コンピュータセキュリティ 9－6
（2000．5．10）

2色木によるオンライン証明書状態検証サーバの実装と評価

安部 謙介[1]　菊池 浩明　中西 祥八郎

東海大学工学部

概要: 公開鍵基盤 PKI において，証明書の失効処理は重要な課題の一つである．古典的なリスト構造による標準的な失効処理方式に代って，一方向性ハッシュ木を用いた効率的なプロトコルが提案されており，[KA00] では，その通信と計算処理に関するコスト削減が検証されている．しかしながら，このハッシュ木によるアプローチは最悪の場合，すなわち，全てのエントリーが降順にソートされているときには，$O(n)$ のオーダにまで悪化してしまう．二色木 (red-black tree) は，節点当り一ビットの追加情報を用いて，検索と挿入を最悪の場合でも $O(\log_2 n)$ で処理することを保証した二分探索木である．本稿では，この二色ハッシュ木を用いたオンライン証明書検証を考察し，通信と処理量についての2分検索木に対するコスト削減を評価する．

Expected Reduction of Cost for Online Certification Status Verification With Red-Black Hash Tree

Kensuke Abe　Hiroaki Kikuchi　Shohachiro Nakanishi

Tokai university, Faculty of Engineering

Abstract: Certificate Revocation is one of the critical issues for a practical public-key infrastructure. A new efficient revocation protocol using one-way hash tree structure instead of the classical list structure, which is known as a standard for revocation, was proposed and examined in communication and computation costs reduction [KA00]. A tree approach, however, might be of $O(n)$ in the worst case when all entries are sorted in descending order. A red-black tree is a binary sorted tree with one extra bit per node, which is used for balancing tree and to guarantee that operations of search and insertion take $O(\log_2 n)$ in the worst case. In this paper, we study the red-black hash tree for online certificate status verification and estimate the reduction of costs against the binary search tree in terms of communication and computation costs in revocation.

1　はじめに

公開鍵証明書の失効処理は公開鍵基盤 PKI における重要な課題の一つである．デファクトスタンダードであるリスト構造による CRL[X.509AM] には，発行間隔が広く即時的な利用に不向きな点と、通信コストが大きいが問題で認識されており，これを改善するために一方向性ハッシュ関数による木構造を用いた証明書廃止木 (CRT:Certificate Revocation Tree) [CRT] が提案されている．

通常の二分探索木 (Binary Search Tree) による CRT では，検索，挿入，削除の基本操作は，廃止する証明書の数 n について平均して $O(\log n)$ で行われる．しかしながら，これは失効する証明書が独立にランダムに発生するときの平均であることに注意しよう．全てのシリアル番号が降順になってしまう最悪の

[1]現在，松下通信工業 (株) 勤務

場合には $O(n)$ の時間がかかる．しかも，現実には，一連のシリアル番号がバルクで失効することは頻繁に生じ得るので，この可能性は十分考慮しなくてはならない．

この問題に対する効果的な対処法は，木の平衡化アルゴリズムである．回転を行ってバランスをとるAVL木，木の節点の次数を変更してバランスをとるB木などが良く知られている [ALG]．本稿では，木の各節点に1ビットの属性 (赤と黒) を持たせて，バランスをとる2色木のアルゴリズムを適用する．この試みにおける我々の興味には次がある．

- 木を2色木としたとき，通信コストと検証処理コストは各々平均してどれだけ削減されるか？
- 通常の2分探索木に対して，2色木にすることで証明書ディレクトリにどれだけオーバヘッドが生じるか．
- 実際に運用されている CRL に対して，2色木の効果はどのくらいか？

そこで我々は，Java を用いて2色木を用いたオンライン証明書状態検証サーバの実装を行った．本稿では，システムの実装とそのパフォーマンスを報告し，通常の2分探索木に対する2色木の証明書失効管理における有効性を議論する．

2 2色木による証明書失効管理

2.1 PKI モデル

本稿で考える PKI モデルは、CA、ディレクトリ (サーバ)、エンドユーザから成る。ディレクトリサーバは、独自の秘密鍵を持ち検証サービスを行う。

2.2 2分検索木

2分検索木は，n 個のノードから成るデータ構造である．各節点は、p, key, $left$, $right$ の属性を持つ．節点 x の右部分木に属する任意の節点 y，左部分木に属する任意の節点 zについて，$key[y] \leq key[x] \leq key[z]$ が成り立つ．

2.3 2色木

2色木とは次の条件を満たす2分検索木である [ALG].

1. 各節点は，赤か黒である．
2. 葉 (NIL) は全て黒である．
3. ある節点が赤ならば，子は両方とも黒である．
4. 根から葉までのどの経路も同じ黒高さ $hb(x)$ をもつ．

ここで，$hb(x)$ は節点 x から葉までの経路上の黒節点の数である．

n 個の内点を持つ2色木は，高々$2\log(n+1)$ の高さをもつことが証明されている [ALG]．一般に，木への挿入や検索などの処理は高さに比例するので，これらも同様の計算量をとることが予想される．実際，挿入に関しては次の効率的なアルゴリズムが開発されている．

```
アルゴリズム RB-Insert(T, x)
1  Insert(T, x)
2  color[x] = Red
3  if color[y] = Red
4     then Rotate1(T, x)
5     else if x が右 (左) の子
6             then Rotate2(T, x)
7             else Rotate3(T, x)
8  color[root[T]] = Black
```

ここで，yはxの叔父，すなわち，$y = right[p[p[x]]]$ または $y = left[p[p[x]]]$ のどちらかである．新に挿入される節点 x は，まず1行目で通常の挿入が行われて赤に塗られる．yによって，3つ (左右対称を区別すると6つ) に場合分されて各々の回転処理 Rotate1,2,3 が施される．Rotate1 の時だけは，$x = p[p[x]]$ と更新されて，根まで処理を溯っていく．従って，この処理も木の高さに比例した時間がかかる．アルゴリズムの詳細と削除の処理は，[ALG] を参照．

図 1に，空の木にデータ $41, 38, 31, 12, 19, 8$ を順に挿入して出来る2分検索木と2色木を示す．アルゴリズムに従って回転を施すことにより，2色木には，根 38 から葉への経路の全てに丁度3つ (NIL を含む) の黒節点が現れている．

2.4 内部データ形式

X.509 CRL において，各廃止証明書は図2の ASN.1 で指定される．従って，これらをハッシュ木で表現するには，各節点に廃止する証明書のシリアル番号 (任

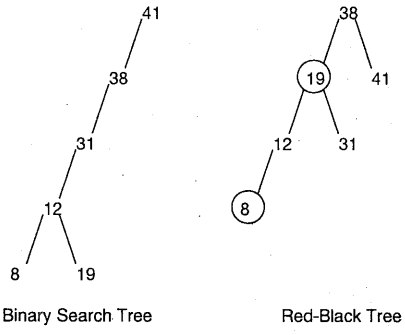

図 1: 2 色木の例 (41,38,31,12,19,8 の順に挿入)

意長) と失効日時の属性が最低限必要である．更に，色の属性とその節点のハッシュ値を加えなくてはならない．節点間の関係は，木に挿入する順序で一意に記述する．すなわち，n 個の節点を持つディレクトリと発行局の木の内部構造は，

$$Z = z_1, z_2, \ldots, z_n$$

の形式をしている．ここで，$i = 1, \ldots, n$ について

$$z_i = k_i||H_i||c_i||s_i||v_i$$

とする．s_i は廃止証明書のシリアル番号 (CertificateSerialNumber), v_i は失効日時 (Time) であり，一方向性ハッシュ関数 h を用いて節点のキーを $k_i = h(s_i)$ とする．c_iは節点の色を表す論理値であり，true が赤，false が黒である．また，記号 || はデータの連結を示し，あいまいさなく各属性を取出すことができるものとする．

節点 x_iのハッシュ値 H_iは，再帰的に

$$H_i = h(x_i) = h(h(left[x_i])||k_i||h(right[x_i]))$$

と定義される．ただし，$h(NIL) =$ null とする．ここで，ハッシュ値の計算は節点の色に無関係に行われることに注意しよう．色の属性は木の更新時にだけ利用される情報であり，それゆえに，クライアントから見た検証処理と通信データは通常の 2 分検索木のものと変わらない．

図 1の 2 色木は，表 1の内部データ形式で表現される．簡単化のため，シリアル番号とキーを同一にし ($k_i = s_i$)，執行日時 s_iを省略している．データの挿入順とは異なり，$38, 19, 12 \ldots$ の根からの順になっていることに注意されたし.(アルゴリズムの元では生成される木は一意なので，必ずこの順序にする必然性はないが，木を再構築する際の平衡化処理を節約できる.)

表 1: 例の 2 色木を表現する内部保存データ

k_i	H_i	c_i
38	6668E1611A4AB23E914331331289A436	true,
19	ADCF72C11A19927D45E3186802694145	false,
12	1AB899864B6F979EB680D52C779AD659	true,
8	C4D9042407EE87310AB5EB633D81EF35	false,
31	7BFF3DE72AC837CF4DD82476688AF75B	true,
41	8D2F06B58513A943BD634C595E85FFBB	true

2.5 通信データ形式

クライアントからの検証要求に対して，ディレクトリは，検証に必要な木の根から該当節点までの冗長でない経路 Pで与えられる部分木を応答する．m 個の節点からなる部分木は，

$$Y = y_1, y_2, \ldots, y_m$$

の形式で定められ，$j = 1, \ldots, m$ について，

$$y_j = \begin{cases} k_i & \text{if } x_i \in P \\ k_i : H_i & \text{if } p[x_i] \in P \end{cases}$$

である．このように，クライアントで根までのハッシュ値を再計算するためには，経路 Pに隣接する節点のハッシュ値が必要である．キーが該当する廃止証明書であるときは，明示的に $[k_i]$ と書きあらわす．

表 2は，図 1の 2 色木の上でキー 19 を証明するための部分木を表す通信データである．この部分木は，表 1の木のハッシュ値を再計算するための必要十分な情報を含んでいる．

表 2: キー 19 を証明する部分木を表す通信データ

k_i	H_i
38	N/A,
[19]	N/A,
12	1AB899864B6F979EB680D52C779AD659,
31	7BFF3DE72AC837CF4DD82476688AF75B,
41	8D2F06B58513A943BD634C595E85FFBB

3 評価

3.1 システム構成

[KA00] のシステムを拡張し，2 色木を用いた証明書状態検証サーバを実現した．本システムの緒言を表 3に示す．システム全体のシステム構成，初期状態，処理命令系などは [KA00] に等しい．

```
revokedCertificates     SEQUENCE OF SEQUENCE  {
        userCertificate         CertificateSerialNumber,
        revocationDate          Time,
        crlEntryExtensions      Extensions OPTIONAL -- if present, shall be v2
                            }  OPTIONAL
```

図 2: 廃止証明書エントリー

表 3: システム緒言

プラットフォーム	Sun Ultra S-7/300U 167MHz Solaris 2.5
開発言語	JAVA Development Kit Ver.1.1
ハッシュ関数	MD5(coded in JAVA)
通信プロトコル	独自ソケット通信方式

3.2 通信コスト

図 3は，検証時にディレクトリからクライアントへ送られる通信量の大きさを示している．2 色木と 2 分検索木のそれぞれについて，対象とする証明書が既に失効されている時 (OK), 未失効である時 (NG) の 2 つの場合に分けて測定した．証明書の失効はランダムに行っている．通信コストは，検証する証明書に対応する節点の高さに大きく依存するため，100 回検索してその平均値を求めた．

失効証明書数 n に対して，対数スケールになっており，その上で線形に増加しているので，理論どおり対数オーダーになっていることが観測できる．また，n に対して失効済か否かの差は無視できるほど小さい．

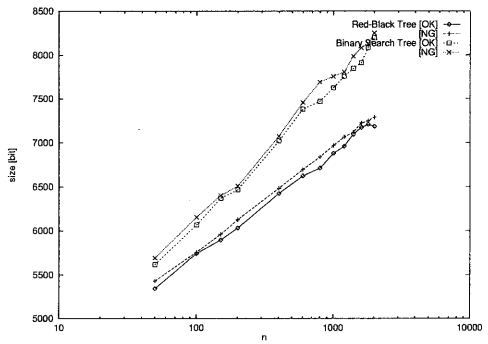

図 3: 通信コストの削減

3.3 計算コスト

2 色木にすることによって，処理時間は検証 (検索) 時に短縮し，挿入時はに増加する．この各々の変化量を，各々，図 4と図 5 に示す．測定には，OS のクロックを用いて 1000 回行った平均を取った．

検索で生じる差は，通信コストほど顕著には出ていない．これは，Java 処理系で引き起こされるガベージコレクションなどのリアルタイム性の不均等が影響していることが予想される．逆に言うと，検索時の処理削減量は OS の影響が問題になるほど微細な差しか生じない．ハッシュ計算の処理は，キャッシングの影響があるが全体としては同様の傾向を示すと考えられる．

一方，挿入時には節点の回転などの平衡化のオーバヘッドがあるため，2 色木が遅くなることが予想されたが，図に示されるように大きな差は生じていない．なお，節点削除の処理も同様である．

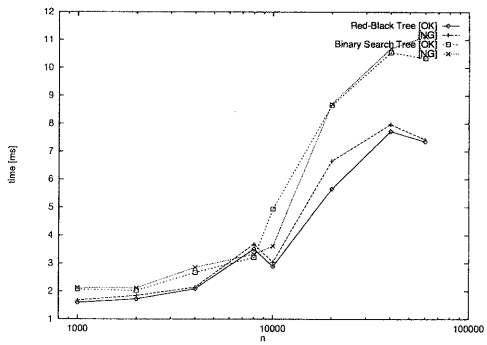

図 4: 検索 (検証) 処理コストの削減

3.4 実際の CRL データに対する評価

表 4の実際の CRL データを，2 色木に入力した場合、どの程度不平衡になるかを調べた．CRL はシリ

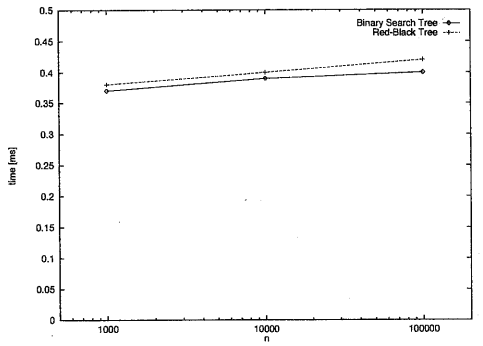

図 5: 挿入処理コストの増加

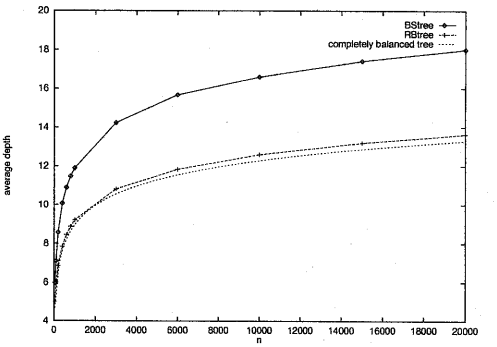

図 6: 実 CRL データに対する平均深さの変化

アル番号でソートされているので，これを失効日時順に並び変えて入力とした．

表 4: CRL 詳細

発行者	VeriSign, Inc.
CRL 名	RSASecureServer.crl
期間	1997.02.14 - 1999.09.27
無効証明書数	20336
シリアル番号のサイズ	128 [bit]

この失効データに対する平均高さ（全節点の高さの平均）は，図6のようになった．2 分検索木，2 色木の実測値に完全に平衡したときの平均高さを加えている．いずれも，$O(\log n)$ で増加しているが，2 色木は 2 分検索木より常に低く，$n = 20000$ の時で，0.75 倍に減少している．検索，挿入などの操作時間と検証時の通信コストは平均高さに比例するので，結局，この差が 2 色木の効果を表す．一方，2 色木は完全平衡木に漸近しており，$n = 20000$ の時ですら，平均深さは完全平衡木の 1.025 倍の大きさである．公開鍵証明書の失効は，独立な事象ではなく，シリアル番号に対してある拘束条件のもとで生起しているであろう，とした本研究の導入で行った仮説はこの結果で裏付けされたといえよう．

4 結論

2 色木を用いて，公開鍵証明書の失効処理を高速化する提案を行った．Java を用いたオンライン証明書状態検証システムを試験実装し，そのパフォーマンスを報告した．通常の 2 分検索木に比べて，通信コスト，検索時の計算コストが削減することを示した．実運用を行っている発行局の CRL データに基づいて，提案方式を評価した結果，$n = 20000$ の時で，2 色木の平均高さは 2 分検索木の 0.75 倍に減少していることが明らかになった．これは，理論上の上限の 1.025 倍でしかない．

参考文献

[ALG] T. Cormen, C. Leiserson and R. Rivest, *Introduction to algorithms*, MIT Press, 1990 (浅野らによる邦訳「アルゴリズムイントロダクション (近代科学社)」あり)

[X.509AM] ITU-T Recommendation X.509—ISO/IEC 9594-8: 1995

[PKIX] R. Housley, W. Ford, W. Polk, D.Solo, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Internet RFC 2459, 1999

[OCSP] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, Internet RFC 2560, 1999

[CRT] P. Kocher, A Quick Introduction to Certificate Revocation Trees (CRTs), `http://www.valicert.com/company/crt.html`

[NN98] Mni Naor and Kobbi Nissim, Certificate Revocation and Certificate Update, in proc. of Seventh USENIX Security Symposium, pp.217-228, 1998

[RFC2510] C. Adams, S. Farrell, Internet X.509 Public Key Infrastructure Certificate Management Protocols, Internet RFC 2510, 1999

[RFC2559] S. Boeyen, T. Howes, P. Richard, Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2, Internet RFC 2559, 1999

[RFC2585] R. Housley, P. Hoffman, Internet X.509 Public Key Infrastructure Operational Protocols, Internet RFC 2585, 1999

[RFC2587] S. Boeyen, T. Howes, P. Richard, Internet X.509 Public Key Infrastructure LDAPv2 Schema, Internet RFC 2587, 1999

[KA98] 菊池, 安部, 中西, 2 分ハッシュ木を用いた証明書廃止・更新システム, 情報研報 Vol.98, No.84,pp.51-56,1998

[SCIS99] 菊池, 安部, 中西, k分ハッシュ木による証明書廃止木 (CRT) の更新方式の提案と評価, 1999 年暗号と情報セキュリティシンポジウム (SCIS'99), Vol.2, pp.621-626, 1999

[KA00] 安部, 菊池, 中西, ハッシュ木を用いたオンライン証明書状態検証サーバ, 情処研報 (2000-CSEC-8), Vol. 2000, No. 30, pp.131-136, 2000

[IWSEC] H. Kikuchi, K. Abe, S. Nakanishi, Performance Evaluation of Public-key Certificate Revocation System with Balanced Hash Tree, proc. of International Workshop on Security (IWSEC'99), 1999

[ISW] H. Kikuchi, K. Abe, S. Nakanishi, Performance Evaluation of Certificate Revocation Using k-Valued Hash Tree, International Information Security Workshop (ISW'99), Springer Lecture Notes in Computer Science 1729, pp.103-117, 1999

[SSLeay] E. Yang, SSLeay, `http://www.ssleay.org`

[VeriSign] VeriSign, `http://www.verisign.com`

[JAVA] Robert Lafore[著], 岩谷 宏 [訳], Java で学ぶアルゴリズムとデータ構造, SOFTBANK, 1999