117 Kitakaname, Hiratsuka, Kanagawa 259-1292

Abstract We implemented a ring signature tool on Java that guarantees anonymity. This tool combines RSA with Schnorr signature to construct a ring signature. We adopt an XML signature format to sign on various objects. In this paper, we show the performance of the tool which proves the practicality of a ring signature.

1 はじめに

近年, 組織内の不正を内部告発することの重要性が認識されてきている. 虚偽の告発文書を防止するにはデジタル署名が有効だが, 告発者の身元が露顕してしまう. そこで, 本研究では, 告発者の匿名性を保証する汎用の匿名署名ツールの開発を目的とする.

匿名性を保証する電子署名には, Rivest らによって提案されたリング署名 [5] や, Chaum らによって提案されたグループ署名 [6] がある. 表 1 にグループ署名とリング署名の比較を示す. グループ署名は, 管理者が存在し, メンバの所属を管理している. グループへの匿名所属証明書

を用いることで, 署名長がグループメンバー数 n に依存しない, 効率的な署名を実現している. 一方, リング署名は管理者を必要としないため, 任意の署名者が, 独自にグループをアドホックに作って署名を行う. 署名長が n に比例する反面, リングを明示的に構成するのでユーザの失効も可能である. 特殊な署名アルゴリズムを仮定することなく, 既存の PKI の任意のアルゴリズムを組み合わせ, 登録不要で署名に利用できる. この性質をアドホック性, または, setup free と呼ぶ.

本研究では, 汎用の匿名署名ツールの開発を目的とするため, 鍵の選択に高い自由度をもつリング署名を用いる. 本ツールは広く用いられ

ている離散対数問題と素因数問題に基づくアルゴリズムを代表して、RSA と Schnorr を混在したリング署名を使用する。

本ツールは、文書、画像、音声などの様々なオブジェクトに対して署名が可能である XML 署名形式 [3] を採用する。しかしながら、[3] では1つの文書に対して2人以上の署名者があるデジタル署名は許されていない。そこで、本論文では、リング署名を表現できるように [3] を拡張した新しいフォーマットを提案する。

高い自由度とアドホック性を持つリング署名ではあるが、署名長が $O(n)$ になる課題があり、そのリングの大きさには制約がある。そこで、本研究では、Java による実装を行い、実測値に基づいて評価を与える。

表 1: グループ署名とリング署名の比較

	グループ	リング
管理者	あり	なし
失効	追加機能	明示的
署名長 (n)	$O(1)$	$O(n)$
アドホック性	×	○

2 従来研究, 関連研究

2.1 グループ署名

グループ署名は、グループのメンバだけが匿名で署名をすることが可能な署名である。不正が発覚したときには、追跡機関により署名者を特定するなど、様々な拡張が行われている [7]。

2.2 リング署名

リング署名は、グループのメンバなら誰でもグループを代表して署名が可能で、検証者に対して匿名性を保証できる署名方式である。検証者を指定する試み [8]、複数のリングが同一の署名者によるものかどうかを調べる試み [9]、非署名者が否認を行う試み [10] など多くの提案が行われている。

2.3 XML 署名

XML 署名は、W3C[3] で定められ、任意のデジタルデータのデジタル署名を XML 形式で記述する技術である。XML 署名には、次の3つの署名形式がある。

- Detached Signature
署名文書と署名対象が別文書になる。署名対象は任意のファイルで、URI で指定する。
- Enveloping Signature
署名対象文書を署名文書の中に埋め込む。
- Enveloped Signature
署名文書が署名対象文書に含まれる。

署名形式としてみたときの XML 署名の大きな特徴は、テキストによる符号化である。S/MIME [11] などのバイナリ符号化に対し、署名データがテキストで符号化されることである。そして XML 署名は、Detached Signature 形式を用いることにより、様々なオブジェクトに対して署名が可能である。加えて、署名対象が XML の場合、文書全体だけでなく、文書の一部に署名を施す部分署名や、複数人の署名をつける多重署名も対応している。以上の S/MIME との比較を表 2 に整理する。

3 “RIN” の設計

3.1 概要

“RIN” は、リング署名を用いることにより、高い匿名性を有する署名ツールである。本ツールは、開発言語に Java を用いた。“RIN” は、生成と検証の2つのツールからなる。署名生成ツールは、グループメンバの公開鍵と署名者の秘密鍵を指定してリング署名を行い、XML 署名を生成する。一方、署名検証ツールは、XML 署名の中に記述されている、メンバの公開鍵を用いて検証を行う。

XML 署名の Detached Signature 形式を用いているため、様々なオブジェクトに対して署名が可能である。

表 2: S/MIME と XML 署名の比較

	S/MIME	XML 署名
符号化	バイナリ	テキスト
構造化	ASN.1, PKCS-7	XML
対象	メール (MIME)	任意のオブジェクト (XML)

3.2 リング署名アルゴリズム

大久保らによるリング署名のアルゴリズム [1] を基にして, RSA と Schnorr の両方を使えるプロトコルを示す.

3.2.1 署名生成, 署名検証

$\mathcal{G}$ をグループメンバの集合 $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$ とする. ここで, 署名者を $\mathcal{U}_i$ で表す.

次の署名生成, 検証において, f, h, k は 3.2.2 で定義する公開鍵アルゴリズム, H は一方向性セキュアハッシュ関数とする.

[署名生成]

文書 m に対する署名 $\sigma[m]$ は, 以下の手順で生成する.

Step 1 i について, 乱数 α を選び

$$\begin{aligned} T_i &= f_i(\alpha), \\ c_{i+1} &= H(m \parallel T_i) \end{aligned}$$

を求める.

Step 2 $j = i + 1, \dots, n, 1, \dots, i - 1$ について, 乱数 s_j を選び

$$\begin{aligned} T_j &= h_j(s_j, c_j), \\ c_{j+1} &= H(m \parallel T_j) \end{aligned}$$

を順次計算する. (j が n を超えたとき 1 に戻る.)

Step 3 (秘密鍵を知っている) i について,

$$s_i = k_i(c_i)$$

を計算する. m に対する署名 $\sigma[m]$ は $\sigma[m] = (c_1, s_1, s_2, \dots, s_n)$ である.

[署名検証]

$j = 1, \dots, n$ まで以下を繰り返す.

$$\begin{aligned} T_j &= h_j(s_j, c_j), \\ c_{j+1} &= H(m \parallel T_j). \end{aligned}$$

$c_1 = c_{n+1}$ ならば受理し, そうでなければ棄却する.

3.2.2 署名アルゴリズム

[RSA]

メンバ $\mathcal{U}_j$ は大きな素数 p_j, q_j を生成し, $n_j = p_j q_j, \lambda(n_j) = \text{LCM}(p_j - 1, q_j - 1)$ を求める. e を決めて公開鍵とし $d_j = e^{-1} \bmod \lambda(n_j)$ を秘密鍵とする.

α, s_j は Z_{n_i} からランダムに選ぶ. f, h, k を次に定義する.

$$\begin{aligned} f_i^R(\alpha) &= \alpha^{e_i} \bmod n_i, \\ h_j^R(s, c) &= s^{e_j} + c \bmod n_j, \\ k_i^R(c) &= (T_i - c)^{d_i} \bmod n_i. \end{aligned}$$

[Schnorr]

メンバ $\mathcal{U}_j$ は, $q_j \mid p_{j-1}$ を満たす大きな素数 $p_j, q_j, Z_{p_j}^*$ の位数 q_j の部分群の生成元となる g_j を生成し, p_j, q_j, g_j を公開する. $x_j \in Z_{q_j}$ を秘密鍵, $y_j = g_j^{x_j} \bmod p_j$ を公開鍵とする.

α, s_j はランダムに Z_{q_j} 選ぶ. f, h, k を次に定義する.

$$\begin{aligned} f_i^S(\alpha) &= g_i^\alpha \bmod p_i, \\ h_j^S(s, c) &= g_j^s y_j^c \bmod p_j, \\ k_i^S(c) &= \alpha - x_i c \bmod q_i. \end{aligned}$$

3.3 署名者の指定

作成した鍵ファイルの種類を表 3 に示す. 拡張子が鍵の種類を表している. 拡張子 rpub, rsec,

spub, ssec はそれぞれ、RSA 公開鍵、RSA 秘密鍵、Schnorr 公開鍵、Schnorr 秘密鍵を表している。鍵のフォーマットは、10 進数の文字列 (ASCII) で、1 行ごとに値が入っている。

ツールでは、これらの拡張子で識別される複数の鍵ファイルを与えることで、暗黙的に署名者を指定する。

表 3: 鍵ファイルの種類

	鍵の種類	拡張子	フォーマット
RSA	公開鍵	rpub	n, e
	秘密鍵	rsec	n, e, d, p, q
Schnorr	公開鍵	spub	p, q, g, y
	秘密鍵	ssec	p, q, g, y, x

3.4 リング署名のための拡張 XML 署名フォーマット

リング署名では、署名が複数のデータで構成され、その順番が重要である。XML 署名では単一の署名者しか想定されていないので、次のように拡張する。

リング署名の署名値 $c, s_1, \dots, s_n$ を入れるため、図 1、図 2 の DTD で定義される RingSignatureValue という要素を追加した。s 要素の順番が署名者の順序になる。[3] では、0 または 1 個と定められていた “KeyInfo?” 要素を複数個指定できる “KeyInfo*” に変更し、リング用の RingSignatureValue (図 2) を追加する。

3.2.1 で示した $\sigma[m] = (c_1, s_1, s_2, \dots, s_n)$ に対応する要素が、図 2 の 1 個の c と複数個の s である。s₁, s₂ を区別するために一意な (ID) 属性 Id を追加して。

また Schnorr の鍵を扱えるようにするため、SchnorrKeyValue という要素を追加した。図 3 に DTD を示す。RSA と Schnorr の鍵の区別は Id に記述された鍵の拡張子によって行われる。

リング XML 署名のサンプルを図 4 に示す。この例では、RSA を用いた署名者候補 kikuchi と Schnorr の鍵をもつ ueyama の 2 人から成るリング署名を表している。署名対象は Word の文書 message.doc である。

```
<!ELEMENT Signature
(SignedInfo,(SignatureValue|RingSignatureValue),
KeyInfo*,Object*)>
```

図 1: 拡張署名者要素 (Signature)

```
<!ELEMENT RingSignatureValue (c,s+)>
<!ELEMENT c (#PCDATA)>
<!ELEMENT s (#PCDATA)>
<!-- ATTLIST s Id ID #REQUIRED -->
```

図 2: 署名値のための拡張要素 (RingSignatureValue)

3.5 平文 (メッセージ)

“RIN” は XML 署名の Detached Signature 形式を用いているため、テキスト、ワード文書など様々なオブジェクトに対して署名が可能である。

4 開発ツールと評価

4.1 仕様

実装したツールの仕様を表 4 に示す。

4.2 インタフェース

署名者やリングを構成するメンバの指定は、ツールの key フォルダに格納される鍵ファイルの拡張子によって行われる。リングへの構成順序は鍵ファイルのアルファベット順である。

図 5 に検証実行画面を示す。署名ファイル名を入力し、検証開始ボタンを押すことにより、署名対象ファイル名、署名ファイルの更新日時、鍵ファイル名、検証結果が表示される。

```
<!ELEMENT SchnorrKeyValue (P,Q,G,Y)>
<!ELEMENT P (#PCDATA)>
<!ELEMENT Q (#PCDATA)>
<!ELEMENT G (#PCDATA)>
<!ELEMENT Y (#PCDATA)>
```

図 3: 追加した要素 (SchnorrKeyValue)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xm1-c14n-20010315" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#ring" />
    <Reference URI="message.doc">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>LTyx0TUyMDIyNzA3MjkxOD...</DigestValue>
    </Reference>
  </SignedInfo>
  <RingSignatureValue>
    <c>NDY2NzQ4NzE4ODM1OTIwMTg0NTkxNj...</c>
    <s Id="kikuchi">MTAyNTc1OTA0MDc4ODUyNjM...</s>
    <s Id="ueyama">NDc5OTI3MjQ2NDk3MTE1NT...</s>
  </RingSignatureValue>
  <KeyInfo Id="kikuchi.rpub">
    <KeyValue>
      <RSAKeyValue>
        <Modulus>MTU0NjUxODEOM...</Modulus>
        <Exponent>NjU1Mzc=</Exponent>
      </RSAKeyValue>
    </KeyValue>
  </KeyInfo>
  <KeyInfo Id="ueyama.spub">
    <KeyValue>
      <SchnorrKeyValue>
        <P>ODc5NzE4ODM1OTIwMTg0NTkxNj...</P>
        <Q>OTk3MjQ2NDk3MTE1NT...</Q>
        <G>NDY2NzE4ODM1OTIwMTg0NTkxNj...</G>
        <Y>ODQwNDc5NzE4ODM1OTIwMTg0NTkxNj...</Y>
      </SchnorrKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

図 4: XML 署名のサンプル ($n = 2$)

表 4: 実装システムの仕様	
対応アルゴリズム	RSA, Schnorr
平文形式	Detached(バイナリ)
実装環境	java (TM) 2 SDK Standard Edition version.1.4.1 Swing

4.3 パフォーマンス

リング署名の実用性を検証するため処理速度を計測する。アルゴリズムの種類の違いによる処理速度を求めるため、ユーザ数 n に対する署名生成、検証時間を、RSA と Schnorr それぞれについて計測した。署名生成、検証時間を図 6 に示す。測定環境は、Celeron M 1.20GHz, 256MB である。このとき署名対象ファイルは 300KB のワードファイルである。

RSA の署名生成時間 $T_{RE}(n)$, 検証時間 $T_{RD}(n)$, Schnorr の署名生成時間 $T_{SE}(n)$, 検証時間 $T_{SD}(n)$ を各々次のように得た。

$$T_{RE}(n) = 3.0762n + 104.32$$

$$T_{RD}(n) = 3.3659n + 31.98$$

$$T_{SE}(n) = 26.234n + 57.217$$

$$T_{SD}(n) = 25.711n + 77.946$$



図 5: 署名検証実行画面

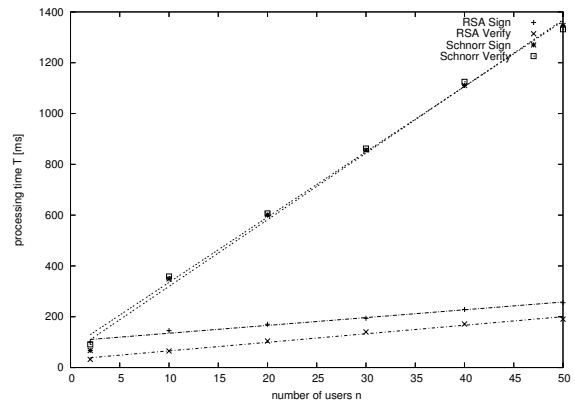


図 6: 署名生成、検証時間

この式より、鍵の違いによる処理時間を表 5 にまとめた。RSA では $e = 2^{16} - 1$ の小さな暗号鍵を用いているので、検証時間が極端に小さい。

表 5 より、秘密鍵の処理は Schnorr の場合が早く、公開鍵の処理は RSA の場合が早いことがわかる。従って、リングの構成によって処理時間が大きく変動するので、サイドチャネル攻撃に対する考慮が必要である。

表 5: 鍵の違いによる処理時間

	秘密鍵 [ms/回]	公開鍵 [ms/回]
RSA	104.3	3.08
Schnorr	57.22	26.23

ユーザ数 n に対する生成された XML 署名ファイル長を図 7 に示す。

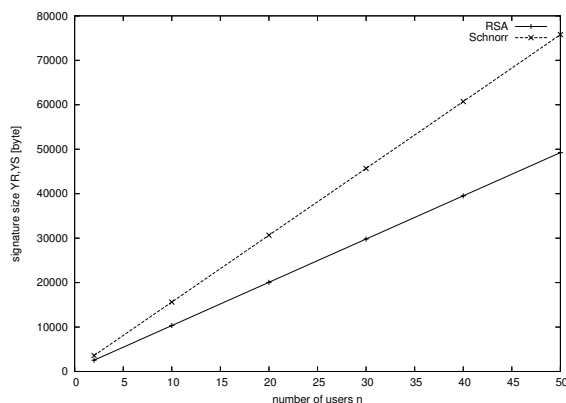


図 7: ユーザ数 n に対する XML 署名ファイル長

5 おわりに

リング署名のアドホック性 (setup-free) という特徴を活かして、署名アルゴリズムを混在する、リング署名アルゴリズムを示した。また実装をして、パフォーマンスを明らかにした。本ツールでは RSA と Schnorr を混在したリング署名を使用した。鍵の違いにより、秘密鍵の処理は Schnorr の場合が早く、公開鍵の処理は RSA の場合が早いことがわかった。

参考文献

- [1] 大久保, 阿部, 鈴木, 辻井, 署名長が短い 1-out-of-n 証明, 暗号と情報セキュリティシンポジウム SCIS2002, pp. 189-193(2002).
- [2] 菊池, 多田, 中西, リング署名における署名者の証明と匿名性破棄プロトコル, 情報処理学会論文誌, Vol. 45, No. 8, pp. 1881-1886(2004).
- [3] W3C XML-Signature Syntax and Processing, (<http://www.w3.org/TR/xmlsig-core/>, 2006 年 2 月 20 日参照).
- [4] 丸山 宏, 「XML と Web サービスのセキュリティ-XML デジタル署名と暗号化」, 共立出版, 2004.
- [5] Rivest, R., Shamir, A. and Tauman, Y. : How to leak a secret, Advances in Cryptology-ASIACRYPT 2001, LNCS, Vol. 2248, pp. 552-565(2001).
- [6] Chaum, D. L. and Van Heyst, E. : Group Signatures, Advances in Cryptology - EUROCRYPTO '91, pp. 257-264, (1991).
- [7] 岡田, 吉田, 加藤, 放送型暗号を利用したグループ署名のリポーク方式, 暗号と情報セキュリティシンポジウム SCIS2004, pp. 1173-1178, (2004).
- [8] 多田, 岡田, 指名確認者リング署名, 暗号と情報セキュリティシンポジウム SCIS2005, pp. 1777-1782, (2005).
- [9] 高橋, 土井, 辻井, 署名者同一性検証可能な 1-out-of-n 署名方式の改良, 暗号と情報セキュリティシンポジウム SCIS2005, pp. 1783-1788, (2005).
- [10] 駒野, 太田, 新保, 川村, 否認機能を持つリング署名方式の再考, 暗号と情報セキュリティシンポジウム SCIS2005, pp. 1789-1794, (2005).
- [11] B. Ramsdell : Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification, Internet RFC 3851, 2004.