XML データベースへの推論攻撃に対する安全性検証システムの開発

石　原　靖　哲　　大阪大学大学院情報科学研究科准教授

1 まえがき

近年，企業等多くの組織でデータベースが運営・活用されている．それらのデータベースには，少なからず機密性の高い情報が格納されており，その情報の安全性を守ることは組織の信頼を確保するためにも重要なことである．データベースでの機密情報の漏洩を防ぐ手法として問合せによるアクセス制御がある．この手法では，データベースはユーザに許可されている問合せに対してのみ，その問合せに対する答えを返す．これにより，機密情報への直接のアクセスを制限し漏洩を防ぐ．しかし，そのように制限していたとしても，ユーザは許可されている問合せとその結果から，許可されていない問合せの結果（機密情報）を推論し，得ることができてしまう場合がある．このような攻撃を推論攻撃と呼ぶ．XML データベースにおける推論攻撃の例を次に示す．

学生の氏名とその出身，貸与されている奨学金の金額の対応を表す XML 文書 D を考える．D は以下のスキーマにしたがっているとする．

学部→学生*
学生→氏名,出身,奨学金
氏名→string
出身→“大阪”｜“京都”
奨学金→“0”｜“5 万”｜“8 万”

T_1 は学生の氏名と出身の組を，T_2 は奨学金が貸与されている（すなわち奨学金の額が 0 以外の）学生の出身と奨学金の組を，それぞれ元の文書における出現順序を保存したまま取り出す問合せとする．今，T_1 および T_2 が許可されており，問合せ結果 $T_1(D)$，$T_2(D)$ がそれぞれ図 1(a),(b)に示すとおりであったとする．このとき，D は図 1(c)に示す文書しかありえないことがわかる．

推論攻撃によりどのような情報が得られるかは一般に自明ではないため，データベース管理者は，データベースの安全性確保のために，推論攻撃によって機密情報が漏洩する可能性の有無をあらかじめ把握しておくことが重要である．

報告者の研究グループではこれまでに，XML データベースを対象とした推論攻撃に対する安全性を定式化し，その検証法を提案してきた[THIF06],[THIF07]．この定式化では，推論によって機密情報の値の候補が絞り込まれないことを安全と考えている．具体的には，与えられたデータベースのスキーマや，許可されている問合せとその結果，機密情報を得るための問合せに対し，機密情報の候補値が有限個にならないことを無限安全，k 個未満に絞り込まれる可能性がないことを k 安全と定式化している．提案されている検証法は，各問合せの結果とスキーマのサイズに対して多項式時間で実行できることが示されているが，安全性の定式化にしたがって単純に検証するナイーブな手法であり，実用的な観点からの効率について検討が不十分であった．

本研究では，この検証法に基づくシステムを実装し，検証法の効率に関する評価を行った．その結果，「#の埋め込み処理」と呼ばれる処理が実用上ボトルネックとなることを明らかにした．さらに，「#の埋め込み処理」を改良した検証法を提案・実装し，検証時間および使用メモリ量の両方が改善されることを確認した．

2 関連研究

XML データベースへの推論攻撃に関する研究としては，文献[FCG04],[YL04]などがある．文献[FCG04]では，攻撃者が機密情報を推論するのに有用な情報を含まない security view という概念を提案しているが，攻撃者の推論の定式化が明確ではない．文献[YL04]では，攻撃者がスキーマ情報と関数従属性を用いて推論を行うという仮定のもとで，安全な極大のビューを求めるアルゴリズムを提案している．これらはいずれもユーザに単一のビューのみを与える環境を想定した研究である．これに対し本研究では，ユーザに複数のビューを与えるという，より可用性が高い環境を想定している．

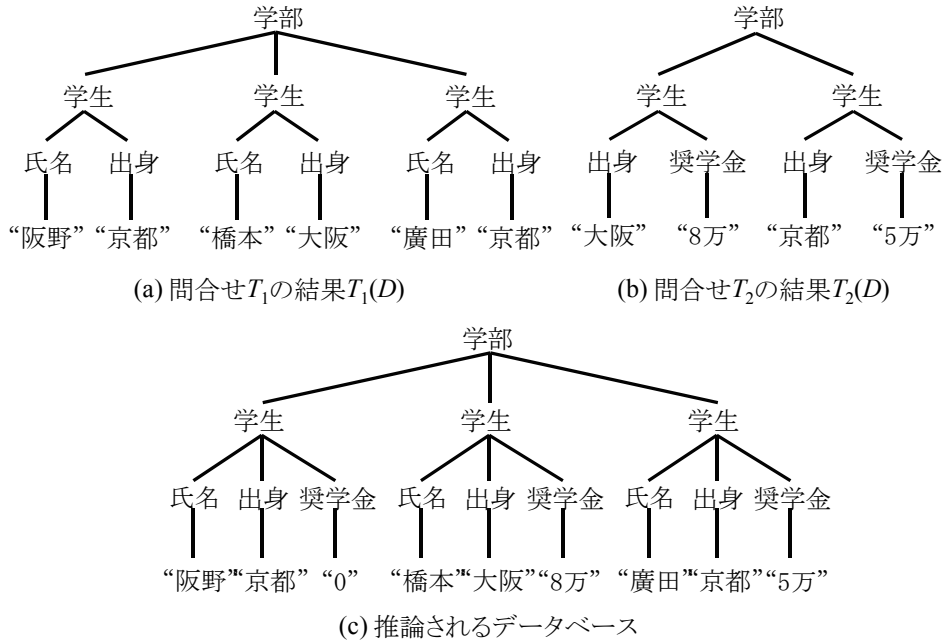

図 1　推論攻撃の例

近年では，プライバシ保護の文脈で推論攻撃に対する安全性を議論する研究も多い．文献[Sw02]では，安全性の一定式化として，k 匿名性の概念を提案している．個人情報を格納する関係データベースにおいて，各個人に該当しうるタプルが k 組以上存在するとき，そのデータベースは k 匿名であるという．一方，文献[MGKV06]は，k 匿名性の概念の欠点を指摘したうえで，ℓ多様性の概念を提案している．この概念では，各個人の機密情報の値の候補が十分に多く（たとえばℓ種類以上）存在することを安全であると考えており，本研究における k 安全性と同じ考え方であると言える．

3 検証で用いるモデル

本節では XML データベースに関するモデルについて述べる．

3-1 XML 文書のモデル

XML 文書のモデルとしてラベル付き順序木を用いる．アルファベットΣ上のラベル付き順序木の集合を $\mathbf{T}_\Sigma$ と書く．$t\in\mathbf{T}_\Sigma$の大きさ$|t|$は t の頂点の数である．以下ではラベル付き順序木を単に木と呼び，しばしば項により表現する．

3-2 XML スキーマのモデル

XML スキーマのモデルには，非決定性木オートマトンを用いる．非決定性木オートマトンは以下の 4 つ組 $A=(Q, \Sigma, q_0, R)$ である：

- Q: 状態の有限集合．
- Σ: 入力アルファベット．
- $q_0\in Q$: 初期状態．
- R: 遷移規則の集合．ただし，$q\in Q$, $a\in\Sigma$とし，e を Q 上の言語を受理する非決定性有限文字列オートマトンとすると，遷移規則は(q, a, e)の形式である．

木 $t=a(t_1 \cdots t_n)$を考える．ここで，$t_1, \ldots, t_n \in\mathbf{T}_\Sigma$である．木 t の根頂点に状態 q が割り当てられたとき $q(t)$と書く．

$q_1 \cdots q_n \in L(e)$であるような$(q, a, e)\in R$が存在するならば，Aは $q(t)$から $a(q_1(t_1) \cdots q_n(t_n))$へ遷移可能であると定義する．ここで$L(e)$は文字列オートマトン eが受理する文字列言語である．$t\in\mathbf{T}_\Sigma$に対して，$q_0(t)$から始めて，以上のような遷移をトップダウンに各部分木について繰り返すことによって，最終的に tへ遷移可能であるとき，Aは tを受理するという．

Aが tを受理することは，XML 文書 tがスキーマ Aにしたがうことに対応する．Aによって受理される全て

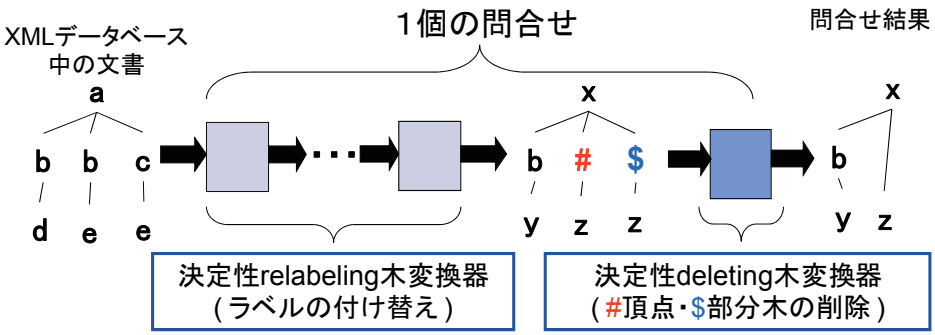

図 2　問合せのモデル

の木の集合を $TL(A)$ と書く．A の大きさ $|A|$ は $|Q|+\Sigma_{(q,a,e)\in R}|e|$ と定義する．ここで，$|e|$ は文字列オートマトン e の大きさであり，状態数と各遷移規則における遷移先の状態集合の大きさの総和と定義される．

3-3 XML 文書への問合せのモデル

本研究では，2 種類の決定性木変換器（決定性 relabeling 木変換器，決定性 deleting 木変換器）の合成を XML 文書への問合せモデルとして用いる．

決定性 relabeling 木変換器は，ラベルの付け替えのみを行う木変換器であり，動作する方向によりトップダウン型とボトムアップ型がある．トップダウン型の場合は先祖の情報のみに基づいて，ボトムアップ型の場合は子孫の情報にのみ基づいて，ラベルの付け替えを行う．一方，決定性 deleting 木変換器は，木を根頂点からトップダウンにたどり，#とラベル付けされている頂点を削除する機能と，$とラベル付けされている頂点以下の部分木を削除する機能をもつ．

形式的には，木変換器は状態集合，入力アルファベット，初期状態，変換規則の 4 つ組で指定される．木変換器のサイズは，木オートマトンと同様に，木変換器の状態数と各変換規則に現れる文字列オートマトンのサイズの総和と定義される．

本研究では，木変換器の合成の形式，ならびに木変換器の機能を次のように限定する．まず，問合せは図 2 のような，relabeling 木変換器の系列と deleting 木変換器の合成で定義されるものを考える．そして，ユーザに許可される問合せ中の deleting 木変換器は，#とラベル付けされた頂点を削除する機能のみ，D 中の機密情報への問合せ T_S 中の deleting 木変換器は，$とラベル付けされた頂点を削除する機能のみをもつものと限定している．

以上のようにモデル化された問合せは，周囲の頂点の情報を基に，指定した位置にある頂点のラベルの付け替えや削除をするフィルタリングの変換を行う機能をもつ．データベースへの問合せの多くは，一部の情報を切り出すフィルタリング機能をもつクラスで表されるため，このモデルは実用的なクラスを対象にしているといえる．この問合せクラスは，XML 変換言語の XSLT の部分クラスである．

4 安全性の定義と検証法

本節では，データベースへの推論攻撃に対する安全性定義と検証法について述べる．

4-1 安全性の定義

ユーザに与えられる情報は以下の通りである．

- $T_1, \ldots, T_n$: ユーザに許可されている問合せ．
- $T_1(D), \ldots, T_n(D)$: データベース中の XML 文書 D に対する問合せ $T_1, \ldots, T_n$ の結果．
- A_G: D がしたがうスキーマ．
- T_S: D 中の機密情報を返す問合せ．

このとき，これらから推論される機密情報の値の候補集合 C は次のように与えられる．

$$C = \{T_S(D') \mid D' \in TL(A_G), T_1(D')=T_1(D), \ldots, T_n(D')=T_n(D)\}.$$

そして，C を用いて以下の 2 種類の安全性を定義する．

- 無限安全性：C の要素数が無限のとき，D の機密情報 $T_S(D)$ は無限安全であるという．
- k 安全性：C の要素数が k 以上のとき，D の機密情報 $T_S(D)$ は k 安全であるという．

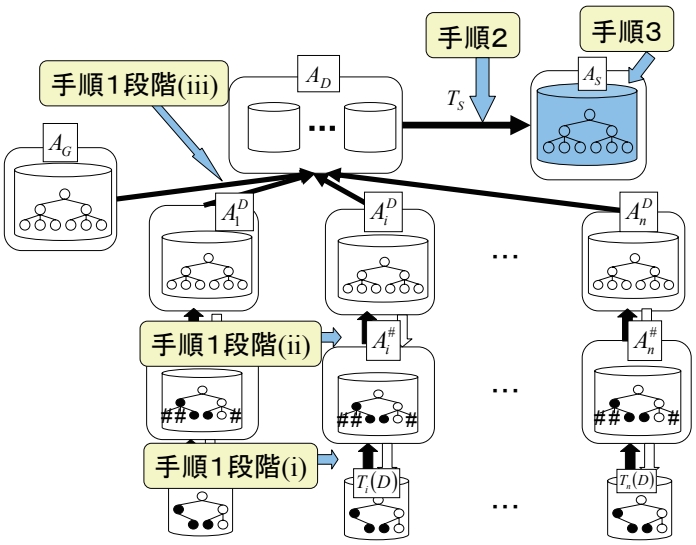

図 3　ナイーブな安全性検証法

4-2 ナイーブな安全性検証法

ナイーブな安全性検証法[THIF06],[THIF07]は以下の 3 つの手順からなる（図 3 参照）.

4-2-1 検証手順 1

検証手順 1 では，ユーザに許可されている各問合せ T_i（i=1, ..., n)とその結果 $T_i(D)$，そして元の XML 文書 D のスキーマ A_Gから，D の候補を受理する木オートマトン A_Dを構成する．この手順はさらに以下の 3 段階に分かれる.

(i) deleting 木変換器についての逆型推論：$T_i(D)$ 中の任意の箇所に#が埋め込まれた木を受理する木オートマトン $A_i^{\#}$を構成する.

(ii) relabeling 木変換器についての逆型推論：(i)で得られた木オートマトン $A_i^{\#}$と，問合せ T_iを構成する relabeling 木変換器の系列から，$T_i(D')=T_i(D)$ となるような XML 文書 D'を受理する木オートマトン A_i^{D}を構成する.

(iii) 各候補集合とスキーマとの交わり：(ii)で得られた木オートマトン A_i^{D}（i=1, ..., n)と，スキーマ A_G との交わりをとることにより，元の XML 文書 D の候補を受理する木オートマトン A_Dを構成する.

4-2-2 検証手順 2

この手順では D 中の機密情報 $T_S(D)$ の候補を求める．手順 1 で求められている A_Dは D の候補を受理する木オートマトンであり，Dの候補に対する問合せ T_Sの結果は，元の XML 文書 D中の機密情報 $T_S(D)$ の候補である．木オートマトン A_Dと T_Sから，$T_S(D)$ の候補を受理する木オートマトン A_Sを構成する方法は以下の 2 段階に分かれる.

(i) relabeling 木変換器についての型推論：Dの候補を受理する木オートマトン A_Dと，問合せ T_Sを構成する relabeling 木変換器の系列から，ラベルを付け替えた後の木を受理する木オートマトン $A_S^{\$}$を構成する.

(ii) deleting 木変換器についての型推論：(i)で得られた木オートマトン $A_S^{\$}$と，問合せ T_S を構成する deleting 木変換器から，$とラベル付けされた頂点を根とする部分木を削除した後の木，すなわち $T_S(D)$ の候補を受理するような木オートマトン A_Sを構成する.

4-2-3 検証手順 3

この手順では手順 2 で求められた木オートマトン A_Sに受理される木の集合の要素が，無限個であるか，あるいは k 個より多いかを検証する．前者の検証（すなわち無限安全性の検証）では，A_Sの遷移規則列が閉路をなすか，または A_Sのある遷移規則(q, a, e)において e が無限長の系列を表現しているかを調べる．後者の検証（すなわち k 安全性の検証）では，A_Sの各状態について，それを初期状態とみなしたときに受理可能な部分木の個数をカウントすることによって，A_Sが受理する木の集合の要素が，0 個，1 個,...，k-1 個，k 個以上のいずれになるかを調べる.

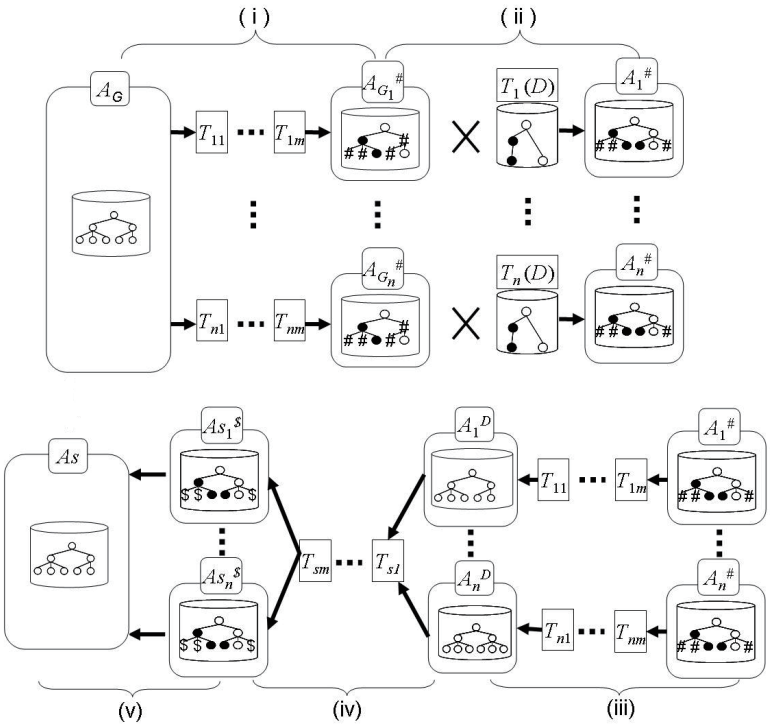

図 4　改良された安全性検証法の手順 1

4-3 ナイーブな検証法の問題点とその改良

次節で述べる実験により,ナイーブな検証法は多大なメモリと時間を要することがわかった.その原因は,手順 1-(i)「deleting 木変換器についての逆型推論」，すなわち，$T_i(D)$のあらゆる箇所に#が埋め込まれた木を受理する木オートマトン $A_i^{\#}$を，逆型推論を用いて構成する処理にあると考えられる．なぜなら，この処理では，$T_i(D)$のあらゆる箇所に#を埋め込むため，作成される木オートマトン $A_i^{\#}$の構成の大部分が，スキーマに一致しないオートマトンとなるからである．ナイーブな検証法では，deleting 木変換器についての逆型推論を行う際，問合せ結果 $T_i(D)$しか用いないため，スキーマに一致するかどうかは，手順 1-(iii)「各候補集合とスキーマの交わりをとる」までわからない．そのため，スキーマに合わないものを非常に多く作成してしまい，多大なメモリや時間を必要としていた．

そこで，スキーマ情報をできるだけ早い時点で利用し，スキーマに合うものだけを生成する新しい#の埋め込み手法を提案するとともに，それに沿った新しい検証法の手順を提案する．この改良された検証法は，以下の 2 つの手順からなる．

4-3-1 改良検証手順 1

各問合せ T_iとその実行結果 $T_i(D)$，スキーマを表すオートマトン A_G，機密情報を求める木変換器 T_Sから，XML 文書 D中の機密情報 $T_S(D)$の値の候補を受理する木オートマトン A_Sを求める．この手順はさらに以下の 5 段階に分かれる(図 4 参照)．

(i) relabeling 木変換器についての型推論：データベースのスキーマ A_Gに対して，問合せ T_i (i=1, ..., n)における決定性 relabeling 木変換器の系列 $T_{i1}, ..., T_{im}$からラベルを付け替え，#とラベル付けされているノードを削除する前の木オートマトン $A_{Gi}^{\#}$を型推論を用いて構成する．

(ii) スキーマ情報を用いた deleting 木変換器についての逆型推論:(i)で得られた木オートマトン $A_{Gi}^{\#}$と，問合せの結果 $T_i(D)$から，スキーマの形に合う#の埋め込みがされた木オートマトン $A_i^{\#}$を構成する．

(iii) relabeling 木変換器についての逆型推論:(ii)で得られた木オートマトン $A_i^{\#}$と,決定性 relabeling 木変換器の系列 $T_{i1}, ..., T_{im}$から,ラベルが付け替えられる前の木,すなわち問合せ T_iの実行結果が $T_i(D)$となるような文書の候補を受理する木オートマトン A_i^{D}を逆型推論を用いて構成する．

(iv) relabeling 木変換器についての型推論：(iii)で得られた木オートマトン A_i^{D}と，機密情報を求める問合せにおける決定性 relabeling 木変換器の系列 $T_{S1}, ..., T_{Sm}$からラベルを付け替え，$とラベル付けされているノードを削除する前の木オートマトン $A_{Si}^{\$}$を型推論を用いて構成する．

(v) 各候補集合の交わりをとる：(iv)で得られた，木オートマトン $A_{Si}^{\$}$ (i=1, ..., n)の交わりをとり，その後，\$とラベル付けされているノードを削除することで XML 文書 D 中の機密情報 $T_S(D)$ の値の候補を受理する木オートマトン A_S を求める．

4-3-2 改良検証手順 2

改良検証手順 1 で求められた木オートマトン A_S が受理する木が，有限個であるか，k 個以下であるかを判定する．この手順 2 はナイーブな検証法の手順 3 と同じである．

5 実装と実験

5-1 実装

改良した検証法の空間計算量がナイーブな検証法に比べて削減されているかを確認するために，検証システムを実装した．実装は Java 言語で行った．規模は 2400 行程度である．

実験を行った際の環境を以下に記す．

- Java: JDK 1.5.0
- CPU: Intel(R) Xeon(R) X5355 2.66GHz
- メモリ: 4GB
- Java VM ヒープ: 1400MB
- OS: Windows XP SP2

5-2 実験に用いた例題

実験には，1 節で述べた例と同じ例題を用いた．ただし，簡単のため，出身要素が取りうる値は 2 種類のみとし，奨学金要素が取りうる値は 0 を含む 3 種類のみとした．また，機密情報を得る問合せ T_S は，与えられた名前を氏名要素に持つ学生要素の氏名要素と奨学金要素を取り出す問合せとした．

5-3 実験 1: 改良の直接的効果の測定

改良した検証法では，手順 1-(ii)「スキーマ情報を用いた deleting 木変換器についての逆型推論」(以降，#の埋め込み処理と記す)で生成される木オートマトンのサイズが，ナイーブな検証法のそれよりも小さくなっていることが期待される．そのことを確認するため，例題の問合せ結果 $T_1(D)$，$T_2(D)$ から#の埋め込み処理を行ったときの出力結果のサイズ等を測定した．

$T_1(D)$ は，「学生の氏名と出身の一覧」を表す木である．この木のみを受理する木オートマトンから，#埋め込み処理を行ったときに出来る木オートマトンサイズを，D 中の学生の数 n を増やしながら測定した(図 5 参照)．この図から，#埋め込み処理後の木オートマトンサイズが大幅に削減されていることがわかる．ナイーブな#埋め込み処理では，スキーマに関係なく木オートマトンのありとあらゆる遷移間に#を処理する遷移を埋め込んでおり，#埋め込み処理後の木オートマトンサイズは $O(n^2)$ であった．一方，改良した#埋め込み処理ではスキーマに合わない#を埋め込まない．改良した#埋め込み処理後の木オートマトンサイズは $O(n)$ となり，理論的に作ることができる最も小さい木オートマトンサイズとほぼ等しかった．

なお，$T_2(D)$ についても同様の結果が得られた．

5-4 実験 2: 改良による検証全体への影響の測定

次に，推論攻撃に対する安全性検証全体にかかる実行時間を測定した．

まず，検証法のメモリ使用量や実行時間が大きくなるのは，$T_2(D)$ に現れる学生の人数が多いときと，検証によって求められた機密情報の値の候補集合 $T_S(D)$ の要素数が多いときであることが,経験上わかっている．これをふまえ，$T_S(D)$ の要素数が多くなるように，各学生要素の子の出身要素の値は同じ値に統一し，貸与される奨学金要素の値も 0 ともう一種類の値のみに制限した．そして，D 中の学生の数を増やしながら，$T_2(D)$ に現れる学生の数が D の半数の場合（このときに $T_S(D)$ の要素数が最も多くなる）についてシステムの実行時間や検証法の各手順終了時の木オートマトンのサイズ等を測定した．

図 6 にナイーブな検証法と改良した検証法で安全性検証を行ったときの実行時間を示す．ナイーブな検証法では，全学生数が 84 人以上になると，手順 1-(ii)「relabeling 木変換器についての逆型推論」でメモリ不足となって検証できない．これは手順 1-(i)「deleting 木変換器についての逆型推論」(#の埋め込み処理)で出力される中間結果の木オートマトンサイズが非常に大きく，それ以降の手順の処理に負担がかかってメモリ不足となったと考えられる．

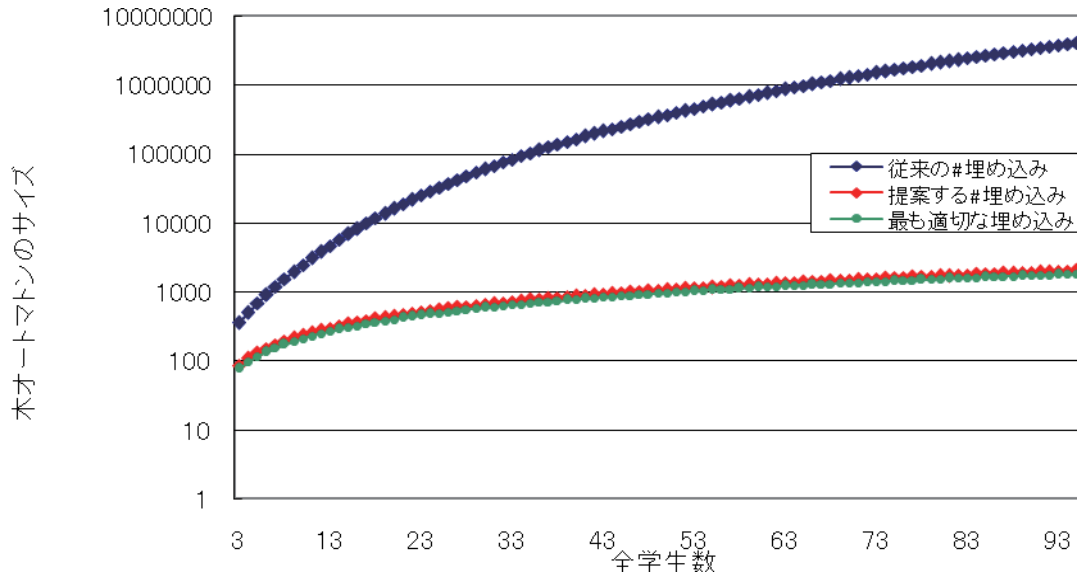

図 5　$T_1(D)$ からの#埋め込み処理後の木オートマトンサイズの比較

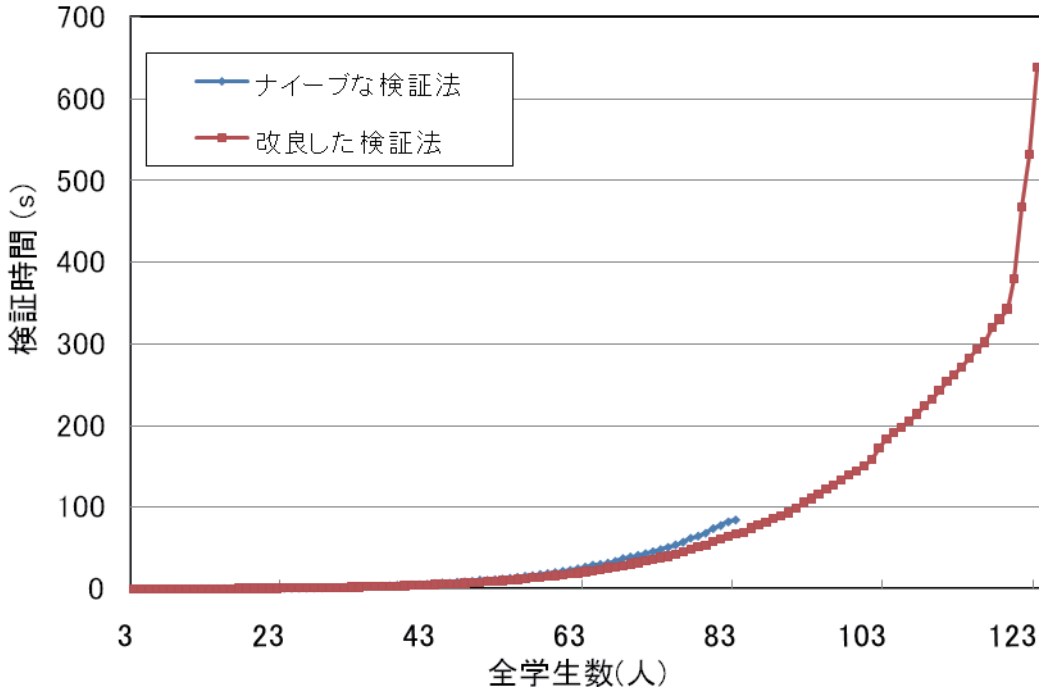

図 6　検証時間の比較

一方，改良した検証法では，全学生数が 124 人以上になると，手順 1-(v)「各候補集合の交わりをとる」でメモリ不足となって検証できなくなった．改良した検証法では，手順 1-(ii)「スキーマ情報を用いた deleting 木変換器についての逆型推論」(#の埋め込み処理)で問合せ結果に加えてスキーマ情報を利用しているので，中間結果が従来に比べて小さくなり，全学生数を 1.5 倍程度に増加させても検証できるようになった．

また，検証の実行時間についても削減できていることがわかる．その理由としては，検証可能人数の増加の理由と同じく，#の埋め込み処理でスキーマ情報を利用することにより，結果の木オートマトンサイズが小さくなって，それ以降に行わなければならない処理が減ったことで，検証時間が削減されていると考えられる．

6 まとめ

本研究では，XML データベースへの推論攻撃に対する安全性検証システムを実装し，効率の評価を行った．その結果，「#の埋め込み処理」が実用上ボトルネックとなることが明らかになった．さらに，「#の埋め込み処理」を改良した手法を提案し，検証時間および使用メモリ量の両方が改善されることを確認した．本報告で述べた例題以外の例題にも適用したところ，データベースの内容にもよるが，1.5 倍から 10 倍程度のサイズのデータベースに対してもメモリ不足にならず，また検証時間についても 20%から 90%程度削減できた．

一方，改良後の検証法においては，手順 1-(v)「各候補集合の交わりをとる」が新たにボトルネックとなることがわかった．現状では，木オートマトンの遷移規則の総当たり的な組み合わせを考えて交わりを求めているため，最終的に不要となる組み合わせを早めにふるい落とす工夫が必要である．

【参考文献】

[FCG04] Fan, W., Chan, C.Y., Garofalakis, M.N.: Secure XML Querying with Security Views, Proc. 2004 ACM SIGMOD Conf., pp. 587-598, 2004.

[MGKV06] Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-Diversity: Privacy Beyond k-Anonymity, Proc. 22nd ICDE, p. 24, 2006.

[Sw02] Sweeney, L.: k-Anonymity: A Model for Protecting Privacy, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, Vol. 10, No. 5, pp. 557-570, 2002.

[YL04] Yang, X., Li, C.: Secure XML Publishing without Information Leakage in the Presence of Data Inference, Proc. 30th VLDB, pp. 96-107, 2004.

[THIF06] 高須賀 史和, 橋本 健二, 石原 靖哲, 藤原 融: XML データベースへの推論攻撃による機密情報特定可能性の形式化とある前提条件のもとでの特定可能性検証法の提案, DBSJ Letters, Vol. 5, No. 2, pp. 21-24, 2006.

[THIF07] 高須賀 史和, 橋本 健二, 石原 靖哲, 藤原 融, XML データベースへの推論攻撃に対する安全性のいくつかの定式化とある前提条件下での安全性検証法の提案, SCIS2007, 4E2-1, CD-ROM (概要集 p. 351), 2007.

〈発 表 資 料〉

題　　名	掲載誌・学会名等	発表年月
XML データベースへの推論攻撃に対する安全性検証システム	情報通信システムセキュリティ研究会予稿集	2007 年 9 月
Verification of the Security against Inference Attacks on XML Databases	Proc. 10th Asia Pacific Web Conference, LNCS 4976	2008 年 4 月