Vol.2016-CSEC-74 No.37
Vol.2016-SPT-19 No.37
2016/7/15

秘密計算フィッシャー正確検定 (1) ～標本数が少ない場合

千田 浩司[1,a)]　長谷川 聡[1]　濱田 浩気[1]　荻島 創一[2,3]　三澤 計治[2,3]　長﨑正朗[2,3]

概要：ゲノム解析等でしばしば用いられるフィッシャー正確検定を秘密計算によって効果的に実現する手法を提案する．フィッシャー正確検定は階乗計算を繰り返し行う必要があり，従来の汎用的な秘密計算では実用が難しい．提案手法は，標本数を固定し事前に全パターンの分割表についてフィッシャー正確検定を行い，検定結果を決定木表現する．これにより本計算を決定木の秘密計算に転換できる．実際に標本数と有意水準をパラメータとして決定木を作成し，標本数が 1,000 程度であれば，一般的な計算資源でも十分実現可能であることを確認した．

キーワード：秘密計算，フィッシャー正確検定，プライバシ保護データマイニング，決定木

Privacy Preserving Fisher's Exact Test(1) - For Small Samples

Koji Chida[1,a)]　Satoshi Hasegawa[1]　Koki Hamada[1]　Soichi Ogishima[2,3]　Kazuharu Misawa[2,3]
Masao Nagasaki[2,3]

Keywords: secure computation, Fisher's exact test, privacy preserving data mining, decision tree

1. はじめに

ICT(Information and Communication Technology) の発達に伴い，多種多様な大量の情報が容易に収集できるようになり，情報の利活用による新たな価値創造への期待が高まっている．そして我々のパーソナルデータ（個人に関するデータ），例えば買物履歴，インターネットアクセス履歴，位置・移動情報，バイタル情報等も事業者によって日々蓄積され，研究やサービス向上のための分析素材として利活用され始めている．しかしパーソナルデータを扱う際はプライバシの保護に十分配慮する必要がある．このような背景から，データ提供者のプライバシを保護しつつ，提供データをデータマイニング等に利活用可能とする**プライバシ保護データマイニング**（Privacy Preserving Data Mining: PPDM）の研究が活発に進められている [1], [2][*1]．

PPDM は Lindell, Pinkas[3] 及び Agrawal, Srikant[4] によって 2000 年に独立に提案された．Lindell, Pinkas は，二者がそれぞれ持つデータセットを暗号化して互いに明かさぬまま，決定木学習を行う手法を提案した[*2]．Agrawal, Srikant も決定木学習を取り上げているが，データセットをランダム化して提供することでデータ提供者のプライバシを保護している．ランダム化されたデータから，ベイズ推定等を用いて誤差の少ない決定木学習を行う手法を提案した．その後の研究で，様々なデータマイニング・統計解析を対象とした PPDM が提案されている [1], [2].

PPDM の研究に先駆けて，本来の入力データを演算実

[1] NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories, 3-9-11, Midori-cho, Musashino-shi, Tokyo 180–8585, Japan
[2] 東北大学東北メディカル・メガバンク機構
Tohoku Medical Megabank Organization, Tohoku University, 2-1, Seiryo-machi, Aoba-ku, Sendai 980-8573, Japan
[3] 東北大学大学院医学系研究科
Graduate School of Medicine, Tohoku, University, 2-1, Seiryo-machi, Aoba-ku, Sendai 980-8575, Japan
[a)] chida.koji@lab.ntt.co.jp

[*1] PPDM はデータマイニングに限らず統計解析等を含むデータ分析全般を対象とすることが多い．
[*2] このような状況は，各事業者が顧客等の個人のデータセットを保持しており，それらを結合してデータ分析する例が挙げられる．また個人のプライバシだけでなく，事業者のプライバシ (機密情報) の保護にも有効といえる．

Vol.2016-CSEC-74 No.37
Vol.2016-SPT-19 No.37
2016/7/15

行者に明かさず計算させることができる**秘密計算** (Secure Computation)[*3]が古くから研究されている [5]．秘密計算は任意の関数について実行可能な手法の研究が進められてきたが，必ずしも処理効率が良くないため，関数を限定した特化型の手法も研究が行われるようになった．Lindell, Pinkas の提案手法もその一例といえる．

本研究では，我々が知る限りこれまで提案されていない，**フィッシャー正確検定** [6] に特化した秘密計算 (以降これを秘密計算フィッシャー正確検定と呼ぶ) に着目する．フィッシャー正確検定はゲノム解析等でしばしば用いられる重要な検定手法だが，階乗計算を繰り返し行う必要があり，従来の汎用的な秘密計算では実用が難しい．提案手法は，標本数を固定して事前に全パターンの分割表についてフィッシャー正確検定を行い，検定結果を決定木表現しておくことを特徴とする．これにより本計算を決定木の秘密計算に転換できる．実際に標本数と有意水準をパラメータとして決定木を作成し，標本数が 1,000 程度であれば，一般的な計算資源でも十分実現可能であることを確認した．

以降，先ず 2.1, 2.2 節でそれぞれフィッシャー正確検定および秘密計算について概説する．次に 2.3 節で秘密計算フィッシャー正確検定の利用形態について考察する．そして秘密計算フィッシャー正確検定を実現する提案手法について，3.1 節で基本アイデアを述べた後，3.2 節で具体的なプロトコルを提案する．4 節では提案プロトコルの評価として，安全性評価 (4.1 節) および実現可能性評価 (4.2 節) を行う．最後に 5 節で課題とともに本稿をまとめる．

2. 準備

2.1 フィッシャー正確検定

表 1 2×2 分割表

	Yes	No	計
カテゴリー A	a	b	X
カテゴリー B	c	d	$N-X$
計	Y	$N-Y$	N

フィッシャー正確検定は，2 つ以上のカテゴリーの独立性の検定を行う手法である．表 1 のような 2×2 の分割表 (度数表) を考えよう．ここで $X = a+b$, $Y = a+c$, $N = a+b+c+d$. すると表 1 の分割表が得られる確率 $P(a)$ は以下の関数 $P(z)$ から得られる：

$$P(z) = \frac{X!Y!(N-X)!(N-Y)!}{N!z!(X-z)!(Y-z)!(N-X-Y+z)!} \quad (1)$$

フィッシャー正確検定 (両側検定) の有意確率 P は，$P(a)$ よりも極端な分割表の確率も考慮し，

$$P = P(a) + \sum_{P(i)<P(a)} P(i) \quad (2)$$

で与えられる．ただし

$$\max(0, X+Y-N) \le i \le \min(X, Y).$$

最終的に有意水準 α との大小関係により統計的な差の有無を得る．具体的には，$P < \alpha$ であれば帰無仮説が棄却され，「カテゴリー A とカテゴリー B には統計的な差が無いとは言えない」と帰結される．α の値は通常 0.05 や 0.01 等が用いられるが，ゲノム解析のように多重に検定を行う場合は，Bonferroni 補正法 [7] 等によって α の値が非常に小さくなることもある．

式 (1) の計算を効率化する工夫として，対数をとる方法がよく知られる．非負整数 n について $\log(n!) = \sum_{i=1}^{n} \log i$ が成り立つことから，$\ell_n := \sum_{i=1}^{n} \log i$ として式 (1) の対数

$$\log P(n) = \ell_X + \ell_Y + \ell_{N-X} + \ell_{N-Y} - (\ell_N + \ell_n + \ell_{X-n} + \ell_{Y-n} + \ell_{N-X-Y-n}) \quad (3)$$

を計算する．特に標本数 N を固定して $\ell_1, \ell_2, \ldots, \ell_N$ を事前計算しておけば，式 (3) を簡便に計算できる．そして $\log P(n)$ から $P(n)$ を求めれば，式 (2) を計算できる．

2.2 秘密計算

秘密計算は一般に，意中の関数の入力データを秘匿化して提供する主体と，その入力データを復元せず当該関数の演算を実行する主体の少なくとも二者が存在する．すなわちデータ提供者と演算実行者が異なっており，提供データを他者に知られたくないが，関数演算は他者の手を借りたいという動機が前提となる．秘匿化の手法として暗号化や秘密分散 [8] 等が知られる．Yao は，組合せ論理回路を実行可能な状態のまま秘匿化する主体と，秘匿化された組合せ論理回路 (Garbled Circuit) を実行する主体からなる秘密計算を提案した [9]．組合せ論理回路によりある程度汎用的な演算が可能となる．なお秘密計算において複数の主体が協調して計算する場合は**セキュアマルチパーティ計算** (Secure Multi-Party Computation: MPC) とも呼ばれる．複数の主体がそれぞれ知られたくないデータを持ち，それらを結合して計算する場合に有効といえる．

加減算や乗算の秘密計算についても多くの研究結果が知られている．Ben-Or ら [10] と Chaum ら [11] は秘密分散を用いて加減算と乗算ができる MPC を提案した．Cramer ら [12] は加法準同型暗号を用いて同様の MPC を実現した．加法準同型暗号は，暗号化したまま加算 (および減算) ができる暗号方式である．言い換えれば，$E(\cdot)$ を加法準同型暗号の暗号化関数としたとき，数値 a, b の暗号文 $E(a), E(b)$ から，それらを復号することなく $a+b$ の暗号文 $E(a+b)$ を求めることができる．加減算と乗算の組合せで回路素子の演算を構成し，前記の MPC を組合せ論理回路による汎用性の高い秘密計算とすることもできる．

更には等号判定や大小比較の秘密計算の研究も見られる．Schoenmakers, Tuyls[13] は，加法準同型暗号を用い

[*3] セキュア計算や秘匿計算とも呼ばれる．

て conditional gate と呼ばれる演算を MPC で構成するとともに，等号判定や大小比較への応用を示した．Damgård ら [14] と Nishide, Ohta[15] は秘密分散を用いて等号判定や大小比較の MPC を構成した．これらは何れも秘密計算の部品として利用することができる．すなわち，加減算，乗算，等号判定，大小比較の結果を秘匿化したまま次の計算の入力にできる．

MPC とは異なるアプローチとして，Gentry は完全準同型暗号を構成した [16]．完全準同型暗号は，加法準同型暗号の性質と乗法準同型暗号 (暗号化したまま乗算ができる暗号方式) の性質を合わせ持った暗号方式である．単一主体で計算可能なことから通信は不要だが，データサイズが大きい，計算が複雑などの欠点もあり，様々な改良研究が進められている [17]．

2.3 秘密計算フィッシャー正確検定の利用形態

PPDM はデータ提供者，演算実行者，演算結果取得者，そして (データ提供者が演算実行者に対して) 秘匿すべき情報によって様々な利用形態が考えられる．また，関数の種類や秘匿方法によってアプローチが異なる場合もある．例えば Agrawal, Srikant[4] は，データ提供者は複数人のパーソナルデータセットを所持している単一組織，演算実行者および演算結果取得者は分析者を想定し (図 1)，秘匿すべき情報はパーソナルデータセットから特定または推定される機微な情報であろう．Lindell, Pinkas[3] は，データ提供者は複数の組織，演算実行者および演算結果取得者も当該複数の組織を想定し (図 2)，秘匿すべき情報は各データ提供者が所持しているデータそのものであり，パーソナルデータの保護や各組織の機密情報保護の両方の目的が考えられる．その他，データ提供者および演算結果取得者は複数人のパーソナルデータセットを所持している単一組織，そして演算実行者は外部のクラウドという形態も挙げられる (図 3)．更にはパーソナルデータが逐次クラウドに集約される状況を想定し，データ提供者は複数の個人，演算実行者はクラウド，演算結果取得者は分析者 (図 4)，そしてクラウドや分析者に対して個々人のパーソナルデータを秘匿するという利用形態も考えられる．

それでは秘密計算フィッシャー正確検定の利用形態についてはどうだろうか．先ず図 4 の形態では，秘匿化されたパーソナルデータがクラウドに集約され，分析者からクラウドに分析要求があると，パーソナルデータを秘匿化したまま分割表を求め，検定を行う方法が考えられる．しかし分割表が個人のプライバシの観点で問題なければ，分割表を復元し，(秘密計算ではなく) 通常の方法で検定を行えばよい．ただし例えば図 2 の形態において，各組織が分割表を作成し，それらを足し合わせた分割表について検定を行う場合，分割表自体が組織の機密情報であれば，足し合わせた分割表も秘匿することが望ましい．その理由は，足し合わせた分割表を復元して各組織に返すと，自組織の分割表を差し引いて他組織の分割表を特定または推定できるという問題が生じ得るためである．このような観点から，本研究では分割表を復元することなく，秘密計算フィッシャー正確検定を実現することを目指している．

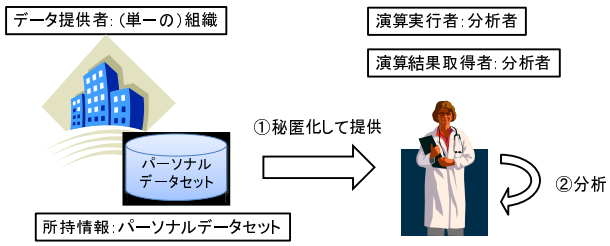

図 1 利用形態 1

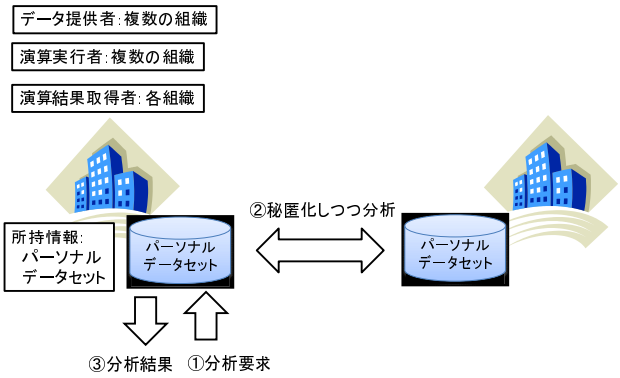

図 2 利用形態 2

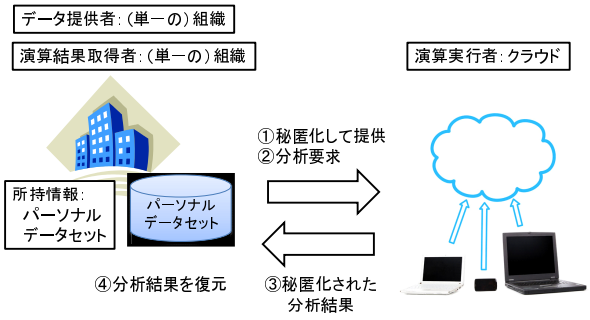

図 3 利用形態 3

3. 提案手法

3.1 基本アイデア

式 (1)(または式 (3)) および式 (2) を汎用的な秘密計算で処理することは現実的とは言い難い．そこで本研究では，事前にフィッシャー正確検定の有意確率が有意水準を下回る ($P < \alpha$ となる) 分割表パターンのリストを作成しておき，本計算では当該リストと実際の分割表を秘密計算によって照合（等号判定）するアプローチを着想した (以降，これをリスト作成方式と呼ぶ)．2×2 の分割表において標

Vol.2016-CSEC-74 No.37
Vol.2016-SPT-19 No.37
2016/7/15

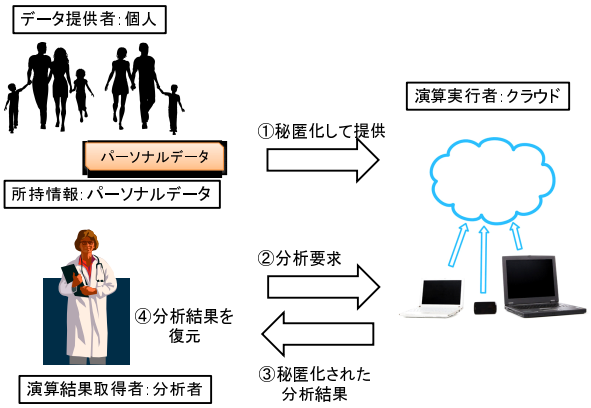

図 4　利用形態 4

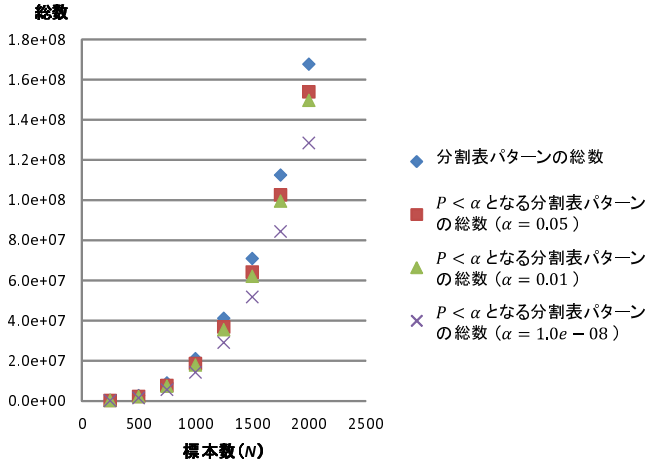

図 5　標本数・有意水準とフィッシャー正確検定の有意確率が有意水準を下回る分割表パターンの総数の関係

本数 N および有意水準 α を固定したとき，およそ N^3 通りの分割表パターンが存在するが，その中で $P < \alpha$ となる分割表パターンを記憶してリストに残す．一般に $O(N^3)$ 個の分割表の記憶と照合は標本数 N が大きければ困難だが，N がそれほど大きくなく，かつ $P < \alpha$ となる分割表パターンの総数が小さければ現実的なアプローチと考えられる．勿論，$P \geq \alpha$ となる分割表パターンの総数の方が小さければ，それらを記憶し照合してもよい．

リスト作成方式について，標本数 N および有意水準 α をパラメータとして，$P < \alpha$ となる分割表パターンの総数を求めた結果を図 5 に示す．なおフィッシャー正確検定の有意確率 P は，分割表の列の入れ替えや対角線で反転させても不変であるため [18]，これらを同一視している．

図 5 より，標本数が 1,000 と小さい場合でも，$\alpha = 10^{-8}$ のとき $P < \alpha$ となる分割表パターンの総数は 1.4×10^7 を超え[*4]，膨大な記憶量が必要となることが分かる．そこでリスト作成方式ではなく，$P < \alpha$ となる分割表パターンを条件式 (大小比較) で表現することについて検討した．条件

[*4] 有意水準が小さいほど，$P < \alpha$ となる分割表パターンの総数は少なくなる．

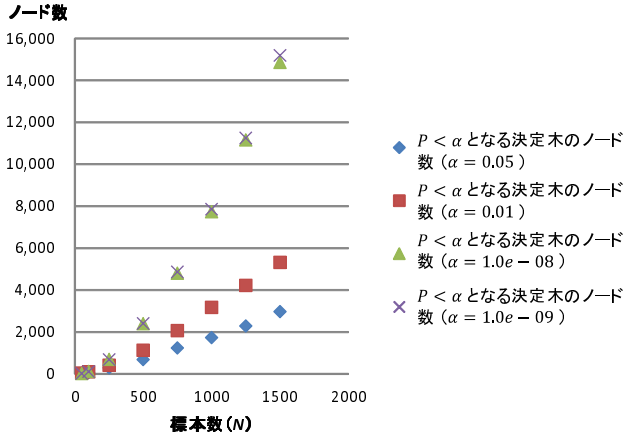

図 6　フィッシャー正確検定の決定木のノード数

式が少なく単純なほど，記憶量や計算量が少なく済むことが期待できる．標本数 N および有意水準 α をパラメータとして，決定木を用いて条件式を作成した (以降，これを決定木作成方式と呼ぶ)．決定木の作成は統計ソフトウェア R の rpart 関数を用い，分割基準は情報量とした．決定木の特徴量は以下とした．

(1) 分割表の度数： a, b, c, d

(2) 2 次単項式： $a^2, b^2, c^2, d^2, ab, ac, ad, bc, bd, cd$

(3) 1 次 2 項多項式： $a+b, a+c, a+d, b+c, b+d, c+d$

(4) χ^2 検定の式の一部: $(ad-bc)^2, (a+b)(a+c)(b+d)(c+d)$

上記によって得られた決定木のノード数を図 6 に示す．また，リスト作成方式における分割表パターンの総数と，決定木作成方式におけるノード数の比較を表 2 に示す．例えば標本数が 1,000，有意水準が 0.01 の場合，決定木のノード数は 3,165 とできるが，これはリスト作成方式における分割表パターンの総数の 1/5,000 以下であり，記憶量の大幅な削減が期待できる．同様に，本計算時の計算量も大幅な削減が期待できる．ただし決定木作成は実装上の課題があり，$N = 1,750$ としたとき，メモリ不足 (100GB 使用) で決定木の作成ができなかった．しかしこの実装上の課題を解決できれば，図 6 のノード数の増加傾向から，標本数が数千程度までであれば，一般的な計算資源でも決定木の記憶や本計算の実行は十分可能であると考えられる．

具体例として $N = 50, \alpha = 10^{-8}$ の決定木を図 7 に示す．ただし $V1 = a$, $V4 = d$, $V10 = ab$, $V12 = ad$, $V13 = bc$, $V15 = cd$, $V17 = a + c$, $V18 = a + d$, $V19 = b + c$, $V22 = (ad - bc)^2$, $V23 = (a+b)(a+c)(b+d)(c+d)$．図 7 の決定木は，TRUE の葉に到達すれば $P < \alpha$ となる．そして $N = 50, \alpha = 10^{-8}$ であれば 18 個のノード (条件式) で決定木を構成できることが分かる．

3.2　提案プロトコル

3.1 節で提案した，決定木作成による秘密計算フィッ

表 2 リスト作成方式における分割表パターンの総数と決定木作成方式におけるノード数の比較

標本数 N	250	500	750	1,000	1,250	1,500
分割表パターンの総数	341,376	2,667,126	8,930,376	21,084,251	41,081,876	70,876,376
リスト作成方式 : 分割表パターンの総数 ($\alpha = 0.05$)	261,528	2,226,028	7,725,555	18,624,152	36,797,921	64,136,493
リスト作成方式 : 分割表パターンの総数 ($\alpha = 0.01$)	238,718	2,095,166	7,363,392	17,878,453	35,494,439	62,078,034
リスト作成方式 : 分割表パターンの総数 ($\alpha = 10^{-8}$)	135,697	1,468,120	5,590,895	14,191,351	29,001,451	51,778,316
決定木作成方式 : 決定木のノード数 ($\alpha = 0.05$)	287	678	1,232	1,733	2,281	2,971
決定木作成方式 : 決定木のノード数 ($\alpha = 0.01$)	410	1,126	2,064	3,165	4,221	5,317
決定木作成方式 : 決定木のノード数 ($\alpha = 10^{-8}$)	689	2,395	4,801	7,734	11,160	14,853

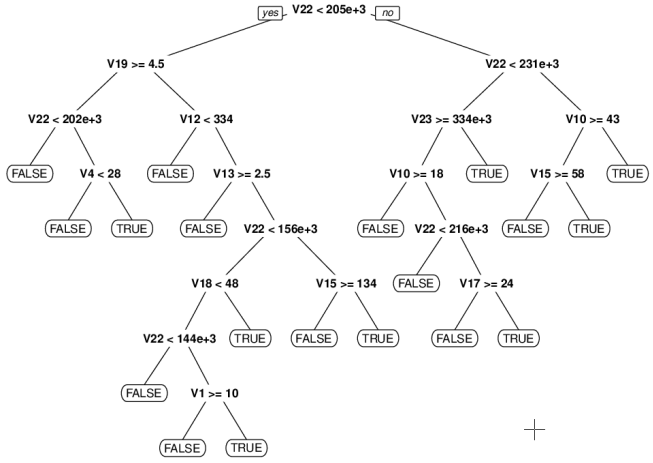

図 7 フィッシャー正確検定の決定木の例 ($N = 50, \alpha = 10^{-8}$)

シャー正確検定のプロトコルを以下に示す．なお標本数 N および有意水準 α は固定とし，決定木は事前に作成しておくものとする[*5]．決定木について，i 段目の j 番目のノードを v_{ij} と表記する．ノード v_{ij} は，条件式が真 (yes) であれば 1 を，偽 (no) であれば 0 を出力するものとする．v_{ij} の出力を $b_{ij} \in \{0,1\}$ とする．k 番目の True の葉に到達するパスを P_k と表記し，P_k に含まれるノードの集合を $V_k \subseteq \{v_{ij}\}$ とする．P_k に含まれる枝のうち，v_{ij} からの枝を w_{ijk} と表記し，v_{ij} の条件式が真の枝を 1，偽の枝を 0 とする．すなわち $w_{ijk} \in \{0,1\}$ となる．なお j と k の順序は適当に定めてよい．入力は分割表の度数 a, b, c, d を秘匿化した値とする (秘匿化関数を $E(\cdot)$，その逆関数を $D(\cdot)$ と表記する)．すなわち 2.3 節で考察した利用形態において，(足し合わした) 分割表が秘匿化された値までは秘密計算によって得られているものとする．これは例えば単純に加算の秘密計算を用いればよい．フィッシャー正確検定の関数を Fisher$(\cdot,\cdot,\cdot,\cdot,\cdot)$ と表記し，出力は

$$\mathrm{Fisher}(\alpha, a, b, c, d) = \begin{cases} 1 & \text{if } P < \alpha \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

とする．

Input: $E(a), E(b), E(c), E(d), \{v_{ij}\}, (P_k, V_k)$.
Output: Fisher(α, a, b, c, d).

[*5] 標本数の上限と有意水準を事前に定め，それ以下の全ての標本数とその有意水準について，決定木を事前作成してもよい．

(1) 全てのノード v_{ij} の条件式の真偽を秘密計算で求め $E(b_{ij})$ を得る．
(2) 全てのパス P_k について以下を行う．
  (a) V_k に含まれる全ての v_{ij} について，

$$c_{ijk} = \begin{cases} E(b_{ij}) & \text{if } w_{ijk} = 1 \\ E(1 - b_{ij}) & \text{otherwise} \end{cases} \tag{5}$$

  を求める．
  (b) 以下を秘密計算によって求める．

$$d_k = \begin{cases} E(1) & \text{if } D(c_{ijk}) = 1 \text{ for all } c_{ijk} \\ E(0) & \text{otherwise} \end{cases} \tag{6}$$

(3) 以下を秘密計算によって求めて出力する．

$$\mathrm{Fisher}(\alpha, a, b, c, d) = \begin{cases} 0 & \text{if } D(d_k) = 0 \text{ for all } d_k \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

上記の提案プロトコルは，事前に作成された決定木について，何れかの TRUE の葉に到達したかどうかを判別している (ただし，どの葉に到達したかは，分割表の情報を与えるため，秘匿されるよう設計されている)．例えば入力の分割表が決定木の全ての条件式を満たし，そのパスの葉は TRUE，当該パスを P_1 としよう．このとき，V_1 に含まれる全ての v_{ij} の出力 b_{ij} およびその枝 w_{ij1} は 1 となる．したがって，式 (5) の全ての c_{ij1} は 1 を秘匿化した値となり，式 (6) も 1 を秘匿化した値となるため，最終出力の式 (7) は正しく 1 を返す．なお TRUE の葉に到達するパスにおいて条件式を満たさないノードがあった場合でも，式 (5) において条件式の真偽結果 b_{ij} を反転させているため，正しい出力を返す．逆に入力の分割表が FALSE に到達する場合は，全てのパスについて式 (6) が 0 を秘匿化した値となるため，最終出力の式 (7) は正しく 0 を返す．

4. 評価

4.1 安全性

3.2 節の提案プロトコルの安全性 (秘匿性) について考察する．秘匿すべき情報は分割表の度数 a, b, c, d であり，これらに関して提案プロトコルが最終出力 Fisher$(\alpha, a, b, c, d) \in \{0,1\}$ 以外の有意な情報を一切漏らさないことが目的となる．

手続き (1) では，条件式の計算とその真偽結果を秘密計

算によって求めており，手続き（1）の出力 $E(b_{ij})$ は秘匿化されていることが分かる．条件式の計算は 3.1 節で与えた決定木の特徴量の形で得られるため，加減乗算の秘密計算によって実現できる．真偽結果は例えば 2.2 節で紹介した大小比較の秘密計算を用いて実現できるため，加減乗算および大小比較の秘密計算を用いれば，手続き (1) は秘匿化された値以外の一切の追加情報を与えない．

手続き (2)(a) では，出力 c_{ijk} は秘匿化されている．また式 (5) の $E(1-b_{ij})$ は，$E(b_{ij})$ を入力として加減算の秘密計算を用いて計算できるため，手続き (2)(a) は秘匿化された値以外の一切の追加情報を与えない．

手続き (2)(b) では，出力 d_k は秘匿化されている．d_k は c_{ijk} を入力として論理積の秘密計算を用いて計算できる．論理積は 2.2 節で述べたように加減乗算で構成できるため，結局，手続き (2)(b) は秘匿化された値以外の一切の追加情報を与えない．

最後に手続き (3) では，d_k を入力として論理和の秘密計算を用いて最終出力 $\mathrm{Fisher}(\alpha, a, b, c, d)$ を計算できる．論理和も 2.2 節で述べたように加減乗算で構成できる．

以上から，加減乗算および大小比較の秘密計算を用いて，a, b, c, d に関して最終出力 $\mathrm{Fisher}(\alpha, a, b, c, d)$ 以外の有意な情報を一切漏らさず計算できる．

なお通常は決定木にデータを入力して真偽を判別する際，何れか一つのパスのみ辿れば判別できる．しかし入力データを秘匿化しても，どのパスを辿ったかが分かると，入力データに関する情報を与えてしまうため，提案プロトコルは全てのパスを計算することで入力データの秘匿性を損ねないよう設計されている．

4.2 実現可能性

提案プロトコルの実現可能性を評価するために，提案プロトコルの実行に必要な記憶量および計算時間を見積もる．ただし見積もりの項目は，記憶量および計算時間それぞれで支配的な部分と考えられる，決定木のサイズ，および手続き (1) の大小比較のみとする．

決定木はノード，枝，葉の集合，およびその構成情報で表現できる．提案プロトコルでは，入力の決定木は秘匿化する必要がない．標本数 N と有意水準 α をパラメータとして，ノードの総数を M としたとき，M は表 2 のようになる．構成情報は，ノードのラベル情報 ($\log_2 M$ 程度の大きさで記述可能)，およびノード間の関係 (次のノードのラベル情報と枝の情報 (1 ビット)) とする．葉の情報は最終段の枝の情報に含まれるので省略できる．以上より，決定木のサイズは $M(\log_2 M + 1)$ ビット程度で表現できることが分かる．例えば $N = 1,000$, $\alpha = 0.01$ のとき，表 2 では $M = 3,165$ となり，決定木のサイズはおよそ 4.7KB と見積もることができる．これは $N = 1,000$ 以下の全ての決定木を記憶しても，十分現実的なサイズといえる．

次に手続き (1) の大小比較に要する計算時間を見積もる．既存の秘密計算実装報告の中でも特に高性能と思われる，菊池ら [19] による秘密分散を用いた秘密計算システムで，20 ビット入力の大小比較の計算時間が測定されている．実験は有線 LAN(1Gbps および 10Gbps) で相互接続された 3 台のコンピュータ (CPU: Core i7 4930K, RAM: 32GB, OS: Ubuntu12.04) で行われ，10^7 回の大小比較に 1Gbps LAN で 7,621 ミリ秒，10Gbps LAN では 3,945 ミリ秒となっている．菊池らが実装した大小比較の秘密計算は，理論上，入力サイズに比例した計算量となる (ただし入力サイズが小さい場合は他のオーバーヘッドの影響を受けやすい)．手続き (1) の大小比較の回数は決定木のノード数 M であり，大小比較の入力長は 3.1 節の決定木の特徴量より高々 $\lfloor \log_2 N^4 \rfloor$ となるため，計算時間 (ミリ秒) は 1Gbps LAN で

$$M \times 7,621 \times 10^{-7} \times \lfloor \log_2 N^4 \rfloor \times 20^{-1} \quad (8)$$

と見積もることができる．例えば前記 $N = 1,000$, $\alpha = 0.01$, $M = 3,165$ を代入すれば，式 (8)$\approx$ 4.8(ミリ秒) となり，十分現実的な数値といえる．

5. まとめ

ゲノム解析等で重要な手法であるフィッシャー正確検定について，入力値を秘匿化したまま検定結果を効率良く求める「秘密計算フィッシャー正確検定」を提案した．フィッシャー正確検定は階乗計算を繰り返し行う必要があり，従来の汎用的な秘密計算では実用が難しいが，提案手法は検定結果の決定木を事前に作成することで，標本数が 1,000 程度であれば一般的な計算資源でも十分実現できることを確認した．ただし標本数が大きいときは実装上の問題により決定木の作成が難しい．また標本数も秘匿化する場合や，標本に欠損があり，それを秘匿化する場合には提案手法を適用できない．これらの課題の対策については，別の文献 [20] にて紹介することとしたい．

謝辞 本稿で紹介した決定木の作成は，東北大学東北メディカル・メガバンク機構のスーパーコンピュータを利用して実行しました．

参考文献

[1] Vaidya, J., Clifton, C.W., and Zhu, Y.M.: Privacy preserving data mining, Springer-Verlag (2005)
[2] Aggarwal, C. and Yu, P.: Privacy-preserving data mining: Models and algorithms, Springer-Verlag (2008)
[3] Lindell, Y. and Pinkas, B.: Privacy preserving data mining, CRYPTO 2000, LNCS 1880, Springer-Verlag, 36-54 (2000)
[4] Agrawal, R. and Srikant, R.: Privacy-preserving data Mining, Proc. of the ACM SIGMOD Conference on Managament of Data, 439-450 (2000)

[5] Yao, A. C.: Protocols for secure computations, Proc. of FOCS '82, 160-164 (1982)
[6] Fisher, R.A.: On the interpretation of χ^2 from contingency tables, and the calculation of P, Journal of the Royal Statistical Society 85(1), 87-94 (1922)
[7] 瀬々潤，濱田道昭: 生命情報処理における機械学習 多重検定と推定量設計, 講談社 (2014)
[8] Shamir, A.: How to share a secret, Communications of the ACM 22(22), 612-613 (1979)
[9] Yao, A. C.: How to generate and exchange secrets, Proc. of FOCS '86, 162-167 (1986)
[10] Ben-Or, M., Goldwasser, S., and Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation, Proc. of STOC '88, 1-10 (1988)
[11] Chaum, D., Crepeau, C., and Damgård, I.: Multiparty unconditionally secure protocols, Proc. of STOC '88, 11-19 (1988)
[12] Cramer, R., Damgård, I., and Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption, EUROCRYPT 2001, LNCS 2045, Springer-Verlag, 280-300 (2001)
[13] Schoenmakers, B. and Tuyls, P.: Practical two-party computation based on the conditional gate, ASIACRYPT 2004, LNCS 3329, Springer-Verlag, 119-136 (2004)
[14] Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., and Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation, TCC2006, LNCS 3876, Springer-Verlag, 285-304 (2006).
[15] Nishide, T. and Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol, PKC 2007, LNCS 4450, Springer-Verlag, 343-360 (2007)
[16] Gentry, C.: Fully homomorphic encryption using ideal lattices, Proc. of STOC 2009, 169-178 (2009)
[17] Soral, A.: Achieving fully homomorphic encryption in security - A survey, SSRG-IJCSE 3(2), 22-27 (2016)
[18] Terada, A., Okada-Hatakeyama, M., Tsuda, K., and Sese, J.: Statistical significance of combinatorial regulations, Proc. Natl. Acad. Sci. 110(32), 12996-13001 (2013)
[19] 菊池亮，五十嵐大，濱田浩気，千田浩司: 改ざん検知機能付きの実用的な秘密計算システム MEVAL2, 第 32 回暗号と情報セキュリティシンポジウム (SCIS2015) 予稿集 (2015)
[20] 濱田浩気，長谷川聡，千田浩司，荻島創一，三澤計治，長崎正朗: 秘密計算フィッシャー正確検定 (2) ～標本数が多い場合, 第 74 回コンピュータセキュリティ研究会 (CSEC) 予稿集 (2016)