修 士 論 文

閾値を設けた秘匿マッチングプロトコルに関する研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

権田 陽彦

平成 29 年 3 月

修士論文

閾値を設けた秘匿マッチングプロトコルに関する研究

1510018　　権田 陽彦

主指導教員　金子 峰雄
審査委員主査　金子 峰雄
審査委員　宮地 充子
審査委員　上原 隆平

北陸先端科学技術大学院大学

情報科学研究科

平成 29 年 2 月

概要

スマートフォンやタブレットなどのモバイル端末の普及による，モバイルソーシャルネットワークの急速的拡大に伴い，そこでの友達のあり方に関心が集まっている．モバイルソーシャルネットワークのサービスには，物理的な距離を超えた交流によって，偶発的な発見または新たな価値観を創造することが期待されている．しかし，見ず知らずの人とネット上で直接会って友達を探すことは気が進まないし，自分のプライバシー情報を秘匿せずに使って友達を探すことは危険である．また，運営サーバを狙ったサイバー攻撃によるプライバシー情報の漏えいも無視することはできない．その一方で，より精度の高いマッチングを行おうとすれば，よりセンシティブなプライバシー情報を使わざるを得ない．これらの懸念を払拭すべく，プライバシー情報を使って安全にマッチングを行う方法として，暗号化した状態でそれらの共通項目や類似度を秘匿計算するという秘匿マッチングプロトコルが提案されている．

これまでに，プロフィールを秘匿したまま共通項目を求めたり類似度を計算し，その上で友達としての条件を満たしているか否かを判定するという秘匿マッチングが多く提案されている．その中でも閾値を用いたマッチングプロトコルには，共通項目数や類似度を秘密計算した後，それと閾値との大小比較の結果を相手に通知するため，共通項目数や比較結果などのマッチングの結果を知ることができるという問題がある．また，両方の条件を同時に判定することで上記の問題を解決した既存研究には，相手の期待する共通項目数と一致しなければマッチしないため，設定する条件が厳しすぎるという問題がある．少なすぎるのはもちろんのこと，期待を超えて多く項目が共通する場合もミスマッチと判定してしまう．

本研究では，プロフィールや興味だけでなく共通項目数などのマッチングの結果につながる部分情報を秘匿してプライバシーを守り，閾値を超えたか否かなどのマッチングの結果を追跡不可能にしたマッチングプロトコルを提案する．また，本研究では，厳しすぎるマッチの条件を閾値を設けることによって緩和し，なおかつ両方の条件を同時に判定することを実現する．このプロトコルは，閾値を超えたか否かを追跡することができない．さらに，マッチングの機能性を充実させるために，マッチしたときだけ共通項目を求めることができるようにプロトコルを設計する．ミスマッチの場合，それ以外の情報が漏れるのは望ましくはないが，マッチの場合は共通項目がわかる方が，共通点は何なのかを心配することなく円滑に関係を構築することができる．

本研究が提案する秘匿マッチングプロトコルは，ユーザやサーバを semi-honest モデルと仮定し，プロトコルの途中でプライバシー情報やマッチングの結果を推測できないように設計されている．なお，ユーザはプロフィールや興味などのプライバシー情報を自己管理するため，運営サーバが攻撃を受けても秘密鍵やプライバシー情報を漏えいすることは決してない．

目次

第1章　はじめに

この章では，本研究の背景と研究目的，そして本論文の構成について述べる．

1.1　背景

VentureBeat などの技術系メディアが，次のトレンドになると予想しているサービスのひとつにソーシャルディスカバリーがある．ソーシャルディスカバリーとは，SNS 上でプロフィールや興味などのプライバシー情報を用いて友達を探して交流するサービスである．物理的距離を超越したグループをつくり，同じ趣味の人や注目している人から得た情報を通じて，偶発的な発見または新たな価値観を創造することが期待されている．

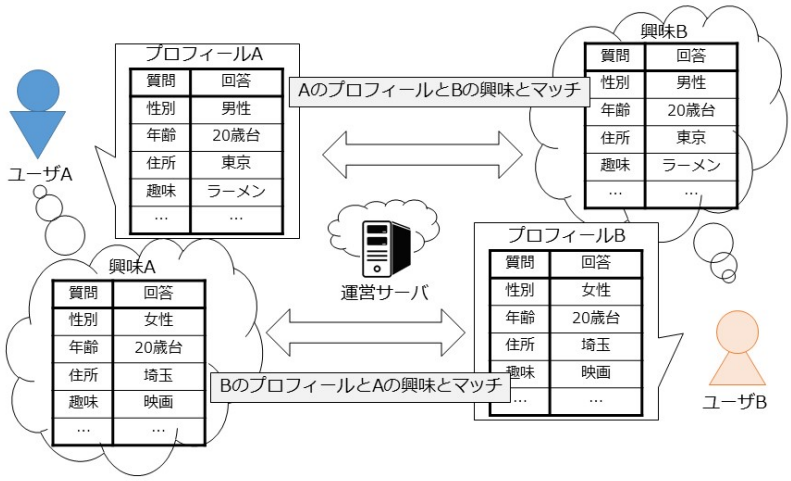

図 1.1: ソーシャルディスカバリーのイメージ図

図 1.1 は，ソーシャルディスカバリーのマッチングのイメージ図である．ユーザ A とユーザ B は，それぞれ自分のプロフィールと興味をサービスに登録する．そして，運営サーバが現在サービスに参加している 2 人を無作為に選びマッチングを行う．興味とは，探している理想の相手のプロフィールである．図 1.1 の場合，男性である A は，20 歳台の埼玉に住み趣味が映画の女性を探している．一方，女性である B は，趣味がラーメン

で 20 歳台の東京に住む男性を探している．そして，A と B はこのサービスのマッチングによってマッチすることで，物理的な距離を超えた交流を果たすのである．

すでに，ソーシャルディスカバリーはいくつか存在している．例えば，SKOUT [1]，Tagged [2]，hi5 [3]，Badoo [4] は，友達を探すために写真とプロフィールを見て Like/No を選択し，互いに Like の場合をマッチと判定し，チャットができるというサービスである．このような選別ゲームは Meet Me と呼ばれている．プロフィールは既婚未婚，性の嗜好，民族性，宗教などであり，公開することもできればすべて非公開にすることもできる．また，POF [5] は 200 問以上の質問に対して 4 択で答えてプロフィールをつくり，それを用いてマッチングを行うサービスである．それらの情報をどのように扱っているのかなどの詳細については公表されていない．

どのサービスにも共通することは，プロフィールを運営サーバにアップしているということである．それらはプライバシー情報として扱う必要がない情報なのかもしれないが，運営サーバを狙ったサイバー攻撃による情報漏えいを防ぐためにも，またセンシティブな情報をマッチングで使うためにも，プライバシー情報は自分で管理し，それらを暗号などによって秘匿して行われるべきである．これを実現するのが秘匿マッチングプロトコルである．

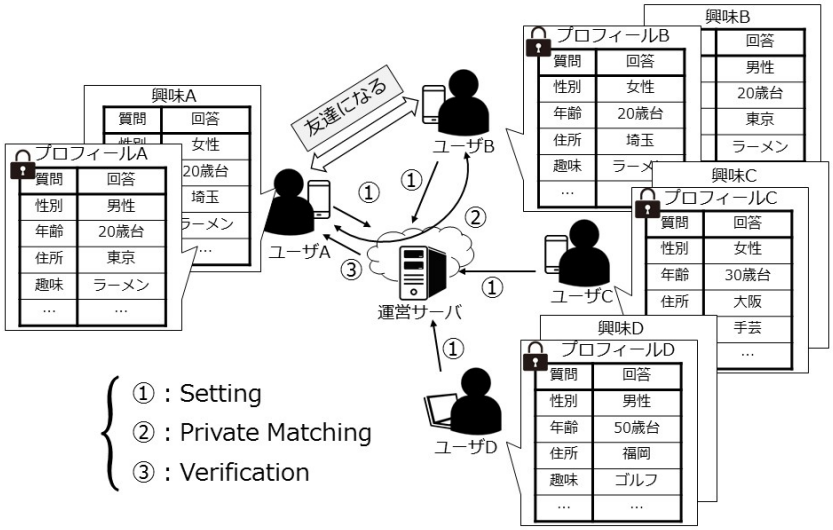

図 1.2: 秘匿マッチングのイメージ図

秘匿マッチングプロトコルは，プロフィールなどのプライバシー情報を秘匿したまま共通項目や類似度を計算し友達の判定を行うプロトコルで，モバイルソーシャルネットワーク上でプライバシー情報を漏えいすることなく使用する技術として注目されている．

図 1.2 は，秘匿マッチングのイメージ図である．まず，サービスに参加するユーザは，自分のプロフィールと興味を生成し，それらを自分で管理する．そして，秘匿マッチングのためにプロフィールを暗号化して運営サーバにアップロードする．基本的に，プロ

フィールや興味などのプライバシー情報は自分で管理するので，運営サーバがユーザのプライバシー情報を漏えいすることは決してない．そして，無作為に選ばれた 2 人のユーザ同士が，プロフィールと興味の共通項目を秘匿計算する．最後に，互いに設定した条件を満足するか否かをサーバが検証することによって，友達の判定を行い通知する．互いに設定する条件とは，例えば共通項目数の一致であったり，閾値であったり，相手は男性に限るなどのように項目の決め打ちなどを想定する．

しかし，これまでに提案された既存の秘匿マッチングは実用面でいくつかの問題がある．多くの研究では，片方の条件を満たしていればマッチするプロトコルであるため，片思いの場合をマッチと判定したり，どちらが片思いなのかを追跡することができる．どちらが片思いなのかということが追跡可能であれば，プロフィールや興味の偽造などの不正行為を生む危険性がある．この問題を [31] は，互いに設定する条件を同時に満たしたときにマッチと判定するプロトコルを提案して片思いの場合をミスマッチと判定するマッチングを実現した．しかし，それは期待する共通項目数との一致しかマッチと判定しないという条件であるため，友達の条件としては厳しすぎる．少なすぎるのはもちろんのこと，期待を超えて多く項目が共通する場合もミスマッチと判定してしまう．

1.2 研究目的

本研究では，プロフィールの共通項目数の下限となる閾値を互いに選択でき，且つその閾値を超えたかどうかの追跡不可能性を満たす二者間秘匿マッチングプロトコルを提案する．このプロトコルは，加法準同型暗号でプロフィールの各項目を暗号化し，共通項目数と閾値との大小を秘匿した状態で比較し，閾値以上を両者が満たすならばマッチを，そうでなければミスマッチを出力する．

これまでに提案されている閾値を用いたプロトコルには，共通項目数を秘密計算した後，それと閾値との大小比較結果を相手に通知するため，共通項目数や通知内容を知ることができるという問題がある．そこで，共通項目数を秘匿してプライバシーを守り，比較結果を追跡不可能にする必要がある．また，初めて追跡不可能性に配慮したプロトコル [31] は，共通項目数と事前に選択する理想の共通項目数が一致しなければミスマッチと判定されるという問題がある．さらに，[12] が指摘するようにマッチまたはミスマッチのみを出力するマッチングではサービスとして物足りない．ミスマッチの場合，それ以外の情報が漏れるのは望ましくはないが，マッチの場合は共通項目がわかる方が，共通点は何なのかを心配することなく関係を構築することができる．

そこで，本研究では上記の既存研究の 2 つの問題を解決し，さらにマッチした場合に限り共通項目を知ることができる秘匿マッチングプロトコルを提案する．ただし，ユーザや運営サーバは semi-honest モデルとし，ユーザはプロフィールなどのプライバシー情報を自己管理すると仮定する．これにより，運営サーバが攻撃を受けても秘密鍵やユーザのプライバシー情報を漏えいすることは決してない．

1.3　本論文の構成

次章以降の構成について述べる．2 章では，準備として本研究で扱うプライバシー情報，システムモデル，プロトコルへの要求事項，攻撃者モデル，プロトコルで使う暗号方式である Paillier 暗号について説明する．3 章では秘匿マッチングに関する関連研究について述べる．4 章では，本研究の提案する秘匿マッチングとそれを構成する 3 つのプロトコルについて説明する． 5 章では，各プロトコルの正当性，秘匿性，効率性の評価に加え，実装評価について述べる．6 章では，考察として既存研究との比較を述べる．7 章では，本研究のまとめを示す．

第2章　準備

この章では，まず本研究で扱うプライバシー情報を明らかにし，秘匿マッチングプロトコルで取り扱いに留意すべき情報を決定する．次に，システムモデルでは，マッチングの全体像を明らかにする．そして，マッチングプロトコルへの要求事項を述べ，さらに攻撃者モデルについて議論する．最後に，Paillier 暗号について述べる．

2.1　本研究のプライバシー情報

本研究におけるプライバシー情報とは，プロフィールと興味，閾値，自分のプロフィールと相手の興味の共通項目，その項目数の他，秘密鍵やマッチングの結果も含める．また，相手の興味に対する自分のプロフィールの評価，すなわち相手の設定する閾値と自分のプロフィールとの共通項目数との大小関係もプライバシー情報として取り扱う．その逆である自分の興味に対する相手のプロフィールの評価もプライバシー情報として扱う．

2.1.1　プロフィールと興味

ユーザは，プロフィール $\boldsymbol{P}$ と興味 $\boldsymbol{I}$ をつくる．$\boldsymbol{P}$ は，n 個の質問 Q に対する回答 $p_1, p_2, \cdots, p_n$ を項目とする．

$$\boldsymbol{P} := [p_1, p_2, \cdots, p_n]. \tag{2.1}$$

また，$\boldsymbol{I}$ は，理想の相手に期待する回答 $i_1, i_2, \cdots, i_n$ を項目とする．

$$\boldsymbol{I} := [i_1, i_2, \cdots, i_n]. \tag{2.2}$$

2.1.2　プロフィールと興味の共通項目数

ユーザ A, B は，それぞれプロフィールと興味のペア $(\boldsymbol{P}_a, \boldsymbol{I}_a), (\boldsymbol{P}_b, \boldsymbol{I}_b)$ を生成する．プロフィールと興味の共通項目とは $\boldsymbol{P}_a \cap \boldsymbol{I}_b, \boldsymbol{P}_b \cap \boldsymbol{I}_a$ であり，共通項目数とは $|\boldsymbol{P}_a \cap \boldsymbol{I}_b|, |\boldsymbol{P}_b \cap \boldsymbol{I}_a|$ である．本研究では，この共通項目数を直接マッチングに使用することはしない．なぜならば，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| = 0$ のような場合，$|\boldsymbol{P}_b \cap \boldsymbol{I}_a|$ を秘匿計算するまでもなく，明らかにミスマッチであると断定することができるためである．したがって，[31] のように $|\boldsymbol{P}_a \cap \boldsymbol{I}_b| + \Delta_b$ のように非零にする．

2.1.3　プロフィールと興味の評価

A, B は，それぞれ閾値 τ_a, τ_b を設定する．B の興味に対する A のプロフィールの評価とは，

$$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| \gtreqless \tau_b. \tag{2.3}$$

一方，A の興味に対する B のプロフィールの評価とは，

$$|\boldsymbol{P}_b \cap \boldsymbol{I}_a| \gtreqless \tau_a. \tag{2.4}$$

前述の通り，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ のような共通項目数を直接的に評価で用いないので，

$$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| + \Delta_b \gtreqless \tau_b + \Delta_b, \tag{2.5}$$
$$|\boldsymbol{P}_b \cap \boldsymbol{I}_a| + \Delta_a \gtreqless \tau_a + \Delta_a \tag{2.6}$$

を評価する．Δ_a, Δ_b の詳細については次節で述べる．マッチングでは，(2.3),(2.4) を (2.5),(2.6) のように評価することでマッチまたはミスマッチを判定する．

$$\begin{cases} |\boldsymbol{P}_a \cap \boldsymbol{I}_b| \geq \tau_b \wedge |\boldsymbol{P}_b \cap \boldsymbol{I}_a| \geq \tau_a & \Longrightarrow \ match, \\ |\boldsymbol{P}_a \cap \boldsymbol{I}_b| < \tau_b \vee |\boldsymbol{P}_b \cap \boldsymbol{I}_a| < \tau_a & \Longrightarrow \ mismatch. \end{cases} \tag{2.7}$$

(2.3),(2.4) の片方を知ればミスマッチが判明する場合があるので，(2.3),(2.4) を両ユーザとも追跡できないようにマッチングを設計する必要がある．ミスマッチの場合，次の 3 つの場合が考えられる．

- $|\boldsymbol{P}_b \cap \boldsymbol{I}_a| \geq \tau_a \wedge |\boldsymbol{P}_a \cap \boldsymbol{I}_b| < \tau_b$: A にとって B はマッチの対象であるが，B にとって A はそうではない場合
- $|\boldsymbol{P}_b \cap \boldsymbol{I}_a| < \tau_a \wedge |\boldsymbol{P}_a \cap \boldsymbol{I}_b| \geq \tau_b$: B にとって A はマッチの対象であるが，A にとって B はそうではない場合
- $|\boldsymbol{P}_b \cap \boldsymbol{I}_a| < \tau_a \wedge |\boldsymbol{P}_a \cap \boldsymbol{I}_b| < \tau_b$: A にとって B はマッチの対象でないし，B にとって A もそうである場合

本研究では，これらのどれに該当するのかを「結果の詳細」あるいは「ミスマッチの原因」と呼び，プライバシー情報として扱う．

2.2　システムモデル

図 2.1 のようなマッチングサービスにリアルタイムで参加しているユーザを運営サーバが適当に 2 人選び出すことで，二者間の秘匿マッチングは行われる．このマッチングをあらゆるユーザに対して行うことで，自分の興味に近いプロフィールを持ち，かつ自分のプロフィールに近い興味を持つユーザを探す．

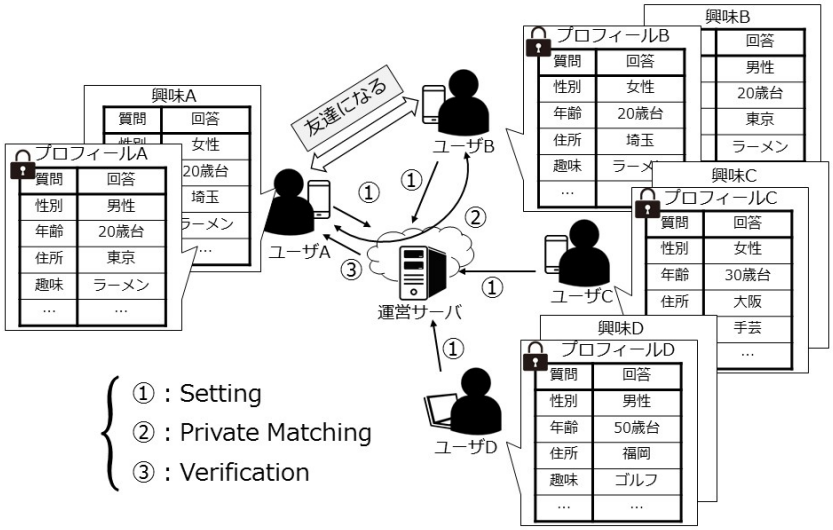

図 2.1: 秘匿マッチングプロトコルのシステムモデル

Setting

まず，ユーザはプロフィールと興味を生成する．次に，プロフィールを暗号化し運営サーバ（以下，サーバと呼ぶ）にアップロードする．プロフィールや興味，秘密鍵はユーザが管理する．

Private Matching

サーバが A, B を選び，A の暗号化プロフィールを B へ，B の暗号化プロフィールを A に送信することで，マッチングは始まる．Private Matching では，プロフィールと興味の共通項目数にセキュリティパラメタを加えた値を秘匿計算する [31]．例えば，A のプロフィールと B の興味の共通項目数を求める場合，B が所持しているのは A の暗号化プロフィールなので，最終的には復号しなければ正しく共通項目数は求められない．そこで，B は自分の興味をブラインドし再現不可能にした上で A に渡し，A が共通項目を計算する．当然，興味のブラインドと同様に暗号化プロフィールをブラインドするため，復号すれば A は正しく $|\boldsymbol{P}_a \cap \boldsymbol{I}_b| + \Delta_b$ を計算することができる．ブラインドの方法については次章で詳しく説明する．

Δ_b とはブラインドによって現れるセキュリティパラメタであり，A は知らないが B は知っている．したがって，A は $|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ を知ることができないので，この時点でマッチ

ングの結果を推測することはできない．一方，B は相手としてふさわしいと判断できる共通項目数を閾値 τ_b とし，それに Δ_b を加えて同じようにブラインドする．

$$\hat{v}_b := |\boldsymbol{P}_a \cap \boldsymbol{I}_b| + \Delta_b,\ v_b := \tau_b + \Delta_b,\ \Delta_b > 0. \tag{2.8}$$

同様に，B は自分のプロフィールと A の興味の共通項目数に Δ_a を加えた値を秘匿計算し，A は閾値 τ_a に Δ_a を加えて同じようにブラインドする．

$$\hat{v}_a := |\boldsymbol{P}_b \cap \boldsymbol{I}_a| + \Delta_a,\ v_a := \tau_a + \Delta_a,\ \Delta_a > 0. \tag{2.9}$$

Verification

Verification では，サーバがブラインドした閾値と共通項目数の大小関係を秘匿計算し結果を判定する．具体的に，マッチは $\hat{v}_b \geq v_b \wedge \hat{v}_a \geq v_a$ が成り立つ場合であり，ミスマッチは $\hat{v}_b < v_b \vee \hat{v}_a < v_a$ が成り立つ場合である．サーバは，$\hat{v}_b \gtreqless v_b, \hat{v}_a \gtreqless v_a$ を判定し，$\hat{v}_b \geq v_b \wedge \hat{v}_a \geq v_a$ ならば「マッチ (True)」を，$\hat{v}_b < v_b \vee \hat{v}_a < v_a$ ならば「ミスマッチ (False)」をマッチングの結果として A と B に通知する．

共通項目を求める

ここまでで基本的にマッチングは完結しているが，本研究ではどの項目が共通したのかを求めたい場合も考慮する．もし A と B がマッチならば，A は自分の興味と B のプロフィールのどの項目が共通したのか，また B は自分の興味と A のプロフィールのどの項目が共通したのかを求める．ただし，この機能は強制されるものではなく，ユーザが使用するか否かを自由に選択する．

2.3 プロトコルへの要求事項

本研究で提案する秘匿マッチングプロトコルには，次の 6 つの事項を要求する．

互いに閾値を設けたマッチングであること：

これまで，[19, 27, 32] のように類似度ベースのマッチングや [7, 12, 31] のように共通項目数ベースのマッチングなど多く提案されているが，互いに閾値を設けてマッチングを行うプロトコルは現在のところ見られない．したがって，本研究では共通項目数ベースの閾値を互いに設ける秘匿マッチングの実現を要求する．

プライバシー情報を安全に管理すること：

全ユーザ共通の質問 Q に対して，各ユーザはプロフィール $\boldsymbol{P}$ と興味 $\boldsymbol{I}$ を生成する．これらは，プライバシー情報として厳重に管理する必要がある．それに加え，秘密鍵も厳重に管理する必要がある．そこで，本研究ではこれらをサーバに預けず，各ユーザが自己管理することでサーバに対するサイバー攻撃への対策を行う．

情報漏えいを防ぐこと：

マッチングでは，プロフィールを暗号化，興味をブラインドして解読困難にすることを要求する．これにより，マッチングの途中でプロフィールや興味などのプライバシー情報の漏えいを防ぐことができる．

プロトコルの途中でマッチングの結果を推測できないこと：

相手の興味に対する自分のプロフィールの評価や，自分の興味に対する相手のプロフィールの評価をサーバによる結果の通知より前に両ユーザともに追跡できないことを要求する．

ミスマッチの原因を追跡することができないこと：

ミスマッチのとき，すなわち片方または両方の閾値を満たさない場合，どちらが閾値未満であるのかという結果の詳細を追跡できないことを要求する．これにより，ユーザ間でいざこざを生まないように配慮することができる．

マッチすれば共通項目が求められること：

マッチの場合に限り，自分の興味と相手のプロフィールの共通項目を求められることを要求する．ただし，ミスマッチの場合は共通項目を全く求められないことを同時に要求する．マッチした後に互いに共通項目がわかれば，関係を築く上で余計な配慮を必要としなくて済む．

2.4 攻撃者モデル

攻撃者は，内部と外部に分けることができる．外部の攻撃者に対しては，[24] のように SSL/TLS などのセキュア通信を用いることで盗聴や中間者攻撃を防ぐ．内部の攻撃者については，semi-honset モデルを仮定する．semi-honset モデルとは，プロトコルに従うが，得られた情報から積極的にプロフィールを推測しようとする攻撃者であり，プロトコルから逸脱したり，他のユーザやサーバと結託して部分情報を得たり横流しをしない [9, 12, 18, 21, 25, 31]．

2.5 Paillier 暗号

次章以降で用いる暗号方式は，Paillier 暗号 [22] である．この節では，[21–23] を参考に Paillier 暗号について述べる．

2.5.1 準備

Paillier 暗号は，次のような性質を利用している．$\forall x \in \mathbb{N}$ のとき，二項定理より，

$$(1+N)^x \equiv 1 + xN \pmod{N^2}. \tag{2.10}$$

そこで，$y := (1+N)^x$ とすると，

$$x = \frac{y-1}{N} \pmod{N^2}. \tag{2.11}$$

ここで，関数 $L(x)$ を次のように定義する．

$$L(x) := \frac{x-1}{N}. \tag{2.12}$$

この関数は，復号アルゴリズムで使う．

2.5.2 鍵生成アルゴリズム

まず，$\gcd(pq, (p-1)(q-1)) = 1$ を満たす 2 つの素数 $p \xleftarrow{U} \mathbb{P}_N, q \xleftarrow{U} \mathbb{P}_N$ を生成する．このとき，公開鍵は $N = pq,\ g = (1+\bar{\alpha}N)\bar{\beta}^N$ であり，秘密鍵は $\lambda = \mathrm{lcm}(p-1, q-1)$ である．ただし，$\bar{\alpha} \xleftarrow{U} \mathbb{Z}^*_{N^2}, \bar{\beta} \xleftarrow{U} \mathbb{Z}^*_{N^2}$ である．

2.5.3 暗号化アルゴリズム

$\gcd(r, N) = 1$ を満たす乱数 $r \xleftarrow{U} \mathbb{Z}^*_{N^2}$ と公開鍵を用いて，次のように平文 $\forall m \in \mathbb{Z}_N$ を暗号化する．c を暗号文，Enc を暗号化関数とすると，

$$c = \mathsf{Enc}(m) := g^m r^N \pmod{N^2}. \tag{2.13}$$

2.5.4 復号アルゴリズム

秘密鍵 λ と (2.12) の関数 L を用いて次のように復号する．Dec を復号関数とすると，

$$\mathsf{Dec}(c) := \frac{L(c^\lambda \pmod{N^2})}{L(g^\lambda \pmod{N^2})} = m \pmod{N}. \tag{2.14}$$

(2.14) を計算するためには，$r^{\lambda N} \equiv 1 \pmod{N^2}$ を使う．

補題 1. $\forall r \in \mathbb{Z}_{N^2}^*. \gcd(r, N) = 1 \Longrightarrow r^{\lambda N} \equiv 1 \pmod{N^2}$.

Proof. Euler の整関数より $\varphi(N^2) = N(p-1)(q-1)$ であり，$\gcd(pq, (p-1)(q-1)) = 1$ なので，原始根定理より $\mathbb{Z}_{N^2}^*$ には，位数 N と位数 $(p-1)(q-1)$ の部分群が存在する．したがって，$\gcd(r, N) = 1$ を満たす $\forall r \in \mathbb{Z}_{N^2}^*$ について，Euler の定理より，

$$r^{\lambda} \equiv 1 \pmod{N}. \tag{2.15}$$

つまり，ある整数 γ を用いて $r^{\lambda} := 1 + N\gamma$ と表すことができる．よって，二項定理より，

$$r^{\lambda N} = 1 + N^2(\gamma + \cdots) \equiv 1 \pmod{N^2}. \tag{2.16}$$

以上より，補題 1 は示された． □

補題 1 を用いて，$L(c^{\lambda} \pmod{N^2})$ を具体的に計算すると，$\gcd(\bar{\alpha}, N) = \gcd(\bar{\beta}, N) = 1$ なので，

$$L(c^{\lambda} \pmod{N^2}) = L(g^{m\lambda} \pmod{N^2}) \equiv \bar{\alpha} m \lambda \pmod{N}. \tag{2.17}$$

2.5.5 加法準同型性

Paillier 暗号の暗号関数 Enc は加法準同型写像である．平文 $\forall m_1 \in \mathbb{Z}_N, \forall m_2 \in \mathbb{Z}_N$ を暗号化した暗号文 $\mathsf{Enc}(m_1)$ と $\mathsf{Enc}(m_2)$ の積は，

$$\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 + m_2) \pmod{N^2}. \tag{2.18}$$

さらに，平文のスカラー倍は次のように表すことができる．$\forall \mathcal{N} \in \mathbb{Z}_N$ において，

$$\{\mathsf{Enc}(m)\}^{\mathcal{N}} = \mathsf{Enc}(\mathcal{N} \cdot m) \pmod{N^2}. \tag{2.19}$$

(2.18), (2.19) のように暗号化した状態で平文の和やスカラー倍を求めることができる性質を**加法準同型性**と呼ぶ．

2.5.6 ベクトルの暗号化・復号の表記

平文ベクトル $\boldsymbol{P} = [p_1, p_2, \cdots, p_n]$ を暗号化するとき，次のように表す．

$$\mathsf{Enc}(\boldsymbol{P}) := [\mathsf{Enc}(p_1), \mathsf{Enc}(p_2), \cdots, \mathsf{Enc}(p_n)].$$

また，暗号文ベクトル $\boldsymbol{C} = [c_1, c_2, \cdots, c_n]$ を復号するとき，次のように表す．

$$\mathsf{Dec}(\boldsymbol{C}) := [\mathsf{Dec}(c_1), \mathsf{Dec}(c_2), \cdots, \mathsf{Dec}(c_n)].$$

2.5.7 安全性

[23] によると，Paillier 暗号は D-CR 仮定 (Decision Composite Residuosity assumption) と D-PDL 仮定 (Decision Partial Discrete Log assumption) の下で IND-CPA 安全である．

D-CR 仮定とは，$\forall x \in \mathbb{Z}_N^*, \exists \hat{\alpha} \in \mathbb{Z}_N^*, \exists \hat{\beta} \in \mathbb{Z}_N^*.\ x^N = \hat{\alpha} + \hat{\beta} \cdot N \pmod{N^2},\ \hat{r} \xleftarrow{U} \mathbb{Z}_N^*$ について，$(\hat{\alpha}, \hat{\beta})$ と $(\hat{\alpha}, \hat{r})$ を識別する問題は手に負えないとする仮定である．また，D-PDL 仮定とは，復号関数 Dec が一方向性であるという仮定である．

また，IND-CPA 安全とは，ある暗号文 c を発行したオラクルから様々な平文に対する暗号文を得て c の解読を試みようとするものの，解読につながる一切の利得 (advantage) は得られないことを意味する．つまり，暗号文を大量に発行して試みる攻撃 (CPA：chosen plaintext attack) では，c を解読することができない．これに大きく貢献する性質は，暗号文の識別不可能性 (IND：indistinguishability) である．識別不可能性とは，2 つの平文 m_1, m_2 とそのどちらかの暗号文 c_j を与えられたとき，c_j が m_1, m_2 のどちらの暗号文かを特定することができないという性質である．Paillier 暗号の場合，暗号化する毎に乱数 r を生成するので，同じ平文を暗号化しても同一の暗号文にならない．したがって，D-CR 仮定と D-PDL 仮定の下で Paillier 暗号は識別不可能性を満たす．

第3章　関連研究

この章では秘匿マッチングプロトコルに関連する既存研究について述べる．まず，秘匿マッチングの既存研究の比較を述べて，その上で代表的な研究について詳しく述べる．さらに，従来の秘匿マッチングにおける問題を整理する．

3.1　既存研究の比較

秘匿マッチングプロトコルは，2004 年に M. Freedman らによって Private Matching (PM) [8] として導入され，後に Private Set Intersection (PSI) と呼ばれるようになった．この PSI は，加法準同型暗号を用いた Oblivious Polynomial Evaluation (OPE) [20] によって実現される．また，[15] はプロフィールなどの要素を素数に写像する Map To Prime (MTP) を用いた PSI を提案した．さらに，[31] はプロフィール項目をベクトルの成分に対応させて，4 つのブラインド操作によって秘匿したまま共通項目を求める Blind Vector Transforming (BVT) という PSI を導入した．

秘匿マッチングは，共通項目ベースと類似度ベースに大別される．共通項目ベースのマッチングは，プロフィールの共通項目や共通項目数を秘匿計算し，ユーザが設定する条件を満足すればマッチと判定する．[6–8] には，片方の条件しか使わないため片思いの場合をマッチと判定してしまったり，2 回プロトコルを行う必要があるという欠点がある．[31] は，互いに設定する条件を同時に満たしたときにマッチと判定する方式を提案することで，これらの欠点を解決した．この方式では，BVT プロトコルで共通項目数を秘匿計算し，FACFM プロトコルで期待する共通項目数と一致しているかを同時に判定する．これによって，片思いの場合をミスマッチと判定する．したがって，どちらが片思いなのか，または両方ともにマッチの対象でなかったのかという情報も秘匿することができるため，プロフィールを偽造してマッチさせようとする不正行為を防ぐことができる．ただし，[31] で設定された条件とは，相手に期待する共通項目数との一致であり，少なすぎるのはもちろんのこと多すぎる場合もミスマッチと判定してしまう．本研究では，この欠点を指摘し互いに閾値を条件とする秘匿マッチングプロトコルを提案する．

通常，マッチングの安全性を高めようとすると，[27, 31] のようにマッチしたか否かのみを出力する方式になってしまう．しかし，実際のマッチングサービスに応用することを考慮すれば，安全であることも重要であるが，マッチしたとき共通項目を互いに共有し合う方が関係を円滑に構築することができる．[12] の提案したプロトコルは，MTP をベースにしたマッチングであり，マッチした場合にどの項目が共通したのかを検証することが

できる．このプロトコルはマッチしなければ何もわからないように設計されている．しかし，[12] では素数の積を平文とするため，プロフィール項目として扱うことができる質問項目数が限られてしまうという欠点がある．本研究では，「マッチしたときのみ共通項目がわかる」マッチングを MTP ベースでない方法で実現し，この欠点を解決する．

類似度ベースのマッチングは，独自に類似度を定義し，それを秘匿計算しユーザが設定する条件を満足すればマッチする．類似度は，プロフィールの項目の Euclid 距離 [14,30] や Manhattan 距離 [10]，重み付き行列式 [19,32]，Social Proximity [27]，射影ベクトルの大きさ [28]，[29] が提案した Dice 類似度 [11] などである．しかし，これらの方式では片方の条件しか使わないため，片思いの場合をマッチと判定してしまう可能性がある．これを解決する手段として考えられる方法は，[31] のようにマッチングの結果を判定するサーバを設けることである．SNS などに応用することを考慮すれば，必ず運営サーバが存在するので，判定を担うようにマッチングサービスを設計することは難しくはない．

以上の秘匿マッチングは，二者間の秘匿マッチングであるが，Shamir の鍵共有 [26] などの Secure Multiparty Computation(SMT) を応用したマルチパーティの秘匿マッチングも提案されている [13,16,17]．

次節以降では，まず [8,12,15,31] の共通項目ベースの秘匿マッチングについて，さらに [27,32] の類似度ベースの秘匿マッチングについて説明する．最後に，まとめとして従来の秘匿マッチングにおける問題点の整理を行う．

3.2 FNP04 [8]

Client の集合を $X = \{x_1, x_2, \cdots, x_{k_c}\}$，Server の集合を $Y = \{y_1, y_2, \cdots, y_{k_s}\}$ とする．それを用いて，Private Matching for Set Intersection PM で $X \cap Y$ を，Private Matching for Set Cardinary PM_C で $|X \cap Y|$ を秘匿計算する．なお，Client と Server は semi-honest モデルを仮定する．

3.2.1 Private Matching for Set Intersection PM

入力： $X = \{x_1, x_2, \cdots, x_{k_c}\},\ Y = \{y_1, y_2, \cdots, y_{k_s}\}$

出力： $X \cap Y$

1. Client

(a) Paillier 暗号などの加法準同型暗号の秘密鍵と公開鍵を生成する．

(b) X の要素を解とする多項式 $P(y)$ の係数 α_j を求める．

$$P(y) = (x_1 - y)(x_2 - y)\cdots(x_{k_c} - y) = \sum_{j=0}^{k_c} \alpha_j y^j. \tag{3.1}$$

(c) k_c+1 個の係数 $\alpha_0, \alpha_1, \cdots, \alpha_{k_c}$ をすべて自分の公開鍵で暗号化して Server に送信する．

$$\mathsf{Enc}(\alpha_0), \mathsf{Enc}(\alpha_1), \cdots, \mathsf{Enc}(\alpha_{k_c}). \tag{3.2}$$

2. Server

(a) Horner の規則を用いて，$\mathsf{Enc}(P(y_1)), \mathsf{Enc}(P(y_2)), \cdots, \mathsf{Enc}(P(y_{k_s}))$ を求める．具体的に，$\alpha_0, \alpha_1, \alpha_2, \alpha_3$ の暗号文の場合，$P(y_1)$ を次のように計算する．

$$\mathsf{Enc}(P(y_1)) = \Big\{\big\{\mathsf{Enc}(\alpha_3)^{y_1} \cdot \mathsf{Enc}(\alpha_2)\big\}^{y_1} \cdot \mathsf{Enc}(\alpha_1)\Big\}^{y_1} \cdot \mathsf{Enc}(\alpha_0). \tag{3.3}$$

(b) 乱数 r を用いて次のように計算し，k_s 個の暗号文を Server に送信する．

$$\mathsf{Enc}(rP(y_i) + y_i) = \{\mathsf{Enc}(P(y_i))\}^r \cdot \mathsf{Enc}(y_i). \tag{3.4}$$

3. Client

すべて復号して $P(y_i) = 0$ ならば，暗号文から y_i を得ることができる．つまり，すべて復号して得られた要素と X の共通要素が $X \cap Y$ である．

3.2.2 Private Matching for Set Cardinary PM_C

入力： $X = \{x_1, x_2, \cdots, x_{k_c}\},\ Y = \{y_1, y_2, \cdots, y_{k_s}\}$

出力： $|X \cap Y|$

1. Client

(a) Paillier 暗号などの加法準同型暗号の秘密鍵と公開鍵を生成する．

(b) X の要素を解とする多項式 $P(y)$ の係数 α_j を求める．

$$P(y) = (x_1 - y)(x_2 - y)\cdots(x_{k_c} - y) = \sum_{j=0}^{k_c} \alpha_j y^j. \tag{3.5}$$

(c) k_c+1 個の係数 $\alpha_0, \alpha_1, \cdots, \alpha_{k_c}$ をすべて自分の公開鍵で暗号化して Server に送信する．

$$\mathsf{Enc}(\alpha_0), \mathsf{Enc}(\alpha_1), \cdots, \mathsf{Enc}(\alpha_{k_c}). \tag{3.6}$$

2. Server

(a) Horner の規則を用いて，$\mathsf{Enc}(P(y_1)), \mathsf{Enc}(P(y_2)), \cdots, \mathsf{Enc}(P(y_{k_s}))$ を求める．

(b) 乱数 r を用いて，次のように計算し， k_s 個の暗号文を Server に送信する．

$$\mathsf{Enc}(rP(y_i)) = \{\mathsf{Enc}(P(y_i))\}^r. \tag{3.7}$$

3. Client

すべて復号して $rP(y_i) = 0$ を満たす暗号文の個数が $|X \cap Y|$ である．

3.3 KLP11 [15]

A の集合を $X_A = \{a_1, a_2, \cdots, a_k\}$，$B$ の集合を $X_B = \{b_1, b_2, \cdots, b_k\}$ とする．まず，A, B はそれぞれ MTP を行い，要素に対応した素数を求め，それらを複数のバケットにグループ分けする．そして，バケットごとに共通の素数から $X_A \cap X_B$ を秘匿計算する．なお，A と B は semi-honest モデルを仮定する．

3.3.1 Map To Prime (MTP)

A, B は写像 $\mathfrak{p} : \{0,1\}^* \to \mathcal{P}_\eta$ を用いて要素を素数に対応させる．

$$\alpha_i = \mathfrak{p}(a_i),\ \beta_i = \mathfrak{p}(b_i), \tag{3.8}$$

さらに，ハッシュ関数 $H : \{0,1\}^* \to \{1, 2, \cdots, l\}$ を用いて各素数をバケット $\{\mathcal{B}_j\}_{j=1}^l$ に分ける．

3.3.2 Mutual PSI Protocol

入力： $X_A = \{a_1, a_2, \cdots, a_k\},\ X_B = \{b_1, b_2, \cdots, b_k\}$

出力： $X_A \cap X_B$

1. A, B は MTP によって要素に対応する素数とバケットをそれぞれ求める．また，加法準同型暗号の公開鍵と秘密鍵をそれぞれ生成する．

2. バケット内の素数をすべて掛け合わせる．

$$A_{H(a_i)} \leftarrow A_{H(a_i)} \cdot \mathfrak{p}(a_i),\ B_{H(b_i)} \leftarrow B_{H(b_i)} \cdot \mathfrak{p}(b_i). \tag{3.9}$$

3. A は $r_1 \xleftarrow{U} \mathbb{Z}_{\lfloor N/4 \rfloor}, r_2 \xleftarrow{U} \mathbb{Z}_{\lfloor N/4 \rfloor}$ のように乱数を生成して，B に $\mathsf{Enc}_b(r_2), \mathsf{Enc}_b(A_j^2), \mathsf{Enc}_b(r_1 A_j^2)$ を送信する．また，B は $s_1 \xleftarrow{U} \mathbb{Z}_{\lfloor N/4 \rfloor}, s_2 \xleftarrow{U} \mathbb{Z}_{\lfloor N/4 \rfloor}$ のように乱数を生成して，A に $\mathsf{Enc}_a(s_1), \mathsf{Enc}_a(B_j^2), \mathsf{Enc}_a(s_2 B_j^2)$ を送信する．

4. A は，加法準同型性を用いて次のように計算する．

$$\mathsf{Enc}_a((r_1+s_1)A_j^2+(r_2+s_2)B_j^2)=\mathsf{Enc}_a(r_1A_j^2)\cdot\mathsf{Enc}_a(s_1)^{A_j^2}\cdot\mathsf{Enc}_a(B_j^2)^{r_2}\cdot\mathsf{Enc}_a(s_2B_j^2). \tag{3.10}$$

B も，加法準同型性を用いて次のように計算する．

$$\mathsf{Enc}_b((r_1+s_1)A_j^2+(r_2+s_2)B_j^2)=\mathsf{Enc}_b(r_1A_j^2)\cdot\mathsf{Enc}_b(A_j^2)^{s_1}\cdot\mathsf{Enc}_b(r_2)^{B_j^2}\cdot\mathsf{Enc}_b(s_2B_j^2). \tag{3.11}$$

5. A, B は復号して $(r_1+s_1)A_j^2+(r_2+s_2)B_j^2$ を得る．

6. A は，$(r_1+s_1)A_j^2+(r_2+s_2)B_j^2$ がバケット $\mathcal{B}_j$ の素数 $\mathfrak{p}(a_i)$ の 2 乗で割り切れるか否か，すなわち $\mathfrak{p}(a_i)^2|((r_1+s_1)A_j^2+(r_2+s_2)B_j^2)$ をひとつずつ検証する．B も同様に $\mathfrak{p}(b_i)^2|((r_1+s_1)A_j^2+(r_2+s_2)B_j^2)$ を検証する．割り切れる素数に対応する要素の集合が $X_A\cap X_B$ である．

3.3.3 バケットを使う理由

プロトコルでは，素数をバケットに分類してからバケットごとに素数を掛け合わせて暗号化するため，合計 l 個の暗号文を生成しなければいけない．暗号化にかかる時間や通信量がバケットの個数だけ倍増されるので，バケットの個数をなるべく減らしたいと考えるのが普通である．例えば，[12,21] は MTP を応用した秘匿マッチングプロトコルをバケットを使わずに提案している．しかし，バケットは集合の大きさと深い関係性がある．

バケットに分ける理由は明記されてないが，k $(=|X_A|=|X_B|)$ の大きさに依存せずに共通要素を求められることを保証するためである．バケットを用いない場合，(3.10),(3.11) で平文の桁あふれを防ぐことができないため，バケットを用意して桁あふれしないようにプロトコルは設計されている．したがって，[12,21] はともにバケットを用いないので k の大きさによって正当性が失われる危険性がある．つまり，素数の積を平文とするためプロフィール項目として扱うことができる質問項目数 k が制限される．

3.4 IO16 [12]

この秘匿マッチングは，相手に求める条件が満たされない場合にミスマッチ (False) と判定し，すべての条件を満たす場合にマッチと判定する．マッチの場合は，共通項目 $X_A\cap X_B$ が出力される．なお，A と B は semi-honest モデルを仮定する．また，次の図 3.1 が表すとおり，A と B のマッチングはサーバ S を介して行われる．S は運営サーバであり，マッチングに条件の反映，通信回数の削減などの役割を担っている．S も A, B と同様 semi-honest モデルを仮定する．

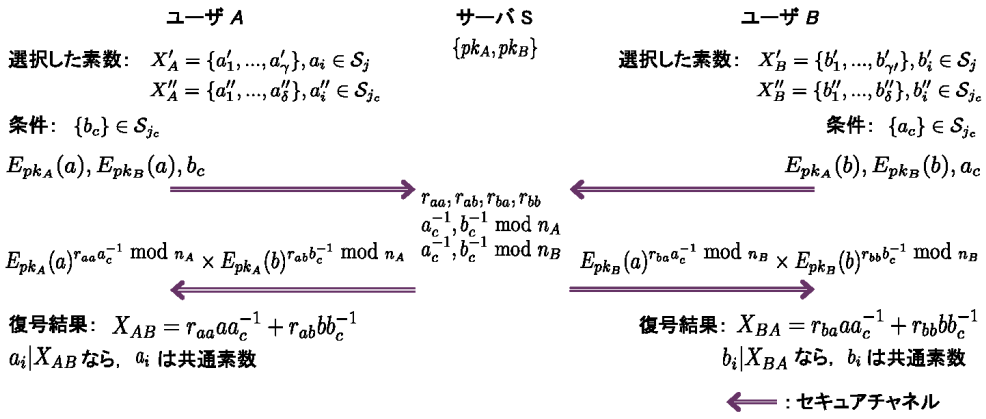

図 3.1: 条件付き秘匿マッチング [12] のシステムモデル

まず，S は k 個のプロフィール用の質問に対応した集合 $\{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k\}$ を用意する．

$$\mathcal{S}_j = \{p_1^{(j)}, p_2^{(j)}, \cdots, p_n^{(j)}\} \in \mathbb{P}^n. \tag{3.12}$$

ただし，$\mathcal{S} = \bigcup \mathcal{S}_j$, $\mathcal{S}_i \cap \mathcal{S}_j = \varnothing$ $(i \neq j)$ とする．A, B は，$\mathcal{S}$ を用いてプロフィールの項目を素数に対応させる．A のプロフィールを $X_A = \{a_1, a_2, \cdots, a_k\}$，$B$ のプロフィールを $X_B = \{b_1, b_2, \cdots, b_k\}$ とする．そして，A の相手に求める条件を $X'_A = \{a'_1, a'_2, \cdots, a'_\delta\}$，$B$ の相手に求める条件を $X'_B = \{b'_1, b'_2, \cdots, b'_{\delta'}\}$ とする．相手に求める条件 a'_i は，B のプロフィールの i 番目の項目が $b_i = a'_i$ であることを A が期待することを意味する．

入力： $X_A,\ X'_A,\ X_B,\ X'_B$

出力： $X_A \cap X_B$/False

1. A, B は Paillier 暗号の公開鍵と秘密鍵をそれぞれ生成する．

2. A は，プロフィール項目と条件の項目を次のようにすべて掛け合わせる．

$$a = \prod_{j=1}^{k} a_j, \quad a_c = \prod_{j}^{\delta} a'_j. \tag{3.13}$$

そして，a を A と B の公開鍵で暗号化し，$\mathsf{Enc}_a(a), \mathsf{Enc}_b(a), a_c$ を S に送信する．B も，

$$b = \prod_{j=1}^{k} b_j, \quad b_c = \prod_{j}^{\delta'} b'_j. \tag{3.14}$$

そして，b を A と B の公開鍵で暗号化し，$\mathsf{Enc}_a(b), \mathsf{Enc}_b(b), b_c$ を S に送信する．

3. S は，4 つの乱数 $r_{aa}, r_{ab}, r_{ba}, r_{bb}$ を生成する．乱数は，$|n_a|, |n_b|$ をそれぞれの暗号文のビット長，t を $p_j^{(i)}$ のビット長とするとき，次のビット長で生成する．

$$|r_{aa}| = |n_a| - (k+\delta)\cdot t - 1, \tag{3.15}$$
$$|r_{ab}| = |n_a| - (k+\delta')\cdot t - 1, \tag{3.16}$$
$$|r_{ba}| = |n_b| - (k+\delta)\cdot t - 1, \tag{3.17}$$
$$|r_{bb}| = |n_b| - (k+\delta')\cdot t - 1. \tag{3.18}$$

そして，サーバは加法準同型性を用いて次のように計算する．

$$\mathsf{Enc}_a(a)^{r_{aa}a_c^{-1}} \cdot \mathsf{Enc}_a(b)^{r_{ab}b_c^{-1}} = \mathsf{Enc}_a(ar_{aa}a_c^{-1} + br_{ab}b_c^{-1}), \tag{3.19}$$
$$\mathsf{Enc}_b(a)^{r_{ba}a_c^{-1}} \cdot \mathsf{Enc}_b(b)^{r_{bb}b_c^{-1}} = \mathsf{Enc}_b(ar_{ba}a_c^{-1} + br_{bb}b_c^{-1}). \tag{3.20}$$

さらに，$\mathsf{Enc}_a(ar_{aa}a_c^{-1} + br_{ab}b_c^{-1})$ を A に，$\mathsf{Enc}_b(ar_{ba}a_c^{-1} + br_{bb}b_c^{-1})$ を B に送信する．

4. A は復号して $ar_{aa}a_c^{-1} + br_{ab}b_c^{-1} \equiv X_{AB}$ を得る．そして，X_{AB} がプロフィール項目 a_j で割り切れるか否か，すなわち $a_j | X_{AB}$ をひとつずつ検証する．もし，条件をすべて互いに満たしていなければ a_c^{-1}, b_c^{-1} の一部または全部が残るため，$a_j | X_{AB}$ の検証で桁あふれするので割り切ることができない．すべての条件を互いに満たす場合，$X_A \cap X_B$ を求めることができる．

5. B も同様に復号して $ar_{ba}a_c^{-1} + br_{bb}b_c^{-1} \equiv X_{BA}$ を得る．そして，X_{BA} がプロフィール項目 b_j で割り切れるか否か，すなわち $b_j | X_{BA}$ をひとつずつ検証する．

3.5 ZDLG13 [31]

3.5.1 システムモデル

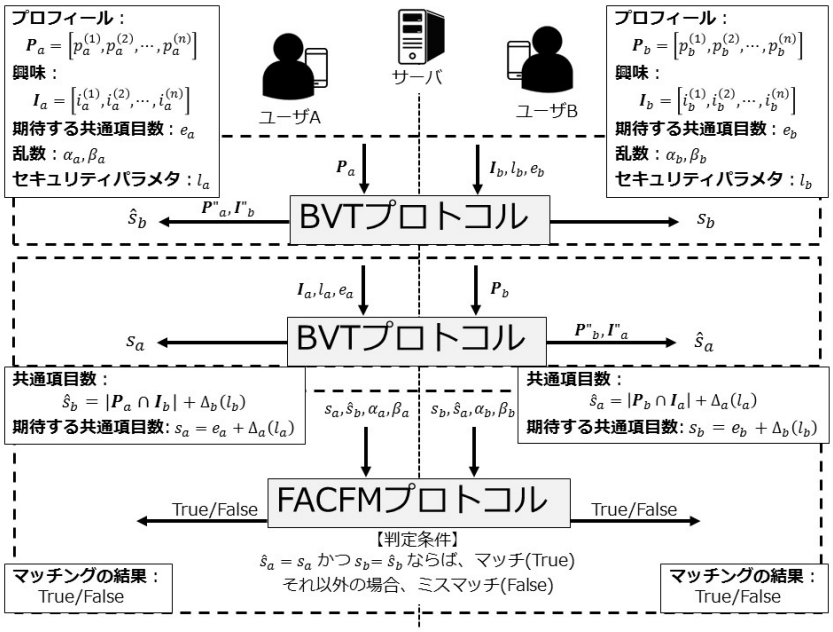

図 3.2: [31] のシステムモデル

この秘匿マッチングは，2 回の BVT プロトコルと 1 回の FACFM プロトコルによって構成される．BVT プロトコルでブラインドされた興味とプロフィールの共通項目を求め，FACFM プロトコルで自分が期待する共通項目数なのかを判定する．ともに興味とプロフィールの共通項目数が期待する値であればマッチであり，それ以外はミスマッチと判定される．したがって，興味とプロフィールの共通項目数が期待する値よりも大きくても小さくてもミスマッチと判定される．

具体的に，秘匿マッチングは Setting，Private Matching，Verification の 3 段階で行われる．まず，Setting として各ユーザはプロフィールと興味の準備や鍵生成などを行い，次に Private Matching として BVT プロトコルを，最後に Verification として FACFM プロトコルを実行する．

Setting

A は，プロフィール $\boldsymbol{P}_a$ と興味 $\boldsymbol{I}_a$ の他に自分の興味と相手のプロフィールの共通項目数の期待する値 e_a と乱数 α_a, β_a，そしてセキュリティパラメタ l_a を用意する．同様に，B

もプロフィール $\boldsymbol{P}_b$ と興味 $\boldsymbol{I}_b$ の他に自分の興味と相手のプロフィールの共通項目数の期待する値 e_b と乱数 α_b, β_b，そしてセキュリティパラメタ l_b を用意する．

Private Matching

A の $\boldsymbol{P}_a$ と B の $\boldsymbol{I}_b, l_b, e_b$ を入力として BVT プロトコルを実行し，A は $\hat{s}_b$ を，B は s_b を出力として受け取る．s_b は e_b に $\Delta_b(l_b)$ を加えてランダマイズした値であり，$\Delta_b(l_b)$ は B のみが知っている値である．したがって，A は $\hat{s}_b$ から $|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ を求めることはできない．同様に，B の $\boldsymbol{P}_b$ と A の $\boldsymbol{I}_a, l_a, e_a$ を入力として BVT プロトコルを実行し，B は $\hat{s}_a$ を，A は s_a を出力として受け取る．s_a は e_a に $\Delta_a(l_a)$ を加えてランダマイズした値であり，$\Delta_a(l_a)$ は A のみが知っている値である．したがって，B は $\hat{s}_a$ から $|\boldsymbol{P}_b \cap \boldsymbol{I}_a|$ を求めることはできない．

Verification

A の $s_a, \hat{s}_b, \alpha_a, \beta_a$ と B の $s_b, \hat{s}_a, \alpha_b, \beta_b$ を入力として FACFM プロトコルを実行し，サーバは $s_a = \hat{s}_a \wedge s_b = \hat{s}_b$ が成り立つか否かを判定し A と B に結果として True/False を渡す．α_a, β_a や α_b, β_b は，プロトコルの途中で $s_a = \hat{s}_a \wedge s_b = \hat{s}_b$ を追跡することができないようにするために用いられる．

3.5.2 BVT プロトコル

入力： A のプロフィール $\boldsymbol{P}_a$， B の興味 $\boldsymbol{I}_b$，B のセキュリティパラメタ l_b と期待する共通項目数 e_b.

出力： A に要求する共通項目数をランダマイズした値 s_b，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ をランダマイズした値 $\hat{s}_b$.

1. A はプロフィールを暗号化し B に送る
A は自分の公開鍵で $\boldsymbol{P}_a$ を暗号化して，B に $\mathsf{Enc}_a(\boldsymbol{P}_a)$ を送信する．

2. A の暗号化プロフィールと B の興味をブラインドする

(i) B は，$\mathsf{Enc}_a(\boldsymbol{P}_a)$ と同じ項目数の乱数ベクトル $\boldsymbol{r}_b$ を生成して，

$$\hat{\boldsymbol{P}}_a := \mathsf{Enc}_a(\boldsymbol{P}_a + \boldsymbol{r}_b), \quad \widetilde{\boldsymbol{I}}_b := \boldsymbol{I}_b + \boldsymbol{r}_b. \tag{3.21}$$

(ii) l_b 個のダミーのベクトル $\boldsymbol{y}_b$ を生成し，$\hat{\boldsymbol{y}}_b := \mathsf{Enc}_a(\boldsymbol{y}_b)$ とする．さらに，$\boldsymbol{y}_b$ の k_b 個の項目を別の値に入れ替えて新しいベクトル $\widetilde{\boldsymbol{y}}_b$ をつくる．ただし，$k_b \xleftarrow{U} \{1, 2, \cdots, l_b\}$.

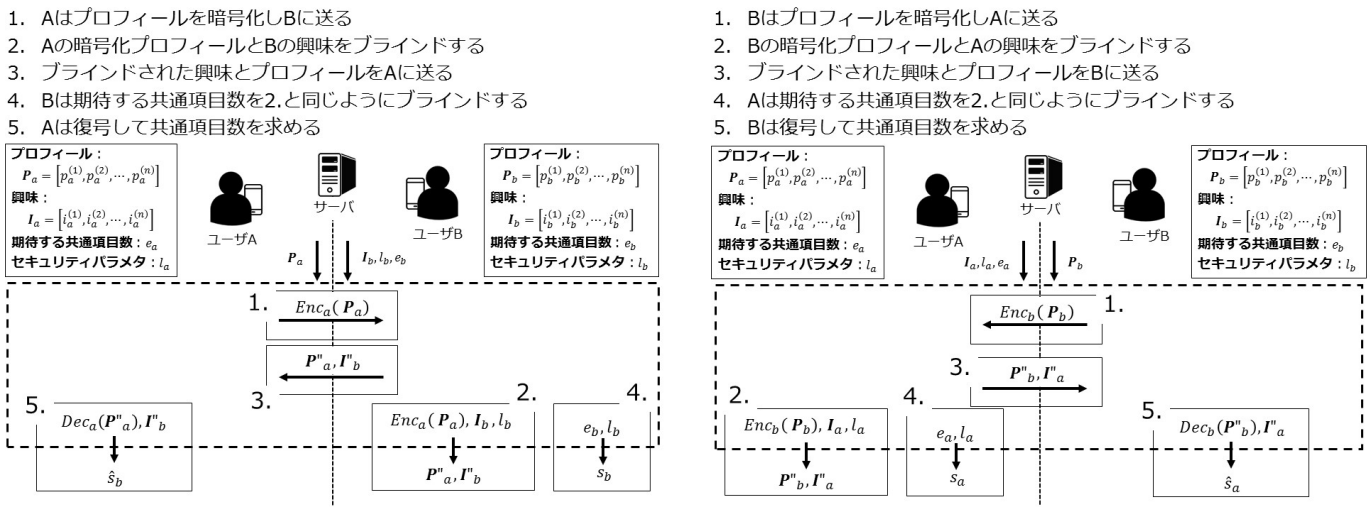

図 3.3: BVT プロトコルの全体像

(iii) $\hat{\boldsymbol{P}}_a, \widetilde{\boldsymbol{I}}_b$ にそれぞれ $\hat{\boldsymbol{y}}_b, \widetilde{\boldsymbol{y}}_b$ を追加し拡張する．

$$\boldsymbol{P}'_a := \hat{\boldsymbol{P}}_a \oplus \hat{\boldsymbol{y}}_b, \quad \boldsymbol{I}'_b := \widetilde{\boldsymbol{I}}_b \oplus \widetilde{\boldsymbol{y}}_b. \tag{3.22}$$

(iv) $\boldsymbol{P}'_a$ と $\boldsymbol{I}'_b$ を同じようにシャッフルする．

$$\boldsymbol{P}''_a := \mathrm{Shuffle}\left[\boldsymbol{P}'_a\right], \quad \boldsymbol{I}''_b := \mathrm{Shuffle}\left[\boldsymbol{I}'_b\right]. \tag{3.23}$$

3. **ブラインドされた興味とプロフィールを** A **に送る**

B は，$\boldsymbol{P}''_a$ と $\boldsymbol{I}''_b$ を A に送信する．

4. B **は期待する共通項目数を** *2* **と同じようにブラインドする**

B は，$\boldsymbol{P}_a$ と $\boldsymbol{I}_b$ の期待する共通項目数 e_b を *2* の *(i)–(iv)* と同じようにブラインドする．

$$s_b := e_b + l_b - k_b. \tag{3.24}$$

5. A **は復号して共通項目数を求める**

A は，$\boldsymbol{P}''_a$ を復号し，$\boldsymbol{I}''_b$ との共通項目数を求め，それを $\hat{s}_b$ とする．

$$\hat{s}_b := |\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b|. \tag{3.25}$$

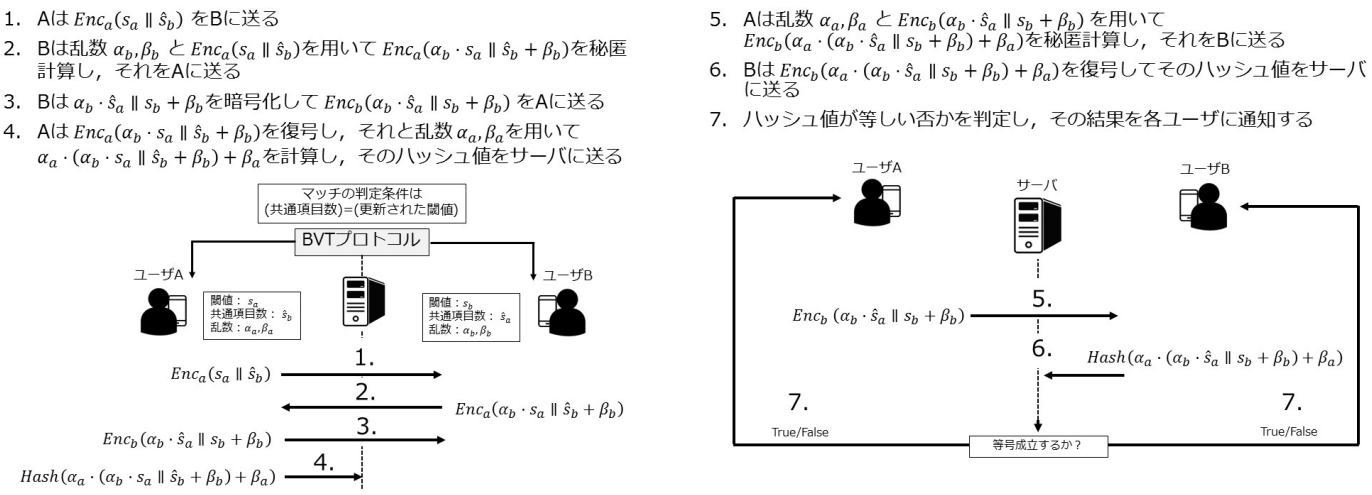

図 3.4: FACFM プロトコルの全体像

3.5.3 FACFM プロトコル

BVT プロトコルによって相手に要求する共通項目数をランダマイズした値と，自分の興味と相手のプロフィールの共通項目数をランダマイズした値のペアをそれぞれ得る．すなわち，A は $(s_a, \hat{s}_b)$ を，B は $(s_b, \hat{s}_a)$ を得たとする．このとき，$s_a = \hat{s}_a \wedge s_b = \hat{s}_b$ を満たすならば，$e_a = |\boldsymbol{P}_b \cap \boldsymbol{I}_a| \wedge e_b = |\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ を満たすので A と B はマッチする．FACFM プロトコルでは，$s_a, s_b, \hat{s}_a, \hat{s}_b$ などの結果につながる部分情報を追跡されないようにした上で $s_a||\hat{s}_b = \hat{s}_b||s_a$ が成り立つか否かをサーバが判定する．そのために，A と B がオフラインでそれぞれ生成した乱数 $(\alpha_a, \beta_a),\ (\alpha_b, \beta_b)$ を FACFM プロトコルで使用する．

入力： $(s_a, \hat{s}_b), (s_b, \hat{s}_a)$，$A$ の乱数 α_a, β_a，B の乱数 α_b, β_b.

出力： $s_a||\hat{s}_b = \hat{s}_a||s_b$ が成り立つか否か.

1. A は，自分の公開鍵で $s_a||\hat{s}_b$ を暗号化し，$\mathsf{Enc}_a(s_a||\hat{s}_b)$ を B に送信する．

2. B は，$\alpha_b, \beta_b, \mathsf{Enc}_a(s_a||\hat{s}_b)$ を用いて次を計算し，その結果を A に送信する．

$$\mathsf{Enc}_a(s_a||\hat{s}_b)^{\alpha_b} \cdot \mathsf{Enc}_a(\beta_b) = \mathsf{Enc}_a(\alpha_b \cdot s_a||\hat{s}_b + \beta_b). \tag{3.26}$$

3. B は自分の公開鍵で $\alpha_b \cdot \hat{s}_a||s_b + \beta_b$ を暗号化し，$\mathsf{Enc}_b(\alpha_b \cdot \hat{s}_a||s_b + \beta_b)$ を A に送信する．

4. A は，$\mathsf{Enc}_a(\alpha_b \cdot s_a||\hat{s}_b + \beta_b)$ を復号して $\alpha_a \cdot (\alpha_b \cdot s_a||\hat{s}_b + \beta_b) + \beta_a$ を計算し，そのハッシュ値をサーバに送信する．

5. さらに A は，$\alpha_a, \beta_a, \mathsf{Enc}_b(\alpha_b \cdot \hat{s}_a || s_b + \beta_b)$ を用いて次を計算し，その結果を B に送信する．

$$\mathsf{Enc}_b(\alpha_b \cdot \hat{s}_a || s_b + \beta_b)^{\alpha_a} \cdot \mathsf{Enc}_b(\beta_a) = \mathsf{Enc}_b(\alpha_a \cdot (\alpha_b \cdot \hat{s}_a || s_b + \beta_b) + \beta_a). \tag{3.27}$$

6. B は，$\mathsf{Enc}_b(\alpha_a \cdot (\alpha_b \cdot \hat{s}_a || s_b + \beta_b) + \beta_a)$ を復号して $\alpha_a \cdot (\alpha_b \cdot \hat{s}_a || s_b + \beta_b) + \beta_a$ のハッシュ値をサーバに送信する．

7. サーバは，受け取った 2 つのハッシュ値が等しいか否かを判断し，等しければ True を，等しくなければ False を A と B に送信する．

3.5.4 正当性

定理 2. BVT プロトコルは，正しく共通項目数を求めることができる．

proof. $\exists x \in \boldsymbol{P}_a \cap \boldsymbol{I}_b, \exists y \notin \boldsymbol{P}_a \cap \boldsymbol{I}_b, \exists y' \notin \boldsymbol{P}_a \cap \boldsymbol{I}_b, \exists y \in \boldsymbol{P}_a, \exists y' \in \boldsymbol{I}_b$ とする．BVT プロトコルでは，次のように 4 つの操作が行われ，$(\mathsf{Enc}_a(\boldsymbol{P}_a), \boldsymbol{I}_b)$ から $(\boldsymbol{P}''_a, \boldsymbol{I}''_b)$ をつくる．

(i) 項目のブラインド
$\forall r \xleftarrow{U} \mathbb{Z}_N$ に対して，Paillier 暗号の加法準同型性より，

$$\begin{cases} \mathsf{Dec}_a(\mathsf{Enc}_a(x) \cdot \mathsf{Enc}_a(r)) = \mathsf{Dec}_a(\mathsf{Enc}_a(x + r)) = x + r, \\ \mathsf{Dec}_a(\mathsf{Enc}_a(y) \cdot \mathsf{Enc}_a(r)) = \mathsf{Dec}_a(\mathsf{Enc}_a(y + r)) \neq y' + r. \end{cases} \tag{3.28}$$

(ii) ダミーの生成と入れ替え
ダミーを d とするとき，$\mathsf{Dec}_a(\mathsf{Enc}_a(\mathsf{d})) = \mathsf{d}$ が成り立つ．また，ダミーを d' $(\neq \mathsf{d})$ に入れ替えるとき，$\mathsf{Dec}_a(\mathsf{Enc}_a(\mathsf{d})) \neq \mathsf{d}'$ が成り立つ．

(iii) ダミーの追加

(iv) シャッフル
ダミーを追加してシャッフルを行うことで，項目の並び順をランダムにする．

以上 *(i)–(iv)* より，

$$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| = |\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b| - (l_b - k_b). \tag{3.29}$$

特に $l_b = k_b$ のとき，BVT プロトコルは正しく共通項目数を求めることができる．ただし，マッチングでは $|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ を秘匿すべきプライバシー情報と見なすため，通常 $l_b \neq k_b$ である． □

定理 3. FACFM プロトコルは，マッチングの結果を正しく出力する．

proof. $s_a = \hat{s}_a$ かつ $s_b = \hat{s_b}$ の場合にのみマッチし，それ以外の場合はミスマッチを出力することを証明すれば，サーバがマッチングの結果を正しく出力することを示すことができる．まず，簡単のため一次関数 f_a, f_b を次のように定義する．f_a を傾き α_a，y 切片 β_a の一次関数とし，f_b を傾き α_b，y 切片 β_b の一次関数とする．f_a を所持するのは A であり，f_b を所持するのは B である．

$$\begin{cases} f_a(x) := \alpha_a \cdot x + \beta_a, \\ f_b(x) := \alpha_b \cdot x + \beta_b. \end{cases} \tag{3.30}$$

A は，Paillier 暗号の加法準同型性を用いて，$f_a(f_b(s_a||\hat{s}_b))$ $(= \alpha_a \cdot (\alpha_b \cdot s_a||\hat{s}_b + \beta_b) + \beta_a)$ を秘匿計算する．まず，$\mathsf{Enc}_a(s_a||\hat{s}_b)$ を B に送り，B は $\alpha_b, \beta_b, \mathsf{Enc}_a(s_a||\hat{s}_b)$ を用いて $\mathsf{Enc}_a(f_b(s_a||\hat{s}_b))$ を計算し，それを A に返す．

$$\mathsf{Enc}_a(s_a||\hat{s}_b)^{\alpha_b} \cdot \mathsf{Enc}_a(\beta_b) = \mathsf{Enc}_a(\alpha_b \cdot s_a||\hat{s}_b + \beta_b). \tag{3.31}$$

そして，復号して f_a に代入して得た結果を x_a とする．

$$x_a := \alpha_a \cdot \mathsf{Dec}_a(\mathsf{Enc}_a(\alpha_b \cdot s_a||\hat{s}_b + \beta_b)) + \beta_a = f_a(f_b(s_a||\hat{s}_b)). \tag{3.32}$$

B も $f_a(f_b(\hat{s}_a||s_b))$ $(= \alpha_a \cdot (\alpha_b \cdot \hat{s}_a||s_b + \beta_b) + \beta_a)$ を秘匿計算する．まず，$f_b(\hat{s}_a||s_b) = \alpha_b \cdot \hat{s}_a||s_b + \beta_b$ を暗号化し A に送る．そして，A は $\alpha_a, \beta_a, \mathsf{Enc}_b(f_b(\hat{s}_a||s_b))$ を用いて $\mathsf{Enc}_b(f_a(f_b(\hat{s}_a||s_b)))$ を計算し，それを B に返す．

$$\mathsf{Enc}_b(\alpha_b \cdot \hat{s}_a||s_b + \beta_b)^{\alpha_a} \cdot \mathsf{Enc}_b(\beta_a) = \mathsf{Enc}_b(\alpha_a \cdot (\alpha_b \cdot \hat{s}_a||s_b + \beta_b) + \beta_a) = \mathsf{Enc}_b(f_a(f_b(\hat{s}_a||s_b))). \tag{3.33}$$

それを復号した値を x_b とすると，

$$x_b := \mathsf{Dec}_b(\mathsf{Enc}_b(\alpha_a \cdot (\alpha_b \cdot \hat{s}_a||s_b + \beta_b) + \beta_a)) = f_a(f_b(\hat{s}_a||s_b)). \tag{3.34}$$

したがって，

- $s_a = \hat{s}_a \wedge s_b = \hat{s}_b$ のとき，すなわち $s_a||\hat{s}_b = \hat{s}_a||s_b$ のとき，$\mathsf{Hash}(x_a) = \mathsf{Hash}(x_b)$.
- それ以外のとき，すなわち $s_a||\hat{s}_b \neq \hat{s}_a||s_b$ のとき，$\mathsf{Hash}(x_a) \neq \mathsf{Hash}(x_b)$.

以上より，FACFM プロトコルは，マッチングの結果を正しく出力する． □

3.5.5 秘匿性

定理 4. BVT プロトコルでは，ユーザは相手の興味を特定することができない．

proof. A が B の興味を特定しようと試みるとする．

相手の興味を特定するためには，$\boldsymbol{I}''_b$ から $\boldsymbol{I}_b$ を解読する必要がある．しかし，BVT プロトコルでは乱数による項目の秘匿，ダミーの追加，シャッフルなどのブラインド操作によって，$\boldsymbol{I}''_b$ から $\boldsymbol{I}_b$ を求めることは極めて難しい．

次に，A は $\mathsf{Dec}_a(\boldsymbol{P}''_a)$ と $\boldsymbol{I}''_b$ との共通項目数 $\hat{s}_b$ から $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を特定しようとする．そこで，k_b を特定する事象を $\mathcal{X}$ とし，$\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を特定する事象を $\mathcal{Y}$ とする．まず，セキュリティパラメタ l_b から k_b の候補を探る．l_b は，$\boldsymbol{P}''_a$ と $\boldsymbol{P}_a$ のサイズから，

$$l_b = |\boldsymbol{P}''_a| - n. \tag{3.35}$$

k_b の候補は $k_b = \{1, 2, \cdots, l_b\}$ なので，確率 $\Pr(\mathcal{X})$ は，

$$\Pr(\mathcal{X}) = \frac{1}{l_b}. \tag{3.36}$$

k_b を求めた上で，$\hat{s}_b = |\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b|$ から $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を特定する．まず，$\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b$ を $\hat{s}_b - (l_b - k_b)$ 個のダミー項目と $l_b - k_b$ 個のそうでない項目に分ける．ダミーでないと推測される組合せの内，必ず 1 通りの正解があるので，この試行をすべての候補に対して繰り返すと，次の確率 $\Pr(\mathcal{Y}|\mathcal{X})$ でダミーでない項目を当てることができる．

$$\Pr(\mathcal{Y}|\mathcal{X}) = \sum_{k_b=1}^{l_b} \frac{1}{{}_n\mathrm{C}_{\hat{s}_b-(l_b-k_b)}}. \tag{3.37}$$

よって，A が $\hat{s}_b$ から $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ の特定に成功する確率 $\Pr(\mathcal{X} \wedge \mathcal{Y})$ は，$n \geq 5$ のとき，

$$\Pr(\mathcal{X} \wedge \mathcal{Y}) = \frac{1}{l_b} \sum_{k_b=1}^{l_b} \frac{1}{{}_n\mathrm{C}_{\hat{s}_b-(l_b-k_b)}} \leq \frac{3}{nl_b}. \tag{3.38}$$

以上より，相手の興味をすべて特定することも，自分のプロフィールと共通した興味も特定することができない． □

定理 5. FACFM プロトコルでは，サーバとユーザが結託していても，サーバがマッチングの結果を判定する前にそれを追跡することはできない．つまり，A がサーバと結託して，サーバから $\alpha_a \cdot (\alpha_b \cdot \hat{s}_a || s_b + \beta_b) + \beta_a$ のハッシュ値を受け取っても $\hat{s}_a || s_b$ を求めることはできない．

proof. FACFM で用いるハッシュ関数 Hash は，ランダムオラクルモデルでないので，総当たり攻撃で $\alpha_a \cdot (\alpha_b \cdot \hat{s}_a || s_b + \beta_b) + \beta_a$ を求めることができる．しかし，A は (α_a, β_a) のみしか知らないので，$\alpha_b \cdot \hat{s}_a || s_b + \beta_b$ から $\hat{s}_a || s_b$ を求める問題が残る．しかし，乱数 α, β から構成される一次関数 $\alpha \cdot x + \beta$ の値から，2 つの乱数を知らずに x を求めることはできない．ゆえに，$\alpha_b \cdot \hat{s}_a || s_b + \beta_b$ から $\hat{s}_a || s_b$ を求めることはできても $\hat{s}_a || s_b$ を求めることはできない． □

3.5.6 効率性

表 3.1 と表 3.2 は，それぞれ BVT プロトコルと FACFM プロトコルの効率性をまとめた表である．BVT プロトコルの計算量と通信量のオーダーは質問項目数 n に比例するが，FACFM プロトコルの計算量と通信量のオーダーは n に依存しない．

表 3.1: 2 回の BVT プロトコルの効率性

	ユーザ A	ユーザ B	サーバ
受信 [bit]	$2t_N \cdot (2n + l_b)$	$2t_N \cdot (2n + l_a)$	-
送信 [bit]	$2t_N \cdot (n + l_a)$	$2t_N \cdot (n + l_b)$	-
計算量オーダー	$\mathcal{O}(n + l_a + l_b)$	$\mathcal{O}(n + l_a + l_b)$	-
通信量オーダー	$\mathcal{O}(n + l_a + l_b)$	$\mathcal{O}(n + l_a + l_b)$	-

表 3.2: FACFM プロトコルの効率性

	ユーザ A	ユーザ B	サーバ
受信 [bit]	$4t_N + 1$	$4t_N + 1$	512
送信 [bit]	$4t_N + 256$	$4t_N + 256$	2
計算量オーダー	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
通信量オーダー	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

3.6 TLSL15 [27]

この秘匿マッチングではプロフィールを使わず，ユーザが所属しているオンラインコミュニティを使って相手との社会的緊密度 (Social Proximity) という類似度を使ってマッチングが行われる．この Social Proximity は，一般的な類似度ではなく独自に定義された数量である．

提案されたマッチングは 3 つのプライバシーレベルによって 3 つに分けられる．プライバシーレベルが 1 番低いマッチング (L1P) とは，オンラインコミュニティの種類をプロフィール項目に見立てた [8] の PM であり，互いに共通のコミュニティを知ることができる．また，プライバシーレベルが 2 番目に低いマッチングは，共通したコミュニティから算出される Social Proximity を閾値判定してマッチすれば互いに共通のコミュニティを知ることができ，ミスマッチの場合は False を得る．プライバシーレベルが一番高いマッチング (L1P) とは，互いに Social Proximity の閾値判定を行う．マッチングで出力される結果が True/False のみであり，自分にとって相手がマッチの対象か否かを知ることができる．なお，いずれのマッチングでもユーザは semi-honest モデルを仮定する．

3.6.1 類似度の定義

コミュニティの優先度

ユーザ i が所属するコミュニティの集合を C_i とすると，

$$C_i = \{C_i^1, C_i^2, \cdots, C_i^{c_i}\}, \quad c_i = |C_i|. \tag{3.39}$$

自分自身と友達を含めたすべてのコミュニティを $\bar{C}_i$ とする．友達の集合を $\mathcal{N}_i$ とすると，

$$\bar{C}_i = \bigcup_{j \in \mathcal{N}_i \cup \{i\}} C_j. \tag{3.40}$$

自分と友達 $j \in \mathcal{N}_i \cup \{i\}$ のコミュニティ $\bar{C}_i$ に対する優先度を $\beta_j(\bar{C}_i)$ とする．ただし，$0 \leq \beta_j(\bar{C}_i) \leq \beta^{\max}$ とし，大きい値の方が優先度が高いことを意味する．

友達の優先度

$\mathcal{N}_i$ を自分の好みでグループ FC_i 分けする．

$$FC_i = \{FC_i^1, FC_i^2, \cdots, FC_i^{f_i}\}, \quad f_i = |FC_i|, \quad \mathcal{N}_i = \bigcup_{j=1}^{f_i} FC_i^j. \tag{3.41}$$

友達 l が属する FC_i のインデックスを $FC(i,l)$ とする．ただし，$FC(i,i) = 0$ とする．

$$l \in FC_i^k \Longleftrightarrow k = FC(i,l). \tag{3.42}$$

各グループ FC_i^k に対する優先度を α_i^k とする．ただし，$0 \leq \alpha_i^k \leq \alpha^{\max}$ とし，大きい値の方が優先度が高いことを意味する．ただし，自分に対する優先度を $\alpha_i^0 = \alpha^{\max}$ とする．

Social Proximity

あるコミュニティ $i \in \bar{C}_R$ に対する R の類似度を $\Psi_{R \leftarrow I}^i$ とすると，

$$\Psi_{R \leftarrow I}^i = \frac{\displaystyle\sum_{j \in \bar{C}_R^i \cap \mathcal{N}_R} \left(\beta_j(\bar{C}_R^i) \cdot \sum_{k \in FC(R,j)} \alpha_R^k \right)}{\displaystyle\sum_{l=1}^{|\bar{C}_R|} \left(\sum_{j \in \bar{C}_R^l \cap \mathcal{N}_R} \left(\beta_j(\bar{C}_R^l) \cdot \sum_{k \in FC(R,j)} \alpha_R^k \right) \right)}. \tag{3.43}$$

R の Social Proximity $\Psi_{R \leftarrow I}$ とは，I と共通するコミュニティの類似度の和である．

$$\Psi_{R \leftarrow I} = \sum_{i \in \bar{C}_R \cap \bar{C}_I} \Psi_{R \leftarrow I}^i. \tag{3.44}$$

(3.44) より，$0 \leq \Psi_{R \leftarrow I} \leq 1$ である．

3.6.2 プロトコル中の表記に関して

このマッチングのプロトコルは，[8] の多項式を用いた秘匿計算を用いている．簡単のため，$\mathsf{Enc}_a(P(x))$ は多項式 $P(x)$ のすべての係数を A の公開鍵で暗号化を表すことに決める．また，$\mathsf{Enc}_a(P(n))$ は $P(x)$ に $x = n$ を代入した $P(n)$ を秘匿計算することを表すことに決める．

3.6.3 L1P

入力： $\bar{C}_R = \{\bar{C}_R^1, \bar{C}_R^2, \cdots, \bar{C}_R^{|\bar{C}_R|}\}, \bar{C}_I = \{\bar{C}_I^1, \bar{C}_I^2, \cdots, \bar{C}_I^{|\bar{C}_I|}\}$

出力： $\bar{C}_R \cap \bar{C}_I$

1. I, R は共通鍵 K を共有する．さらに，I は加法準同型暗号の秘密鍵と公開鍵を生成する．

2. I は，多項式 $P_I(x)$ の係数を自分の秘密鍵ですべて暗号化して R に送信する．

$$P_I(x) = \prod_{k=1}^{|\bar{C}_I|} (\bar{C}_I^k - x). \tag{3.45}$$

3. R は Horner の規則に従って $\mathsf{Enc}_I(P(\bar{C}_R^i))$ $(i \in \{1, 2, \cdots, |\bar{C}_R|\})$ を秘匿計算する．そして，後でどの項目が共通したのかを求めるために，乱数 R_i を各暗号文中の平文に加法準同型性を用いて加える．それらを I に送信する．

$$\mathsf{Enc}_I(P(\bar{C}_R^i)) \cdot \mathsf{Enc}_I(R_i) = \mathsf{Enc}_I(P(\bar{C}_R^i) + R_i). \tag{3.46}$$

4. I は暗号文をすべて復号し，K で暗号化して R に送信する．

5. R は暗号文をすべて復号し，$P(\bar{C}_R^i) + R_i \stackrel{?}{=} R_i$ を検証することで $\bar{C}_R \cap \bar{C}_I$ を求め，K で暗号化して I に送信する．

6. I は，復号して $\bar{C}_R \cap \bar{C}_I$ を得る．

3.6.4 L2P

入力： $\bar{C}_R, \bar{C}_I, \{\Psi_{R \leftarrow I}^i\}, \Psi_{R_\tau}$

出力： $\bar{C}_R \cap \bar{C}_I$/False

1. I, R は共通鍵 K を共有する．さらに，I, R は加法準同型暗号の秘密鍵と公開鍵を生成する．

2. I は，多項式 $P_I(x)$ の係数を自分の秘密鍵ですべて暗号化して R に送信する．

3. R は乱数 D_R, ρ_i, R_i $(i \in \{1, 2, \cdots, |\bar{C}_R|\})$ を生成して，次のような 2 つの暗号文を秘匿計算する．そして，2 つの暗号文と R_i を I に送信する．

$$\mathsf{Enc}_I(\rho_i \cdot P(\bar{C}_R^i)) = \mathsf{Enc}_I(P(\bar{C}_R^i))^{\rho_i}, \tag{3.47}$$

$$\mathsf{Enc}_I(P(\bar{C}_R^i) + \mathsf{Enc}_R(\Psi^i_{R\leftarrow I} \cdot D_R)) = \mathsf{Enc}_I(P(\bar{C}_R^i)) \cdot \mathsf{Enc}_I(\mathsf{Enc}_R(\Psi^i_{R\leftarrow I} \cdot D_R)). \tag{3.48}$$

4. I は，まず 1 つ目の暗号文を復号して $\rho_i \cdot P(\bar{C}_R^i) = 0$ を満たすインデックスを求め，その集合を $\mathcal{J}$ とする．そして，2 つ目の暗号文の内 $\mathcal{J}$ に含まれるインデックスのみの暗号文を復号する．すなわち，$j \in \mathcal{J}$ とするとき，I は $\mathsf{Enc}_R(\Psi^j_{R\leftarrow I} \cdot D_R)$ を得る．このようにして求めた暗号文を R に送信する．

5. R はすべて復号し，インデックスから $\bar{C}_R \cap \bar{C}_I$ を求める．次のように平文の和と閾値の大小を比較して，閾値以上ならば $\bar{C}_R \cap \bar{C}_I$ を K で暗号化して I に送信する．そうでなければ，False を送信する．

$$\sum_{j\in\mathcal{J}} \Psi^j_{R\leftarrow I} \cdot D_R \gtreqless \Psi_{R_\tau} \cdot D_R. \tag{3.49}$$

6. I は，$\mathsf{Enc}_K(\bar{C}_R \cap \bar{C}_I)$ を受け取ったらマッチであり，復号して $\bar{C}_R \cap \bar{C}_I$ を求める．False を受け取ったらミスマッチである．

3.6.5 L3P

入力： $\bar{C}_R, \bar{C}_I, \{\Psi^i_{R\leftarrow I}\}, \{\Psi^i_{I\leftarrow R}\}, \Psi_{R_\tau}, \Psi_{I_\tau}$

出力： True/False

1. I, R は加法準同型暗号の秘密鍵と公開鍵を生成する．

2. I は乱数 D_I を生成して，R に $\mathsf{Enc}_I(P_I(x)), \mathsf{Enc}_I(D_I)$ を送信する．

3. R は乱数 D_R を生成して，I に $\mathsf{Enc}_R(P_R(x)), \mathsf{Enc}_R(D_R)$ を送信する．

4. I は乱数 ρ'_i を生成して，次のように 2 つの暗号文を秘匿計算する．そして，2 つの暗号文と $\mathsf{Enc}_I(\Psi_{I_\tau} \cdot D_I)$ を R に送信する．

$$\mathsf{Enc}_R(\rho'_i \cdot P_R(\bar{C}_I^i)) = \mathsf{Enc}_R(P_R(\bar{C}_I^i))^{\rho'_i}, \tag{3.50}$$

$$\mathsf{Enc}_R(\rho'_i \cdot P_R(\bar{C}_I^i) + \mathsf{Enc}_I(\Psi^i_{I\leftarrow R} \cdot D_I)) = \mathsf{Enc}_R(\rho'_i \cdot P_R(\bar{C}_I^i)) \cdot \mathsf{Enc}_R(\mathsf{Enc}_I(\Psi^i_{I\leftarrow R} \cdot D_I)). \tag{3.51}$$

5. R は乱数 ρ_i を生成して，次のように 2 つの暗号文を秘匿計算する．そして，2 つの暗号文と $\mathsf{Enc}_R(\Psi_{R_\tau} \cdot D_R)$ を I に送信する．

$$\mathsf{Enc}_I(\rho_i \cdot P_I(\bar{C}_R^i)) = \mathsf{Enc}_I(P_I(\bar{C}_R^i))^{\rho_i}, \tag{3.52}$$

$$\mathsf{Enc}_I(\rho_i \cdot P_I(\bar{C}_R^i) + \mathsf{Enc}_R(\Psi_{R\leftarrow I}^i \cdot D_R)) = \mathsf{Enc}_I(\rho_i \cdot P_I(\bar{C}_R^i)) \cdot \mathsf{Enc}_I(\mathsf{Enc}_R(\Psi_{R\leftarrow I}^i \cdot D_R)). \tag{3.53}$$

6. I は，L2P のように共通項目のインデックスの集合 $\mathcal{J}_I$ を求める．そして，乱数 r_1', r_2' を生成して，$j \in \mathcal{J}_I$ において次のように秘匿計算する．

$$\mathsf{Enc}_R(r_1' \cdot \Psi_{R\leftarrow I}^j \cdot D_R + r_2') = \mathsf{Enc}_R(\Psi_{R\leftarrow I}^j \cdot D_R)^{r_1'} \cdot \mathsf{Enc}_R(r_2'), \tag{3.54}$$

$$\mathsf{Enc}_R(r_1' \cdot \Psi_{R_\tau} \cdot D_R + r_2') = \mathsf{Enc}_R(D_R \cdot \Psi_{R_\tau})^{r_1'} \cdot \mathsf{Enc}_R(r_2'). \tag{3.55}$$

7. R は，L2P のように共通項目のインデックスの集合 $\mathcal{J}_R$ を求める．そして，乱数 r_1, r_2 を生成して，$j \in \mathcal{J}_R$ において次のように秘匿計算し I に送信する．

$$\mathsf{Enc}_I(r_1 \cdot \Psi_{I\leftarrow R}^j \cdot D_I + r_2) = \mathsf{Enc}_I(\Psi_{R\leftarrow I}^j \cdot D_I)^{r_1} \cdot \mathsf{Enc}_I(r_2), \tag{3.56}$$

$$\mathsf{Enc}_I(r_1 \cdot \Psi_{I_\tau} \cdot D_I + r_2) = \mathsf{Enc}_I(D_I \cdot \Psi_{I_\tau})^{r_1} \cdot \mathsf{Enc}_I(r_2). \tag{3.57}$$

8. I は，すべての暗号文を復号して平文の和が閾値以上か否かを検証する．閾値以上ならば True であり，そうでなければ False である．

$$\sum_{j\in\mathcal{J}_I}(r_1 \cdot \Psi_{I\leftarrow R}^j \cdot D_I + r_2) \gtreqless r_1 \cdot \Psi_{I_\tau} \cdot D_I + r_2 \tag{3.58}$$

9. R は，すべての暗号文を復号して平文の和が閾値以上か否かを検証する．閾値以上ならば True であり，そうでなければ False である．

$$\sum_{j\in\mathcal{J}_R}(r_1' \cdot \Psi_{R\leftarrow I}^j \cdot D_R + r_2') \gtreqless r_1' \cdot \Psi_{R_\tau} \cdot D_R + r_2' \tag{3.59}$$

3.7 ZLJCL14 [32]

この秘匿マッチングは，多肢選択式のプロフィールを用いた類似度ベースのマッチングである．まず各ユーザは，プロフィール行列を次のようにつくる．例えば，ユーザ A がつくる行列を $\mathcal{A} = (a_{ij})$ とするとき，列 j はプロフィールの質問に相当し，行 i は好みの度合いに相当する．好みの度合いの最大値を l とし，質問の項目数を n とするとき，

$$\forall i \in \mathbb{N} \cap [1, l], \quad \forall j \in \mathbb{N} \cap [1, n]. \tag{3.60}$$

Algorithm 1: Initialization algorithm for private configuration

Input : *Initiator*'s attribute matrix $A_{l\times n}$
Output: Encrypted matrix $A^{\star}_{l\times n}$

Choose two large primes p, q, where $|p| = 256$, $q > (n+1)l^2p^2$;
Randomly generate two matrixes $C_{l\times n}$ and $R_{l\times n}$, $\forall c_{ij} \in C_{l\times n}$, $\forall r_{ij} \in R_{l\times n}$, $\sum_{i=1}^{l}(\sum_{j=1}^{n} c_{ij}) < (p - ln)$, $|r_{ij}q| \approx 1024$;
$\forall a_{ij} \in A_{l\times n}$, $\forall a^{\star}_{ij} \in A^{\star}_{l\times n}$, $k_i \in \overrightarrow{K}$, do:
for $(i = 1;\ i \le l;\ i++)$ **do**
    $k_i = 0$;
    for $(j = 1; j \le n; x++)$ **do**
        if $a_{ij} = 1$ **then**
            $a^{\star}_{ij} = p + c_{ij} + r_{ij}q$;
        else
            $a^{\star}_{ij} = c_{ij} + r_{ij}q$;
        end
        $k_i = k_i + (r_{ij}q - c_{ij})$;
    end
end

具体的に $j = 2$ の質問を「映画はどのくらい好きか?」とすると，a_{i2} のいずれかに 1 を代入し，それ以外の列には 0 を代入する．i が大きければ大きいほど優先度が高いことを意味する．同様に，ユーザ B もプロフィール行列 $\mathcal{B} = (b_{ij})$ をつくる．つまり，プロフィール行列は要素が $\{0, 1\}$ から成る $n \times l$ 行列である．

A は，Confusion Matrix Transformation(CMT) と呼ばれている Algorithm 1 によってプロフィール行列 $\mathcal{A}$ を秘匿して行列 $\mathcal{A}^*$ と復号に使う鍵 $\overrightarrow{K}$ を得る．同様に，B も Algorithm 1 によって $\mathcal{B}$ を秘匿して $\mathcal{B}^*$ を得る．

A は B に $\mathcal{A}^*$ を送り，B は Algorithm 2 によって $\mathcal{A}^*$ と $\mathcal{B}$ から行列 $\mathcal{D}$ を得る．B は，それを A に返して $\mathcal{D}$ と復号に使う鍵 $\overrightarrow{K}$ から行列 $\mathcal{T}^*$ をつくる．次のような重み行列 $\mathcal{W} = (w_{ij})$ を $\mathcal{T}^*$ に掛けて行列 $\mathcal{H} = (h_{ij})$ をつくる．そして，各要素の合計 $\tau = \sum\sum h_{ij}$ を計算する．

$$w_{ij} = \begin{cases} i & (i = j), \\ i - |i - j| & (i - |i - j| > 1), \\ 1 & (i - |i - j| \le 1). \end{cases} \tag{3.61}$$

最後に，A は τ と自分の閾値 τ_A とを比較して $\tau \ge \tau_A$ ならば True を，$\tau < \tau_A$ ならば False を B に送る．

以上のように，類似度ベースのマッチングが行われるが，A は $\mathcal{D}$ から容易に $\mathcal{B}$ を特定することができてしまう．なぜならば，

$$(\mathcal{A}^*)^{-1} \times \mathcal{D} = (\mathcal{A}^*)^{-1} \times \mathcal{A}^* \times \mathcal{B} = \mathcal{B}. \tag{3.62}$$

この課題を解決するために，さらに Algorithm 3 を提案している．Algorithm 3 では，B が重み行列 $\mathcal{W}$ との掛け算を行い，さらに行列の各要素の和を計算するため，A は $\mathcal{B}$ を推測することはできない．

Algorithm 2: EWPM for achieving the **Level-I privacy**

Input : $A^{\star}_{l\times n}$, $B_{l\times n}$
Output: The match value τ

Compute $D=(d_{ij})_{l\times l}=A^{\star}_{l\times n}\times B^{\mathrm{T}}_{l\times n}$ following the operations bellow:
for $(i=1; i\leq l; i++)$ **do**
  for $(j=1; j\leq l; j++)$ **do**
    $d_{ij}=0$;
    for $(x=1; x\leq n; x++)$ **do**
      if $(b_{ix}=1)$ **then**
        $d_{ij}=d_{ij}+p\cdot a^{\star}_{ix}$;
      else
        $d_{ij}=d_{ij}+a^{\star}_{ix}$;
      end
    end
  end
end
The *responder* sends computing results $D_{l\times l}$ to the *initiator*;
The *initiator* computes $T=(t_{ij})_{l\times l}$, $t_{ij}=(d_{ij}+k_i) \mod q$, for $d_{ij}\in D_{l\times l}$;
The *initiator* makes a further transformation to get $T^{\star}=(t^{\star}_{ij})_{l\times l}$ where $t^{\star}_{ij}=\frac{t_{ij}-(t_{ij} \mod p^2)}{p^2}$;
The *initiator* considers the corresponding weights and computes $H_{l\times l}$ according to Formula-2: $H_{l\times l}=T^{\star}_{l\times l}\cdot W_{l\times l}$;
Up to now, we can get the match value:
$\tau=\sum_{i=1}^{l}\sum_{j=1}^{l}h_{ij}$.

Algorithm 3: EWPM for achieving the **Level-II privacy**

Input : After the *responder* computes $D_{l\times l}$
Output: The match value τ

The *responder* computes $T=(t_{ij})_{l\times l}=D_{l\times l}\cdot W_{l\times l}$ according to Formula-2 and we get $t_{ij}=d_{ij}w_{ij}$;
Calculate $\delta=\sum_{i=1}^{l}\sum_{j=1}^{l}t_{ij}$;
Send δ to the *initiator*;
Upon receiving the message δ, the *initiator* decrypts the matching value via the following operators:
$\tau_1=(\delta+l(\sum_{i=1}^{l}k_i)) \mod q$;
$\tau=\frac{\tau_1-(\tau_1 \mod p^2)}{p^2}$.

3.8 従来の秘匿マッチングにおける問題の整理

これまでに提案されている閾値を用いたプロトコルには，共通項目数や類似度を秘密計算した後，それと閾値との大小比較結果を相手に通知するため，共通項目数や比較結果などのマッチングの結果を知ることができるという問題があることを前に述べた．

表 3.3: 関連研究と 2 つの課題

	課題 1	課題 2	サーバ	備考
FNP04 [8]	×	△	無	初の秘匿マッチング
IO16 [12]	◯	×	有	条件付きの秘匿マッチング
LUB14 [18]	×	△	有	不正なユーザを検出できる
SX15 [25]	×	×	有	サーバはパラメタをつくるだけ
TLSL15 [27]	×	△	無	類似度によるマッチング
ZDLG13 [31]	◯	×	有	結託耐性のある秘匿マッチング
ZLJCL14 [32]	×	△	無	類似度によるマッチング

例えば，表 3.3[1] のような閾値を用いたプロトコルは，[8,18,27,32] などいくつか提案されているが，どれも共通項目数あるいは類似度を秘密計算した後，それと閾値との大小比較結果を相手に通知するため，共通項目数や通知内容を知ることができてしまう．そこで，共通項目数を秘匿してプライバシーを守り，閾値を超えたか否かなどのマッチングの結果を追跡不可能にする必要がある．

また，[31] には，共通項目数と事前に選択する理想の共通項目数が一致しなければミスマッチと判定されるという問題があることも前に述べた．

整理すると，既存研究の秘匿マッチングには主に次の 2 つの課題が存在する．

1. 共通項目数や類似度，あるいは閾値を超えたか否かなどのマッチングの結果につながる部分情報がマッチングの途中で漏れないようにすること

2. 互いに設定する閾値を同時に満たすことをマッチの条件にすること

そこで，本研究では上記の 2 つの課題を解決し，さらにマッチングの機能性の向上のため，マッチした場合に限り共通項目を知ることができる秘匿マッチングプロトコルを提案する．ただし，ユーザや運営サーバは semi-honest モデルとし，ユーザはプロフィールなどのプライバシー情報を自己管理すると仮定する．

[1] 凡例：◯ · · · 解決できている，△ · · · 仮定をつけることで解決，× · · · 解決または配慮していない

第4章　提案手法

この章では，本研究で提案する二者間の秘匿マッチングの全体像とそれを構成する各プロトコルについて説明する．秘匿マッチングは，図 4.1 のように 3 つのプロトコルを合計 5 回実行することで行われる．まず，秘匿マッチングの全体像を説明し，次に [31] の BVT プロトコルをマッチの場合に限り共通項目を求められるように改良したプロトコル，そして閾値判定プロトコル，さらに共通項目を求めるプロトコルを説明する．

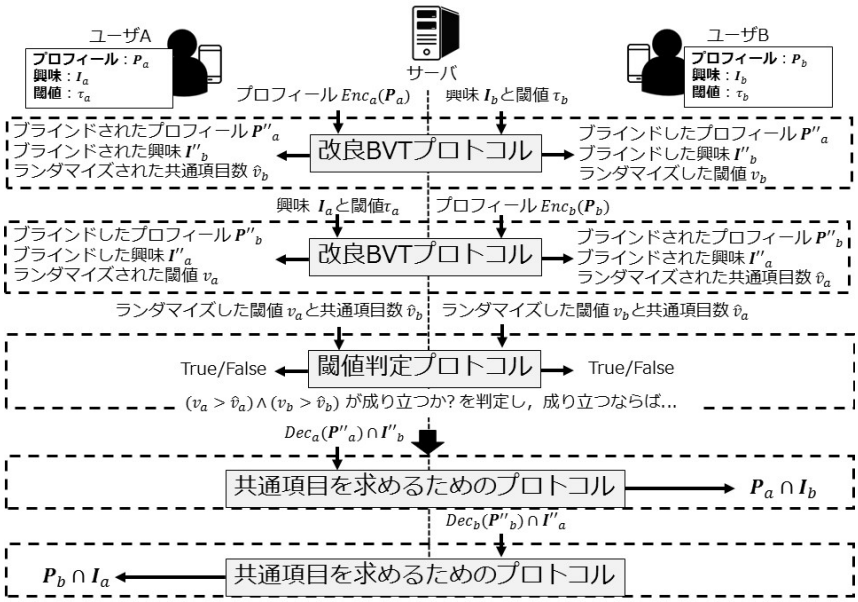

図 4.1: 秘匿マッチングの全体像

4.1　秘匿マッチングの全体像

この秘匿マッチングでは，図 4.1 のように 2 回の改良 BVT プロトコルと 1 回の閾値判定プロトコル，さらにマッチの場合にのみ 2 回の共通項目を求めるプロトコルによってマッチングが行われる．改良 BVT プロトコルでブラインドされた興味とプロフィールの共通項目を求め，閾値判定プロトコルで自分が期待する共通項目数以上か否かを判定す

る．マッチングの判定は，以前の閾値判定プロトコルを用いる．さらにマッチの場合，共通項目を求めるプロトコルによって自分の興味と相手のプロフィールの共通項目をインデックステーブルを用いて求める．

Setting

A は，プロフィール $\boldsymbol{P}_a$ と興味 $\boldsymbol{I}_a$ の他に自分の興味と相手のプロフィールの共通項目数の閾値 τ_a と乱数 $\alpha_a, \beta_a, \gamma_a, \xi_a$，セキュリティパラメタ l_a，そしてダミーの項目とプロフィールの項目を識別し，正しく共通項目を求めるために 2 つのインデックステーブル $\boldsymbol{\eta}_a := [\eta_a^{(1)}, \eta_a^{(2)}, \cdots, \eta_a^{(n)}], \boldsymbol{\eta}'_a := [\eta'^{(1)}_a, \eta'^{(2)}_a, \cdots, \eta'^{(n)}_a]$ を用意する．同様に，B もプロフィール $\boldsymbol{P}_b$ と興味 $\boldsymbol{I}_b$ の他に自分の興味と相手のプロフィールの共通項目数の閾値 τ_b と乱数 $\alpha_b, \beta_b, \gamma_b, \xi_b$，セキュリティパラメタ l_b，そして正しく共通項目を求めるために 2 つのインデックステーブル $\boldsymbol{\eta}_b := [\eta_b^{(1)}, \eta_b^{(2)}, \cdots, \eta_b^{(n)}], \boldsymbol{\eta}'_b := [\eta'^{(1)}_b, \eta'^{(2)}_b, \cdots, \eta'^{(n)}_b]$ を用意する．そして，A は $\mathsf{Enc}_a(\boldsymbol{P}_a)$ を，B は $\mathsf{Enc}_b(\boldsymbol{P}_b)$ をサーバにアップロードする．

Private Matching

A の $\boldsymbol{P}_a$ と B の $\boldsymbol{I}_b, l_b, \tau_b, \boldsymbol{\eta}_b, \xi_b$ を入力として改良 BVT プロトコルを実行し，A は $\hat{v}_b$ を，B は v_b を出力として受け取る．v_b は τ_b に $\Delta_b(l_b)$ を加えてランダマイズした値であり，$\Delta_b(l_b)$ は B のみが知っている値である．したがって，A は $\hat{v}_b$ から $|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ を求めることはできない．同様に，B の $\boldsymbol{P}_b$ と A の $\boldsymbol{I}_a, l_a, \tau_a, \boldsymbol{\eta}_a, \xi_a$ を入力として改良 BVT プロトコルを実行し，B は $\hat{v}_a$ を，A は v_a を出力として受け取る．v_a は τ_a に $\Delta_a(l_a)$ を加えてランダマイズした値であり，$\Delta_a(l_a)$ は A のみが知っている値である．したがって，B は $\hat{v}_a$ から $|\boldsymbol{P}_b \cap \boldsymbol{I}_a|$ を求めることはできない．

Verification

A の $v_a, \hat{v}_b$ と B の $v_b, \hat{v}_a$ を入力として閾値判定プロトコルを実行し，運営サーバは $v_a \geq \hat{v}_a \wedge v_b \geq \hat{v}_b$ が成り立つか否かを判定し A と B に結果として True/False を渡す．$\alpha_a, \beta_a, \gamma_a$ や $\alpha_b, \beta_b, \gamma_b$ は，プロトコルの途中で $v_a \geq \hat{v}_a \wedge v_b \geq \hat{v}_b$ を追跡することができないようにするために用いる．

共通項目を求める

A の $\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}_b$ と B の $\boldsymbol{\eta}_b, \boldsymbol{\eta}'_b, \xi_b$ を入力として共通項目を求めるためのプロトコルを実行し，B は $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を出力として受け取る．同様に，B の $\mathsf{Dec}_b(\boldsymbol{P}''_b) \cap \boldsymbol{I}_a$ と A の $\boldsymbol{\eta}_a, \boldsymbol{\eta}'_a, \xi_a$ を入力として共通項目を求めるためのプロトコルを実行し，A は $\boldsymbol{P}_b \cap \boldsymbol{I}_a$ を出力として受け取る．

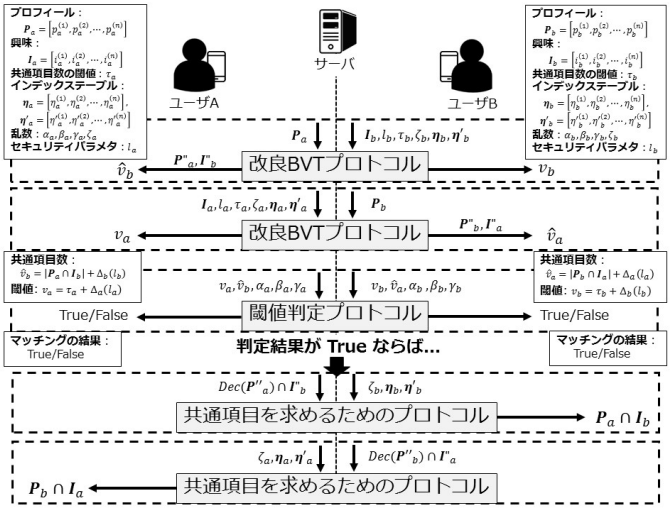

図 4.2: 詳細な秘匿マッチングの全体像

4.2 改良 BVT プロトコル

4.2.1 改良の目的

まず，BVT プロトコルを改良する目的について述べる．[31] のマッチングでは，結果として得られるのはマッチしたか否かだけでどの項目が一致したのかを求めることはできない．マッチまたはミスマッチ以外の情報が得られないので，プライバシーを保護しながらマッチングを行うという要件は満たされたが，それだけでは機能的に乏しい．そこで，マッチならば共通項目を求めることができ，ミスマッチならばそれ以外の情報を一切得られないというマッチングを提案する．そのために，ブラインド操作の中でインデックステーブルを埋め込み，後でインデックステーブルの一部または全部を正しく求めることで，正しく共通項目を求められるように BVT プロトコルを改良する．次節では，インデックステーブルという概念を導入する．

4.2.2 インデックステーブルの導入

インデックステーブルとは，次のように定義され，各要素は表 4.1 のように順番と対応付けされる．

$$\boldsymbol{\eta}_b := [\eta_b^{(1)}, \eta_b^{(2)}, \cdots, \eta_b^{(n)}], \quad \eta_b^{(i)} \neq \eta_b^{(j)} \ (i \neq j). \tag{4.1}$$

表 4.1: インデックステーブル

No.	1	2	$\cdots$	n
$\boldsymbol{P}_b$	$p_b^{(1)}$	$p_b^{(2)}$	$\cdots$	$p_b^{(n)}$
$\boldsymbol{I}_b$	$i_b^{(1)}$	$i_b^{(2)}$	$\cdots$	$i_b^{(n)}$
$\boldsymbol{\eta}_b$	$\eta_b^{(1)}$	$\eta_b^{(2)}$	$\cdots$	$\eta_b^{(n)}$

インデックステーブルには，項目のインデックスをランダムな値に対応させることで，自分以外には真のインデックスを追跡させないようにする役目がある．インデックステーブルの一部または全部を求めることで，B のみが何番目の項目が一致したのかを把握することができる．そのために，$\mathrm{Dec}_a(\boldsymbol{P}_a'') \cap \boldsymbol{I}_b''$ からインデックステーブルの一部の要素が得られるように BVT プロトコルを改良する．すなわち，次節のように $\boldsymbol{P}_a''$ や $\boldsymbol{I}_b''$ を計算する過程を改良する．

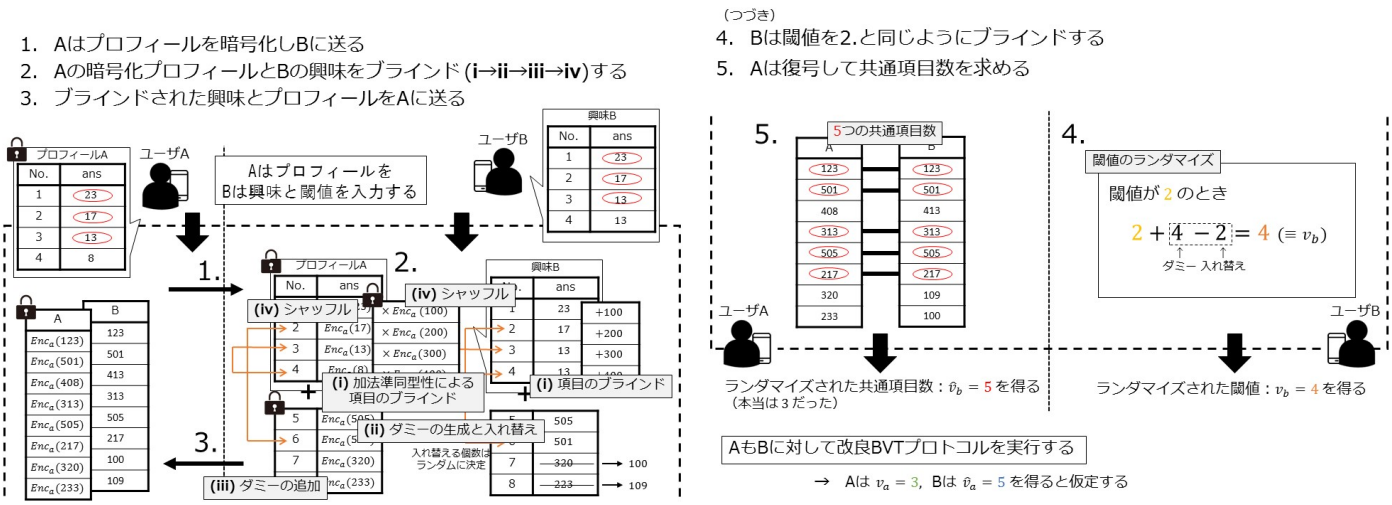

図 4.3: 改良 BVT プロトコルの全体像

4.2.3 プロトコルの詳細

入力： A の公開鍵で暗号化された A のプロフィール $\mathrm{Enc}_a(\boldsymbol{P}_a)$，$B$ の興味 $\boldsymbol{I}_b$，B のセキュリティパラメタ l_b，B のインデックステーブル $\boldsymbol{\eta}_b$，ダミーのインデックステーブル $\boldsymbol{\eta}_b'$，B の乱数 ξ_b.

出力： 共通項目数の閾値をランダマイズした値 v_b，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ をランダマイズした値 $\hat{v}_b$.

1. **サーバは A の暗号化プロフィールを B に送る**

サーバは A と B の秘匿マッチングを行うことを決めて，B に $\mathsf{Enc}_a(\boldsymbol{P}_a)$ を送信する．

2. **A の暗号化プロフィールと B の興味をブラインドする**

(i) B は，$\mathsf{Enc}_a(\boldsymbol{P}_a)$ と同じ項目数の乱数ベクトル $\boldsymbol{r}_b$ を生成して，

$$\hat{\boldsymbol{P}}_a := \mathsf{Enc}_a(\xi_b \cdot (\boldsymbol{P}_a + \boldsymbol{r}_b) + \boldsymbol{\eta}_b), \tag{4.2}$$

$$\widetilde{\boldsymbol{I}}_b := \xi_b \cdot (\boldsymbol{I}_b + \boldsymbol{r}_b) + \boldsymbol{\eta}_b. \tag{4.3}$$

ただし，$\xi_b \in \mathbb{Z}_N,\ r_b^{(i)} \in \mathbb{Z}_N,\ {\eta_b}^{(i)} \in \mathbb{Z}_N,\ {\eta_b}^{(i)} < \xi_b,\ \widetilde{I}_b^{(i)} < N$ を満たす．

(ii) l_b 個のダミーのベクトル $\xi_b \cdot \boldsymbol{y}_b + \boldsymbol{\eta}'_b$ を生成し，$\hat{\boldsymbol{y}}_b := \mathsf{Enc}_a(\xi_b \cdot \boldsymbol{y}_b + \boldsymbol{\eta}'_b)$ とする．さらに，$\xi_b \cdot \boldsymbol{y}_b + \boldsymbol{\eta}'_b$ の乱数 k_b 個の項目を $\mathbb{Z}_N$ 上の別の値に入れ替えて新しいベクトル $\widetilde{\boldsymbol{y}}_b$ をつくる．ただし，$\boldsymbol{\eta}_b$ と $\boldsymbol{\eta}'_b$ の各要素はすべて異なり，かつ ${\eta'_b}^{(i)} \in \mathbb{Z}_N,\ {\eta'_b}^{(i)} < \xi_b$ を満たす．また，$k_b \overset{U}{\leftarrow} \{1, 2, \cdots, l_b\}$．

(iii) $\hat{\boldsymbol{P}}_a, \widetilde{\boldsymbol{I}}_b$ にそれぞれ $\hat{\boldsymbol{y}}_b, \widetilde{\boldsymbol{y}}_b$ を追加し拡張する．

$$\boldsymbol{P}'_a := \hat{\boldsymbol{P}}_a \oplus \hat{\boldsymbol{y}}_b, \quad \boldsymbol{I}'_b := \widetilde{\boldsymbol{I}}_b \oplus \widetilde{\boldsymbol{y}}_b. \tag{4.4}$$

(iv) $\boldsymbol{P}'_a$ と $\boldsymbol{I}'_b$ を同じようにシャッフルする．

$$\boldsymbol{P}''_a := \mathrm{Shuffle}\left[\boldsymbol{P}'_a\right], \quad \boldsymbol{I}''_b := \mathrm{Shuffle}\left[\boldsymbol{I}'_b\right]. \tag{4.5}$$

3. **ブラインドされた興味とプロフィールを A に送る**

B は，$\boldsymbol{P}''_a$ と $\boldsymbol{I}''_b$ をサーバを介して A に送信する．

4. **B は閾値を *2* と同じようにブラインドする**

B は，$\boldsymbol{P}_a$ と $\boldsymbol{I}_b$ の共通項目数の閾値 τ_b を ***2*** の *(i)–(iv)* と同じようにブラインドする．

$$v_b := \tau_b + l_b - k_b. \tag{4.6}$$

5. **A は復号して共通項目数を求める**

A は，$\boldsymbol{P}''_a$ を復号し，$\boldsymbol{I}''_b$ との共通項目数を求め，それを $\hat{v}_b$ とする．

$$\hat{v}_b := |\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b|. \tag{4.7}$$

4.3 閾値判定プロトコル

法 N のビット長を t_N とする．A, B は，乱数 $\alpha_i, \beta_i, \gamma_i\ (i = a, b)$ を次のようにオフラインで生成する．

$$\alpha_i \in \mathbb{Z}_N, \beta_i \in \mathbb{Z}_N, \gamma_i \in \mathbb{Z}_N, \tag{4.8}$$

$$\alpha_i \cdot (n + l_i) + \beta_i < N - 2^{t_N - 1},\ \alpha_i > \beta_i > 0 \tag{4.9}$$

改良 BVT プロトコルによって，ランダマイズされた閾値と，自分の興味と相手のプロフィールの共通項目数をランダマイズした値のペアをそれぞれ得る．すなわち，A は $(v_a, \hat{v}_b)$ を，B は $(\hat{v}_a, v_b)$ を得たとする．このとき，$v_a \leq \hat{v}_a \wedge v_b \leq \hat{v}_b$ を満たすならば，$\tau_a \leq |\boldsymbol{P}_b \cap \boldsymbol{I}_a| \wedge \tau_b \leq |\boldsymbol{P}_a \cap \boldsymbol{I}_b|$ を満たすので A と B はマッチする．閾値判定プロトコルでは，プライバシー情報 $v_a, v_b, \hat{v}_a, \hat{v}_b$ を漏らさずに $v_a \leq \hat{v}_a \wedge v_b \leq \hat{v}_b$ が成り立つか否かをサーバが判定する．まず，$\mathbb{Z}_N$ を正の範囲 $(0, 2^{t_N-1})$ と負の範囲 $[2^{t_N-1}, N)$ に分ける．そして，閾値との差が負の数ならばビット長が t_N なので，サーバは $v_a \leq \hat{v}_a \wedge v_b \leq \hat{v}_b$ が成り立つか否かをビット長を計算することによって判定する．

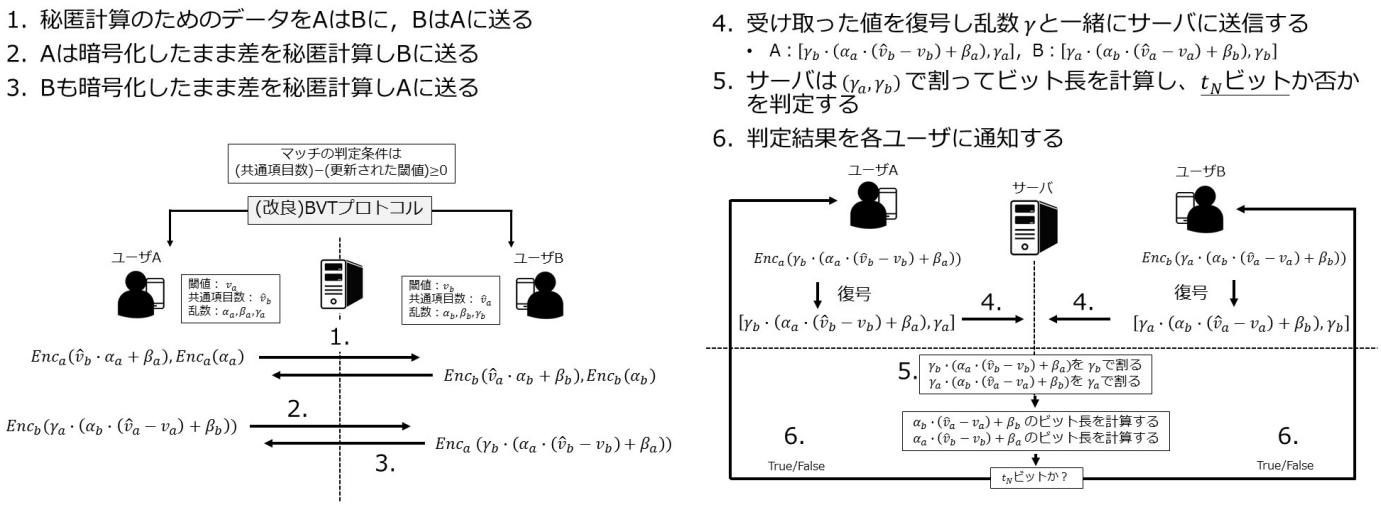

図 4.4: 閾値判定プロトコルの全体像

4.3.1 プロトコルの詳細

入力： $(v_a, \hat{v}_b), (v_b, \hat{v}_a)$，乱数 $\alpha_a, \alpha_b, \beta_a, \beta_b, \gamma_a, \gamma_b$.

出力： $v_a \leq \hat{v}_a \wedge v_b \leq \hat{v}_b$ が成り立つか否か.

1. **秘匿計算のためのデータを A は B に，B は A に送る**

A は $\mathsf{Enc}_a(\alpha_a \cdot \hat{v}_b + \beta_a), \mathsf{Enc}_a(\alpha_a)$ を B に，B は $\mathsf{Enc}_b(\alpha_b \cdot \hat{v}_a + \beta_b), \mathsf{Enc}_b(\alpha_b)$ を A に送信する.

2. **A は暗号化したまま閾値との差を計算し B に送る**

A は，γ_a, v_a と $\mathsf{Enc}_b(\alpha_b \cdot \hat{v}_a + \beta_b), \mathsf{Enc}_b(\alpha_b)$ を用いて $\mathsf{Enc}_b(\gamma_a \cdot (\alpha_b \cdot (\hat{v}_a - v_a) + \beta_b))$ を秘匿計算し，それを B に送信する.

$$\begin{aligned} &\left\{\mathsf{Enc}_b(\alpha_b \cdot \hat{v}_a + \beta_b) \cdot \mathsf{Enc}_b(\alpha_b)^{-v_a}\right\}^{\gamma_a} \\ &= \mathsf{Enc}_b(\gamma_a \cdot (\alpha_b \cdot (\hat{v}_a - v_a) + \beta_b)). \end{aligned} \tag{4.10}$$

3. B は暗号化したまま閾値との差を計算し A に送る

B は，γ_b, v_b と $\mathsf{Enc}_a(\alpha_a \cdot \hat{v}_b + \beta_a), \mathsf{Enc}_a(\alpha_a)$ を用いて $\mathsf{Enc}_a(\gamma_b \cdot (\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a))$ を秘匿計算し，それを A に送信する．

$$\begin{aligned} &\left\{\mathsf{Enc}_a(\alpha_a \cdot \hat{v}_b + \beta_a) \cdot \mathsf{Enc}_a(\alpha_a)^{-v_b}\right\}^{\gamma_b} \\ &\quad = \mathsf{Enc}_a(\gamma_b \cdot (\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a)). \end{aligned} \tag{4.11}$$

4. 受け取った値を復号し γ_i と一緒にサーバに送る

暗号文を復号し，A は $[\gamma_a, \gamma_b \cdot (\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a)]$ を，B は $[\gamma_b, \gamma_a \cdot (\alpha_b \cdot (\hat{v}_a - v_a) + \beta_b)]$ をサーバに送信する．

5. サーバはビット長を計算し t_N ビットかを判定する

サーバは，受け取ったデータから次のように $\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a,\ \alpha_b \cdot (\hat{v}_a - v_a) + \beta_b$ を計算する．そして，それらのビット長を計算し t_N ビットか否かを判定する．

$$\gamma_b \cdot (\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a) \cdot \gamma_b^{-1} = \alpha_a \cdot (\hat{v}_b - v_b) + \beta_a, \tag{4.12}$$

$$\gamma_a \cdot (\alpha_b \cdot (\hat{v}_a - v_a) + \beta_b) \cdot \gamma_a^{-1} = \alpha_b \cdot (\hat{v}_a - v_a) + \beta_b. \tag{4.13}$$

6. 判定結果を各ユーザに通知する

どちらか一方でも t_N ビットならば A と B に False を送信する．それ以外の場合は True を送信する．

4.4 共通項目を求めるためのプロトコル

A と B はマッチと判定されたと仮定する．すでに，A はブラインドされた B の興味との共通項目 $\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b$ を持っており，B は 2 つのインデックステーブル $\boldsymbol{\eta}_b, \boldsymbol{\eta}'_b$ と乱数 ξ_b を持っている．これらを用いて B は，次のように A のプロフィールと B の興味の共通項目 $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を求める．同様に $\mathsf{Dec}_b(\boldsymbol{P}''_b) \cap \boldsymbol{I}''_a$ とインデックステーブル $\boldsymbol{\eta}_a, \boldsymbol{\eta}'_a$，そして乱数 ξ_a を用いれば，A も B のプロフィールと A の興味の共通項目 $\boldsymbol{P}_b \cap \boldsymbol{I}_a$ を求めることができる．

4.4.1 プロトコルの詳細

入力 $\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b$，インデックステーブル $\boldsymbol{\eta}_b$，改良 BVT プロトコルで使った乱数 ξ_b.

出力 $\boldsymbol{P}_a \cap \boldsymbol{I}_b$.

1. B は ξ_b を A の公開鍵で暗号化して A に送る

2. A は ξ_b に対する $\boldsymbol{I}''_b \cap \mathrm{Dec}_a(\boldsymbol{P}''_a)$ の剰余を求める

$\hat{\boldsymbol{I}}_b := \boldsymbol{I}''_b \cap \mathrm{Dec}_a(\boldsymbol{P}''_a)$ とすると，

$$\hat{I}_b^{(i)} \equiv \begin{cases} \eta_b^{(i)} & (\mathrm{mod}\ \xi_b), \\ \eta'^{(i)}_b & (\mathrm{mod}\ \xi_b), \\ \eta & (\mathrm{mod}\ \xi_b). \end{cases} \tag{4.14}$$

ただし，η は $\boldsymbol{\eta}_b, \boldsymbol{\eta}'_b$ の如何なる要素とも等しくない．

3. A はすべての剰余を B の公開鍵で暗号化して送る

4. B は暗号文をすべて復号し剰余を求める

5. 剰余とインデックステーブルから共通項目を求める

B は，受け取った剰余が自分のインデックステーブル $\boldsymbol{\eta}_b$ の何番目の要素と等しいのかをひとつひとつ確認することによってインデックスを得る．

1. Bは ξ_b を暗号化してAに送る
2. Aは復号して，ブラインドされた興味との共通項目 $Dec_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b$ を ξ_b で割った剰余を求める
3. Aはすべての剰余を暗号化してBに送る
4. Bは暗号文をすべて復号し剰余を求める
5. Bは剰余とインデックステーブルから共通項目を求める

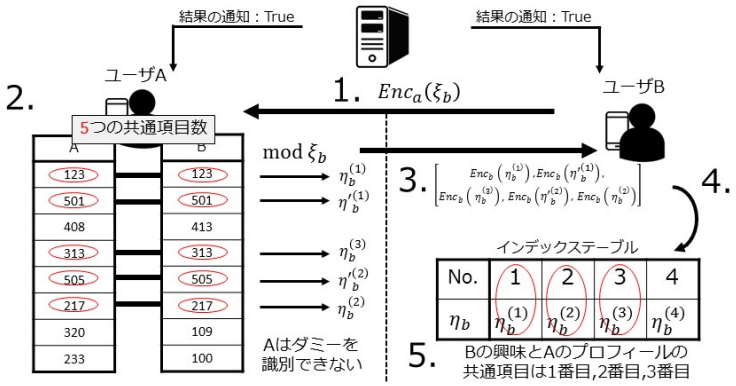

図 4.5: B が A のプロフィールと B の興味の共通項目 $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を求める例．

第5章　評価

この章では，前章で提案した3つのプロトコルの正当性と秘匿性，そして効率性の観点から評価を行い，最後にタブレット端末で実装評価を行う．

5.1　正当性

定理 6. 改良BVTプロトコルは，$l_b = k_b$ のとき正しく共通項目数を求めることができる．

proof. BVTプロトコルの正当性 [31](定理2) により正しく共通項目数を求めることができる． □

定理 7. 閾値判定プロトコルは，次の式 (5.1) を満たすように乱数 α_i, β_i を生成するとき，マッチングの結果を正しく出力する．$\alpha_i \in \mathbb{Z}_N, \beta_i \in \mathbb{Z}_N, \gamma_i \in \mathbb{Z}_N$ のとき，

$$\alpha_i \cdot (n + l_i) + \beta_i < N - 2^{t_N - 1}, \quad \alpha_i > \beta_i > 0. \tag{5.1}$$

proof. $\mathbb{Z}_N$ を正の範囲 $(0, 2^{t_N-1})$ と負の範囲 $[2^{t_N-1}, N)$ に分ける．つまり，$\forall x \in \mathbb{Z}_N$ のとき，x が正の数ならば $t_x \neq t_N$ であり，x が負の数ならば $t_x = t_N$ である．したがって，次の2つの式を同時に満たせば，$\alpha_i \cdot (\hat{v}_i - v_i) + \beta_i$ のビット長から $\hat{v}_i \gtreqless v_i$ を判定することができる．

$$\begin{cases} \alpha_i \cdot (\hat{v}_i - v_i) + \beta_i < 2^{t_N - 1}, \\ \alpha_i \cdot (\hat{v}_i - v_i) + \beta_i < N - 2^{t_N - 1}. \end{cases} \tag{5.2}$$

一般的に $2^{t_N-1} \geq N - 2^{t_N-1}$ なので，

$$\alpha_i \cdot (\hat{v}_i - v_i) + \beta_i < N - 2^{t_N - 1} \leq 2^{t_N - 1}. \tag{5.3}$$

を満たすように α_i, β_i を生成するとき，$\hat{v}_i \gtreqless v_i$ を次のように判定することができる．

- $\hat{v}_i \geq v_i$ のとき，$\alpha_i \cdot (\hat{v}_i - v_i) + \beta_i$ は t_N ビットでない．
- $\hat{v}_i < v_i$ のとき，$\alpha_i \cdot (\hat{v}_i - v_i) + \beta_i$ は t_N ビットである．

以上より，$\alpha_i \cdot (\hat{v}_i - v_i) + \beta_i$ が両ユーザともに t_N ビットであるか否かを判定することで，サーバはマッチングの結果を正しく出力することができる． □

補題 8. インデックステーブル $\boldsymbol{\eta}_b$ と乱数 ξ_b を次のように生成すれば，ブラインドされた興味 $\boldsymbol{I}''_b$ から $\boldsymbol{\eta}_b$ のすべての要素を求められる．$i \neq j$ において，

$$\xi_b \in \mathbb{Z}_N,\ \eta_b^{(i)} \in \mathbb{Z}_N,\ 0 < \eta_b^{(i)} < \xi_b,\ \eta_b^{(i)} \neq \eta_b^{(j)}, \tag{5.4}$$

$$\widetilde{I}_b^{(j)} = \xi_b \cdot (i_b^{(j)} + r_b^{(j)}) + \eta_b^{(j)} < N. \tag{5.5}$$

proof. ブラインドされた興味 $\boldsymbol{I}''_b$ を改良 BVT プロトコルで得たとし，そこで使ったインデックステーブルを $\boldsymbol{\eta}_b$，乱数を ξ_b とする．$\boldsymbol{I}''_b$ は，$\widetilde{\boldsymbol{I}}_b$ と $\widetilde{\boldsymbol{y}}_b$ から構成されており，次のような関係が成り立つ．

$$\widetilde{\boldsymbol{I}}_b = \xi_b \cdot (\boldsymbol{I}_b + \boldsymbol{r}_b) + \boldsymbol{\eta}_b. \tag{5.6}$$

それらの要素を $\widetilde{I}_b^{(i)}$ とすると，

$$\widetilde{I}_b^{(i)} \equiv \eta_b^{(i)} \pmod{\xi_b}. \tag{5.7}$$

確かに $\boldsymbol{\eta}_b$ のすべての要素を求めることができる． □

定理 9. 共通項目を求めるためのプロトコルでは，ξ_b と $\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b$ とインデックステーブル $\boldsymbol{\eta}_b$ から共通項目 $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を正しく求めることができる．

proof. 補題 8 より，$\mathsf{Dec}_a(\boldsymbol{P}''_a) \cap \boldsymbol{I}''_b$ からインデックステーブル $\boldsymbol{\eta}_b$ の一部の要素を求めることができる．それらが $\boldsymbol{\eta}_b$ の何番目の要素と等しいのかを確かめることによって共通項目 $\boldsymbol{P}_a \cap \boldsymbol{I}_b$ を求めることができる． □

5.2 秘匿性

定理 10. 改良 BVT プロトコルでは，ユーザは相手のプロフィールや興味を特定することができない．

proof. A が B のプロフィールや興味を特定しようと試みると仮定する．

- プロフィールについて
  プロフィールを特定するためには，Paillier 暗号を解読する必要がある．しかし，Paillier 暗号は IND-CPA 安全なので暗号文からプロフィールを識別することができない．したがって，A は $\mathsf{Enc}_b(\boldsymbol{P}_b)$ から $\boldsymbol{P}_b$ を特定することはできない．

- 興味について
  BVT プロトコルの秘匿性 [31](定理 4) より，相手の興味をすべて特定することも，プロフィールとの共通項目も特定することができない．

以上より，ユーザはプロフィールや興味を特定することができない． □

定理 11. ユーザとサーバを semi-honest モデルと仮定するとき，閾値判定プロトコルではサーバがマッチングの結果を判定する前に誰もそれを追跡することはできない．

proof. A が，$\hat{v}_b \gtrless v_b$ を求めるためには，$\mathsf{Enc}_a(\gamma_b \cdot (\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a))$ から $\mathsf{Enc}_a(\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a)$ を求める問題，または $\gamma_b \cdot (\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a)$ から $\alpha_a \cdot (\hat{v}_b - v_b) + \beta_a$ を求める問題のどちらかを解く必要がある．しかし，A は γ_b を知らないのでどの問題も解けない．したがって，A は $\hat{v}_b \gtrless v_b$ を求められない．また，Paillier 暗号の識別不可能性より，$\mathsf{Enc}_b(\alpha_b \cdot \hat{v}_a + \beta_b)$ から $\hat{v}_a$ を求めることは不可能なので，A は $\hat{v}_a \gtrless v_a$ も求められない． □

定理 12. 共通項目を求めるためのプロトコルでは，共通していない項目まで求めることはできない．

proof. A は $\boldsymbol{\eta}_b$ と $\boldsymbol{\eta}'_b$ を識別することができないので，$\boldsymbol{I}''_b$ からダミーを除いた $\widetilde{\boldsymbol{I}}_b = \xi_b \cdot (\boldsymbol{I}_b + \boldsymbol{r}_b) + \boldsymbol{\eta}_b$ を求めることができない．仮に識別して $\boldsymbol{I}_b + \boldsymbol{r}_b$ を求めることができても，$\boldsymbol{r}_b$ を求めることはできないので，A は $\boldsymbol{I}_b + \boldsymbol{r}_b$ から $\boldsymbol{I}_b$ を求めることはできない．一方，B は $\boldsymbol{\eta}_b$ と比較することでどれが一致しているのかを求めるため，直接的に共通項目を求めているわけではないので，共通していない項目まで求めることはできない． □

5.3 効率性

3 つのプロトコルについて，次のように計算量と通信量の評価を行う．

5.3.1 計算量

表 5.1: 本研究と [31] の計算量のオーダー

	ユーザ	サーバ
改良 BVT	$\mathcal{O}(n + l_a + l_b)$	-
BVT [31]	$\mathcal{O}(n + l_a + l_b)$	-
閾値判定	$\mathcal{O}(1)$	$\mathcal{O}(1)$
FACFM [31]	$\mathcal{O}(1)$	$\mathcal{O}(1)$
共通項目を求める	$\mathcal{O}(\hat{v}_a + \hat{v}_b)$	-

表 5.1 は，改良 BVT プロトコルと閾値判定プロトコル，そして共通項目を求めるためのプロトコルの計算量のオーダーである．[31] のプロトコルの計算量とともに掲載した．[31] と同様で，改良 BVT プロトコルの計算量のオーダーは質問項目数 n に比例するが，閾値判定プロトコルの計算量のオーダーは n に依存しない．また，共通項目を求めるプロトコルの計算量のオーダーは，ランダマイズされた共通項目数に依存する．

5.3.2 通信量

表 5.2，表 5.3，表 5.4 は，改良 BVT プロトコルと閾値判定プロトコル，そして共通項目を求めるためのプロトコルのの通信量とそのオーダーである．改良 BVT プロトコルの通信量やそのオーダーは，BVT プロトコルのそれら（表 3.1）と同じである．また，閾値判定プロトコルの通信量は，FACFM プロトコルに比べて約 2 倍になったが，オーダーは n に依存しない．さらに，共通項目を求めるプロトコルの通信量やそのオーダーは，ランダマイズされた共通項目数に依存する．

表 5.2: 改良 BVT プロトコルの通信量 [bit]

	ユーザ A	サーバ
受信	$2t_N \cdot (2n + l_b)$	-
送信	$2t_N \cdot (n + l_a)$	-
オーダー	$\mathcal{O}(n + l_a + l_b)$	-

表 5.3: 閾値判定プロトコルの通信量 [bit]

	ユーザ	サーバ
受信	$8t_N + 1$	$16t_N$
送信	$8t_N$	$16t_N + 2$
オーダー	$\mathcal{O}(1)$	$\mathcal{O}(1)$

表 5.4: 共通項目を求めるプロトコルの通信量 [bit]

	ユーザ A	サーバ
受信	$2t_N \cdot (\hat{v}_a + 1)$	$2t_N \cdot (\hat{v}_a + \hat{v}_b + 2)$
送信	$2t_N \cdot (\hat{v}_b + 1)$	$2t_N \cdot (\hat{v}_a + \hat{v}_b + 2)$
オーダー	$\mathcal{O}(\hat{v}_a + \hat{v}_b)$	$\mathcal{O}(\hat{v}_a + \hat{v}_b)$

5.4 実装評価

5.4.1 目的

モバイル端末は，ラップトップPCに比べ依然として処理能力が劣るが，近年では2GB RAMや3GB RAMまで実装され徐々に追いつく兆しが見られる．そのため，これまで非力なデバイスで敬遠されてきた公開鍵暗号を使った新しい技術が広く世に出る可能性も低くない．したがって，本研究では改良BVTプロトコル，閾値判定プロトコル，共通項目を求めるためのプロトコルをAndroid Studio 2.2で実装し，マッチングのアプリケーションを作成し，実行時間をOSがAndroid 6.0.1，CPUがQualcomm Snapdragon S4，2GB RAMのNexus 7のタブレット端末で計測した．

5.4.2 設定

ユーザA, Bとサーバが本研究の秘匿マッチングを行うと仮定する．具体的に，A, Bは改良BVTプロトコルをそれぞれ1回ずつ行い，その結果を基に閾値判定プロトコルによってAとBがマッチかミスマッチかをサーバが判定する．ここではマッチしたと仮定し，さらに共通項目を求めるためのプロトコルをそれぞれ1回ずつ行う．プロフィールと興味の項目数を$n = 10, 20, 30, \cdots, 90$まで変化させながら，各プロトコルの実行時間をそれぞれ計測する．ただし，実行時間には通信時間は含まれない．

改良BVTプロトコル

改良BVTプロトコルは，AとBの相互間で通信を伴って行われるため，それぞれの側で処理時間を分けて計測する．具体的に，次のようにプロトコルの手順で分けて[31]のBVTプロトコルと改良BVTプロトコルとを比較する．ただし，データの通信にかかる時間は除外する．

名称	改良BVTプロトコルの手順番号	BVTプロトコル[31]の手順番号
Update	-	***1***
Process	***2***,***4***	***2***,***4***
Decrypt	***5***	***5***

改良 BVT プロトコルの乱数を次のように設定する．ただし，$i = a, b$ とする．

$\boldsymbol{r}_i$	108 bit の整数が n 個から成るベクトル
$\boldsymbol{\eta}_i$	108 bit の整数が n 個から成るベクトル
$\boldsymbol{\eta'}_i$	108 bit の整数が n 個から成るベクトル
ξ_i	36 bit の整数
l_i	10
k_i	1 から 10 までの整数

閾値判定プロトコルと共通項目を求めるためのプロトコル

どちらもプロトコルの実行は，多くの通信とともにそれぞれの側で細かく行われるため，ここでは全体の時間を計測する．ただし，閾値判定プロトコルで行われるサーバの処理は除く．閾値判定プロトコルの乱数を次のように設定する．ただし，$i = a, b$ とする．

α_i	36 bit の整数
β_i	108 bit の整数
γ_i	1024 bit の整数

5.4.3 改良 BVT プロトコルの実行時間

改良前と後の BVT プロトコルを実装したアプリの実行時間をタブレット端末で計測したところ，平均の実行時間は次の表 5.5 のようになった．

表 5.5: 改良前後の BVT プロトコルの平均実行時間 [msec]

項目数 $(n+l)$	20	30	40	50	60	70	80	90	100
Update	1449.6	2154.6	2891.1	3624.4	4345.5	5059.5	5812.9	6475.0	7197.46
本研究の Process	2229.2	3667.3	5079.3	6605.1	8002.2	9585.9	11104.3	12439.4	13966.7
[31] の Process	1476.9	2215.9	2932.6	3671.0	4431.0	5126.0	5909.2	6635.5	7378.9
Decrypt	2848.0	4242.6	5655.8	7059.8	8472.7	9872.1	11316.4	12697.7	14105.6

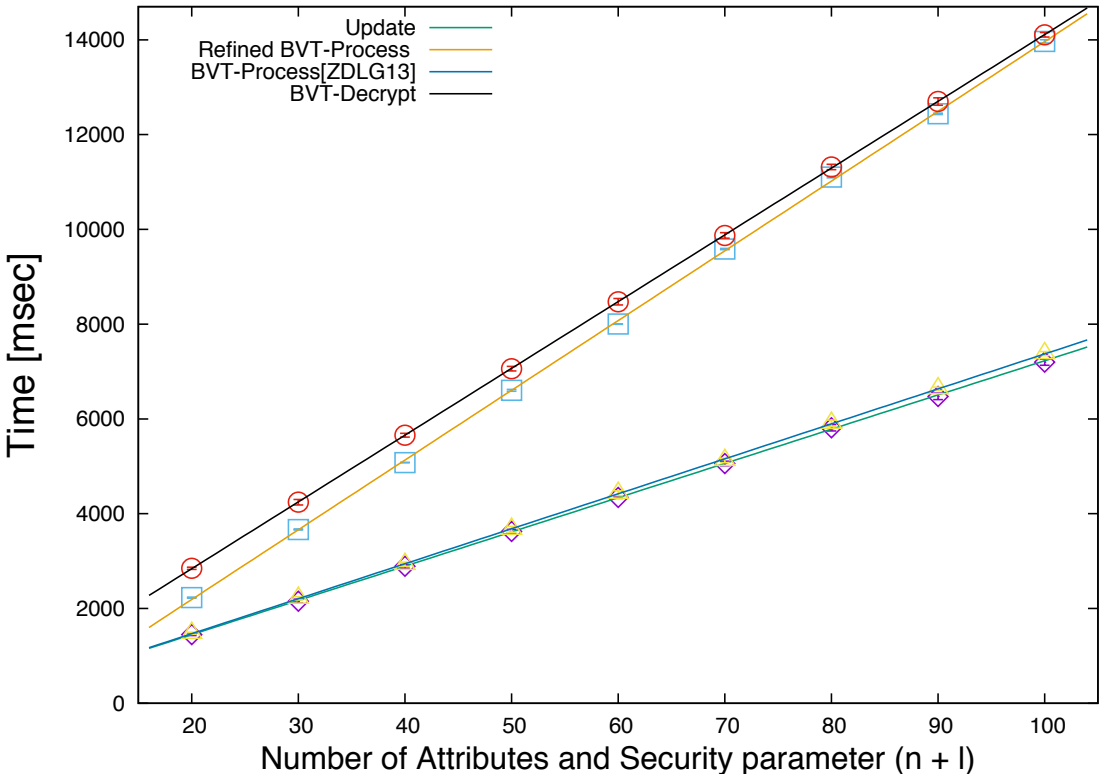

図 5.1: BVT プロトコル [31] と改良 BVT プロトコルの実行時間

図 5.1 は，改良 BVT プロトコルと BVT プロトコル [31] の実行時間のプロフィールの項目数 n を増やしたときの変化のようすである．どちらのプロトコルも計算量のオーダーが $\mathcal{O}(n+l_a+l_b)$ なので，図のように線形回帰を行った．Update と Decrypt の 2 つは，改良前後で変わっていない処理なので実行時間に変化はないが，Process はグラフのように改良に伴って約 1.5 倍遅くなった．これは加算が $n+l_i$ 回，乗算が $2n+l_i$ 回，さらに指数計算が n 回追加されるためである．しかし，改良 BVT プロトコルではマッチしたらどの項目が一致したのかを求めることができるように機能性を向上させた．一般に，機能性と効率性はトレードオフの関係にあるが，計算量のオーダーは $\mathcal{O}(n+l_a+l_b)$ であり，改良前と等しい．よって，計算量のオーダーを変えることなく，マッチングの機能性の向上を実現することができた．

5.4.4 閾値判定プロトコルの実行時間

FACFM プロトコル [31] と閾値判定プロトコルを実装したアプリの実行時間をタブレット端末で計測したところ，平均の実行時間は次の表 5.6 のようになった．

表 5.6: FACFM プロトコル [31] と閾値判定プロトコルの平均実行時間 [msec]

項目数 $(n+l)$	20	30	40	50	60	70	80	90	100
本研究の Process	751.1	769.5	755.2	770.1	760.3	767.1	752.9	764.4	758.3
[31] の Process	593.2	589.3	603.4	592.1	592.0	592.9	589.5	579.3	593.9

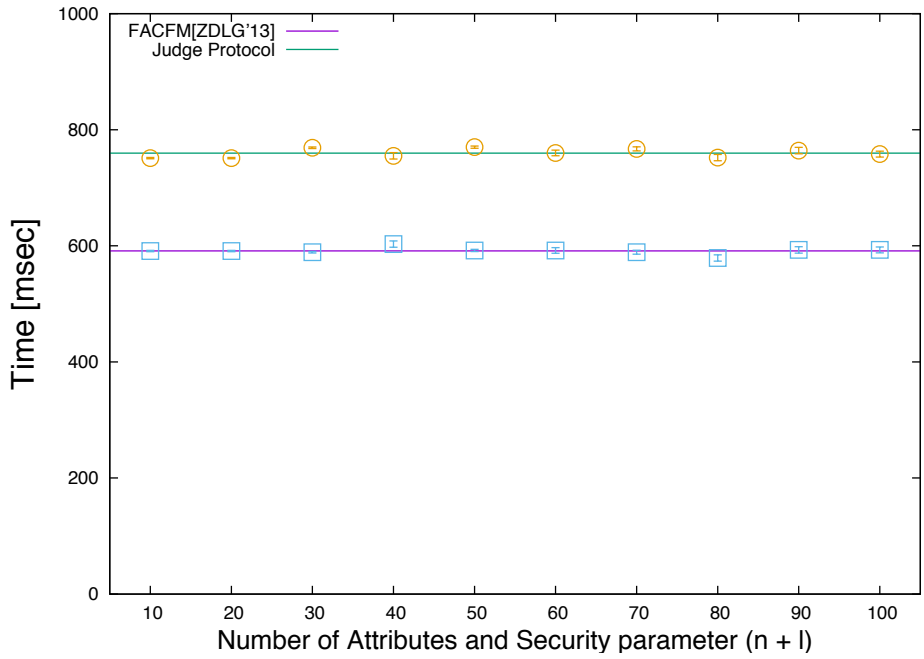

図 5.2: FACFM プロトコル [31] と閾値判定プロトコルの実行時間

図 5.2 は，閾値判定プロトコルと FACFM プロトコル [31] の実行時間のプロフィールの項目数 n を増やしたときの変化のようすである．どちらのプロトコルも計算量のオーダーが $\mathcal{O}(1)$ なので，図のように線形回帰を行った．閾値判定プロトコルは，FACFM プロトコルよりも約 180 ms 遅くなった．これは，乗算，指数計算，そして逆元計算が 1 回ずつ追加されたためである．しかし，閾値判定プロトコルでは二値の一致しか判定することができなかった FACFM を改良して二値の大小比較の判定を導入することで実用性の向上を実現した．マッチングにおいては，一致を判定の基準にするよりも閾値を判定の基準にする方が実用性が高い．

5.4.5 閾値判定と [31] の判定を出力結果で比較する

次のようなユーザ A, B を想定し，既存研究 [31] と本研究の判定プロトコルの出力の差について述べる．

$$\begin{cases} A & : \ \boldsymbol{P}_a = [1,0,0,0,0],\ \boldsymbol{I}_a = [1,1,0,0,0], \\ B & : \ \boldsymbol{P}_b = [1,1,0,0,1],\ \boldsymbol{I}_b = [1,0,0,1,1]. \end{cases} \tag{5.8}$$

このとき，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| = 3,\ |\boldsymbol{P}_b \cap \boldsymbol{I}_a| = 4$ である．

表 5.7: FACFM プロトコルの出力と期待する共通項目数の関係 ($n = 5$)

A \ B	1	2	3	4	5
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	**True**	False	False
5	False	False	False	False	False

表 5.7 は，FACFM プロトコル ([31] の判定プロトコル) を用いたマッチングの出力結果である．A は，共通項目数がいくつになるのか予測して BVT プロトコルを行う必要がある．その予測値が，B に期待する共通項目数である．例の場合では，$|\boldsymbol{P}_b \cap \boldsymbol{I}_a| = 4$ になることを事前に予測してマッチングを行わなければいけない．さもなくば，真の共通項目数が少なすぎても多すぎても FACFM プロトコルはミスマッチを出力してしまう．しかし，予測するための事前情報が一切ないため，マッチング前に予測のしようがない．B についても同様で，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| = 3$ になることを事前に予測する必要がある．

表 5.8: 閾値判定プロトコルの出力と閾値の関係 ($n = 5$)

A \ B	1	2	3	4	5
1	**True**	**True**	**True**	False	False
2	**True**	**True**	**True**	False	False
3	**True**	**True**	**True**	False	False
4	**True**	**True**	**True**	False	False
5	False	False	False	False	False

表 5.8 は，閾値判定プロトコル (本研究の判定プロトコル) を用いたマッチングの出力結果である．閾値で判定すれば，事前に共通項目数がいくつになるのか予測する必要はない．$|\boldsymbol{P}_b \cap \boldsymbol{I}_a| = 4$ の場合では，$\tau_a = 1, 2, 3, 4$ のいずれかを閾値として選択すればマッチが出力されるし，それ以外はミスマッチが出力される．このプロトコルでは，真の共通項目数が閾値よりも多すぎてもミスマッチを出力されることは決してない．つまり，既存研究で A に要求されていた不可能な予測は必要ない．B についても同様で，$|\boldsymbol{P}_a \cap \boldsymbol{I}_b| = 3$ になることを事前に予測する必要はない．このように提案方式では $\tau_a > |\boldsymbol{P}_b \cap \boldsymbol{I}_a|$ の場合をミスマッチとし，マッチを $\tau_a = |\boldsymbol{P}_b \cap \boldsymbol{I}_a|$ から $\tau_b \leq |\boldsymbol{P}_a \cap \boldsymbol{I}_a|$ まで拡張することによって，既存研究の欠点を修正した．

5.4.6 共通項目を求めるためのプロトコルの実行時間

共通項目を求めるためのプロトコルを実装したアプリの実行時間をタブレット端末で計測したところ，平均の実行時間は次の表 5.9 のようになった．

表 5.9: 共通項目を求めるためのプロトコルの平均実行時間 [msec]

共通項目数	10	20	30	40	50	60	70
本研究	2374.0	4506.5	6605.6	8781.3	11037.6	13092.4	15208.3
[31]	N/A	N/A	N/A	N/A	N/A	N/A	N/A

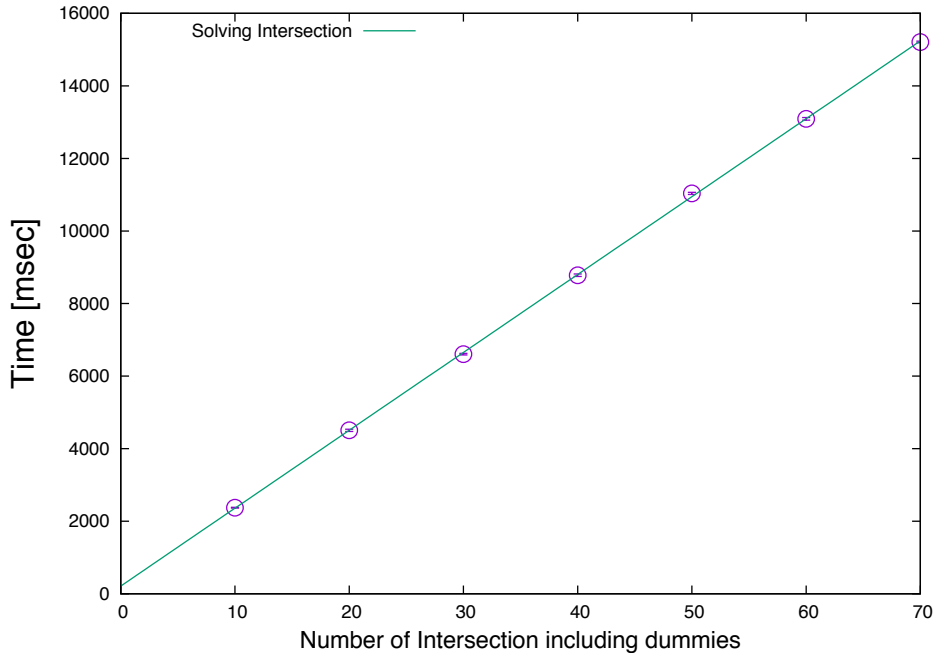

図 5.3: 共通項目を求めるためのプロトコルの実行時間

図 5.3 は共通項目を求めるためのプロトコルの実行時間のダミーを含んだ共通項目数 $|\mathsf{Dec}(\boldsymbol{P}'') \cap \boldsymbol{I}''|$ を増やしたときの変化のようすである．計算量のオーダーはダミーを含んだ共通項目数に線形で依存するので，図のように線形回帰を行った．共通項目を求めるためのプロトコルの実行時間は，プロフィールの項目数 n との直接的な関係はなく，ユーザ同士の共通項目が多いほど，あるいはセキュリティパラメタ l_i が大きいほど実行時間が増える．

第6章　考察

この章では，関連研究との比較について考察する．従来の秘匿マッチングと比較し，さらに最も近い研究 [31] と比較する．

6.1　従来の秘匿マッチングとの比較

前述のとおり，既存研究の秘匿マッチングには主に次の 2 つの課題が存在する．

1. 共通項目数や類似度，あるいは閾値を超えたか否かなどのマッチングの結果につながる部分情報がマッチングの途中で漏れないようにすること
2. 互いに設定する閾値を同時に満たすことをマッチの条件にすること

表 6.1: 本研究と関連研究の比較

	課題 1	課題 2	サーバ	備考
本研究	◯	◯	**有**	**互いの閾値を同時に判定する**
FNP04 [8]	×	△	無	初の秘匿マッチング
IO16 [12]	◯	×	有	条件付きの秘匿マッチング
LUB14 [18]	×	△	有	不正なユーザを検出できる
SX15 [25]	×	×	有	サーバはパラメタをつくるだけ
TLSL15 [27]	×	△	無	類似度によるマッチング
ZDLG13 [31]	◯	×	有	結託耐性のある秘匿マッチング
ZLJCL14 [32]	×	△	無	類似度によるマッチング

表 6.1[1] のように本研究で提案した秘匿マッチングプロトコルは，課題 1 も課題 2 も解決している．改良 BVT プロトコルから得られるのは，ブラインドされたプロフィールと興味，そしてランダマイズされた閾値と共通項目数のみであり，定理 10 よりこれらからマッチングの結果を推測することはできない．さらに，定理 11 より閾値判定プロトコルでも結果が通知されるまで推測することはできない．また，閾値判定プロトコルでは，サーバに判定を任せることによって，同時に閾値を満たすか否かを判定するように設計した．

[1] 凡例：◯ · · · 解決できている，△ · · · 仮定をつけることで解決，× · · · 解決または配慮していない

したがって，本研究では，共通項目数や閾値を超えたか否かなどのマッチングの結果につながる部分情報をマッチングの途中で漏らすことなく，互いの閾値を同時に満たすか否かをサーバが判定する秘匿マッチングプロトコルを設計することができた．このことは，表6.1の通り既存研究では実現されていないことである．

6.2 ZDLG13 [31] との比較

表 6.2: プロトコルの比較 (1)

	BVT [31]	改良 BVT
攻撃者モデル	malicious	semi-honest
計算量	$\mathcal{O}(n+l_i)$	$\mathcal{O}(n+l_i)$
通信量	$\mathcal{O}(n+l_i)$	$\mathcal{O}(n+l_i)$
マッチした共通項目の検証	×	◯

表6.2は，[31] のBVTプロトコルと共通項目を求める機能を追加した改良BVTプロトコルの比較である．計算量オーダー，通信量オーダーは同じであるが，加算が $n+l_i$ 回，乗算が $2n+l_i$ 回，さらに指数計算が n 回追加されるので，改良前と後では実行時間が遅くなる．しかし，改良BVTではマッチしたらどの項目が一致したのかを求めることができるように機能性の向上を実現した．

表 6.3: プロトコルの比較 (2)

	FACFM [31]	閾値判定
攻撃者モデル	malicious	semi-honest
計算量	$\mathcal{O}(1)$	$\mathcal{O}(1)$
通信量	$\mathcal{O}(1)$	$\mathcal{O}(1)$
閾値による判定	×	◯

表6.3は，[31] でマッチングの判定プロトコルとして提案されたFACFMプロトコルと本研究で提案した閾値判定プロトコルの比較である．計算量オーダー，通信量オーダーは同じであるが，セキュリティレベルを低下してしまった．しかし，前に述べたとおりFACFMプロトコルは，相手に期待する共通項目数と一致しなければマッチと判定されない厳しいマッチングであり，閾値判定を導入することによってこの問題点を克服し実用性の向上を実現した．

第7章　おわりに

本稿では，期待する共通項目数を閾値とし，互いに条件を満たす場合に限りマッチとサーバが判定する秘匿マッチングプロトコルを提案した．そして，マッチしたらどの項目が共通したのかを検証するためのプロトコルも提案した．

これまでに提案されている閾値を用いたマッチングプロトコルには，共通項目数や類似度を秘密計算した後，それと閾値との大小比較結果を相手に通知するため，共通項目数や比較結果などのマッチングの結果を知ることができるという問題があった．本研究では，プロフィールや興味だけでなく共通項目数などのマッチングの結果につながる部分情報を秘匿してプライバシーを守り，閾値を超えたか否かなどのマッチングの結果を追跡不可能にしたマッチングプロトコルを提案した．また，既存研究では，両方の条件を同時に判定することができても，相手の要求するマッチの個数と一致しなければマッチしないという問題があった．本研究では，この厳しすぎるマッチの条件を閾値を設けることによって緩和し，なおかつ両方の条件を同時に判定する閾値判定プロトコルを提案した．このプロトコルは，閾値を超えたか否かを追跡することができない．さらに，マッチングの機能性を充実させるために，マッチしたときだけ共通項目を求めることも実現した．これらのプロトコルは，ユーザやサーバを semi-honest モデルと仮定し，プロトコルの途中でプライバシー情報やマッチングの結果を推測できないように設計されている．

今後の課題としては，セキュリティレベルの強化や計算コストの削減が挙げられる．本研究では，プロトコルに従うユーザやサーバを考えたが，実世界ではプロトコルに従わない可能性が十分にあり得る．そのため，プロトコルから逸脱した場合にプライバシー情報につながる一切の情報も得られないような，つまり何の利得もない秘匿マッチングを設計することが課題として考えられる．具体的には，[31] と同等のセキュリティのレベルの閾値判定プロトコルを実現する必要がある．

また，マッチングにかかる実行時間の短縮は，迅速なサービスを目指す上では非常に重要視される．たとえ，プライバシーが保護されて安全にマッチングのサービスが享受できるとしても，マッチングを実行したのにいつまで経っても画面とにらめっこをしなければいけないようではサービスとは言えない．そのためには，プロトコルにおける計算量と通信量を抑えるための手法を提案することも課題として挙げることができる．

第8章　対外発表

8.1　国内発表

- 権田 陽彦，面 和成，“閾値を設けた秘匿マッチングプロトコルに関する考察”，SCIS2016，2C1-3，2016.
- 権田 陽彦，面 和成，“閾値を設けた秘匿マッチングプロトコルに関する考察 (2)”，SCIS2017，3B3-3，2017.

謝辞

本研究を進めるにあたって，懇切丁寧に御指導下さった面和成准教授に感謝致します．また，面准教授の異動に伴い新たな研究室として受け入れて下さった金子峰雄教授には，様々なご支援を賜りました．誠にありがとうございました。また，所属する研究室の同期である，大家政胤君とは，切磋琢磨して非常に有意義な2年間が過ごすことができました．この場を借りて，御礼申し上げます．さらに，宮地充子教授，上原隆平教授には副指導教官として様々なご支援を賜りました．ここに感謝の意を示します．ありがとうございました．

参考文献

[1] http://www.skout.com/.

[2] http://www.tagged.com/.

[3] http://www.hi5.com/.

[4] https://www.badoo.com/.

[5] http://www.pof.com/.

[6] L. Batina, J. Hermans, J. Hoepman, and A. Krasnova. High-Speed Dating Privacy-Preserving Attribute Matching for RFID. *RFIDSec 2014*, pages 19–35, 2014.

[7] J. Camenisch and G. M. Zaverucha. Private Intersection of Certified Sets. *FC 2009*, 5628:108–127, 2009.

[8] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. *EUROCRYPT*, 3027:1–19, 2004.

[9] P. Gasti and K. B. Rasmussen. Privacy-preserving User Matching. *WPES 2015*, pages 111–120, 2015.

[10] D. He, Z. Cao, X. Dong, and J. Shen. User Self-controllable Profile Matching for Privacy-preserving Mobile Social Networks. *IEEE ICCS 2014*, pages 248–252, 2014.

[11] Y. He, F. Li, B. Niu, and J. Hua. Achieving Secure and Accurate Friend Discovery Based on Friend-of-Friend's Recommendations. *IEEE ICC 2016*, pages 1–6, 2016.

[12] Y. Ishikuro and K. Omote. Privacy-preserving profile matching protocol considering conditions. *NSS 2016*, pages 171–183, 2016.

[13] C. Clifton J. Vaidy. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.

[14] S. Jiang, X. Zhu, L. Guo, J. Liu, R. Hao, and B. Yang. Efficient Private Matching based on Blind Signature for Proximity-based Mobile Social Networks. *IEEE ICC 2015*, pages 3246–3251, 2015.

[15] M. Kim, H.T. Lee, and J.H. Cheon. Mutual Private Set Intersection with Linear Complexity. *WISA 2011*, pages 219–231, 2011.

[16] M. Li, N. Cao, S. Yu, and W. Lou. FindU: Privacy-Preserving Personal Profile Matching in Mobile Social Networks. *IEEE INFOCOM 2011*, pages 2435–2443, 2011.

[17] M. Li, S. Yu, N. Cao, and W. Lou. Privacy-Preserving Distributed Profile Matching in Proximity-Based Mobile Social Networks. *IEEE Trans. on Wireless Communications*, 12(5):2024–2033, 2013.

[18] X. Liao, S. Uluagac, and R. A. Beyah. S-MATCH: Verifiable Privacy-preserving Profile Matching for Mobile Social Services. *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 287–298, 2014.

[19] E. Luo, Q. Liu, and G. Wang. NMHP: A Privacy Preserving Profile Matching Protocol in Multi-hop Proximity Mobile Social Networks. *ICA3PP 2015*, pages 463–474, 2015.

[20] B. Pinkas M. Naor. Oblivious Polynomial Evaluation. *SIAM J. Comput.*, pages 1254–1287, 2006.

[21] K. Omote. 秘匿多肢選択マッチングプロトコルの提案と評価. *CSS 2014*, pages 659–666, 2014.

[22] P. Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. *EUROCRYPT*, pages 223–238, 1999.

[23] P. Paillier and D. Pointcheval. Efficient Public-key Cryptosystems Provably Secure Against Actirve Adversaries. *ASIACRYPT*, pages 165–179, 1999.

[24] F. Qi and W. Wang. Efficient Private Matching Scheme for Friend Information Exchange. *ICA3PP 2015*, pages 492–503, 2015.

[25] S. Sarpong and C. Xu. Privacy-preserving Attribute Matchmaking in Proximity-based Mobile Social Networks. *International Journal of Security and Its Applications*, 9(5):217–230, 2015.

[26] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[27] A. Thapa, M. Li, S. Salinas, and P. Li. Asymmetric Social Proximity Based Private Matching Protocols for Online Social Networks. *IEEE Trans. on Parallel and Distributed Systems 2015*, 26(6):1547–1559, 2014.

[28] Y. Wang, X. Chen, Q. Jin, and J. Ma. LIP3: A Lightweighted Fine-Grained Privacy-Preserving Profile Matching Mechanism for Mobile Social Networks in Proximity. *ICA3PP 2015*, pages 166–176, 2015.

[29] Z. B. Chen Z. Some formal analysis of rocchio's similarity-based relevance feedback algorithm. *Algorithm and Computation*, 19(69):108–119, 2000.

[30] R. Zhang, Y. Zhang, J.S. Sun, and G. Yan. Fine-grained Private Matching for Proximity-based Mobile Social Networking. *IEEE INFOCOM 2012*, pages 1969–1977, 2012.

[31] H. Zhu, S. Du, M. Li, and Z. Gao. Fairness-Aware and Privacy-Preserving Friend Matching Protocol in Mobile Social Networks. *IEEE Transactions on Emerging Topics in Computing*, 1(1):192–200, 2013.

[32] X. Zhu, J. Liu, S. Jiang, Z. Chen, and H. Li. Efficient Weight-based Private Matching for Proximity-based Mobile Social Networks. *IEEE ICC 2014*, pages 4114–4119, 2014.