NAIST-IS-DT9961011

博士論文

電子透かしと暗号技術を用いたネットワークサービス構成法に関する研究

北川 隆

2003年 11月 6日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

北川 隆

審査委員： 関 浩之 教授
渡邉 勝正 教授
伊藤 実 教授

電子透かしと暗号技術を用いたネットワークサービス構成法に関する研究*

北川 隆

内容梗概

インターネットに代表されるコンピュータネットワークの普及によって, さまざまなサービスがネットワーク上で行われるようになっている. この際, ネットワーク上を流通するコンテンツの著作権管理やサービス利用者のプライバシ保護などが大きな問題となる. そこで, 不正者による盗聴・改ざん・なりすましなどが行われた場合でも, 利用者のプライバシが保護され, システムが正しく動作するように, 暗号技術やネットワーク技術を利用しサービスを設計しなければならない. 本論文では, ネットワーク上でのサービスの安全な実現のために必要となる技術の中で, 電子透かし法, ソフトウェアの従量制課金法, 匿名アンケートプロトコルについて議論する.

第 2 章では, 電子透かしに関する研究について述べる. 近年, 広く行われつつあるネットワーク上でのコンテンツ配信システムにおける, 著作権保護, とくに, 音楽や映画, コンピュータプログラムなどのコンテンツの不正配布が大きな問題となっている. しかし, これらのコンテンツのコピーは容易であり, 不正コピーを原理的に防止することは困難である. 次善の策として, 電子透かしを利用した不正コピー抑止法が考えられる. 本論文では, まずコンピュータプログラムに対する電子透かし法を提案する. コンピュータプログラムに対する電子透かしの安全性について形式的に定義し, 定義した安全性を満足する電子透かし法を提案する. 提案方式は, Java, C, C++などの言語で書かれたプログラムで利用可能であり, コンピュータプログラムの不正コピー対策に役立つと考えられる. 次に, 電子透かしで用いられる ID 符号化法について議論する. ID が埋め込まれているコンテンツの不正配布を考えるユーザは, 電子透かしの除去や改ざんを試みると考えられる. 電子透かしに対する有効な攻撃法として, 複数のユーザがコンテンツを比較し, 電子透かしの

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DT9961011, 2003 年 11 月 6 日.

埋め込み位置を特定し, 改ざん等を行う結託攻撃がある. これまでに結託攻撃を防ぐために結託耐性符号と呼ばれる ID 符号化法が提案されてきた. 本論文では, 結託耐性符号の性質を利用した新しい結託攻撃法を提案し, その攻撃法が有効であることをコンピュータシミュレーションを利用して示す. さらに, 従来提案されていた符号の問題点について考察し, その問題点を解決した新しい符号を提案する. 提案した符号を利用することで符号長を大きくすることなく, 従来の符号に比べ検出誤り確率が劇的に減少することを示す.

第 3 章では, ソフトウェアの回数券型課金法を提案する. 個人ユーザがソフトウェアを購入する場合は, 無期限で利用できるライセンスを購入するという形式が一般的である. しかし, 高価なソフトウェアや頻繁には利用しないソフトウェアの場合, このようなライセンスはユーザにとって負担となる. ソフトウェアを利用した分だけ代金を支払う方式が実現できれば, ユーザの負担も減り, ソフトウェアを利用しやすくなる. 本論文では, ソフトウェアの難読化技術と電子署名法を用いて, 回数券型の従量制課金法を提案する. 提案法は, ソフトウェア技術のみで容易に実現可能である. 提案手法を利用することによって, ユーザは比較的低コストでソフトウェアを利用できるようになる.

最後に第 4 章では, 安全な匿名電子アンケートプロトコルを提案する. 近年, 多くの大学で学生による講義評価が行われている. 講義評価アンケートをネットワーク上で実現できれば, 集計等の手間が削減できるなど多くの利点がある. しかし, 安易な匿名アンケートシステムを利用すると, その匿名性ゆえに回答率が極端に低下するという問題が発生する. この問題を解決するために, 匿名かつ回答していないユーザを特定できるアンケートプロトコルを提案する. 提案プロトコルでは, 回答していないユーザを特定できるため, 回答を促すことが可能である. 従来の投票プロトコルでは, 匿名性を保つために, 匿名ネットワーク等が必要であったが, 提案プロトコルでは全体を 3 つのフェーズで分けて実行することとブラインド署名を利用することにより, MIX-NET などの匿名ネットワークを用いることなく実現できる. 本論文では, 実装したプロトコルを実際に講義評価アンケートに利用し, 実証実験を行った. その結果, 以前単純な匿名アンケートシステムを用いたときの回答率が 6%, 17 %であったのに対して, 49%, 35% へと大幅に上昇した.

キーワード

情報セキュリティ, 電子透かし, 従量制課金, 匿名アンケートプロトコル

Digital Watermaking and Cryptographic Techniques for Secure Network Services*

Takashi Kitagawa

Abstract

For providing secure network services, it is important that the services are constructed to be secure against any attacks by malicious users. In this thesis, three techniques for secure network services, watermarking systems, a pay-per-use pricing system for software and an anonymous questionnaire system are discussed.

Contents distribution systems have been widely used. In such systems, copyright protection is a very important matter. Especially, illegal re-distribution of audio-visual contents and computer software has been a serious problem. Digital watermarking technique is one of the practical methods to prevent illegal copying. In chapter 2, two topics on digital watermarking technique are discussed. First, a digital watermarking technique for a computer program is proposed. We formally define a condition on a digital watermarking for a computer program to be secure, and propose a digital watermarking method which satisfies this condition. The proposed method can be applied to Java, C and C++ programs. This method will help us inhibit illegal duplications of computer programs. Second, a new collusion attack for collusion secure fingerprinting code and a new collusion secure fingerprinting code, Randomized c-secure CRT Code are proposed. A malicious user who wants to make an illegal copy of contents may try to delete or modify the watermarked information. One of the techniques which malicious users can use is collusion attack: Some users collude and compare their contents. Differences between their contents suggest the positions where the watermark

*Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9961011, November 6, 2003.

is embedded. The users then modify the contents using the knowledge of the positions. Encoding user ID by a collusion secure fingerprinting code and embedding the codeword into the content, the server can detect one of the colluders from the attacked contents. In this thesis, we propose a new attack against c-secure CRT Code which was considered one of the most practical codes. Based on the analysis of the effectivity of this attack, we propose a new ID coding scheme which is secure against new attack.

In chapter 3, a pay-per-use pricing system for computer software is proposed. Typical licenses which come with most software sold today provide a user with the right to execute the software as many times as he/she wishes. However, for software which is expensive or is not used frequently, a user may feel the license fee is too expensive. It might cause an illegal duplicate of software. If there exists a pay-per-use pricing system, users are willing to pay for the software rather than use illegal copies or pirated software. In this thesis, a coupon-ticket like pricing system is proposed. The system is constructed by using only software components and cryptographic technique. Thus, the system is easy to realize and suitable for mobile terminals.

In chapter 4, a secure protocol for a faculty courses questionnaire (FCQ) in universities is proposed. A network questionnaire system will be convenient for a student to send an answer. However, a simple anonymous questionnaire system may allow malicious operations of students, and students do not feel guilty even if they do not answer the questionnaire. It will be convenient if we can utilize an anonymous FCQ system with which the server can distinguish those students who have answered the questionnaire from those who have not answered. Using this function, a university can ask students who have not answered the questionnaire to do so. The proposed protocol consists of three sub protocols and uses a blind signature scheme, and thus it does not need an anonymous network such as MIX-NET. The protocol has been implemented and used experimentally. The result of the experiment shows that the ratio of respondents increased from 6 %, 17% to 49 %, 35 % respectively.

Keywords:

information security, digital watermark, pay-per-use pricing software, anonymous questionnaire protocol

目 次

図 目 次

表 目 次

第1章　まえがき

インターネットに代表されるコンピュータネットワークの普及によって, さまざまなサービスがネットワーク上で行われるようになっている. ネットワーク上でサービスを実現する場合には, ネットワーク上を流れるコンテンツの著作権管理やネットワークサービスを利用するユーザのプライバシ保護などが大きな問題となる. また, ネットワーク上では, 不正者による盗聴・改ざん・なりすましなどの不正が行われる可能性がある. そのような不正が行われた場合でも, 利用者のプライバシは保護され, システムは正しく動作するように, 暗号技術やネットワーク技術を利用しサービスを設計しなければならない. 図 1.1

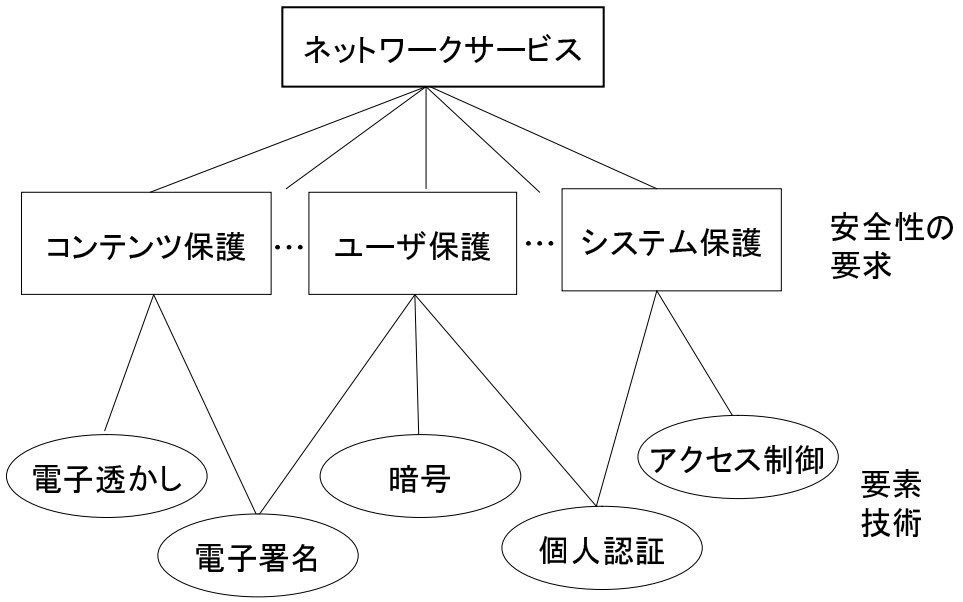

図 1.1　ネットワークサービスに要求される安全性と要素技術

に示したように, ネットワークサービスに要求される安全性を実現するための要素技術は数多く提案されており, それらに関する研究は広く行われている. しかし, あるネットワークサービスを設計する際には, 必要となる要素技術を適切に組み合わせなければ安全に実現することはできない. 現在, 安全なネットワークサービスを設計するための汎用的な方法は知られていないため, 各サービスについて個別に設計を行う必要がある.

本論文第 2 章では, ネットワーク上でのサービスの安全な実現のために必要となる要素技術の一つである電子透かし法について議論する. 第 3 章と第 4 章では, 要素技術を利用して設計したネットワークサービスであるソフトウェアの回数券型課金方式, 匿名アンケートプロトコルについて議論する.

電子透かし法

本論文第 2 章では, 電子透かし法について議論する. ネットワークを利用したコンテンツ配信システムにおいては, ディジタルコンテンツの著作権保護が重要な問題となる. ディジタルコンテンツの著作権を保護する方法の一つとして電子透かし法が提案されている. 電子透かし法とは, ディジタル情報に対して, 一般のユーザには感知されないように何らかの情報を埋め込む技術である. 電子透かし法を以下のように利用することで, コンテンツの不正コピーの抑止に利用できる. コンテンツ配信サーバは, コンテンツを配信する前に, コンテンツを受け取るユーザの ID 情報（ユーザを特定できる情報）をコンテンツに埋め込み, ID を埋め込んだコンテンツを配信する（図 1.2）. あるユーザがコンテンツを不正コピーし, コンテンツを不正に配布したとする. もし, 不正コピーの 1 つが配信サーバに発見された場合, そのコンテンツに埋め込まれている ID を調べることで, 不正を行ったユーザを特定することができる（図 1.3）. このように, 不正コピーを配布したユーザに対してリスクを負わせることで, 不正コピーの抑止効果が期待できる.

一方, コンピュータプログラムの不正コピーは従来より大きな問題であった. コンピュータプログラムの不正コピーを防ぐことは重要な課題であるが, 原理的にコピーを防ぐことは困難である. 不正コピーを抑止するために, プログラムに対する電子透かし法を利用する方法が考えられる. これまでに, 画像データや音声データなどに対する電子透かし法が広く研究されている. 画像や音声データはその冗長度が高く, データ内容の一貫性がそれほど強く要求されないこと等の理由により, 電子透かしを構成するための自由度がかなり大きく, 望ましい性質を持った透かしの実現も比較的容易である. また, 電子透かしを埋

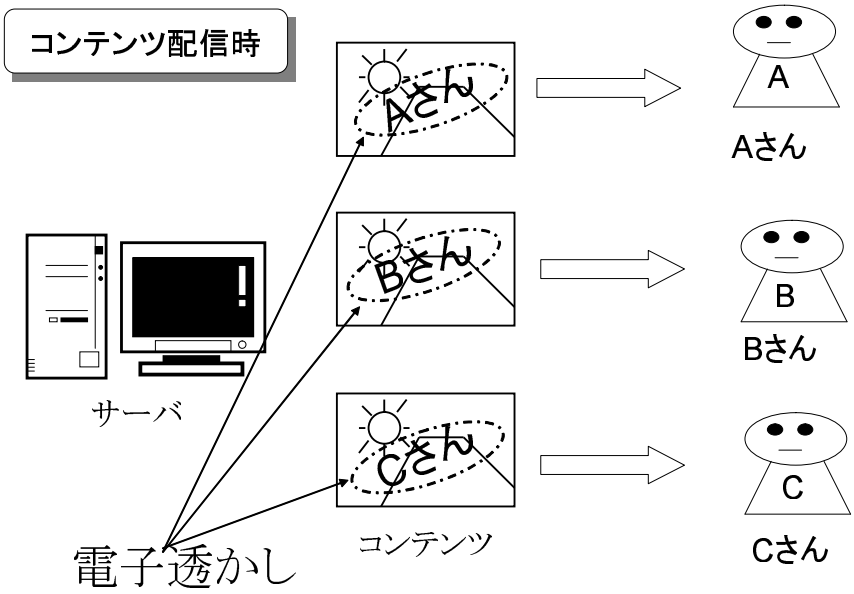

図 1.2 電子透かしを埋め込んだコンテンツの配信

め込むことによってもとのデータが持っていた情報量が減少するため, 悪意を持ったユーザがどれだけの資源を費やしても, オリジナルデータを得ることはできない. 一方, コンピュータプログラムは画像などのデータと比較すると, 冗長度が低く, さらに内容の一貫性や完全性が強く要求されるため, 電子透かしを埋め込む自由度はそれ程大きくない. また, 透かしの埋め込まれたプログラムは元のプログラムと同じ動作をする必要があるため, 透かしを埋め込むことで情報量の欠損を発生させることはできない. むしろ, 透かしを埋め込んだプログラムには, 本来のプログラム動作には必要ない冗長な部分が含まれてしまうことが避けられない. したがって, コンピュータプログラムの透かしは, 画像などに対する透かしとは異なり, 悪意を持ったユーザが十分な資源を費やせば透かしを除去することが可能である. たとえば, プログラムの仕様や意味を完全に理解し, 等価な動作をするプログラムを一から作成することで, 透かしの入っていないプログラムと等価なものを構成することが原理的には可能である. すなわち, 理論的な観点からは, プログラムに対する安全な電子透かし法は実現できないことになる. しかし, 工学的な立場からすると,

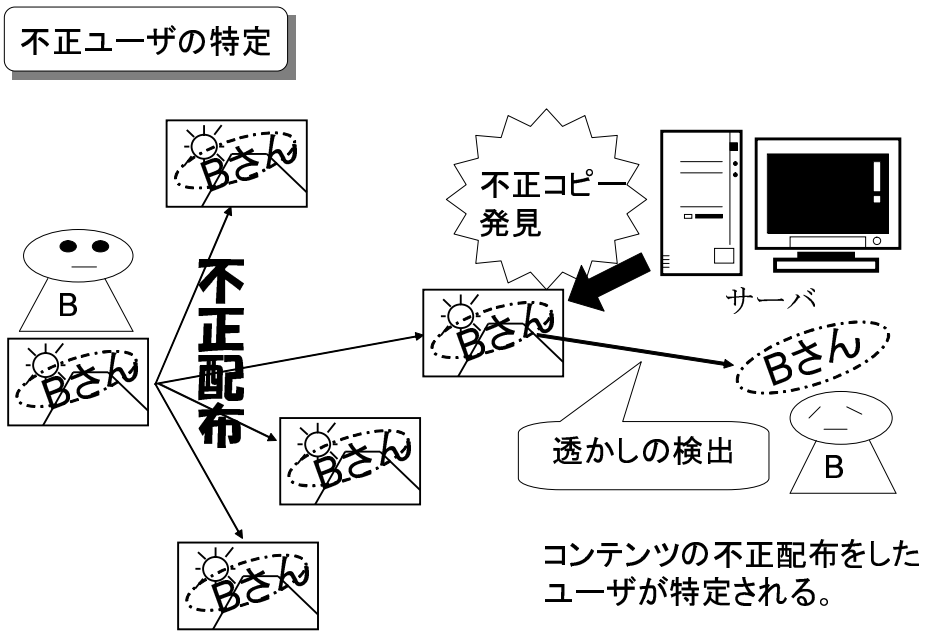

図 1.3 電子透かしを用いた不正ユーザの特定

たとえ不正ユーザがそのような手段によって透かしの入っていないプログラムを構成できるとしても, あまり現実的な問題にはならないと考えられる. コンピュータプログラムに対する電子透かし法では, 安全性の高さとそのコストとのバランスを考えて, 現実的な透かしの実現法を考えることが重要である.

本論文では, まずコンピュータプログラムに対する電子透かしの安全性を次の 3 種類に分類し, それぞれを形式的に定義する.

- **改変不可能性** 不正ユーザが（実行可能な）操作をプログラムに行ったとしても, サーバは埋め込まれている透かしを正しく検出できる性質.
- **消去不可能性** 不正ユーザが（実行可能な）操作をプログラムに行ったとしても, 不正ユーザが透かし情報を完全に消去することができない性質.
- **偽造不可能性** 不正ユーザが（実行可能な）操作をプログラムに行ったとしても, 他者の透かしを偽造できないという性質.

さらに Java 言語を例に電子透かしの実現方式を提案する. 提案方式は, プログラム中に透かし情報を保持するための変数を導入し, その変数に透かし情報を格納する. その後, 格納した透かし情報を, 不正ユーザによる改変から保護するために, 難読化操作等を行う. 透かしの検出には, 検出サーバのみが持つ特別なクラスファイルを用いてプログラムを実行することにより, 透かしを検出できる. 論文では, 提案方式が形式的に定義した改変不可能性を満たすクラスに属することも示す. 提案方式は, Java 言語だけではなく, C や C++などの言語でも利用可能であり, ソフトウェアの不正利用の抑止に有効であると考えられる.

次に, 電子透かしのための結託耐性符号に対する新しい攻撃法と新しい結託耐性符号の提案を行う. 先ほど述べたように電子透かしを利用することで, 不正コピーを行った利用者を特定することが可能となる. 従って, 不正を行おうと考えるユーザはコンテンツから電子透かしを取り除いたり, 電子透かしを改変したりしようと試みると考えられる. 画像データや音声データなどに対する電子透かし法への攻撃の一つに結託攻撃がある. 結託攻撃とは, 複数のユーザがそれぞれの ID が電子透かしとして埋め込まれているコンテンツを持ち寄り, それらのコンテンツを比較することで電子透かしが埋め込まれている位置を特定し, 埋め込まれている電子透かしを除去したり改変したりするという攻撃である. 結託攻撃を防ぐ方法としては, 結託攻撃に対して安全な電子透かしアルゴリズムを利用する方法や結託耐性符号と呼ばれる符号を利用してユーザ ID を符号化する方法が提案されている.

本論文 2.3 節では, 結託耐性符号 c-secure CRT 符号 [14] に対する新しい攻撃法を提案し, さらにその攻撃に対しても安全となるように c-secure CRT 符号を改良した Randomized c-secure CRT 符号を提案する. c-secure CRT 符号とは次のような性質を持つ符号である. コンテンツ配信サーバが c-secure CRT 符号を用いて符号化したユーザ ID をコンテンツに埋め込み, そのコンテンツを配信する. c 人以下のユーザによる結託攻撃で作成されたコンテンツをサーバが発見した場合, サーバは c-secure CRT 符号のトレースアルゴリズムを利用することで, 結託に加わった不正ユーザのうち少なくとも一人は検出することができる.

提案する攻撃は, c-secure CRT 符号の符号語が持つ情報の偏りと, トレースアルゴリズムが用いる閾値を悪用した攻撃法である. この攻撃を行った符号語からは高い確率で検出誤りが起こることをコンピュータシミュレーションによって示す. 次に, この攻撃が効果的である理由を考察し, 新しい結託耐性符号 Randomized c-secure CRT 符号を提案

する．Randomized c-secure CRT 符号は, 符号語を生成する際に用いられる情報に対してランダムな置換を施すことにより, 符号語が持つ情報の偏りを解消する．Randomized c-secure CRT 符号の性能を評価するためにコンピュータシミュレーションを行い, その安全性を示す.

ソフトウェアに対する回数券型課金方式

本論文第 3 章では, ソフトウェアの従量制課金方式を提案する．ソフトウェアを購入して利用する場合, 利用者はソフトウェアの販売者とソフトウェアの使用形態などに関する契約を結び購入する．個人向けのソフトウェアの場合, ユーザはそのソフトウェアを無制限に利用できる契約でソフトウェアを購入するという形態が主流である．頻繁に利用するソフトウェアの場合, このような契約でも問題はない．しかし, あまり頻繁に利用しないソフトウェアの場合, このようなライセンス形態は, 利用者に対し割高感を感じさせ, またその結果不正コピーを引き起こす原因の一因にもなると考えられる．あまり頻繁に利用しないソフトウェアでは, 無制限に利用できるライセンスよりも, ユーザが利用した回数や時間に応じて料金を支払う従量制課金方式の方が適切であると考えられる.

提案方式では, 利用者はソフトウェアと共にそのソフトウェアを利用するための回数券型のライセンスを購入する．利用者は買ったライセンスの回数だけソフトウェアを利用することができる．このような回数券型ライセンスを実用化するには, ユーザによる不正行為をいかに防止するかが問題となる．すなわち, 悪意を持ったユーザが自分のコンピュータを不正に操作することで, 契約回数以上にソフトウェアを利用しようとすることが考えられる．公正な課金方式の実現のためには, このような不正が行われないよう十分注意を払ってシステムの設計を行う必要がある.

同種の目的を持った方式として, 超流通システム [18, 19] の研究がある．超流通システムとは, 特殊なシステム（ハードウェア）を仮定し, そのシステム上でしかコンテンツが利用できないように機構を構成した上で, コンテンツの利用に対して課金するシステムである．超流通システムでは, 電子的流通によりコンテンツの流通コストを抑えることができるため, ユーザに対して低価格でコンテンツを提供できるという利点がある．しかし, 超流通システムでは通常, 耐タンパ性を有した特殊なハードウェアの存在を仮定しているため, 汎用性やコストの面について問題がある.

また, 超流通システムで利用されているような特殊なハードウェアの代わりに, ネット

ワークへの常時接続性を仮定しても, 同種の方式を実現することが可能である. すなわちユーザの使用する計算機がネットワークに常時接続されており, ライセンスを管理するサーバと無制限に通信を行なうことが可能であるならば, 従量制課金の仕組みを実現することは比較的容易であると考えられる. たとえば, ソフトウェアの起動のたびにネットワーク上に存在するライセンス管理サーバと通信を行ない, ユーザのソフトウェア起動回数や実行時間をライセンス管理サーバが常に監視しておく方式を実現すれば, ユーザの不正を完全に防ぎつつ従量制課金方式を実現できる. 実際, 一部のソフトウェアでは, ソフトウェア起動時にライセンスサーバと交信を行い, ユーザのソフトウェア使用権をチェックする方式が実用化されている. しかし, この方式では, サーバやネットワークに障害が発生した場合にソフトウェアを使用できなくなるという問題点がある. このように, ユーザのコンピュータに特殊なデバイスが装備されていたり, ユーザのコンピュータとソフトウェア販売者のコンピュータとがネットワークで常時接続されている場合にはソフトウェアの従量制課金方式は比較的容易に実現可能である. しかし, そのような特殊なデバイスの利用は利便性に欠け, サーバとの常時接続もモバイル PC での利用では不可能なこともある.

本論文では, 特殊なデバイスやサーバとの常時接続を仮定しない環境においてソフトウェア技術のみで従量制課金方式を実現する方式を提案する. 提案方式では, ソフトウェアの起動回数（回数券の残り度数）をユーザのコンピュータ上に保管する. 課金対象となるソフトウェアの起動時には, そのソフトウェア自身がユーザコンピュータ上の回数券の残り度数を確認し, 更新する. ただし, ユーザは自分のコンピュータ上のファイル等を操作して残りチケット数をごまかす等の不正操作を行い得るので, 暗号化や難読化等の手法を用いて情報の保護を行う. また, 不定期に, あるいはソフトウェアの実行環境に不審な点が発見された場合, ソフトウェアはライセンスを管理するサーバに接続し, ユーザの行為の正当性を確認する. 本方式でもネットワークへの接続性は必要となるが, その頻度はかなり小さくすることが可能であり, ダイアルアップ回線やモバイル端末のような環境でも負担はそれほど大きくないと考えられる. 一方, 本方式では回数券の残り度数などの重要な情報をユーザのコンピュータ上で管理することになるため, セキュリティの低下は免れない. 本論文では, 提案方式が不正ユーザによる簡単な攻撃に対しては安全であることを示し, 不正を行うためにはユーザは多大な手間をかける必要があることを示す. 提案方式は, ソフトウェアのみで構成されているため実現が容易であり, ユーザにとっては高価なソフトウェアも手軽に利用でき, またソフトウェア販売者側にとってもユーザの不正

コピー等の減少により, 利益が得られると考えられる.

匿名アンケートプロトコル

オークションやアンケートなどの投票をネットワーク上で行う研究は従来より広く行われている. さまざまな投票をネットワーク上で行う場合, 不正者による盗聴, 情報の改ざん, なりすましなどの攻撃を防ぐ必要がある. また, 投票者のプライバシなどを保護しつつ, 正しい結果を保証する必要があるなど, さまざまな要求がある. これらの要求を満たすように, 電子投票方式では, ネットワーク技術と暗号技術を適切に組み合わせて, プロトコルを設計する必要がある. 本論文第 4 章では, 投票方式の一つであるアンケートに着目し, 匿名電子アンケートプロトコルを提案する.

近年, 多くの大学で学生による講義評価が行われている. 講義評価アンケートをコンピュータネットワーク上で実現できれば, 集計等の手間も削減できる. しかし, 安易な匿名アンケートシステムを利用し, 講義評価アンケートを行った場合, その匿名性ゆえに回答率が極端に低下するという問題が発生する. 大学での講義評価に限らず, 学会での役員選挙など匿名で投票を行いたいが, できるだけ高い回答率を得たい, もしくは回答が義務付けられているというアンケートは多く存在する. この問題を解決する一手法として, 匿名かつ回答していないユーザを特定できるアンケートプロトコルがあれば, 集計サーバが, 回答していないユーザを特定できるため, 回答期間終了後に回答していないユーザに対し回答を促すことが可能である. 一方, 同種の研究として電子投票プロトコルがある [9, 22, 24, 27]. 講義評価アンケートは電子投票の特殊な場合と考えることができるが, 電子投票と講義評価アンケートでは要求される事項や前提条件に異なる点も多い. また, 現在提案されている電子投票システムでは, MIX-NET [5] などの匿名通信路が必要になることも多い. しかし, MIX-NET は結託をしない複数のサーバから構成されるため, 大学がサーバをの管理を行うとするならば, サーバが結託していないことを学生に納得させることは困難である. これらの理由から, 本論文では大学での講義評価に適したアンケートプロトコルを提案する. 匿名かつ, 回答していないユーザを特定するという機能を実現するために, 提案プロトコルは, 全体を 3 つのフェーズから構成し, ブラインド署名法を利用した. ユーザが 3 つのフェーズを異なる計算機上で実行することで, インターネットなど通常のネットワーク上でも匿名性を実現することが可能となる. 提案プロトコルは, 比較的単純なプロトコルであり, 容易に実装可能である. 本論文では, 実装したプロトコ

ルを実際に講義評価アンケートとして利用した実証実験の結果についても述べる.

第2章　コンピュータプログラムに対する電子透かしと結託耐性符号

2.1. 電子透かし

電子透かしとは, デジタル情報に対して, 一般のユーザには感知されない形式でなんらかの情報を埋め込んでおき, ある種の特権を持った管理者のみが埋め込まれた情報を取り出せるような仕組み, あるいはそのように埋め込まれた情報のことをいう. 電子透かしは主として, 以下のような2つの用途に利用可能であると考えられる.

1 つ目の用途は, 第三者に対して, 情報の著作権を主張するというものである. ある有価値情報を一般に公開する前に著作権者の名前などを埋め込んでおき, その情報のオリジナリティなどが問題になったときに, 埋め込んだ情報を取り出すことで, 本来の著作権を主張するのに利用できると考えられる. 2 つ目の用途は, 有価値情報の転売や再配布が行われた場合, その流通経路を特定するというものである. すなわち, 著作権者がユーザに対して有価値情報を配布・販売する際に, オリジナルの有価値情報をそのままユーザに渡すのではなく, そのユーザを特定することが可能な情報を電子透かしとして埋め込むことで, ユーザ毎に微妙に異なる情報を配布・販売する. もし, 有価値情報の不正コピー・不正使用が発見された場合, その不正に流通している情報から電子透かしを取り出すことで, 情報の漏洩元となったユーザの特定が可能である. 情報の漏洩元となったユーザに対し, 不正コピーによって生じる損害賠償責任を課すような社会的機構が実現されれば, 有価値情報の不正コピーを抑止することができると考えられる.

悪意を持ったユーザは, 配布された有価値情報から透かしを取り除いたり, あるいは透かしを破壊してから, 情報の転売や再配布を試みる可能性がある. もし, 透かし情報が正しく復元できないように情報を操作されてしまうと, どのユーザが不正に荷担したか特定できなくなるため, 不正使用抑止効果が低下することになる. したがって電子透かしは, 一般のユーザには除去することも改変, 破壊することもできないようになっていることが望まれる.

2.2 節では, コンピュータプログラムに対する電子透かし法について議論する. まず, コンピュータプログラムに対する電子透かしの特徴や性質について議論し, 電子透かしの安全性について形式的に定義する. 次に, その安全性を満たす電子透かしの実現法について述べる.

2.3 節では, 電子透かしのための結託耐性符号ついて述べる. 結託攻撃とは電子透かし法に対する攻撃法の一つであり, 複数のユーザが電子透かしが埋め込まれたコンテンツを持ち寄り, それらのコンテンツを比較して透かしを改変する攻撃である. 結託攻撃を防ぐために結託耐性符号と呼ばれる符号化法が提案されている. 2.3 節では, 従来効率がよいと考えられていた結託耐性符号である c-secure CRT 符号に対する新しい攻撃法を提案し, さらにその攻撃に対して安全であるような Randomized c-secure 符号を提案する.

2.2. コンピュータプログラムに対する電子透かし法

2.2.1 はじめに

コンピュータプログラムの不正コピー, 不正使用は, 従来より大きな社会的問題であった. とくに近年, 安価なパーソナルコンピュータが大量に普及するのに伴い, 企業や大学などでも, ユーザー自身が計算機管理を行うような状況が増加している. 専門的な教育を受けた計算機管理者に比べ, 一般ユーザのソフトウェア利用に関する倫理意識は十分とはいえないため, 第三者のチェック機構を持たない企業や大学などは, ソフトウェアの不正使用の温床となる可能性がある.

ソフトウェアの不正使用を防止・抑止するようなアプローチは, 少なくとも 3 種あると考えられる. 最初のアプローチは, 磁気テープや CD-ROM 等, ソフトウェア配布媒体の物理的な複製を困難にするような, いわゆるコピープロテクトと呼ばれる手法である. コピープロテクトの実現方法は様々であるが, ソフトウェア情報の重要な部分を, オペレーティングシステムにおける標準的な記録方式とは異なる方式で媒体に格納することが多い. コピープロテクト技術はあくまでも媒体のコピーを防止する技術であり, たとえば同一の配布媒体を複数のユーザが何度も使い回すような不正行為には無力である. 不正抑止の 2 番目のアプローチは, ソフトウェア使用時に, ユーザが本当にソフトウェアの使用権を持っているかどうかを検査する方式である. たとえば, ネットワーク上に設置された

ライセンスサーバと交信する, CD-ROM 等の配布媒体の存在を確認する, 計算機内に暗号化して保護された情報を検査する等の手段により, ユーザの正当性を検査する方式がこれにあたる. このアプローチはソフトウェアの不正使用防止にかなり有効であるが, いくつか問題点もある. たとえばライセンスサーバ方式は, ユーザ数増大に応じてサーバの負荷が大きくなることや, ネットワーク障害に対する耐性が損なわれること等の問題がある. この媒体検査方式は, CD-ROM 等の配布媒体の複製がそれほど困難ではないことを考慮すると, 必ずしも十分な安全性を有しているとは言えない. また, 詳しい説明は省略するが, 暗号化された情報を利用する方式では, 計算機ごとにユニークで改変不可能な識別情報を, 暗号化された情報に埋め込んでおく必要がある. しかし, 全ての計算機にそのような識別情報を割り当てることは, 現時点ではあまり現実的でないと考えられる. ソフトウェアの不正使用を抑止する 3 番目のアプローチは, 不正行為を行ったものに対して処罰が与えられるよう, 社会的な仕組みを構成することである. 不正行為に対して相応のリスクを負わせることで, ユーザの不正行為への関与を抑止できると期待される.

以上 3 つのアプローチは排他的なものではなく, むしろ相補的な関係にあり, ソフトウェアの不正使用を防止するには, これらの技術を適切に組み合わせて使用することが肝要であると考えられる. 3 つのアプローチのうち, 最初の 2 つについては既にさまざまな形態で実用化されている. 一方, 第 3 のアプローチについては, 未だ十分な議論が行われているとは言いがたい. ソフトウェアの不正使用を効果的に抑止するためには, 第 3 のアプローチに基づく方式の実用化が不可欠であると考えられる.

第 3 のアプローチを実現するためには, 不正に関与したユーザの特定が可能でなければならないが, 現段階ではそのような技術が確立されているとはいえない. 一方, 画像・音声などのデジタルデータに対しては, その不正流通を抑止することを目的として「電子透かし [11, 13]」が盛んに研究されている. 電子透かしとは, 画像などのデジタルデータに, 一般のユーザには感知されない形式でなんらかの情報を埋め込んでおき, ある種の特権を持った管理者（たとえば著作権者など）のみが埋め込んだ情報を取り出せるような仕組み, あるいはそのように埋め込まれた情報のことをいう. 画像や音声などのデジタルデータの場合, 冗長度が高いこと, データ内容の一貫性がそれほど強く要求されないこと等の理由により, 電子透かしを構成するための自由度がかなり大きく, 望ましい性質を持った透かしの実現も比較的容易であると考えられる. また, 電子透かしを埋め込むことでオリジナルの有価値情報が持っていた情報量が減少するため, 悪意を持ったユーザがどれだけの資源を費しても, 元のオリジナル情報を得ることはできない.

一方, コンピュータプログラムは, 画像などのデジタルデータに比べると冗長度が低く, さらに内容の一貫性や完全性が強く要求されるため, 透かしを埋め込むための自由度はそれほど大きくない. また, 透かしの埋め込まれたプログラムはオリジナルのプログラムと同等の動作をする必要があるため, 透かしを埋め込むことで情報量の欠損を発生させることはできない. むしろ, 透かし入りプログラムには, 本来のプログラム動作には必要のない冗長な部分が含まれてしまうことが避けられない. したがって, どのような透かし方式であったとしても悪意を持ったユーザが十分な資源を費せば, 冗長部分である透かしを除去することが可能となってしまう. たとえば, プログラムの仕様や意味を完全に理解し, 等価な動作をするプログラムを一から作成することで, 透かしの入っていないプログラムと等価なものを構成することが, 原理的には可能である. すなわち, 理論的な観点からは, プログラムに対する安全な電子透かし法は実現しないことになる. 一方, 工学的な立場からすると, たとえ不正ユーザがそのような方式で透かしの入っていないプログラムを構成できるとしても, あまり現実的な問題にはならないと考えられる. 完全無欠な電子透かし法を目指すよりも, 安全性の高さとそのコストのバランスを考えて, より現実的な透かしの実現法を模索することが肝要であると考えられる. 2.2.2 節では, プログラムに対する電子透かしの安全性について, やや形式的に議論する.

これまで, プログラムに対する電子透かし的な技術がまったく存在しなかったわけではない. たとえば一部のゲームソフトなどでは, プログラムの実行中に特殊な操作（たとえば, 特定の複数のボタンを同時に押す等の操作）を行うと, あらかじめ仕組まれたメッセージが表示されるような仕組みが存在する. そのような, いわゆる「隠しコマンド」的な技術を汎用のプログラムに対して適用すれば, プログラムに対する電子透かしが実現できると考えられる.

本論文 2.2 節の目的は 2 つある. まず最初の目的は, プログラムに対する電子透かしが満たすべき条件を形式的に表現することである. 詳しくは本文中で議論するが, プログラムに対する電子透かしには, 画像等のデータに対する電子透かしとはやや異なる性質が要求されると考えられる. 第 2.2.2 節では, プログラムに対する電子透かしが, 従来のデータに対する電子透かしとどのような点において異なるかを議論し, その満足すべき性質を形式的に与える. 本論文で述べる条件はそれほど複雑なものではなく, むしろ多くの人が直観的に思い描くようなものである. しかし, プログラムに対する電子透かしを議論する出発地点においてその目標を明確にすることは, きわめて重要であると考えられる. 本研究の 2 番目の目標は, プログラムに対する電子透かしの具体的な実現例を示すことで

ある. 第 2.2.2 節でも議論するが, コンピュータプログラムに対しては, 絶対的に安全な電子透かしは実現できない. プログラムの価値や攻撃者（不正を行なおうとするユーザ）の能力, 透かし実現にかけるコストなどの要素を総合的に判断し, 具体的な透かしを実現する必要があると考えられる. 本論文第 2.2.3 節では, きわめて単純な攻撃だけを想定し, 比較的低コストで実現可能な電子透かしの一例を与える. 本例で与える方式は, あくまでも人工的な環境における一実現例に過ぎない. しかし, 第 2.2.3 節で行われるような議論を現実に則して綿密に展開することで, 実用的な電子透かしが実現できると考えられる.

2.2.2 プログラムに対する電子透かしの安全性定義

透かしの妥当性

本節ではプログラムに対する電子透かしのモデルについて説明する. ここでは, 実行環境が異なれば, プログラムの動作も変化するような仕組みを持つシステムを考える. 例えば, プログラム中から環境変数の値を参照し, その値によってプログラムの動作を切替えることが可能なシステムや, 動的リンク機構を使うことにより, プログラム内で使用するモジュールが実行時に決定されるようなシステムを考える. このような仕組みは多くのオペレーティングシステムで採用されており, 妥当な仮定であると考えれられる.

本節では, まずプログラムに対する電子透かしの妥当性の形式的定義を示す. P を全てのプログラムからなる集合とし, E をプログラムの実行環境全てからなる集合とする. また, 環境 $e(\in E)$ におけるプログラム $p(\in P)$ の動作を $e[p]$ と書く. 2 つのプログラム p_1 と p_2 がある環境 $e \in E$ において, どのような入力や操作に対しても同じ動作をする時に, この 2 つのプログラムは環境 e のもとで等価であると言う. 上で定義した記号を使うと, プログラム p_1 と p_2 が環境 e において等価であるということは, $e[p_1] = e[p_2]$ と表すことができる. 以下では $e[p_1] = e[p_2]$ を $p_1 \equiv_e p_2$ と記述する. 2 つのプログラム p_1, p_2 がどのような実行環境でも等価であるとき, すなわち, 任意の $e \in E$ に対して $p_1 \equiv_e p_2$ であるとき, その 2 つのプログラムは等価であると言う. プログラム $p \in P$ に対し, 透かし w を埋め込んで得られるプログラムを p_w と表記する. また, 電子透かしを埋め込んでいないプログラム（ p そのもの）をとくに p_ϕ と書く. 透かしを検出する者（特権ユーザ, 著作権者, 調停機関等, 以下ではサーバと呼ぶ）は, E の中から, 一般ユーザの実行環境となる確率が非常に低いもの（例えば, 環境変数に特殊な値が設定されているような環境）を一つ選びだし, これを $s(\in E)$ として区別しておく. 電子透かしが妥当であるとは, 以

下の 2 条件が成立するときをいう.

条件 1 任意の透かし w, （s 以外の）任意の実行環境 $u \in E - \{s\}$ に対し, $p_w \equiv_u p_\phi$.

条件 2 任意の異なる透かし w, w' （どちらか片方は ϕ でも良い）に対し, $p_w \not\equiv_s p_{w'}$.

条件 1 は, 電子透かしが埋め込まれたプログラムとオリジナルプログラムは, どの一般ユーザの実行環境においても等価な動作をすることを意味している. すなわち, 透かしを入れたことにより, ユーザにとってのソフトウェアの価値, 品質が劣化していないことを保証する条件である. 条件 2 は, 異なる透かしの埋め込まれたプログラムは, サーバによって区別することが可能であることを意味している. 条件 2 の系として, 任意の透かし $w(\neq \phi)$ に対し $p_w \not\equiv_s p_\phi$, すなわち, 透かしの入ったプログラムは, サーバの実行環境においては, オリジナルのプログラムとは異なる動作をすることが保証される.

電子透かしの安全性

電子透かし法は, 前節で述べた妥当性を持つように設計されなければならない. 一般に, プログラムの不正な再配布を行なおうとするユーザ（以下では不正ユーザと呼ぶ）は, 埋め込まれた透かしを削除しようとしたり, 改変しようとしたりする恐れがあるため, そのような不正ユーザの操作に対しても妥当性が保証される方式であることが望ましい. 不正ユーザが, 透かしの除去や改変を意図してプログラムに対して行なう操作すべてからなる集合を A とし, プログラム $p \in P$ に対して操作 $a \in A$ を行なった結果得られるプログラムを $a(p)$ と書く. プログラムに対する透かしの安全性は, 以下のように定式化できる.

- 改変不可能性

  任意の透かし w と任意の操作 $a \in A$ に対し $a(p_w) \equiv_s p_w$ ならば, 透かし方式は改変不可能性を有するという. これはすなわち, 不正ユーザが（実行可能な）どのような操作を行なったとしても, 透かしを破壊したり消去することができず, サーバは元の透かしを認識できるような性質である.

- 消去不可能性

  任意の透かし w と任意の操作 $a \in A$ に対し $a(p_w) \not\equiv_s p_\phi$ ならば, 透かし方式は消去不可能性を有するという. これは, 不正ユーザが透かし情報を完全に消去することができないような性質を定式化したものである.

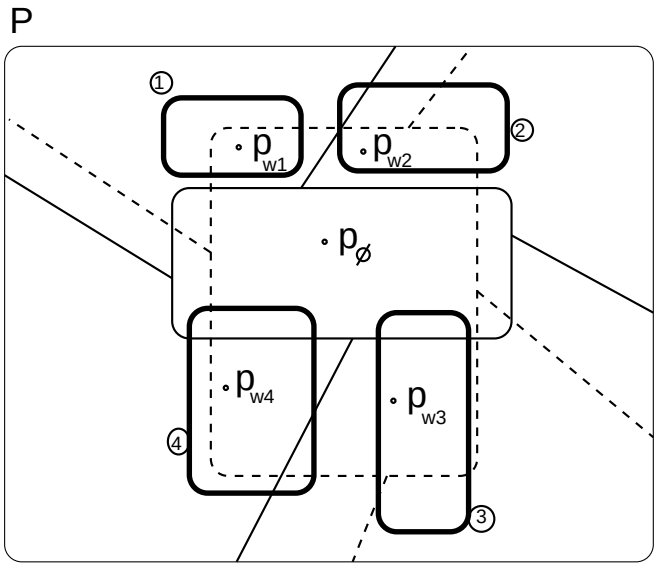

図 2.1 電子透かしの安全性

- 偽造不可能性

  W を, 可能な透かし全体のある部分集合とする. 任意の透かし $w, w' \in W$ と任意の操作 $a \in A$ に対し $a(p_w) \not\equiv_s p_{w'}$ ならば, 透かし方式は W に関して偽造不可能性を有するという. これは, 不正ユーザが他者の透かしを偽造できないような性質を定式化したものである.

妥当性を満たすある透かし方式が改変不可能性を有するならば, その方式は消去不可能性や偽造不可能性も有することが, 定義より容易に証明できるが, 逆は必ずしも成り立たない. これより, 改変不可能性を強い安全性, 消去不可能性や偽造不可能性を弱い安全性と位置付けることができる.

これらの安全性を模式的にあらわすと図 2.1 のようになる. すべてのプログラムからなる集合 P は, 関係 $\equiv_s$ を用いて同値類に分割できる. 図 2.1 における細い実線は, $\equiv_s$ による同値類分割をあらわす. 同様に, ある環境 $u \in E - \{s\}$ に対し, $\equiv_u$ を用いて P を同値類分割することも可能である. 一般に, $\equiv_s$ による同値類分割と $\equiv_u$ による同値類分割は異なるものとなる. 図 2.1 における破線は, $\equiv_u$ による同値類分割をあらわす. 関係 $\equiv_u$ $(\equiv_s)$ による同値類分割において, プログラム p の属する同値類を $p^{\equiv_u}(p^{\equiv_s})$ と書くことにする. いま, 妥当な電子透かし法を用いて透かし $w(\neq \phi)$ をプログラム p に埋め込み, p_w を得た

とする. 透かしの妥当性より, $p_{\phi}^{\equiv^s} \neq p_w^{\equiv^s}$ かつ $p_{\phi}^{\equiv^u} = p_w^{\equiv^u}$ である. すなわち, ユーザの視点からは p_w は p_ϕ と等価であるが, サーバの視点からは p_w は p_ϕ と異なるものであると識別できる. ここで, $A(p_w) = \{a(p_w)|a \in A\}$ と定義し, 図 2.1 では $A(p_w)$ を p_w を含む太線で表現するものとする. もし, 任意の透かし w に対し, $A(p_w) \subset p_w^{\equiv^s}$ ならば（図 2.1 中の①）, その透かし法は改変不可能性をもつ. この場合, 不正ユーザが A に属する任意の操作を行なったとしても, サーバの視点からは何の問題もなく $a(p_w)$ から p_w を特定できる. また, 任意の透かし w に対して $A(p_w) \cap p_{\phi}^{\equiv^s} = \phi$ ならば（図 2.1 中の①,②）, その透かし法は消去不可能性を持つ, さらに, 透かしの部分集合 $W = \{w_1, w_2, w_3, w_4\}$ と W に属する任意の透かし $w, w' \in W$ に対して $A(p_w) \cap p_{w'}^{\equiv^s} = \phi$ ならば（図 2.1 中の①,③）, その透かし法は偽造不可能性をもつ.

2.2.3 プログラムに対する電子透かしの構成法

本節では, プログラムに対する電子透かしの一実現例を考える. 本節で述べる手法は, 基本的にはどのようなプログラミング言語に対しても適用可能であるが, ここでは特に, ネットワークを介して流通する機会の多い Java プログラムを対象として議論を行う.

Java 言語

ここではまず, Java 言語のうち, 透かしの実現と関連する事項について簡単に述べる. Java の実行形式プログラムは, 複数の「クラスファイル」と呼ばれるファイルからなる. クラスファイルは C 言語のオブジェクトファイルに相当し, Java のソースコードをコンパイルすることによって得られる. Java プログラムの実行時には, あるクラスファイルの内部から, 他のクラスファイルに記述されたメソッドやオブジェクトを参照することが可能である. この性質を利用すると, 一部のクラスファイルを入れ替えることにより, プログラムの動作を変化させることも可能となる. 後に詳しく述べるが, 提案手法ではこの性質を利用して透かしの抽出を行う.

Java プログラムは通常, そのプログラムを構成するクラスファイルの集合として配付されることが多い. しかし, クラスファイルには, Java 仮想マシンの命令セットであるバイトコードの他に, クラス名やメソッド名などの情報も含まれているため, 非常に高い精度で, クラスファイルから元のソースコードを得ることが可能である. 実際, クラスファイルの配付を受けたユーザは, Mocha [17], JODE [12], DECAFE [8] などの優秀な逆コ

ンパイラを利用することで, プログラムのソースコードを得ることができる. したがって, Java プログラムに対する電子透かしを議論する際には, 不正行為を行おうとするユーザがソースコードを参照可能であることを仮定して, 安全性の議論を行う必要がある.

不正ユーザの行い得る操作

前節で述べた改変不可能性, 消去不可能性, 偽造不可能性等の性質を議論するためには, 不正ユーザの実行可能な操作をあらかじめ定式化しておく必要がある. これは, 現実世界では非常に難しい問題であり, プログラムの規模や複雑さ, 工学的な技術水準やプログラム自体が持つ金銭的価値などを総合的に判断して決定されるべき事項である. ここでは簡単のため, 不正ユーザは, 以下に述べるような比較的単純かつ低コストの攻撃のみを行うものと仮定し, それらの攻撃に対する耐性を持った透かし方式を考察する.

1. 変数名, メソッド名やクラス名を置換する: 不正ユーザは, クラスファイルを逆コンパイルしてソースコードを入手する. ソースコード中に復元された変数, メソッドやクラスの名前を変更し, 変更後のソースコードをコンパイルして変造プログラムを得る.

2. 互いに依存しない操作の実行順序を変更する: ソースコード中に, たとえば 「`a = 3; b = 5;`」(変数 a に 3 という値を代入し, その後, 変数 b に 5 という値を代入する) という記述があったとする. この場合, 該当箇所を「`b = 5; a = 3;`」としてもプログラムの動作には影響しない. 不正ユーザは同様の操作により、プログラムの改変をはかる.

3. 最適化器により冗長を除去する: 電子透かしは, 本来のプログラムの動作には必要のない, 冗長な部分であると考えることができる. 不正ユーザは, 逆コンパイルして得られたソースコードに最適化操作を加え, 透かしの除去を試みる. ただし, オプティマイザ (最適化器) はプログラムに対して, どのような実行環境においても等価な動作をするような変換しか行なわないと仮定する. 実際に多くのオプティマイザはこのような動作しか行なわないと考えられる.

透かしの基本要素

本実現方式においては, 透かしは以下の 4 つの要素から構成される.

1. 透かし変数

2. トラップメソッド

3. トラップメソッドの特殊定義

4. 透かし出力部

透かし変数は, 透かし情報を格納しておくための変数であり, 透かし検出部から参照できる位置において宣言されている必要がある. 透かし変数はかならずしも独立した変数である必要はなく, たとえば配列内の未使用要素や, ポインタ等によって指定された領域を, 透かし変数として利用してもよい. トラップメソッドは, プログラム中のメソッドの一種であり, ユーザの実行環境において取り得る戻り値があらかじめ予想可能であるようなメソッドである. 一方, トラップメソッドの特殊定義は, 一般のユーザの実行環境では考えられないような値を戻り値として持つよう, トラップメソッドを再定義するものである. トラップメソッドの一例としては, たとえばプログラム実行時の西暦を戻り値とするようなメソッドが考えられる. 通常のユーザの実行環境では, そのようなメソッドは必ず正の整数値を返すことがわかっている. この場合, トラップメソッドの特殊定義としては, たとえば西暦として負の整数を返すような再定義が考えられる. トラップメソッドを用いてある種の条件判定を行い, 一般ユーザの環境では何も実行されないが, サーバの実行環境では透かし変数の値が出力されるようなコードを透かし出力部とする. ユーザにプログラムを配付する際には, オリジナルプログラムに対して透かし変数へ透かし情報の代入を行う処理を加え, また透かし出力部を追加したプログラムを配付する. その際, 透かし変数は何か, どのメソッドがトラップメソッドでその特殊定義が何か, といった情報は秘密にしておく.

図 2.2 に, 非常に簡単な電子透かし入りプログラムの例をあげる. この例では `w` が透かし変数となっており, ここでは 27 という値を透かし情報として与えている. トラップメソッドは `Detect` クラスのメソッド `de` であり, 一般のユーザの実行環境では 0 から 3 の整数が戻り値となる. 一方, このメソッドの特殊定義では, 戻り値が常に 5 となるようメソッドを再定義している. 透かし出力部は `main` メソッド中の `if` 文である. 一般ユーザの実行環境では `d.de` は常に 4 未満の値を返すため `System.out.println` は実行されないが, サーバ環境において `Detect` クラスを特殊定義に置き換えて実行することで, 透かし変数の値が出力されることになる.

```
public static void main(String[] args){
  int w;
  w = 27;
  ⋮
  Detect d = new Detect(5);
  if (d.de() > 4) System.out.println(w);
  ⋮
}
```

一般ユーザ向けクラス定義

```
class Detect{
  int m = 1;
  public int de(){
    return m;
  }
  Detect(int n){
    m = n % 4
  }
}
```

特殊定義

```
class Detect{
  public int de(){
    return 5;
  }
}
```

図 2.2 電子透かしの例

この方法は, 不正ユーザがクラス名を変更すると, そのままの形では利用できなくなる. 不正ユーザが行なうクラス名の変更に対する対策について述べる.

まず, サーバはユーザにプログラムを配布する前に, 全てのクラスファイル内にダミーの main メソッドを埋め込んでおく. 通常のソフトウェアの利用法では, このダミーのメインメソッドは利用されない. そして, その main メソッド内にそのクラス名を特定するためのなんらかの情報（例えばクラス名をサーバが持つ秘密鍵で暗号化したもの）を出力するコードを書いておく. このような仕掛けを埋め込んでおくことにより, 不正ユーザがクラス名を変更したとしても, サーバはそのクラスを単体のプログラムとして実行することで, 元の名前を知ることができる. したがって, サーバはクラス名を元どおりに復元することができるため, クラス名が変更されても電子透かしの検出は可能である.

偽装コードによる透かしの保護

前節の例のような単純な電子透かしでは, 不正ユーザによる比較的単純なプログラム解析によって, 透かしの除去あるいは改変が可能となるおそれがある. そこで, 埋め込んだ透かしの除去や改変が困難となるよう, プログラムに偽装コードを埋め込むことを考える. 一般的な難読化手法に加えて, 以下のような偽装コードが考えられる.

- 透かし情報の導出を複雑にする: 前節の例では, 透かし情報として 27 という即値を透かし変数に代入していたが, これを即値代入の形式ではなく, より複雑な計算の結果得られる値として, 透かし変数に代入するようプログラムの改変を行う. この偽装により, どの変数が透かし変数であるか, 不正ユーザにわかりにくくする効果が期待される.

- 重要な演算を, 透かし情報に依存するよう変更する: プログラム中で重要な役割を果たす定数などを, 透かし変数を用いて表現する. たとえば前例のプログラムにおいて 5 という定数を表現している箇所があった場合, それを「w の 1 の位マイナス w の 10 の位」と置き換えるような操作を考える. 透かし変数の値 27 がユーザによって操作されない限り, この置換によってプログラムの動作が変化することはない. 逆に, 透かし変数の値を操作しようとするとプログラムが異常な動作をすることになるため, このような偽装コードは, 不正ユーザによる透かし改変を抑止すると考えられる.

- 恒真または恒偽となる論理演算の混入: 透かし情報の値により, 常に真または常に

偽となる論理式が考えられる. そのような論理式を利用することで, プログラムの動作を変えることなく, 複雑な条件分岐などをプログラム中に混入させることが可能となる. もし不正ユーザが透かし変数の値を操作した場合, 当初とは異なる条件分岐などが発生し, プログラムの動作が異常となるため, この偽装コードも不正ユーザによる透かし改変を抑止すると考えられる.

透かしの安全性

2.2.3 節では, 不正ユーザの行い得る操作として 3 種類の攻撃を仮定した. タイプ 1 の攻撃, すなわち変数名やメソッド名を置換するような操作の集合を A_1, タイプ 2, すなわち互いに依存しない操作の実行順序を変更するような操作の集合を A_2, タイプ 3 の攻撃, すなわち最適化器により冗長を除去するような操作の集合を A_3 とする. 提案方式がこれらの攻撃に対して強い安全性（改変不可能性）を有することを示すためには, 任意の操作列 $a_1, a_2, \ldots, a_m \in A_1 \cup A_2 \cup A_3$ および透かしを埋め込んだプログラム p_w に対し, $a_1(a_2(\cdots a_m(p_w)\cdots)) \equiv_s p_w$ であることを示せば良い.

補題 1 任意の $a \in A_1 \cup A_2 \cup A_3$ に対し, $a(p_w) \equiv_s p_w$ である.

証明 まず $a \in A_1$ のときを考える. 変数名やメソッド名の出現などを機械的に置換した場合, ソースファイルとしては元のプログラムとは異なるものとなるが, プログラムの動作は変化しない. また, クラス名の変更については, 2.2.3 節で述べたように, サーバが元の名前に戻すことができる. よって, $a(p_w) \equiv_s p_w$ である. 次に $a \in A_2$ のときを考える. この際, 厳密には, 2 つの操作が「互いに依存しない」とはどういうことであるか定式化する必要があるが, ここでは, その 2 つの操作の実行順序を入れ換えてもプログラムの動作が変化しないような操作の対である, と定義する. この場合あきらかに $a(p_w) \equiv_s p_w$ である. 最後に $a \in A_3$ のときを考える. 透かし変数は, 本来のプログラムの動作とは無関係であり, 本質的に冗長な存在であると考えられる. しかし, 基本的に最適化器は, あらかじめ定められたコードの等価変換しか行わないため, 十分な複雑さを有する偽装コードを使用することで, 最適化器が透かし変数の冗長性を検知する確率を格段に小さくすることが可能となる. よって $a(p_w) \equiv_s p_w$ としてよい.

系 1 任意の操作列 $a_1, a_2, \ldots, a_m \in A_1 \cup A_2 \cup A_3$ に対し, $a_1(a_2(\cdots a_m(p_w)\cdots)) \equiv_s p_w$ である.

```
public static void main(String[] args){
  int w = 27; // Watermark
  try{
   FileInputStream in = new FileInputStream(``/secret'')
   in.close();
  }catch(FileNotFoundException e){
  ここに本来のプログラムの動作を記述する.
   System.exit(-1);
  }catch(IOException ioe){
   System.exit(-1);
  }
  System.out.println(w);
  }
}
```

図 2.3 ファイルの存在をチェックして透かしを検出する方式

他の電子透かしの実現例について

2.2.3 節では, ユーザに配布したクラスファイルの一部をサーバの持つ秘密のクラスファイルに置き換えてプログラムを実行することで電子透かしを検出する方法について説明した. しかし, その方法はあくまで電子透かし検出法の一つの例であり, 他の電子透かし検出方式を利用することも可能である.

図 2.3 は, 特定のファイルがあるかどうかをチェックして電子透かしを検出するプログラムの例である. このプログラムでは, `/secret` というファイルが存在するかどうかをチェックし, もしそのファイルが存在すれば, 電子透かしを表示し, そのファイルが存在しなければ, 本来のプログラムの動作のみを行うという仕組みになっている. ユーザにこのプログラムを配布する際には, `/secret` というファイルは配布しないようにしておくことで, ユーザは通常通りプログラムを利用することができ, サーバが電子透かしを検出するときには, `/secret` というファイルが存在する環境のもとでこのプログラムを実行することにより, 電子透かしを検出することができる.

2.2.4 まとめ

コンピュータプログラムに対する電子透かしについて議論した．画像や音声に対する透かしとは異なり, プログラムに対する透かしは原理的には人手による解析で取り除くことが可能である．したがって, 工学的見地からは, 除去, 改変に非常に大きなコストが必要であるような透かしの実現が求められる．本論文では透かしの妥当性や安全性を形式的に議論し, プログラムに対する電子透かしが目指すべき方向性を与えた．また Java プログラムに対する電子透かしの簡単な実現例も示した．

提案手法は, Java 以外の他のプログラム言語に対しても適用可能である．実際, C, C++, Pascal 等多くのプログラム言語に対して, 2.2.3 節で示した透かしの埋め込み方法が適用できる．また, Windows の動的リンクライブラリ (DLL) 等, 実行時にプログラムの動作を決定できる機構はかなり一般的に利用されているため, 提案した透かし検出法も広い環境で利用可能である．

2.3. 結託攻撃に耐性を持つ電子透かし用符号

2.3.1 はじめに

ディジタルコンテンツ配信システムにおいては, コンテンツの不正コピーを防止することが重要である．2.1 節でも述べたように, 電子透かし法を用いて, 不正コピーを行ったユーザを特定できるようにすることで, 不正コピーの抑止を行うことができる．不正コピーを行おうとするユーザは, あらかじめコンテンツに埋め込まれている電子透かしの除去, 改ざんを試みると考えられる．画像や音声データなどに対する電子透かしへの攻撃の一手法に結託攻撃がある．結託攻撃とは, 複数のユーザが結託し, それぞれのコンテンツを比較することで電子透かしが埋め込まれている位置を特定し, 電子透かしの除去, 改ざんを行うという攻撃である．

Boneh らは, 結託攻撃を防ぐために c-secure 符号 with ε error [3, 4] という符号を提案した．この符号を用いてユーザ ID を符号化し, 電子透かしとして埋め込むことで, Marking Assumption と呼ばれる仮定のもとで c 人（c は符号生成時に定めるパラメータ）までの結託に対して確率 $1-\varepsilon$ で安全であることが示されている．しかし, この符号

の符号長は結託者数 c が大きくなると非常に大きくなることが文献 [14] で指摘されている. 一方, 文献 [10] においてGuthらによって, 結託攻撃を行って作成したコンテンツに対し, さらにランダムエラーを付加する攻撃が提案され, またランダムエラーが付加された場合にも安全な $O(\log n)$ c-secure 符号が提案された. しかし, $O(\log n)$ c-secure 符号は, ユーザ数 n と 結託者数 c が大きい場合, 符号長が非常に大きくなるという問題がある. 文献 [14] で, 村谷によって c-secure CRT 符号 with ε error が提案された（以下では c-secure CRT 符号と呼ぶ）. 符号長は $O(n^{1/k})$ （ k はセキュリティパラメータ）で, 大きな n と大きな c に対しても符号長が比較的短いという性質を持つ. また, この符号はパラメータ c の値を変化させることによって, ランダムエラーに対しても耐性を持たせることが可能である.

コンテンツ配信システムが広く利用されるためには, 電子透かしが埋め込まれたコンテンツに対し不正ユーザが何らかの攻撃を加えても, サーバは正しく不正者を特定できる必要がある. サーバによる不正者の特定の失敗は 2 種類考えられる. 1 つは, サーバが結託者を誰一人特定できない場合, もう 1 つは, サーバが誤って結託に加わっていないユーザを特定してしまう場合である. 後者は無実のユーザに罪を着せてしまうもので, 決して起こってはならない誤りであるといえる. c-secure CRT 符号では, コンテンツにランダムエラーが付加される場合には, 符号長を非常に大きくしなければ, 無実のユーザを誤って検出する確率を十分小さくすることができない. そこで, 符号長はそのままで検出誤りを減少させるために, c-secure CRT 符号の新しいトレースアルゴリズム A_3 が提案された [28, 30]. トレースアルゴリズムとは, コンテンツから取り出された透かし情報から, ユーザ ID を特定するアルゴリズムである.

2.3.2 c-secure CRT 符号

本節では, まず c-secure CRT 符号 [14] の構成とトレースアルゴリズム [14] について説明し, 次にランダムエラーを考慮したトレースアルゴリズム A_3 [28, 30] について説明する.

n をコンテンツ配信システムにおいて想定しているユーザの人数とし, 各ユーザには $\mathbb{Z}_n$ から選ばれた一意のユーザ ID が与えられているものとする. また, c をシステムが想定する結託者数の上限とする.

ユーザ ID を中国人剰余定理を利用して, 剰余の組で表現する. 各剰余は, $O(n)$ n-secure 符号 [3] によって符号化する. c-secure CRT 符号 は $O(n)$ n-secure 符号により符号化さ

れた剰余を連接した符号である.

法

$\varepsilon_1, \varepsilon_2$ を $0 < \varepsilon_1 < 1, 0 < \varepsilon_2 < 1, (1-\varepsilon_1)(1-\varepsilon_2) > 1-\varepsilon$ を満たす正の数とする. ここで, ε は, 想定しているトレースエラーの上限である. セキュリティパラメータ k を選択し, 次式を満たす整数 l を求める.

$$[1 - \prod_{i=0}^{l-1}\{1 - (1 - \frac{1}{p_{k+i}})^c\}]^{\lceil c(k+l)/2 \rceil C_{k+l} \times 2^{k+l}} \geq 1 - \varepsilon_2.$$

次に, $k' = \lceil c(k+l)/2 \rceil$ と置き, 次の 2 つの条件を満たすように, k' 個の相異なる整数 $p_0, p_1, \cdots, p_{k'-1}$ を選ぶ. 一般性を失うことなく $p_0 < p_1 < \cdots < p_{k'-1}$ と仮定する.

条件 1 $i \neq j$ なる p_i と p_j は互いに素.

条件 2 $\prod_{i=0}^{k-1} p_i \geq n$.

$p_0, p_1, \cdots, p_{k'-1}$ はユーザ ID を剰余の集合に分解する際の法として用いられる. 以降では, $p_0, p_1, \cdots, p_{k'-1}$ の平均値を $\bar{p}$ で表す. $\bar{p} = (p_0 + p_1 + \cdots + p_{k'-1})/k'$ である.

剰余

各ユーザ ID u に対し, $r_i \equiv u \pmod{p_i}$ を満たす $r_i \in \mathbb{Z}_{p_i} (0 \leq i \leq k'-1)$ を計算する. 法を選ぶ際に用いた 2 つの条件より, u の剰余が k 個与えられれば, 中国人剰余定理によってユーザ IDu を計算することができる.

内部符号

各法 p_i に対して, 内部符号 $\Gamma_0(p_i, t)$ が定義される [3]. $\Gamma_0(p_i, t)$ の符号語 $w_i^{(j)}$ は次のように定義される:

$$w_i^{(j)} = \underbrace{00\cdots0}_{t\times j} \ \underbrace{11\cdots1}_{t\times(p_i-j-1)} \ .$$

ここで, t は $t \geq -\log_2[1-(1-\varepsilon_1)^{\frac{1}{2k'}}]$ を満たす最も小さい整数である. 内部符号中の t 個づつのビット列の集まりをブロックと呼ぶ. 符号語 $w_i^{(j)}$ は, j 個の 0 からなるブロック

と $(p_i - j - 1)$ 個の 1 からなるブロックにより構成される. 以降では, 一番左のブロックを 0 番目, 一番右のブロックが $p_i - 2$ 番目のブロックというように左から順番に呼ぶこととする.

値	内部符号
0	11111 11111 11111
1	00000 11111 11111
2	00000 00000 11111
3	00000 00000 00000

表 2.1 法 4, ブロックの大きさ 5 の内部符号

内部符号の例として, 法 4 でブロックの大きさが 5 の内部符号 $\Gamma_0(4,5)$ を表 2.1 に示す.

c-secure CRT 符号

c-secure CRT 符号はユーザ ID の各剰余を内部符号によって符号化したものを連接した符号である. 以降では, c-secure CRT 符号を $\Gamma(p_0, p_1, \cdots, p_{k'-1}; n, t)$ と記述する. ユーザ ID $u \in \mathbb{Z}_n$ に対する c-secure CRT 符号の符号語 $W^{(u)}$ は次のように定義される:

$$W^{(u)} = w_0^{(r_0)} \| w_1^{(r_1)} \| \cdots \| w_{k'-1}^{(r_{k'-1})}.$$

ここで, $r_i \equiv u \pmod{p_i} (0 \leq i \leq k' - 1)$ であり, $\|$ は連接を表す. $\prod_{i=0}^{k-1} p_i \geq n$ が満たされているので, 任意の k 個の内部符号が正しく復号できれば, 中国人剰余定理を利用することで, ユーザ ID u を復号できる.

c-secure CRT 符号のトレースアルゴリズム

次に, c-secure CRT 符号のトレースアルゴリズムについて説明する. c-secure CRT 符号のトレースアルゴリズムは次の 5 つのステップからなる.

STEP 1: コンテンツから埋め込まれている透かしを取り出す. 取り出したデータを x で表す. x の長さは $\bar{p}k't$ である.

STEP 2: x を各内部符号に対応するように k' 個に分割する.

$$x = x_0 \| x_1 \| \cdots \| x_{k'-1}.$$

ここで, x_i の長さは $t(p_i - 1)$ である.

STEP 3: 各 x_i に対し, 次のアルゴリズムを実行する.

input x_i;
for $(min = 0;\ min < p_i - 1;\ min ++)$
    if $(H_{min}(x) > 0)$ break;
for $(max = p_i - 1;\ max > min;\ max --)$
    if $(H_{max-1}(x) < t)$ break;
output min and max;

ここで, $H_{min}(x_i)$ は, x_i の min 番目のブロックのハミング重みを表す. このアルゴリズムの出力 min, max をそれぞれ $r_i^{(-)}, r_i^{(+)}$ と表す.

STEP 4: ユーザ ID u に対して, $\mathcal{D}(u)$ を次のように定義する.

$$\mathcal{D}(u) = |\{i \in \mathbb{Z}_{k'} | (u \equiv r_i^{(-)} \pmod{p_i}) \vee (u \equiv r_i^{(+)} \pmod{p_i})\}|.$$

各ユーザ ID u に対して, $\mathcal{D}(u)$ を計算する.

STEP 5: $\mathcal{D}_{th} = k + l$ と定義する. $\mathcal{D}(u) > \mathcal{D}_{th}$ が成り立つならば, ユーザ u が結託に加わったと判断する.

このトレースアルゴリズムは, c 人までの結託攻撃に対しては十分安全なトレースアルゴリズムである. しかし, 結託攻撃に加えて符号語にランダムエラーが加えられた場合, このトレースアルゴリズムでは高い確率で検出誤りが起こる. c の値を変更することによって, c-secure CRT 符号をランダムエラーが加えられても安全なようにすることは可能であるが, 符号長が増加するという問題が起こる. この問題を解決するために, 新しいトレースアルゴリズム A_3 が提案された [28, 29, 30]. 次に トレースアルゴリズム A_3 について説明する.

トレースアルゴリズム A_3

トレースアルゴリズム A_3 は, c-secure CRT 符号のトレースアルゴリズムの STEP 3 を変更したアルゴリズムである. ここでは, 符号語に対し, 一様に確率 e でランダムエラーが付加されているものとし, さらにトレースを行うサーバはあらかじめ何らかの方法でエラー確率 e を求めることができるものと仮定する. トレースアルゴリズム A_3 では, ランダムエラー確率 e からスレッシュホールド w_{th} を以下のように計算する: コンテンツから取り出された c-secure CRT 符号の符号語（攻撃を受けた符号語）の各内部符号部分を考える. 各内部符号は次の 3 つの種類のブロックから構成されていると考えられる. (1) 結託攻撃が行われたブロック（Type 1 と呼ぶ）, (2) 全て 0 からなるブロックにランダムエラーが加えられてできたブロック（Type 2）, (3) 全て 1 からなるブロックにランダムエラーが加えられてできたブロック（Type 3）. これら 3 つのタイプのブロックそれぞれのハミング重みの確率分布は次のようになる:

$$
\begin{aligned}
p_{Type1}(w) &= \binom{t}{w}\left(\frac{1}{2}\right)^t, \\
p_{Type2}(w) &= \binom{t}{w}e^w(1-e)^{t-w}, \\
p_{Type3}(w) &= \binom{t}{w}e^{t-w}(1-e)^w.
\end{aligned}
$$

w_{th} を $p_{Type2}(w) > p_{Type1}(w)$ を満たす最大の整数 w と定義する. つまり, あるブロックにおいて, そのブロックのハミング重みが w_{th} より大きければ, そのブロックは, 全て 0 のブロックにランダムエラーが加えられてできたブロック (Type 2) である可能性より, 結託攻撃によって作られたブロック (Type 1) である可能性のほうが大きいということを意味する. 最後に, サーバは小さい正整数 ad_{th} を選ぶ.

トレースアルゴリズム A_3 の STEP 3 は以下である.

STEP 3 of Algorithm A_3:

input x;
for $(min = 0;\ min < p_i - 1;\ min{+}{+})$
  if $((H_{min}(x) > w_{th}) \wedge \cdots \wedge (H_{min+ad_{th}-1}(x) > w_{th}))$
    break;
for $(max = p_i - 1;\ max > min;\ max{-}{-})$
  if $((H_{max-1}(x) < t - w_{th}) \wedge \cdots \wedge (H_{max-ad_{th}}(x) < t - w_{th}))$

break;

output min and max;

2.3.3 結託耐性符号に対する攻撃

本節では, まず従来想定されていた結託攻撃について簡単に説明する. 従来, 結託耐性符号は, この攻撃に対して安全であるように設計されてきた. 次に, c-secure CRT 符号とトレースアルゴリズム A_3 に対する新しい結託攻撃を提案する. この攻撃法は, 符号化法とトレースアルゴリズムの性質を利用した攻撃法である. さらに, この攻撃の効果についても述べる. 最後に, この攻撃が効率的である理由について考察する.

Marking Assumption [3] により, 結託したユーザが異なるコンテンツを比較して見つけた透かしの埋め込み位置には, 0 と 1 がそれぞれ埋め込まれている. 結託したユーザが行える操作は, 見つかった各埋め込み位置に対して 0 と 1 に対応する値のどちらかを選んで新しいコンテンツを作成することである.

文献 [14, 28, 29, 30] での安全性評価の際には, 次のランダムエラー付加型結託攻撃を考えている. 今後, この攻撃を Uniform Selection Attack と呼ぶ. Uniform Selection Attack は次に示す 3 つのステップからなる.

Uniform Selection Attack

1. 結託者は自分たちが持つコンテンツを比較して電子透かしが埋め込まれている位置を見つける.

2. その検出した位置に 0 に対応する情報が埋め込まれているコンテンツと 1 に対応する情報が埋め込まれているコンテンツの 2 種類がある. 結託者は, どちらのコンテンツが 0 に対応する情報が埋め込まれているのか, あるいは 1 に対応する情報が埋め込まれているかは分からない. したがって, 結託者は, その検出した位置について 2 つのコンテンツのどちらかを採用して新しいコンテンツを作ることはできるが, 作成したコンテンツのその位置に埋め込まれている情報が 0 なのか 1 なのかは分からない (Marking Assumption).

   そこで, それぞれの検出した位置において, どちらのコンテンツを採用するかを 1/2 の確率で決定する.

3. 前ステップで作成したコンテンツにランダム誤りを付加する.

本論文では, この攻撃よりも検出誤り確率が増加する攻撃を提案する. 提案する攻撃は, トレースアルゴリズム A_3 で用いていたスレッシュホールド w_{th} についての知識を利用したものである. 今後, この攻撃を Threshold Based Attack と呼ぶ. Theshold Based Attack は以下の 5 ステップからなる.

Threshold Based Attack

1. 結託者は自分たちが持つコンテンツを比較して電子透かしが埋め込まれている位置を見つける.

2. 結託者の中で, 最も多くの位置で異なるコンテンツを持つ 2 人を選ぶ.

3. コンテンツに付加するランダムエラー確率 e を決める. トレースアルゴリズムで用いられるスレッシュホールド w_{th} を e から計算する.

4. 2 人の持つコンテンツから新しいコンテンツを作成する. 透かしが埋め込まれていることを検出した位置それぞれについて, 2 人のユーザのうちどちらの持つコンテンツの値を利用するかは, $w_{th} : t - w_{th}$ の割合で選択することにする. ここで, t はブロックの大きさである.

5. 作成したコンテンツに対し, ステップ 3 で決定したランダムエラー確率 e でランダムエラーを加える.

Threshold Based Attack を行った際の c-secure CRT 符号のトレースエラー確率をコンピュータシミュレーションによって計算した. 図 2.4 はその結果である. 図の $\diamond$ で示されている値は結託に加わっていないユーザを誤って検出した確率を示し, $+$ で示されている値は結託者のうち誰一人検出できなかった確率を示している. これらの値はそれぞれ 1000 回行った実験の平均値である. このシミュレーションで用いたパラメータは文献 [28, 29, 30] で用いられている値と同じである: $c = 15, k' = 52, k = 2, t = 25, p_0 = 100, n = 1.0 \times 10^4, l = 5, L = 2.77 \times 10^6, ad_{th} = 3$. 各パラメータの内容は表 2.2 に示す. 図 2.4 における不連続な部分は, w_{th} の値が変わる境界である.

この攻撃は非常に人工的な攻撃のように見え, また結託者のうちの誰も検出されない確率が Uniform Selection Attack よりも小さいことなどから, 現実にはあまり効果的でな

いように思えるかもしれない. 事実, ランダムエラー確率 e が小さい領域においては, 確率 1 で結託者の一人が検出されている. しかし, 誤って無実のユーザを特定する確率は十分に小さいことが求められる. もし, 検出誤りの確率を十分下げることができなければ, そのシステムは広く利用されないと考えられる. また, この攻撃によって検出誤り確率を増やすことができれば, それだけでシステムを混乱させる要因となり, 敵対者にとってこの攻撃を行う価値があると考えられる.

c	想定している結託者の数
k, m, l	パラメータ (法の項を参照)
t	ブロックの大きさ
p_0	最小の法 $(p_1 = 101, \ldots, p_{k'-1} = 359)$
n	ユーザの総数
L	符号長
ad_{th}	A_3 アルゴリズムにおけるパラメータ

表 2.2 シミュレーションで用いたパラメータ

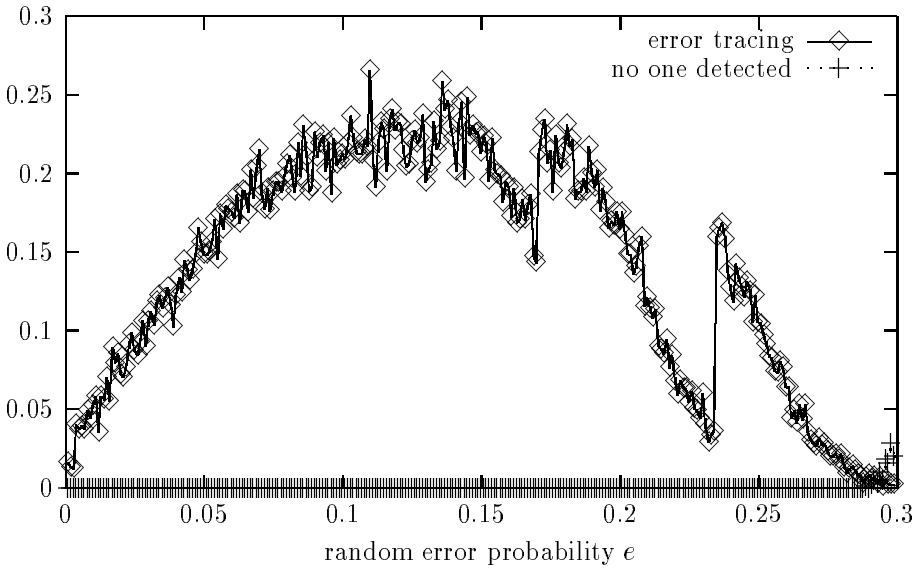

図 2.4 c-secure CRT 符号のトレースエラー確率

考察

本節では, Threshold Based Attack が c-secure CRT 符号と A_3 アルゴリズムに対して効率的である理由を考察する.

Threshold Based Attack は, 透かしに対して結託者が行った改変とランダムエラーとの区別をサーバにつけにくくして検出エラーを起こしている. この攻撃が成功するためには, 改変された内部符号の一定数以上からある特定の無実のユーザが誤って検出されるようにしなければならない. Threshold Based Attack では, このような改変が小さくない確率で起こっている. このことを例を用いて示す. ユーザ ID が連続する 2 人のユーザを考える. ここでは 2 人の ID をそれぞれ 123, 124 とする. c-secure CRT 符号で用いる法を $\{p_i\} = \{\ldots, 57, \ldots, 101, \ldots\}$ とする. この場合, 法 57,101 に対する 2 人の ID の剰余は表 2.3 に示すようになる. このように, 隣り合う ID では, その剰余も高い確率で隣り合い, 対応する内部符号も隣り合うこととなる.

ここで, ID 123 と ID 150 の 2 人のユーザが結託したとする. 表 2.4 に示したように, 法 57 に対する内部符号で攻撃が成功するためには, X で表されているブロックの左端の改変がランダムエラー付加によるものであるとトレースアルゴリズムが認識するときである. この改変された内部符号から無実のユーザ u $(124 \leq u \leq 149)$ が誤って検出される確率は小さくはない. なぜなら, Threshold Based Attack の STEP 4 において, 片方のユーザの符号ともう一人のユーザの符号の組み合わせる割合は $w_{th} : t - w_{th}$ であり, このブロックにおけるランダムエラーによる “0” と “1” の割合と高い確率で同じであるからである. したがって, 高い確率で, X の左端の “1” の数が w_{th} を超えない. よって, トレースアルゴリズムは, X の左端の改変はランダムエラーの付加によるものであると高い確率で考え, トレースエラーを引き起こす.

各内部符号に対し, 同様に考えることができる. 隣り合うユーザ ID の剰余と内部符号は関係があるため, 結託者が十分多くの内部符号において同じ ID（例えば 124 など）を検出されるように改変することはそれほど困難ではないと考えられる.

ID	mod 57 での剰余	mod 101 での剰余
123	9	22
124	10	23

表 2.3 2 つの ID の剰余間の関係

ID	mod 57 における内部符号
123	$\underbrace{00\cdots0}_{t\times9}\underbrace{11\cdots1}_{t\times47}$
150	$\underbrace{00\cdots0}_{t\times36}\underbrace{11\cdots1}_{t\times19}$
結託攻撃を受けた符号語	$\underbrace{00\cdots0}_{t\times9}\underbrace{XX\cdots X}_{t\times29}\underbrace{11\cdots1}_{t\times19}$

表 2.4 内部符号に対する改変

2.3.4 Randomized c-Secure CRT 符号

前節で議論した結果から, 新しい ID 符号化法 Randomized c-secure CRT 符号を提案する. この符号化法は c-secure CRT 符号に基づくものである. 前節で述べたように c-secure CRT 符号においては, 隣り合う ID の剰余と内部符号が相関を持つことが問題であった. Randomized c-secure CRT 符号では, この問題を解決するためにランダム置換を導入する. 提案符号では, 各剰余に対して, ランダムな置換を掛けたものを内部符号を用いて符号化する. また, Randomized c-secure CRT 符号のトレースアルゴリズムも, c-secure CRT 符号のトレースアルゴリズムにランダム置換の逆操作を加えたものとなっている.

一方, 隣り合う ID の剰余が持つ相関を打ち消す方法としては, 例えば ID を $100, 200, 300, \ldots$ というように, 間隔を大きくとる方法が考えられる. この解決法は単純で効果的であるが, 全体として ID が大きな値となり, 符号長が大きくなる問題がある. このような理由から, この方法は採用せず, ランダム置換を用いることとした.

ランダム置換

各法 $p_i(0 \leq i \leq k'-1)$ に対して 0 から $p_i - 1$ までのランダム置換 P_i を定義する. このような k' 個のランダム置換のテーブルをサーバは作成し, 保存しておく.

ここで利用するランダム置換を簡単な例を用いて説明する. 法 $p = 4$ とする. このとき法 4 における剰余は 0,1,2,3 の 4 つである. この 4 つの数字に対して, 表 2.5 のようなテーブルを作成する.

この例では, 0 は 1 に, 1 は 3 にというように置換される. テーブルを保存しておくため, サーバは後で逆置換を行うことも可能となる.

元の値	置換された値
0	1
1	3
2	0
3	2

表 2.5 法 4 におけるランダム置換の例

Randomized c-secure CRT 符号

Randomized c-secure CRT 符号を $\Gamma_R(p_0, p_1, \cdots, p_{k'-1}; n, t)$ で表す. 各ユーザ ID $u \in \mathbb{Z}_n$ に対し, Randomized c-secure CRT 符号の符号語 $W_R^{(u)}$ は次のように定義される:

$$W_R^{(u)} = w_0^{(P_0(r_0))} \| w_1^{(P_1(r_1))} \| \cdots \| w_{k'-1}^{(P_{k'-1}(r_{k'-1}))}$$

$r_i \equiv u \pmod{p_i}, 0 \leq i \leq k'$. Randomized c-secure CRT 符号の符号長は c-secure CRT 符号の符号長と同じ $\bar{p}k't$ である.

トレースアルゴリズム

Randomized c-secure CRT 符号のトレースアルゴリズムを以下に示す. トレースアルゴリズムは 5 つのステップからなり, A_3 アルゴリズムとは STEP 4 と 5 が異なっている.

STEP 1: コンテンツから埋め込まれている透かしを取り出す. 取り出した透かしデータを x で表す. x の長さは $\bar{p}k't$ である.

STEP 2: x を内部符号の符号語に対応するように k' 個に分割する.

$$x = x_0 \| x_2 \| \cdots \| x_{k'-1}.$$

ここで, x_i の長さは $t(p_i - 1)$ である.

STEP 3: 各 x_i に対し, 次のアルゴリズムを実行する.

input x;
for $(min = 0;\ min < p_i - 1;\ min++)$
　　if $((H_{min}(x) > w_{th}) \wedge \cdots \wedge (H_{min+ad_{th}-1}(x) > w_{th}))$
　　　　break;
for $(max = p_i - 1;\ max > min;\ max--)$
　　if $((H_{max-1}(x) < t - w_{th}) \wedge \cdots \wedge (H_{max-ad_{th}}(x) < t - w_{th}))$
　　　　break;
output min and max;

ここで, $H_{min(x_i)}$ は x_i の min 番目のブロックのハミング重みを表す. このアルゴリズムの出力 min, max をそれぞれ $r_i^{(-)}, r_i^{(+)}$ で表す.

STEP 4: ユーザ ID u に対して, $\mathcal{D}_R(u)$ を次のように定義する.

$$\mathcal{D}_R(u) = \left|\{i \in \mathbb{Z}_{k'} | (u \equiv P_i^{-1}(r_i^{(-)}) \pmod{p_i}) \vee (u \equiv P_i^{-1}(r_i^{(+)}) \pmod{p_i})\}\right|.$$

ここで, P_i^{-1} はランダム置換 P_i の逆置換を表す.

各ユーザ ID u に対して, $\mathcal{D}_R(u)$ を計算する.

STEP 5: $D_{th} = k + l$ と定義する. $\mathcal{D}_R(u) > D_{th}$ が成り立つならば, ユーザ u が結託に加わったと判断する.

2.3.5 安全性

本節では, Randomized c-secure CRT 符号の安全性について議論する. 図 2.5 は Randomized c-secure CRT 符号に対して Threshold Based Attack を行ったときのトレースエラー確率をコンピュータシミュレーションで調べた結果である. 図の各点はそれぞれ 10000 回の平均値である. 本シミュレーションで用いたパラメータは文献 [30] で用いられている値と同じ値を利用した. 本シミュレーションの結果では, ランダムエラー確率が $e < 0.3$ のときには, 誰一人検出できない確率は 0 となった. したがって, 図 2.5 ではプロットしていない. この結果から, トレース性能が格段に向上しているといえる. 前節で述べたように Randomized c-secure CRT 符号ではランダム置換を用いているため, 隣り合った ID の剰余は非常に高い確率で隣り合うことはない. このため, Threshold Based Attack を行ったとしても, 攻撃した符号語の各内部符号から同じ無実のユーザの ID が検

出される確率が大幅に減少する. その結果, トレースエラー確率は劇的に減少した. ここで, ランダムエラー確率が 0.1 の場合のトレースエラー確率を検定する. シミュレーションでは, 10000 回の試行において 4 回トレースエラーが発生した. この確率を Pr とおく. 今, 真のトレースエラー確率 $\bar{Pr}$ がシミュレーション結果の 2 倍の 0.0008 であったと仮定する. このとき, 10000 回の試行においてトレースエラーが起こる回数がシミュレーション結果の 4 回より小さい確率 Pr' は

$$\sum_{i<4} \binom{10000}{i} \bar{Pr}^i (1-\bar{Pr})^{10000-i}$$

となる. ここで, $E(\bar{Pr}, Pr) = -\bar{Pr}\log_2(Pr/\bar{Pr}) - (1-\bar{Pr})\log_2((1-Pr)/(1-\bar{Pr}))$ とおくと,

$$\sum_{i<k} \binom{n}{i} \bar{Pr}^i (1-\bar{Pr})^{n-i} < 2^{-nE(\bar{Pr},Pr)}$$

が成り立つ. この式を用いて Pr' を評価すると $Pr' < 0.213$ となる. したがって, 真のトレースエラー確率が 0.0008 より大きい確率は 20% 程度であると考えられる. また, 同様に真のトレースエラー確率を 0.0010 とおくと, $Pr' < 0.042$ となり, 真のトレースエラー確率が 0.0010 より大きい確率は 4% 程度であると考えられる.

Randomized c-secure CRT 符号は, c-secure CRT 符号の各内部符号値に対してランダム置換を加えたものであり, 各内部符号でのトレース誤り確率は変わっていない. したがって, 従来考えられていた Uniform Selection Attack に対しても安全な符号である.

2.3.6 まとめ

c-secure CRT 符号に対する新しい結託攻撃法である Threshold Based Attack を提案た. Threshold Based Attack は c-secure CRT 符号とトレースアルゴリズム A_3 の性質を利用した攻撃法であり, この攻撃により無実のユーザを誤って検出してしまう確率が高いことをコンピュータシミュレーションによって示した. さらに, Threshold Based Attack に対しても安全であるように c-secure CRT 符号を改良した Randomized c-secure CRT 符号を提案した. Randomized c-secure CRT 符号は c-secure CRT 符号の内部符号の値にランダム置換を施した符号であり, 符号長は c-secure CRT 符号と同じである. 結果として, Randomized c-secure CRT 符号は, コンテンツ配信システムにおける電子透かしの符号化法として最も適したものの一つであると考えられる.

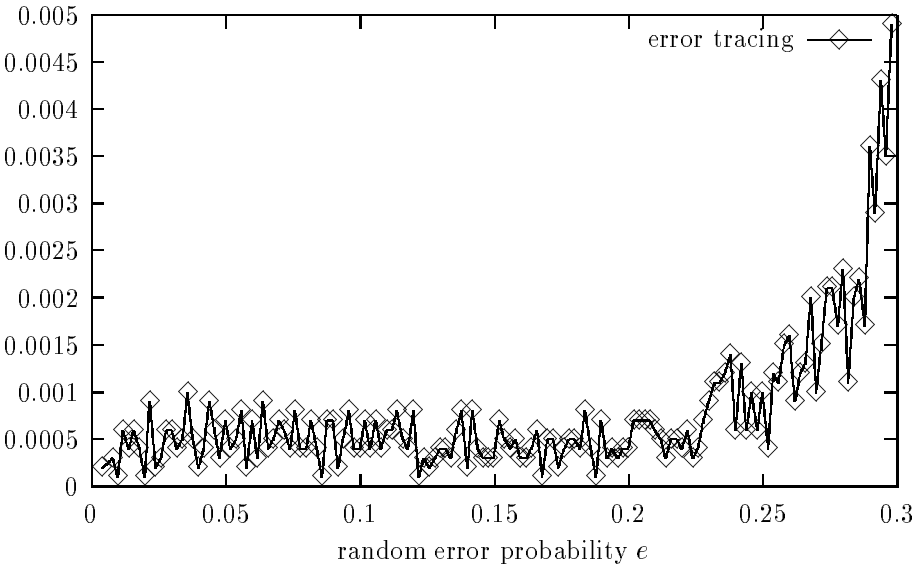

図 2.5 Randomized c-secure CRT 符号のトレースエラー確率

また, 本研究によって, 電子透かしの符号化法を設計する際には, 符号化の知識を利用した攻撃に対しても安全であるように設計しなければならないということを示した.

第3章　ソフトウェアに対する回数券型課金システム

3.1. はじめに

コンピュータのソフトウェアを購入して利用する際には, ソフトウェアの販売者とユーザとの間で, ソフトウェアの使用形態や管理責任などに関する契約（以下ではライセンス契約と呼ぶ）が結ばれる. これまで, 個人ユーザを対象として販売される多くのソフトウェアでは, そのソフトウェアを個人で無制限に利用できるライセンスをユーザが購入するという形態が主流であった. 頻繁に利用するソフトウェアの場合はそのような契約でも問題はない. しかし, それほど頻繁に利用しないソフトウェアの場合, 無制限に利用できるライセンス形態ではユーザは割高感を感じ, ソフトウェアの不正使用を引き起こす一因ともなる. すなわち, ソフトウェアのコピー等違法行為を犯すリスクよりも, 不正使用を行なうことによって得られる利益の方が大きいと判断した場合, そのユーザは正当な製品の購入を見送り, 海賊版や不法コピー版を使用することにもつながりかねない. このような不正を防ぐためにも, あまり頻繁に利用しないソフトウェアに対しては, ソフトウェアの利用回数や利用時間に応じて料金を支払う従量制課金方式が適切であると思われる. 従量制課金方式が実現できれば, ユーザはソフトウェアの利用単価を低く抑えることができる. また, 正規のソフトウェアを合法的かつ安価に使用することができれば, ソフトウェアの不正使用抑止にもつながることになる.

従量制課金方式としてはいくつかのタイプが考えられるが, ここでは, ユーザに対して限られた回数だけソフトウェアを実行する権利を販売するような形態を考える. すなわち, N を適当な正整数とし, ユーザに対して N 回までソフトウェアを起動することを許すような契約形態である. もし, ユーザが N 回以上ソフトウェアを起動したい場合は, 使用権を新たに買い足すこととなる. 同種のサービスとしては列車等の回数券が考えられる. すなわち, ユーザに対してソフトウェア使用権に関する回数券的なものを発行し, ユーザが回数券を使い果たすまでソフトウェアの利用を認めるような形態である.

このような形態の契約では, ユーザによる不正行為をいかに防止するかが問題となる. すなわち, 悪意を持ったユーザが自分のコンピュータを不正に操作することで, 契約回数以上にソフトウェアを利用しようとすることが考えられる. 公正な課金方式の実現のためには, このような不正が行われないよう十分注意を払ってシステムの設計を行う必要がある.

同種の目的を持った方式として, 超流通システム [18, 19] の研究がある. 超流通システムとは, 特殊なシステム（ハードウェア）を仮定し, そのシステム上でしかコンテンツが利用できないように機構を構成した上で, コンテンツの利用に対して課金するシステムである. 超流通システムでは, 電子的流通によりコンテンツの流通コストを抑えることができるため, ユーザに対して低価格でコンテンツを提供できるという利点がある. しかし, 超流通システムでは通常, 耐タンパ性を有した特殊なハードウェアの存在を仮定しているため, 汎用性やコストの面について問題がある.

また, 超流通システムで利用されているような特殊なハードウェアの代わりに, ネットワークへの常時接続性を仮定しても, 同種の方式を実現することが可能である. すなわちユーザの使用する計算機がネットワークに常時接続されており, ライセンスを管理するサーバと無制限に通信を行なうことが可能であるならば, 従量制課金の仕組みを実現することは比較的容易であると考えられる. たとえば, ソフトウェアの起動のたびにネットワーク上に存在するライセンス管理サーバと通信を行ない, ユーザのソフトウェア起動回数や実行時間をライセンス管理サーバが常に監視しておく方式を実現すれば, ユーザの不正を完全に防ぎつつ従量制課金方式を実現できる. 実際, 一部のソフトウェアでは, ソフトウェア起動時にライセンス管理サーバと交信を行ない, ユーザのソフトウェア使用権のチェックを行なう方式が実用化されている. しかしこの方式では, サーバやネットワークに障害が発生した場合にソフトウェアが使用できなくなるという問題点がある. また, ダイヤルアップ回線やモバイル端末を使用している場合等, ネットワークへの常時接続が困難な場合があることを考えると, 極度にネットワークに依存する方式は, あまり汎用的であるとはいえない.

以上で述べたように, 耐タンパ性をもつ特殊なハードウェアやネットワークとの常時接続性を仮定すれば従量制課金方式は比較的容易に実現できるが, これらの仮定は一般的な個人ユーザにとっては利便性に欠ける仮定であるといえる. 例えば家庭で使用するソフトウェアの課金情報管理を考えるうえにおいては, 適切な仮定とはいえない.

本研究では, 耐タンパ性を有する特殊なハードウェアや常時ネットワーク接続性を仮定

することなく, 上で述べたような回数券方式を実現することを目標とする. 提案方式では, ソフトウェアの起動回数（回数券の残り度数）をユーザのコンピュータ上に保管する. 課金対象となるソフトウェア起動時には, そのソフトウェア自身がユーザコンピュータ上の残りチケット数を確認し, 更新する. ただし, ユーザは自分のコンピュータ上のファイル等を操作して残りチケット数をごまかす等の不正操作を行ない得るので, 暗号化や難読化等の手法を使って重要な情報の保護を行なう. また, 不定期的に, あるいはソフトウェアの実行環境に不審な点が発見された場合, ソフトウェアはライセンスを管理するサーバに接続し, ユーザの行為の正当性を確認する. 本方式でもネットワークへの接続性は必要となるが, その頻度はかなり小さくすることが可能であり, ダイアルアップ回線やモバイル端末のような環境でも負担はそれほど大きくないと考えられる. 一方, 提案方式では残りチケット数等の重要な情報を全てユーザのコンピュータ上で管理することになるため, セキュリティの低下は免れない. 3.4.2 節では提案方式の安全性について議論し, 悪意を持ったユーザによる簡単な攻撃に対して提案方式が安全であること, 提案方式において不正行為が成功するためには, ユーザは多大な手間をかける必要があることを示す.

3.2. 前提とする環境と条件

提案方式では, ユーザのソフトウェア使用権を管理するライセンスサーバと, ユーザの計算機および, ユーザの存在を考える. 現実世界において提案方式を運用するためには, ユーザとライセンスサーバの間には適当な決済手段が必要であるが, 簡単のため決済方式には触れないものとする.

ユーザは購入したソフトウェアを自分の計算機上で実行する者である. 全てのユーザが信頼できるわけではない. すなわち, 購入した回数券の実行回数よりも多くの回数, ソフトウェアを実行しようとするユーザが存在し得る. 以下ではそのような不正を行なうユーザのことを不正ユーザと呼ぶ. ユーザが使用する計算機としては, 通常のパーソナルコンピュータを仮定する. ユーザの計算機は常にネットワークに接続されている必要はないが, 必要であれば, ネットワークを介してライセンスサーバと通信を行うことが可能であるとする. 現実の世界では, ネットワーク上において情報の改竄や盗聴など, 悪意のある第三者による妨害操作が起こり得るが, 暗号技術を適切に利用することで仮想的に安全な通信路を確保することが可能であると考えられる. したがって本研究では, ユーザの計算機とライセンスサーバとの間の通信は, 安全で信頼できるものと仮定する. 一方,

ユーザの計算機は完全にユーザの管理下にあるため, ユーザは計算機上のファイルに対して自由にコピー, 変更, 削除などの操作が行える. またユーザは計算機をネットワークから切り離した状態で利用したり, 内部の時計の時刻を操作するなど, 一般の計算機に対して行なうことができる操作はすべて行ない得ると考える. ただし, ユーザは, 実行形式ファイルの内部を理解したり, 自分の意図した動作をするように改変することはできないものとする. この仮定は, 近年盛んに研究されているソフトウェアに対する難読化の技法 [15, 16, 20] を利用することによって実現可能であると考えられる.

ここで仮定した条件はいずれも一般的かつ自然なものであり, 現在広く流通している Windows システムや各種の UNIX オペレーティングシステムは, 本節で述べた条件を全て満足する.

3.3. 提案方式

3.3.1 提案方式の概要

本研究にて対象とするソフトウェアは, ワードプロセッサや表計算ソフト等に代表されるような, 通常のアプリケーションプログラムである. 従量制課金を可能にするため, プログラムは, ユーザがプログラムを実行する権利を有するかどうか検査する「実行権検査部」を内包する. プログラム起動時には実行権検査部が最初に呼び出され, そのプログラムが実行されるプラットフォーム（ユーザの計算機）に, 回数券に相当する情報があるかどうかを検査する. もし回数券に相当する情報が存在しない場合や, 回数券の残り度数がゼロの場合, また, 回数券情報の内容が不正であると判明した場合は, ユーザにサービスを提供することなくプログラムは終了する. また, 実行権検査部が必要と判断した場合には, ソフトウェアはライセンスサーバと通信を行い, 回数券情報の確認と登録を行う. この際, ライセンスサーバと交信できなかったり, ライセンスサーバからプログラム実行を拒否するよう指示された場合は, やはりユーザにサービスを提供することなくプログラムは終了する. 実行権検査部が各種の検査を通過したときには, 回数券の残り度数を 1 度数減らし, ユーザに対して（ワープロ, 表計算等の） 本来のサービスを提供する.

プログラムおよび回数券情報はユーザの計算機上にて保管されるため, 不正ユーザはこれらの中身を改変して, 不当にプログラムの実行を試みる可能性がある. したがって, 回数券情報の正当性を暗号技術によって保証する仕組みを導入する. この際, プログラム内で暗号の鍵を保持しておく必要があるが, プログラム自体を難読化することで鍵の保全を

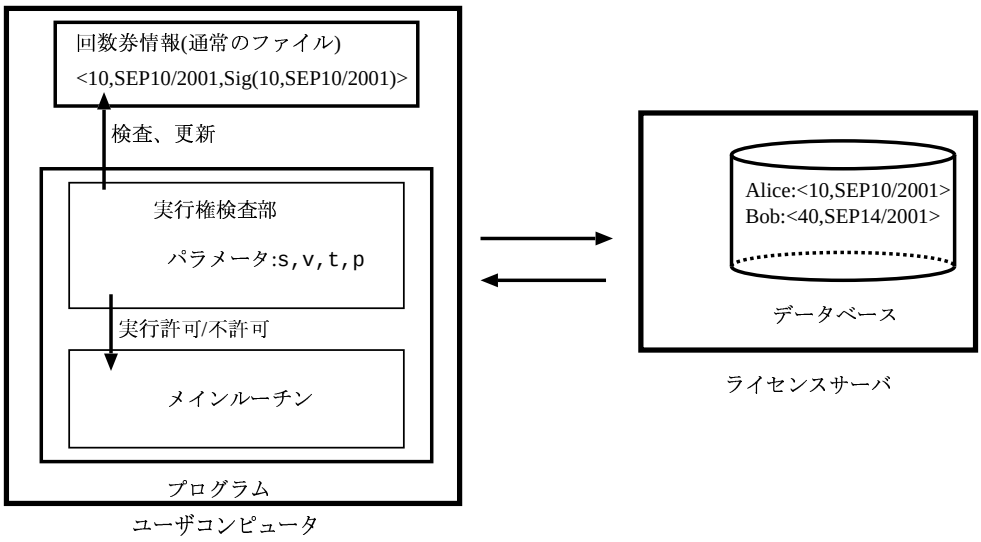

図 3.1 システムの構成

図る.

プログラムおよび回数券情報の配付形態についてはとくに規定しないが, たとえばプログラム自体は誰でも自由に入手可能にしておき, 回数券情報のみをユーザに販売する等のソフトウェア流通形態が考えられる.

3.3.2 ソフトウェアの構成要素

提案方式では, 回数券情報の正当性を保証するため, 電子署名方式を利用する. 電子署名方式では署名鍵 s および検証鍵 v を用いて, 署名の生成と検証が行われる. 通常, 署名鍵 s は署名を行う者だけが秘密に保持する値であり, ある情報（データ）に対して署名を生成できるのは署名鍵 s を知っているものだけである. 逆に, 検証鍵 v は公開されており, 誰でも署名の正当性を検証することができるようになっている. また, 公開されている検証鍵 v の値から署名鍵 s を特定することは計算量的に困難であるとの仮定をおく. 良く知られている RSA 暗号 [26] を利用した署名方式は, ここにあげた条件を満足するものと考えられている.

提案方式では, ライセンスサーバが証明鍵 s と検証鍵 v の組を生成し, v の値を公開しておく. また, 後にも述べるが, ユーザに配付するプログラムの内部に s と v の値を保持

しておき, プログラムが単独で（サーバの協力なしでも）署名を生成できるようにする. 回数券情報は常に署名を含んだ形式で運用され, その正当性の検査は署名の正当性検査によって行われる. 正当な署名を生成できるのはライセンスサーバかプログラムだけであり, よって正当な署名を含んだ回数券情報は, 信頼できるものであると判断される. 逆に, 正当な署名が与えられていない回数券情報は, 不正ユーザによって偽造されたものと判断される.

提案方式における回数券情報は 3 字組 $\langle N, T, Sig_s(N,T)\rangle$ として与えられる. ここで N はプログラムを起動可能な残り度数, T は前回そのプログラムを起動したときの日時, $Sig_s(N,T)$ は N および T に対して署名鍵 s を用いて生成した署名である. ユーザがライセンスサーバから回数券を購入する際には, T には販売日時が格納されているものとする. たとえば, あるユーザがライセンスサーバに対し, プログラムを 100 回起動できるような回数券を 2001 年 9 月 10 日に購入したいと申し出たとすると, ライセンスサーバは署名鍵 s を用いて $Sig_s(100, \mathrm{SEP10/2001})$ を計算し, 3 字組 $\langle 100, \mathrm{SEP10/2001}, Sig_s(100, \mathrm{SEP10/2001})\rangle$ を回数券情報としてユーザに販売する. ユーザがこの回数券情報を用いて, たとえば 2001 年 9 月 11 日にプログラムを実行したとすると, 回数券情報は $\langle 99, \mathrm{SEP11/2001}, Sig_s(99, \mathrm{SEP11/2001})\rangle$ に更新される. 以下, 残り度数 N の値が 0 になるまで, ユーザはプログラムを実行可能である.

上に述べたような回数券情報の更新を可能にし, ユーザの不正行為を検知するため, プログラムは以下のパラメータを保持するものとする. 繰り返しになるがプログラムは難読化技法によって保護されており, ユーザはこれらパラメータの値を知ったり改変することはできないものと仮定する.

- 署名鍵 s および検証鍵 v: プログラム起動時に回数券情報が正当かどうか検査するため, プログラムは検証鍵 v を保持しておく必要がある. また, 回数券情報を更新する際, 新しい署名を生成する必要があるので, 署名鍵 s もプログラム中で保持する.

- 接続確率 p: p は 0 から 1 の間の実数である. 後に詳しく述べるが, プログラム起動時には確率 p でライセンスサーバと通信を行なう. p を大きくすると通信頻度が大きくなるが, 方式の安全性も向上する. 逆に p を小さくすると, 方式の安全性は低下する.

- 強制接続期限 t: 提案方式では, 後述するような回数券情報のバックアップ攻撃を困難にするため, 回数券情報の残り度数とともにソフトウェアの起動日時を逐一記録

しておく. 記録されている前回起動日時と今回の起動日時との間に t 以上の隔たりがある場合は, 強制的にライセンスサーバと通信を行ない, 回数券情報の検査を行なう.

ライセンスサーバはユーザに発行した回数券情報をライセンスサーバ内のデータベースに保管しておく. また, ユーザの計算機から回数券情報の検査を依頼された場合には, ユーザの計算機側の回数券情報とデータベース内の情報を比較し, ユーザが不正を行なっていないかどうかを判定する. もし不正が発見されず, 回数券情報が正当と判断されれば, ユーザの計算機に対し, ソフトウェアの実行許可を与える. もし送られてきた回数券情報が不正なものであった場合は, 実行許可を与えない. 実行許可を与えた場合, ユーザから送られてきた回数券情報をデータベースに保存しておく. 詳細は次節で述べる. 以上の概略をまとめると図 3.1 のようになる.

3.3.3 回数券情報の正当性検査

本節では, 回数券情報の正当性検査の方式について説明する. ユーザがプログラムを起動するとまず, 実行権検査部が起動される. 実行権検査部は以下の手順で回数券情報の正当性の検査を行なう.

1. 回数券情報 $\langle N, T, Sig_s(N,T)\rangle$ の署名が正しいかどうか検査する. もし回数券情報が存在しなかったり, 署名が正しくない場合は, プログラムを終了する.

2. $N > 0$ であることを確認する. もし, $N = 0$ であれば, 残り度数が 0 であることをユーザに通知したあとプログラムを終了する.

3. 次の二つの条件のうちどちらかが成立した場合, 4 に進む. もし, どちらの条件も成立しなければ 5 に進む.

   - T と計算機内部の時計が示す現在時刻 T' の差が強制接続期限 t より大きい.
   - $0 < r < 1$ なる乱数 r を発生させ, $r < p$ が成り立つ.

4. 実行権検査部はユーザの計算機とライセンスサーバとの間に接続を確立し, 回数券情報 $\langle N, T, Sig_s(N,T)\rangle$ をライセンスサーバに送信する. その後, ライセンスサーバから実行許可が得られた時点で 5 に進む. サーバとの接続ができなかったり, 実行許可が得られなかった場合, プログラムを終了する.

5. 実行権検査部は, 新しい回数券情報として $\langle N-1, T', Sig_s(N-1, T')\rangle$ を構成し, 古い回数券情報と置き換える. その後, プログラムの本来のメインルーチンに実行を移す.

実行権検査部のステップ 4 では, ユーザの計算機上の回数券情報がライセンスサーバに送信される. ライセンスサーバは, 送られてきた回数券情報と自分のデータベース中の回数券情報とを比較し, ユーザ計算機上の回数券情報が正当かどうか判定する. いま, 送られてきた回数券情報を $\langle N, T, Sig_s(N, T)\rangle$, データベースに保管されていた回数券情報を $\langle N', T', Sig_s(N', T')\rangle$ とする. ここで N', T' は, 前回, ユーザ計算機からライセンスサーバに対して回数券情報の正当性検査があったとき（ステップ 4 が前回実行されたとき）の残り度数およびそのときの日時である. ユーザが回数券情報の改変等不正行為を行っていない場合, 回数券情報に記録された残り度数は単調に減少する. すなわち $N < N'$ が成立するはずである. 同様に, ユーザがユーザ計算機の内部時計を操作していない限り, $T' < T$ が成立する（日時の比較は, より最近の日時のほうがより大きいと解釈する）. これらの不等式が成立している場合, ライセンスサーバはユーザ計算機に対してプログラムの実行許可を送る. 同時に, データベース中の回数券情報を, 今回送られてきた $\langle N, T, Sig_s(N, T)\rangle$ に置き換える. もし $N \geq N'$ または $T' \geq T$ であった場合, そのユーザは, 回数券情報の置き換えや計算機の内部時計を操作する等の不正を行ったものと考えることができる. この場合, プログラムの実行を不許可としてユーザに警告を与える.

3.4. 提案方式の評価

3.4.1 手続きの利便性について

提案方式では, ユーザの計算機がネットワークに接続できないような状態や, ライセンスサーバに障害が発生しているような状態であっても, ある程度はプログラムの実行が可能になることを目標としている. 本小節では, ユーザがなんら不正を行っておらず, 回数券情報内の残り度数が 1 以上である場合, すなわち, ユーザはプログラムを実行する正当な権限を有する場合を想定し, 提案方式が所期の目標を達成していることを示す. ユーザが不正を行っている場合についての議論は, 3.4.2 節で行う. 以下, ユーザ計算機上の回数券情報を $\langle N, T, Sig_s(N, T)\rangle$ とする.

ユーザの計算機がライセンスサーバに接続可能な場合

実行権検査部は, ユーザ計算機上の回数券情報を検査し, $N > 0$ であることを確認して, ステップ 4 を実行するか否かを決定する. もしステップ 4 を実行しない場合, プログラムは問題無く起動され, ユーザに通常のサービスが提供される. ステップ 4 を実行した場合でも, ユーザが不正行為を行っていなければサーバ側で咎められることはないため, 通常どおりプログラムが実行されることになる. したがって, ユーザの計算機がライセンスサーバに接続可能な場合は, 問題無くプログラムが実行可能である.

ユーザの計算機がライセンスサーバに接続不能で, 同じプログラムを最近使用していた場合

実行権検査部は, ユーザ計算機上の回数券情報を検査し, $N > 0$ であることを確認して, ステップ 4 を実行するか否かを決定する. 同じプログラムを最近使用していた場合, T は現在時刻に近い値となっているはずであり, ステップ 3 の 2 条件のうち最初の条件は成立しないと考えてよい. よって実行権検査部は乱数を発生させ, その大小によってステップ 4 を実行するか否かを決定する. もしステップ 4 を実行しない場合, プログラムは問題無く起動され, ユーザに通常のサービスが提供される. 一方, ステップ 4 が実行された場合, サーバとの接続ができないため, 本来の機能を提供することなくプログラムは終了する. したがってこの場合, ユーザは正当な実行権限を持っているにもかかわらず, プログラムの起動に失敗することになる.

しかし, このような事態が発生した場合でも, ユーザはプログラムを再起動することで, かなり高い確率でプログラムを利用することが可能となる. プログラムを再度実行したとき, 実行権検査部が発生する乱数値は, 基本的に前回発生した乱数値とは無関係である. したがって, たとえ前回はライセンスサーバに接続するような乱数値が発生したとしても, プログラム再起動時には別の乱数値が得られ, ステップ 4 を経由せずにプログラムが実行される可能性がある. k 回連続してライセンスサーバに接続しようとする確率は r^k であり, これは k に対して指数的に減少する値である. すなわち, ユーザは再起動を繰り返すことで, 高い確率でプログラムが実行可能である. また, ライセンスサーバに接続できなくてプログラムが終了する場合, 回数券情報は更新されないので, たとえ再起動を繰り返したとしても, ユーザのプログラム使用残り度数が不当に消費されることはない.

ユーザの計算機がライセンスサーバに接続不能で, 同じプログラムをしばらく使用していなかった場合

実行権検査部は, ユーザ計算機上の回数券情報を検査し, $N > 0$ であることを確認して, ステップ 4 を実行するか否かを決定する. 同じプログラムをしばらく使用していなかった場合, T は現在時刻からは遠い値となっており, ステップ 3 の 2 条件のうち最初の条件が成立すると考えられる. この場合, 実行権検査部はライセンスサーバに接続しようとするが, 接続に失敗して, 本来の機能を提供することなくプログラムは終了する. さきほどの場合と異なり, たとえユーザがプログラムを再起動したとしても, 回数券情報内の時刻情報は変わらない. よってユーザがプログラムの再起動を繰り返したとしても, 毎回ライセンスサーバに接続しようとすることになる. したがってこの場合, ユーザはライセンスサーバに接続できる環境に移行しない限り, プログラムの実行ができない. これは, 方式の安全性を保証するためにはやむを得ないことではあるが, ユーザにとっては不便なことである. この問題をいかに解決するかは, 今後の課題として検討したい.

3.4.2 手続きの安全性について

提案方式においてステップ 4 が実行されない場合, ユーザがプログラムの実行権を有するかどうかの判定は, ユーザの計算機内で完結する. したがって, もし回数券情報がユーザに操作されてしまい, ユーザの計算機をネットワーク接続できない場所へ物理的に移動されてしまうと, ユーザがプログラムを際限無く利用できてしまうことにもなりかねない. そこで提案方式では回数券情報を電子署名によって保護し, ユーザにとって都合の良い内容の回数券情報を偽造することを困難にしている. 一方, ユーザは自分の計算機上のファイルを自由に操作することができるため, 回数券情報を含んだファイルのバックアップコピーをあらかじめ作成しておき, 必要に応じて現時点の回数券情報と置き換えることにより, 不正をはたらく可能性もある. たとえば, あるユーザ A が 2001 年 9 月 10 日にプログラムを 100 回起動できるような回数券情報$\langle 100, \mathrm{SEP10/2001}, Sig_s(100, \mathrm{SEP10/2001})\rangle$を購入し, そのバックアップコピーを取っておいたとする. その後ユーザ A は通常の手続きに則ってプログラムを何回か使用し, 回数券情報内の残り度数が少なくなったときに, バックアップしていた回数券情報 $\langle 100, \mathrm{SEP10/2001}, Sig_s(100, \mathrm{SEP10/2001})\rangle$ を復元するようなケースが考えられる. 本稿ではこのような不正攻撃を, ユーザによるバックアップ攻撃と呼ぶ. バックアップ攻撃は提案手法に固有のものではなく, ユーザ計算機上に使

用回数等を管理する情報を保存するような方式では, 常に考慮すべき攻撃方法である.

もしプログラムの実行権検査部が回数券情報の残り度数しか検査しない場合, 上で述べたバックアップ攻撃により, ユーザはプログラムを 100 回使用する権限を不正に再取得することになる. このような不正行為を防止するため, 実行権検査部はなんらかの方法でバックアップ攻撃を検知しなければならない. とくに提案方式では, ユーザの計算機に特殊なデバイスを設置しないこと, ユーザの計算機がネットワークに接続できない場合でもプログラムの使用が可能であることを目標としているため, 実行権検査部は, 純粋にユーザの計算機が保有するリソースと情報だけを使用して, バックアップ攻撃を検知する必要がある. 本節ではまず, ユーザが自分の計算機上の内部時計を不正に操作しない場合を仮定し, この仮定の下では, バックアップ攻撃が高い確率で露呈する（サーバに検知される）ことを示す. ユーザが内部時計を操作する場合については 3.4.3 節で議論することとする.

バックアップ攻撃を防止するため, 提案方式は 2 種類の不正防止機構を設けている. ひとつは, 回数券情報に埋め込まれたプログラムの前回実行時間である. もしユーザがバックアップ攻撃を行ったとすると, 回数券情報内に記録されている日時は, 現在時刻よりかなり遡った過去のものに相当する日時となっていることが予測される. 提案手法では, 回数券情報内の日時と現在時刻との差が強制接続期限 t より大きい場合, 強制的にライセンスサーバと接続することで回数券情報の正当性の検査を行う. このとき, ライセンスサーバ側に, ユーザがバックアップコピーを作成した日時よりもあとの回数券情報が記録されていれば, ユーザの不正行為がライセンスサーバ側に露見することになる. たとえば, 回数券情報 $\langle 100, \mathrm{SEP10/2001}, Sig_s(100, \mathrm{SEP10/2001})\rangle$をバックアップしていたユーザが, 2001 年 11 月 15 日に, その時点の回数券情報をバックアップしていた情報と置き換えたとする. 強制接続期限 t が 60 日と設定されていた場合, 次にプログラムが起動されるときには, ユーザの計算機は必ずライセンスサーバと通信を行うことになる. もし, ライセンスサーバ側にユーザの使用記録として$\langle 25, \mathrm{NOV01/2001}, Sig_s(25, \mathrm{NOV01/2001})\rangle$が残っていた場合, すなわち, 2001 年 11 月 1 日現在では回数券の残り度数が 25 にまで減っていたことが記録されていた場合, ユーザ計算機からライセンスサーバに（バックアップコピーから復元された）回数券情報が送られた時点で, サーバはユーザの不正を検知することが可能である. 以上の機構により, 不正を行おうとするユーザは, 非常に古い回数券情報を再利用することはできないようになっている.

上で述べた時刻検査の機構だけでは, バックアップ攻撃の防止策としては不十分である. 実際, 回数券情報内の日時と現在時刻の差が t を越えるまでライセンスサーバに接続

しないことがわかっていれば, 不正を行おうとするユーザはこまめにバックアップコピーを取っておき, 現在時刻から t だけ遡った期間内に作成したバックアップコピーの中で, 最も古いもの（残り度数の大きいもの）を復元して利用することで, 不当に多くの回数プログラムを実行可能となってしまう. たとえば, 購入直後の回数券情報をバックアップしておき, 残り回数が少なくなったときにバックアップコピーを復元してやることを繰り返すことで, 回数券情報購入後 t の期間内は, プログラムを何度でも起動できることになってしまう. このような不正行為を防ぐため, 提案方式は第 2 の不正防止機構, すなわち乱数を用いたライセンスサーバとの強制接続機構を有する. 本機構により, 回数券情報内の時刻が比較的新しく, 現在時刻との差が t 以内であったとしても, 回数券情報をライセンスサーバに送信する場合が発生する. このときユーザが不正な回数券情報を使用していた場合, その不正はライセンスサーバに検知され得る. したがって, 上で述べたような, 購入直後の回数券情報を何度も使い回すような不正を行っていた場合, その不正行為はサーバの知るところとなる可能性が高い.

それでも不正を行おうとするユーザは, 乱数を用いたサーバとの接続機構を無効化するため, たとえばあらかじめネットワークケーブルを抜いておくなど, 自分の計算機をネットワークに接続できないような設定にしておいて, バックアップ攻撃を行うことも考え得る. すなわち, ユーザは回数券情報$\langle N, T, Sig_s(N,T)\rangle$ のバックアップコピーを作成しておき, 以降はネットワークケーブルを抜いた状態でプログラムを使用するような形態である. 前節で述べたように, ネットワークに接続できない場合でも, 再起動を繰り返すことでプログラムの使用は実質的に可能である. 回数券情報内の残り度数が少なくなってきたら, ユーザはバックアップコピーから $\langle N, T, Sig_s(N,T)\rangle$ を復元し, 再度 N 回の実行権を不正に取得する. このような行為を繰り返すことで, 時刻 $T+t$ までの期間, ユーザは無制限にプログラムを実行することが可能となってしまう. しかし, 時刻 $T+t$ を過ぎると, たとえバックアップコピー $\langle N, T, Sig_s(N,T)\rangle$ の情報を復元したとしても, 第 1 の不正防止機構がはたらいてプログラムの実行ができなくなってしまう. この状態から抜け出すには, ユーザは自分の計算機をネットワークに再接続するしかない. 再接続後のプログラム起動時には回数券情報 $\langle N, T, Sig_s(N,T)\rangle$ がサーバに送信され, 回数券情報は（時刻 $T+t$ に再接続を行った場合）$\langle N-1, T+t, Sig_s(N-1,T+t)\rangle$ に更新される. 更新された回数券情報は新しい時刻情報を含むが, 残り度数は 1 減っていることに注意されたい. この更新された回数券情報を利用してユーザは同様の不正を繰り返す可能性があるが, 同様の議論により, 時刻 $T+2t$ には再度サーバに接続する必要がある. 以下同様の

議論を繰り返すことにより, ユーザは時刻 $T + Nt$ を越えては回数券情報を使用することができない（正確には, ユーザのプログラム使用時間の総数は Nt を越えない）. 提案手法は, 残念ながらユーザの不正を完全に防ぐという理想には達していないが, ユーザのプログラム不正使用に一定の歯止めを与えるものとなっている.

3.4.3 ユーザ計算機の内部時計に対する不正操作

ユーザの計算機はユーザ自身によって管理される. よって, 不正を行おうとするユーザは, 計算機の内部時計も不正に操作した上で, 前節で述べたような各種の攻撃を行う可能性もある. たとえばバックアップ攻撃により古い回数券情報を復元し, 計算機の内部時計を遅らせて（昔の日時に設定して）, プログラムの実行権検査部を欺こうとすることも考えられる. 実行権検査部は, ユーザ計算機の内部時計を利用して回数券情報の古さを判定しているため, 内部時計そのものを操作されてしまうと, 回数券情報に対する不正防止機構のうち最初のもの, すなわち回数券情報内の時刻情報を利用したものが実質的に無効化されてしまう. この場合でも, ユーザの計算機がネットワークに接続できるのであれば, 第 2 の不正防止機構, すなわち乱数を用いたライセンスサーバとの強制接続機構により, ユーザの不正を検知することは可能である. 一方, 内部時計が操作されたうえで, さらにネットワークに接続できないような環境にユーザ計算機が設置された場合, 提案手法ではユーザの不正を検知することは不可能である. すなわち, 不正を行おうとするユーザは, 内部時計を戻し, バックアップしていた回数券情報を復元し, さらにネットワーク接続ができないようにケーブル等を抜いた上でプログラムを起動すれば, サーバに検知されることなくプログラムを不正使用することが可能である. プログラムの起動毎にこれらの作業を行うことは, 不正を行おうとするユーザに取ってもかなりの負担になることが予想される. とくに, 高度に複雑化した今日の計算機システムにおいて不用意に内部時計を操作することは, 他のトラブルの原因にもなりかねない. すなわち, 提案手法において不正を完遂するには, ある程度の心理的かつ技術的な障壁が存在すると言ってよい. ソフトウェアの使用料金等を適切に設定することにより, 大多数の潜在的不正ユーザは, 大きな手間と時間をかけて不正をはたらくことよりも, 正当な料金を支払って回数券情報を購入することを選択すると期待される.

3.5. まとめ

コンピュータソフトウェアに対する従量制課金を実現する一方式として, 回数券型の課金方式を提案した. 提案方式の特徴として, 特殊な耐タンパデバイスの存在を仮定しないこと, ユーザ計算機の常時ネットワーク接続性を仮定しないことがあげられる. 耐タンパデバイスやネットワーク接続性は, システムの安全性を保証するためには有用であるが, 実現時のコスト増加や（とくにモバイル端末の） ユーザの利便性を損なうことになる. 提案方式では暗号技術を利用することにより, ユーザの利便性をそれほど損なうことなく, 実用的なレベルでの安全性を実現している. また, 提案法は純粋にソフトウェア的な要素のみにより構成されているため, 経済的かつ簡単に実現することが可能である.

第4章　回答証明が可能な匿名アンケートプロトコル

4.1. はじめに

本節では, 主として情報セキュリティの観点から, 大学等における講義評価アンケートシステムを電子的に実現する方法について議論する.

近年多くの大学において, 学生が自分の受講した講義の内容を評価する, いわゆる講義評価が盛んに行われている [1]. 講義評価を実施する目的は多様であるが, たとえば評価結果を教官にフィードバックすることで, 講義内容の改善や向上が期待できる. 奈良先端科学技術大学院大学においても, 数年前よりアンケート方式にて学生による講義評価を行っている. 当初, 試験実施時などにアンケート用紙を学生に配布し, 試験の解答と同時にアンケート用紙の回収を行っていた. 紙ベースでのアンケートの場合, 教官の目の前で, 試験の解答用紙だけを提出してアンケート回答用紙を提出しないのは学生も気まずく感じるのか, アンケート回収率（試験受験者に対するアンケート回答者数の割合）は 100 パーセントに近かった. しかし, 紙ベースでアンケートを行った場合, 集計等は手作業で行う必要があるため, アンケートの機械化・システム化による省力化の要望が出てきた.

この要望に対処するため, 平成 12 年度より試験的に Web 上でアンケートを実施するシステムを導入し, 約 2 年間の実験的な運用を行った. システムの導入により, 当初の狙いどおり集計の手間自体は軽減したが, 一方でアンケートの回収率が極端に低下するという問題が発生した. アンケート回収率が低下した原因としては, システムの安直な匿名性実現方式が考えられる. このシステムでは, 回答者の匿名性を保証するために誰でも無記名で（個人認証を行うことなく）アンケート回答フォームに書き込めるようになっている. 大学側では, 誰がアンケートに回答して誰が回答していないのか把握できないため, 学生にとっては, アンケートに回答しないことに対する心理的な障害が小さくなったものと考えられる. たとえば回答時に個人の認証を行う等, いわゆる記名式のアンケートにすれば回答率自体は向上すると考えられるが, 記名回答では学生が批判的な意見を書きにく

くなるのではないかとの懸念もある. もし, 回答内容については匿名性が保証されているが, 回答の事実については大学側で記録をとることが可能な方式があれば, 講義評価等のアンケートには有用であると考えられる.

一方, 同種の研究として電子投票プロトコルがある [9, 22, 24, 27]. 講義評価アンケートは電子投票の特殊な場合と考えることができるが, 電子投票と講義評価アンケートでは要求される事項や前提条件に異なる点も多い. また, 現在提案されている電子投票システムでは, MIX-NET [5] などの匿名通信路が必要になることが多い. 大学内において, 結託を行わない複数のサーバによって MIX-NET を実現することは可能であるが, それらサーバの管理を大学が行うのであれば学生に対して説得力を持たない.

これらの理由から, 大学での講義評価システムに適した電子アンケートプロトコルを提案する. 提案方式は, 既存の電子投票プロトコルの簡易版とも考えられるが, 匿名通信路の問題にも十分考慮した方式となっている. さらに, Java を用いて Web 上にアンケートシステムを実装し, 講義評価アンケートの用に供することで, 実証実験を行う.

4.2. 準備

4.2.1 匿名アンケートプロトコルに求められる性質

本節では, 主としてセキュリティ的な観点から, 講義評価のためのアンケートプロトコルが満たすべき性質を示す. 以下では, アンケートの集計を行う大学側エンティティをサーバと呼び, アンケートに回答する学生をユーザと呼ぶ.

ユーザの不正回答防止: 一人のユーザが二回以上回答を行なったり, アンケートに回答する権利のないユーザが回答を行なえないこと.

匿名性: サーバは, どのユーザがどの回答を行なったか特定できない（当て推量以上の確率で言い当てることができない）こと.

回答者と未回答者の識別: サーバは, どのユーザが回答を行なって, どのユーザが回答を行なっていないかを検出できること.

設問設定の柔軟性: アンケートの設問は多者択一方式に限定されず, たとえば回答者による自由記述が可能である等, 柔軟に設定できること.

上で述べた性質は, 通常の電子投票等とも共通する要求仕様である. 一方, 本研究で対象としている講義評価アンケートでは, 以下の条件を仮定できる.

サーバの不正行為に関する仮定: サーバは, 回答内容の改ざんや回答の水増し等, いわゆる「能動的攻撃」を行わないと仮定する. 大学（サーバ）が講義評価アンケートを実施する目的は, 学生の率直な意見を収集することであり, 回答結果を不正に操作したとしても, 大学として得るものはない. したがって, 大学が能動的な攻撃を行う動機や必然性は低く, アンケートプロトコルは, これらの不正行為に対する耐性を有する必要はない. 一方, サーバの「受動的攻撃」（正当に入手した情報から他の情報, たとえば誰がどのような回答を行ったか等の情報を導出するような攻撃）については, プロトコルとして耐性を有する必要がある.

通信路に関する仮定： サーバとユーザとの間には, 盗聴や通報の改変ができないような安全な通信路が存在すると仮定する. 大学内ネットワークはファイアウォール等により学外から保護されており, 物理的にも大学の管理下にある. また, 取り扱う情報の経済的な価値もそれほど大きくはないため, 第三者による大規模な攻撃の対象となることもない. たとえばSSL [23] 等の既存技術を利用することで, 通信路については実用上十分な安全性が得られると考えられる.

通常の電子投票では必ずしも上記の仮定を置くことができないため, 投票プロトコル自体が, たとえばサーバの能動的攻撃への保護機能や, 通信メッセージの暗号化等の機能を有する必要があった. その意味で, 講義評価アンケートに要求されるセキュリティ要件は若干弱く, 逆に, それら安全性実現のための機構の一部を省略することで, より簡潔で実現に適したプロトコルが得られる可能性がある.

4.2.2 既存の電子投票について

ネットワーク上で匿名の投票や選挙を行うための方式については, 文献 [9, 22] で提案されているブラインド署名 [6, 7] を用いる方式や, 文献 [24, 27] などで提案されているMIX-NET [5] と呼ばれる匿名通信路を用いる方式など, 多くの理論的な研究が行われている. MIX-NET は, 匿名通信路を実現する一手法で, 結託しない複数のサーバ（MIX サーバ）によって構成される. 大学内に複数の MIX サーバを設置することは技術的には困難でないと考えられるが, 学生に対し, 大学が管理する MIX サーバが結託していないこと

を納得させるのは困難である. 一方, ブラインド署名を用いる方式の場合には, 投票者の匿名性を保つために, サーバと各投票者間に匿名通信路が必要である. 匿名通信路については現在でも盛んに研究されているが, インターネット等の実用的な環境において匿名通信を実現する仕組みについては, 残念ながらまだ確立されているとはいえない. また, ブラインド署名を用いる方式では, サーバの能動的攻撃を防止するため, 公開掲示板のような仕組みが必要になることも多い.

4.3. 提案プロトコル

4.3.1 提案プロトコルの概要

本研究では, ブラインド署名を利用することで, 必要最小限の機能を持ったアンケートプロトコルを構成する. ブラインド署名を利用する方式については, 前節で述べたように若干の問題点が知られているが, 他の方式に比べて, より実用的なアプローチであると考えられる. プロトコルの構成にあたっては, たとえば文献 [21] などで提案されているような 2 フェーズ型プロトコルを基本として検討を行う. 文献 [21] のプロトコルでは, ANDOS (All-or-Nothing Disclosure of Secrets) プロトコルを利用することで, サーバ（集計者）から各ユーザ（投票者）に対して, 投票権を有することの証明書を発行する. ユーザは, 実際の投票内容にこの証明書を添付してサーバに送ることで, 自分の投票の正当性を主張する. 集計結果に関する情報は掲示板等で公開され, サーバの能動的攻撃はユーザによって監視される.

文献 [21] の方式はシンプルで安全性が高い反面, 少なくとも以下に示すような問題があると考えられる. (1)ANDOS プロトコルは, 複数のユーザが同期・協力して実行する必要があるが, 実際の投票者にそれを期待するのは現実的でない, (2) 公開掲示板が必要となる, (3) 匿名通信路が必要となる. 本研究では, (1) に関して, ANDOS の代わりにブラインド署名を利用することで, 各ユーザが非同期的に投票（アンケートに回答）できるようにする. また, (2) について, 講義評価アンケートではサーバの能動的攻撃を想定外としているので, 公開掲示板自体が不要となる. 一方, (3) については依然として大きな問題であり, 一般的な解決策を提示することは困難であるが, 運用上の工夫で対応することを検討する. たとえば, ユーザがプロトコル実行の一部を他の計算機上で行うことで, 回答者と回答内容の関連付けが事実上できないようにする等の対応を考える. そのためには, プロトコルの各操作の独立性をできるだけ高くしておき, 各操作間でのデータの移行等の

作業ができるだけ小さくなるよう, プロトコルおよびシステムを構成する必要がある.

ブラインド署名等を用いて正当性に関する証明をあらかじめ入手し, 後日, 以前の操作とは独立した環境において入手した証明を行使するという方式は, 電子マネーの匿名性実現においてしばしば用いられるアプローチである. 一方, 著者の知る範囲では, 従来の電子投票に関する研究において, 電子マネー型の単純な匿名性実現方式はそれほど重要視されてこなかった. 原因はいくつか考えられるが, 単純に操作の独立性を高める方式では, 電子投票の研究において重要視されている「投票者による集計者監視」の仕組みとうまく整合しないことが考えられる. 講義評価アンケートでは, 集計者の不正監視が不要になるため, 電子マネー型の単純な方式でも有効であると考えられる.

提案プロトコルはハンドル登録フェーズ, 回答フェーズ, 受領書送信フェーズの３つのフェーズからなる. ユーザはまずハンドル登録フェーズを実行し, その後回答フェーズを実行し, 最後に受領書送信フェーズを実行する. 各フェーズの概要は以下のとおりである.

ハンドル登録フェーズ: ユーザはランダムにハンドルを選び, ハンドルに対するサーバの電子署名を入手する. この際, ブラインド署名を用いて署名の計算を行うこととし, サーバがユーザとハンドルとの対応をつけることができないようにする. ハンドルに対する署名は, 次の回答フェーズで利用する.

回答フェーズ: ユーザはハンドルとその署名を提示することで, 自分に回答権があることをサーバに提示する（この際, 自分の本名は明かす必要はない）. その後, サーバにアンケートの回答を送り, サーバからアンケートに回答したことを証明する受領書を受けとる. 受領書の授受にもブラインド署名を利用し, たとえユーザが受領書を公開しても, そこから回答内容が露呈することはないようにする.

受領書送信フェーズ: ユーザは回答フェーズで手に入れた受領書をサーバに送信する.

4.3.2 提案プロトコルの詳細

以下では RSA ブラインド署名 [6, 7] を用いた一実現法を示す. 他のブラインド署名法を用いてもほぼ同様のプロトコルを得ることが可能である.

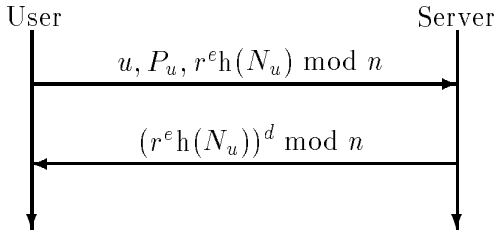

図 4.1 ハンドル登録フェーズ

ハンドル登録フェーズ

準備 サーバは RSA 暗号の鍵組を計算し, 署名検証鍵（暗号化鍵） (e, n) を公開, 署名作成鍵（復号鍵） d を秘密にしておく. またサーバは一方向性ハッシュ関数 h を定め, 全てのユーザに通知する. サーバは回答権のあるユーザに対し, ユーザ ID u と初期パスワード P_u を発行しておく.

1. ユーザはハンドル N_u をランダムに作成する. ユーザは, ハンドル N_u にブラインド署名をもらうために乱数 $r(1 < r < n)$ を選び, $r^e \mathrm{h}(N_u) \bmod n$ を計算する.

2. ユーザは ID u , 初期パスワード P_u , ステップ (1) で計算した $r^e \mathrm{h}(N) \bmod n$ をサーバに送信する.

3. サーバはユーザから送られてきた ID u と初期パスワード P_u をチェックする. もし, パスワードが正しくなかったり, そのユーザに回答権がなかったり, あるいは既にハンドル登録フェーズを実行していたならば, プロトコルを終了する.

4. サーバは $(r^e \mathrm{h}(N_u))^d \bmod n$ を計算し, ユーザに送信する.

5. ユーザはサーバから受けとった $(r^e \mathrm{h}(N_u))^d \bmod n$ に $r^{-1} \bmod n$ をかけ, $\mathrm{h}(N_u)^d \bmod n$ を得る. $\mathrm{h}(N_u)^d \bmod n$ はハンドル N_u に対するサーバの署名となっているが, 以降では, この $\mathrm{h}(N_u)^d \bmod n$ を回答用パスワードと呼ぶ. 回答用パスワードは, 次の回答フェーズで利用する.

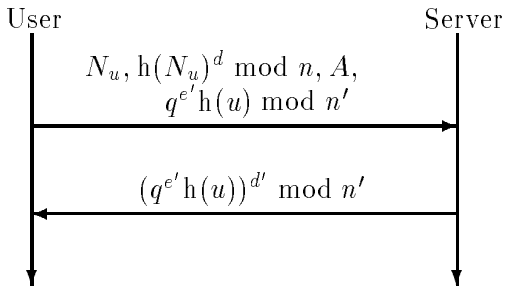

図 4.2 回答フェーズ

回答フェーズ

準備 サーバは RSA 暗号の鍵組を計算し, 署名検証鍵（暗号化鍵） (e', n') を公開, 署名作成鍵（復号化鍵） d' を秘密にしておく. これらの鍵はハンドル登録フェーズで用いた鍵とは異なるものを利用する. また, ハンドル登録フェーズと同様に一方向性ハッシュ関数 h を選び, 全てのユーザに通知する. 一方向性ハッシュ関数は, ハンドル登録フェーズで用いたものと同じ関数でかまわない.

1. ユーザは以下の 4 つのデータをサーバに送信する.
   - ハンドル登録フェーズで選んだハンドル N_u
   - 回答用パスワード $\mathrm{h}(N_u)^d \bmod n$
   - アンケートの回答 A （自由に記述した文書でよい）
   - 受領書を計算するためのデータ $q^{e'}\mathrm{h}(u) \bmod n'$. ここで, q はユーザが任意に選んだ乱数で, $1 < q < n'$ とする. また u はユーザ ID を表す.
2. サーバはユーザから送られたハンドル N_u から $\mathrm{h}(N_u)^d \bmod n$ を計算し, 回答用パスワードと一致することを確認する. パスワードが一致し, かつハンドル N_u を持つユーザからまだ回答を受けとっていないならば, 回答 A を受理する. もし, 以前に同じハンドルを持つユーザが回答をしていた場合には回答は受理せず, プロトコルを終了する.
3. サーバは回答を受理したのち, $(q^{e'}\mathrm{h}(u))^{d'} \bmod n'$ を計算しユーザに送信する.
4. ユーザはサーバから送られてきた $(q^{e'}\mathrm{h}(u))^{d'} \bmod n'$ に $q^{-1} \bmod n'$ をかけて, $\mathrm{h}(u)^{d'} \bmod n'$ を得る. これが受領書となる.

受領書送信フェーズ

ユーザは受領書をサーバに送信する.

4.4. 提案プロトコルの安全性

4.3.2 節で紹介したプロトコルは, 文献 [21] で議論されているような通常の電子投票プロトコルから不要な機能を削除して, 簡単化したものと考えることもできる. 本節では, たとえ前述のような簡単化を行ったとしても, 講義評価アンケートの実現に要求されるセキュリティ要件は損なわれていないことを示す.

不正回答防止

ユーザによる不正回答としては, 一人のユーザが二回以上回答をする多重回答と, 回答権のないユーザが回答を行う非権利者による回答が考えられる. 提案プロトコルにおいて回答をサーバに受理してもらうには, ハンドルとそのハンドルに対する回答用パスワードが必要となるため, 不正を行おうとするユーザは, 本来ならば入手できないはずの回答用パスワード（および対応するハンドル）をあらかじめ入手しておく必要がある. 一方, 提案プロトコルにおいて回答用パスワードを発行できるのはサーバのみである. サーバは, 回答用パスワードの発行に当たって常にユーザの正当性を検査するため, ユーザがサーバから不正に回答用パスワードを入手することはできない. また, サーバの助けを借りずに回答用パスワードを構成することは, 電子署名を偽造することに相当するため, （RSA 署名が安全である限り）ユーザがこれに成功することはない. よって, ユーザが不正に回答用パスワードを入手・構築することはなく, 不正回答を行うことはできない.

匿名性

ハンドル登録フェーズにおいて, ユーザは ID u, 初期パスワード P_u, $r^e \mathrm{h}(N_u) \bmod n$ をサーバに送っている. ここで, ハンドル N_u は一方向性ハッシュ関数によって処理され, しかもブラインド化乱数 r が掛かっているため, サーバはこれらの情報からユーザ u のハンドル N_u を知ることができない. 投票終了後, サーバはアンケートの回答に利用された全てのハンドルを知ることができるが, それでもハンドル登録フェーズでは $\mathrm{h}(N_u)$ を

ブラインド化して送っているため, サーバはユーザ u が使用したハンドルを求めることはできない.

次に, 回答フェーズでは, ユーザはハンドル N_u とアンケートの回答 A を送っているため, サーバはハンドルと回答とを対応付けることが可能である. しかし, 先に述べたように, サーバはハンドルとユーザ ID の対応をつけることができないため, 結局ユーザ ID とアンケートの回答を対応付けることはできない. また, 受領証の発行にもブラインド署名を用いているため, 受領書送信フェーズでユーザが受領証をサーバに送信しても, 受領証の情報からユーザ ID と回答とを対応付けることはできない.

上述したプロトコルにおける唯一の懸念は, 回答フェーズにおいて, サーバがユーザの ID を特定できないことを保障しなければならない点にある. 現実世界のコンピュータネットワークの多くでは, 自分が現在通信している相手が誰なのか, ある程度特定することが可能である. もし, 悪意を持ったサーバがこの特質を悪用すると, ユーザとその回答内容の関連付けが可能となってしまう. たとえば, あるユーザがハンドル登録フェーズと回答フェーズを同一の計算機上でほぼ同時刻に行ってしまうと, サーバに大きな手がかりを与えることになる. 理想的には, 回答フェーズの実行にはいわゆる匿名ネットワークを利用することが推奨される. しかし残念ながら, 匿名ネットワークはまだ研究段階の技術であり, 現在, 実用的・標準的な方式が確立されているとはいえない. 事実, ネットワークの匿名性の問題は, 電子投票プロトコルの実現における最大の技術的課題のひとつであると考えられる.

本研究では, この問題に対し, 運用上の工夫で問題を回避することを検討する. 上述したアンケートプロトコルは, 従来の電子投票プロトコルに比べて非常に単純なものとなっており, 各フェーズ間で継承しなければならない情報はきわめて少量である. もし, ユーザがそれら継承すべき情報を自分の手で直接取り扱うことができれば, ユーザは各フェーズを好きなときに任意の計算機から実行することが可能である. たとえば, ハンドル登録フェーズおよび受領証送信フェーズは自分専用の計算機から実行し, 回答フェーズのみ, 自分が特定されないような計算機を利用することで, 擬似的に匿名ネットワークを実現することが可能である. 実際, 回答フェーズの実行においてはユーザ ID の提示は要求されないため, たとえば大学内に設置されている（認証なしで誰でも利用可能な）共有パソコンを利用することや, 大学とは無関係のインターネットプロバイダ経由でサーバにアクセスすること, あるいは, 近年増加している無線 LAN のホットスポットを利用すること等により, 上記のような運用を行うことは十分現実的であると考えられる.

回答者と未回答者の識別

回答フェーズを実行したのち, ユーザは受領書 $\mathrm{h}(u)^{d'} \bmod n'$ を入手できる. ユーザはサーバの秘密鍵 d' を知らないので受領書を偽造することは不可能である.

また, ハンドル登録フェーズと回答フェーズで異なる RSA 鍵を利用しているため, ユーザは, ハンドル登録フェーズの署名機構を悪用して受領書の偽造を行うこともできない. もし両フェーズで同じ鍵を利用したとすると, 次のようにして受領書を偽造できる. ユーザはハンドル登録フェーズのステップ 2 で $r^e\mathrm{h}(N) \bmod n$ の代わりに $r^e\mathrm{h}(u) \bmod n$ をサーバに送信する. サーバはステップ 4 で, ユーザに $\mathrm{h}(u) \bmod n$ を送信する. これは受領書と同じ値となり, ユーザは回答フェーズを実行することなく受領書を入手できる. この不正を防ぐために, ハンドル登録フェーズと回答フェーズでは異なる鍵を利用する必要がある.

このように, ユーザは回答フェーズを実行しない限り受領書を手に入れることは不可能であるため, 受領書を保持していることはアンケートに回答したことの証明となる. 逆に, 受領書を提示できないユーザは, アンケートに回答していないものと判断できる.

4.5. 講義評価アンケートシステムの実装と評価

4.5.1 実証実験の目的と概要

4.4 節でも議論したように, 一定の前提条件のもとでは, 提案したアンケートプロトコルは実用上十分な安全性を有すると考えられる. 一方, 現実の世界でアンケートシステムを実現する際には, 前節で考えたような前提条件を実現することが難しかったり, あるいは何らかの理由で, プロトコル中の操作を他の操作で置き換える（近似する）必要が生じることもある. 通常, これらの要因はプロトコルの安全性を損なう方向に働くと考えられるが, その影響がどの程度のものになるのか, 事前に予測することは難しい. 本節では, 提案プロトコルの試作と実証実験を通じて, プロトコル実現の際の問題点の抽出と, 現実環境における提案方式の安全性について考察を行う. また, 電子的な講義評価アンケートを実際に利用する場合, 安全性以外の問題についても考慮する必要があるが, 本研究では以下に述べる 3 つの点に的を絞り, 実証実験を通じて評価を行った.

安全性要件以外で評価すべき点の第一は, プロトコルのユーザビリティである. 前節でも述べたように, 提案方式において匿名性を確保するためには, （匿名ネットワークが利

用可能でない限り） プロトコルの各フェーズを独立に, 異なる計算機上で実行する必要があり, ユーザに若干の負荷をかけることとなる. 実証実験を通じて, ユーザの負荷がどの程度のものとなるのかを評価する.

評価すべき点の第二は, 提案方式のスケーラビリティである. 一般に, 大学には多数の学生が在籍し, 同時に複数の講義科目を受講している. したがって, 講義評価アンケートシステムは多数の回答を処理する必要があり, システムが対象とする規模によっては, サーバにかかる負荷が問題になる可能性もある. 提案方式が比較的大規模な使用にも耐えるか否かを検証するため, 試作したシステムの処理性能の計測や, 実証実験中の動作ログ解析などを行う.

実証実験で評価する最後の点は, アンケートの回収率である. 冒頭でも書いたように, 本研究の目的は匿名性を確保しつつ, 高い回収率が実現できるような電子アンケートシステムを実現することである. アンケート未回答者を識別可能であるという提案方式の特性がアンケート回収率にどの程度影響するかを評価する.

安全性に加えて以上 3 点の評価を行うため, 2 種類のシステムを試作し, 2 種類の実験を行った. システムの試作にあたっては, 鍵長 1024 ビットの RSA 暗号を利用した. 一方向性ハッシュ関数は MD5 [25] を, 通信路の保護には SSL [23] を用いた. 提案方式では, ユーザ（学生）の側でもブラインド署名等の計算が必要になるため, ユーザの計算機上でなんらかのプログラムを実行する必要がある. 本研究では, ユーザの利便性や保守の手間についても検討し, ユーザ側プログラムを Java アプレットとして実装した. したがって, ユーザの視点からは, Web ブラウザを使用する延長として提案システムを利用することが可能である. また, 特別なソフトウェアのインストールや設定等の作業が不要になるため, ユーザは自宅の計算機や街中に設置されているインターネット端末からでも, アンケートに回答することが可能となる. 一方, サーバ（大学）については, Java アプレットとの親和性を考え, Java サーブレットにより実装を行った. システム内容の詳細については, それぞれの実験の節にて述べる.

4.5.2 実験 1 : システムのユーザビリティ評価

本実験の概要

実験 1 ではユーザビリティ評価を行う. 本実験で用いる試作システムでは, 提案プロトコルの各フェーズごとに別のプログラム（Java アプレット）を準備してユーザに提供す

る. ユーザは, 自分の手でフェーズ間のデータ移行を行うことになる. 比較的小規模な講義科目において, 実際の受講学生に本システムを利用してもらい, 使用感などについて聞き取り調査を行った.

ただし, 大学では一人の学生が多数の科目を受講していることが多いため, 前節で述べたプロトコルを科目ごとに実現するのでは, 被験者（学生） にとっての負荷が重くなり過ぎるのではないかと考えられる. そこで, 学生の負荷軽減を目的として, 回答用パスワードの内部構造に科目情報が反映されるようプロトコルを若干変更して, 提案方式の試作を行った. この変更により, 学生はハンドル登録フェーズを一度実行するだけで, 全ての受講科目に対する回答用パスワードを得ることができる. 一方, 本変更によりプロトコルのセキュリティ的要件はやや低下する. 複数の学生が結託して不正を行ったり, 一人の学生が, 自分の回答権を犠牲にして受講していない科目の回答用パスワードを得ることが可能になる場合がある. また, 学生は複数科目にわたって同一のハンドルを使用することになるため, どのハンドルがどの科目のアンケート回答に使われたかを分析することで, ハンドルと学生の実体との推測が可能になるおそれもある. 本格的な選挙であればこれらは大きな問題であるが, 講義評価アンケートでは一般の選挙ほどの安全性は要求されないことを考慮し, ここではユーザの利便性を優先することとした.

試作システムについて

本実験で用いる試作システムでは, プロトコルの各フェーズごとに, それぞれ独立したプログラム（Java アプレット）を用意する. 各フェーズを担当するプログラムはそれぞれ独立しているため, ハンドルや回答用パスワード, 受領書等の中間情報をなんらかの方法で直前フェーズから受け取る必要がある. 一方, ユーザの計算機保護のため, Java アプレットによるユーザ計算機のリソースアクセスは強く制限されており, たとえば上記の中間情報をユーザ側計算機上のファイルとして残すことは極めて困難である. したがって, ユーザは自分自身の手で, ハンドルや回答用パスワード, 受領書情報などを取り扱う必要がある. 本来ならば, これらの情報はできるだけ乱雑で大きな 2 進値であることが望ましいが, ユーザの取り扱いの利便性を考え, 以下のように対処することとした. まずハンドルは, ユーザ自身によって好きな文字列を選定することとした. 機械的にハンドルを選定する場合と比較して, 異なるユーザが同一のハンドルを選ぶ確率が若干増加する可能性があるが, ユーザ自身はハンドルを覚えやすいという利点がある. また, 回答用パスワードおよび受領書については, S/Key 等の使い捨てパスワードの実装で見られるように, 2 進

値をハッシュして適当な英単語に置き換える手法を採用することとした.

サーバ側では, RSA 鍵転送, ユーザ認証および回答用パスワード発行, 回答受付および受領書発行, 受領書受付の各機能ごとに Java サーブレットを用意し, ユーザ側 Java アプレットからの要求に応じてサービスを提供する. これらサーブレットの構成は, 後述する実験 2 のものとほぼ同様であるため, 詳細はそちらで述べることとする.

実験の詳細と結果

実験 1 は, 平成 13 年度第 IV 期に開講された小規模な講義科目において, 前節で述べたシステムを利用して行った. 被験者の数は 12 名で, 必ずしもセキュリティの専門知識を有しない. アンケートの実施にあたっては, プロトコルの概要や試作システムの使い方等について簡単な説明を行い, アンケート回答期間として約一週間を設定した. 学生は操作方法などを概ね理解したようであり, 操作上の問題で回答できなかったというケースはなかった. ただし, 回答用パスワード等の中間情報の取り扱いが面倒である, 回答科目数が多いときには学生の負荷が大きくなりすぎるのではないか等, 操作性に関して疑問を投げかけるコメントも何件か寄せられた.

また, ハンドルとして自分のユーザ名や本名を使用する者や, 同一の計算機上ですべてのフェーズを実行する者も多くみられた. 少なくとも今回の実験では, 匿名性実現のために余計な手間をかけるよりも, より簡便に操作ができることを望む学生が多数を占めることがわかった. 本実験では被験者数も少数であり, 観察された傾向を単純に一般化することはできないが, より多くのユーザに利用してもらうためには, システムの操作性が非常に重要であると考えられる. その意味で, 提案方法をそのまま実現し, データ移行等の作業を全てユーザに任せてしまうのは, 残念ながらあまり現実的であるとは言えない.

本問題を解決するための一手法として, IC カード等のデバイスを利用し, 大学における事務作業や手続きとリンクして, アンケート実施に必要な情報の授受を行うことが考えられる. たとえば, 学期の最初の履修登録作業の一部として, 事務局の端末などでハンドル登録フェーズを実行することを考える. ハンドルの選択や, 一連のブラインド署名関連の計算などは IC カードが行い, ハンドルと回答用パスワードはカード内に蓄積しておく. 学期終了時には, 個人の端末と IC カード内に蓄積された情報を用いて回答フェーズを実行し, 受領書は再度 IC カード内で記録する. 成績発行時には, 事務局の端末等で受領書の確認を行えば, アンケート回答者と未回答者を識別可能である. 以上のように IC カードを利用することで, ユーザは中間情報に触れる必要がなくなり, 操作性は格段に改

善すると考えられる. また, ユーザは無意識のうちに, 各フェーズで異なる計算機を利用することになるため, 匿名性の実現上も好ましいといえる.

4.5.3 実験 2 : スケーラビリティとアンケート回収率の評価

本実験の概要

実験 2 では, より多くの被験者を対象とし, スケーラビリティとアンケート回収率の評価を行う. 本来であれば, 前節でも考察したように, IC カード等を導入して操作性を改善してからスケーラビリティ評価を行うべきところであるが, 今回の実験ではそこまでの環境設定が難しいこと, サーバの負荷を評価するためには, IC カードの有無は本質的でないことを考慮し, スケーラビリティ評価に特化したシステムを試作して利用することとした. 本実験で使用する試作システムでは, ユーザは全てのフェーズを同一のプログラム (Java アプレット) 内で実行する. したがって, 通信ログ等を解析することにより, どの学生がどの回答を行ったか追跡することが可能である. 一方, ユーザは中間データに触れる必要がないため, ユーザが利用するプログラムの操作性は格段に向上する. 匿名性の問題は, 先にも述べたように IC カード等の導入で解決できること, 今回の実験の目的を考えると, 操作性を高めて, より多くの学生に実験に参加してもらうことのほうが意義が大きいことをふまえて, 次節で述べるようなシステムを構築した.

試作システムについて

本実験では, ユーザに対して 1 個の Java アプレットを提供する. この 1 つのアプレットが, ユーザ認証, ハンドル選定, 回答用パスワードの取得, 回答の送信と受領書の構築など, 一切の作業を行う. ユーザは中間情報に触れる必要が無いため, 実験 1 のシステムに比べて, 使いやすさは格段に上昇することが期待される. また, ユーザが署名鍵, ハンドル, 回答用パスワード, 受領証情報などに直接触れる必要がないため, 実験 1 で考えたような, セキュリティを若干犠牲にして操作性を高める工夫をする必要がなく, 安全性の観点からも優れていると考えられる.

サーバ側では, 実験 1 と同様, RSA 鍵転送, ユーザ認証および回答用パスワード発行, 回答受付および受領書発行, 受領書受付の各機能ごとに Java サーブレットを用意し, ユーザ側 Java アプレットからの要求に応じてサービスを提供する. 実験では, パーソナル

コンピュータ （AMD Athlon 1GB プロセッサ, 主記憶容量 512MB, OS は FreeC85D 4.5-RELEASE） 上にサーバを構築した. Java サーブレットのコンテナとしては, tomcat (バージョン 3.2.3） を採用し, 署名の作成および検証には Java の BigInteger オブジェクトを使用する. サーブレットの機能, プロトコル上での役割, 性能は以下のとおりである. ただし, 以下で述べる実行時間の中には, サーブレット起動のために OS やコンテナ (tomcat) で消費される時間は含まれない. 時間の計測にあたっては, Java の Date オブジェクトを使用して実時間を計測した. したがって, 同じ環境であっても, サーバの負荷状況によっては実行時間に多少の変動があるものと考えられる.

- RSA 鍵転送: 4.3.2 節各フェーズの「準備」段階における, 鍵を公開する機能を提供するサーブレットである. クライアントとなる Java アプレットから講義科目名を受け取り, 対応する科目のアンケート回答に使用される鍵 （ハンドル登録フェーズ用と回答フェーズ用） を返送する. 本サーブレットは, 1 秒あたり約 400 のリクエストを処理することが可能である. 単純計算では, 1 リクエストの処理に必要となる時間は約 2.5 ミリ秒となる.

- ユーザ認証および回答用パスワード発行: ハンドル登録フェーズの第 3 および第 4 ステップに相当する機能を提供するサーブレットである. ユーザ ID および初期パスワードとしては, 学内で統一的に管理されている学生のユーザ名とログインパスワードを利用する. 本サーブレットは, Java アプレットからユーザ名, パスワード, 講義科目名, ブラインド化されたハンドルパスワード （ステップ 2 の $r^e h(N_u) \pmod{n}$ ） を受け取り, パスワードが正しいか, その学生が指定された講義を受講しているか, まだ回答用パスワードを受け取っていないか等を検査し, 問題がなければ署名を作成してクライアントに返送する. 本サーブレットの実行には, 1 リクエストあたり約 200 ミリ秒が必要となる. ただしこの時間の中には, パスワード認証を行うために NIS (YP) データベースを参照する時間も含まれている. ユーザ認証機構をネットワークに依存しない形で実現すれば, 効率は改善可能である. パスワード認証にかかる時間を除外すると, 実行時間は約 120 ミリ秒となる.

- 回答受付および受領書発行: 回答フェーズの第 2 および第 3 ステップに相当する機能を提供するサーブレットである. クライアントから, ハンドル, 回答用パスワード, 講義科目名, 設問への回答, 受領証計算のための情報 ($q^{e'} h(u) \pmod{n'}$) を受け取り, ステップ 2 で定められた検査を行った後, 受領証に署名をして返送する.

本サーブレットの実行には, 1 リクエストあたり約 220 ミリ秒が必要となる. この処理では, 回答用パスワードの検査と受領証への署名を行う必要があり, 実行時間のほとんどが, BigInteger オブジェクトのベキ乗剰余演算に使われていると考えられる.

- 受領書受付: 受領書送信フェーズに相当する. クライアントからユーザ名と受領書を受け取り, 受領書の正しさを検証して記録する. 受領書の正しさを確認してから記録するため, 1 リクエストあたり約 100 ミリ秒が必要となる.

各サーブレットの実行時間は数百ミリ秒単位であり, クライアントからのリクエストに対して, きわめて短時間で各処理を完了することが可能である. 数秒おきにリクエストが来るような環境であれば, サーバにはほとんど負荷がかからないと予想される. 万一, あるサーブレットの処理中に他のリクエストが到着した場合でも, サーブレットコンテナが別スレッドを生成して処理を行うため, ユーザに対する処理が滞ることはない.

実験の詳細と結果

実験 2 は, 平成 14 年度第 I 期, 第 II 期に開講された全ての講義を対象とし, 原則として受講生全員参加で実施した. 第 I 期の講義の場合, 少なくとも 1 つの講義を履修している学生 165 名が対象となり, 開講科目数は 20 科目である. 履修登録件数は全部で 1383 件であった. 第 II 期では, 学生数 164 名, 科目数 25 で履修件数は 1616 件であった. 第 I 期, 第 II 期の講義期間終了後, それぞれ約一週間の期間を設定してアンケートに回答するよう学生に依頼した. ただし, 学生には事前に, アンケートに回答したかどうかが担当教官には把握できる旨をアナウンスした.

その結果, 第 I 期については 680 件, 第 II 期については 559 件の回答が寄せられた. 履修登録数に対するアンケート回収率は, それぞれ 49%, 35% となった. 実際には, 学期途中で講義を放棄する学生が少なからず居るため, 最後まで出席していた学生数に対する回収率は, これよりも大きな値となる. 前年度同時期には, セキュリティ的要件を考慮しない無記名電子アンケートシステムを使用していたが, そのときの回収率はそれぞれ 6%, 17% であった. 昨年度と比べて回収率が大幅に上昇しており, 所期の目的は達成したことになる.

動作ログを確認したところ, サーバに到着するリクエストは, 多い時でも数十秒から数分おきであった. この程度の規模のアンケートであれば, 通常のパーソナルコンピュータ

程度でも全く問題なくサーバとして利用可能である. 実際, 著者の主観的な判断ではあるが, アンケートシステム稼働中でもサーバ計算機のパフォーマンス低下はみられず, 性能的にはまだ余裕があると思われる. もっと規模の大きなアンケートであっても, 現システムで十分対応可能であると予測できる.

4.5.4 実験の総括

2 つの実験を通じて, セキュリティ以外の要件について, 提案プロトコルの評価を行った. 実験 1 の結果より, 提案プロトコルに忠実に従うと, ユーザの負荷が大きくなることが明らかとなった. しかし, この問題については, 本文中でも述べたように IC カード等を併用することで回避可能であると考えられる. 近年, 私立大学を中心とした多くの大学において学生証を IC カードに置き換える動きがあることを考慮すると, この問題回避法は十分現実的であると考えられる. 一方, 実験 2 を通じて, 提案方式のスケーラビリティについての検討を行った. また, 今回の実験でアンケート回収率を向上させることに成功したが, これは, 試作システムが有する未回答者識別機能を, 学生が強く意識しているためと考えられる.

4.6. まとめ

大学等における講義評価アンケートを念頭におき, 匿名性の保証と回答者・未回答者の識別が可能なアンケートプロトコルを提案した. また, 提案したプロトコルに基づいてアンケートシステムの実装を行い, 実際に講義評価アンケートとして試験的に運用を行った. 本方式は講義評価だけにとどまらず, さまざまな場面で応用が可能であると考えられる. たとえば, 海外の学会における役員選挙等では, 投票率を上げるため, 投票した人に抽選で景品を授与するような試みもさかんに行われているが, 提案方式はそのような選挙にも応用可能である. また提案方式は, 電子的な投票方式と従来の紙ベース投票方式とを併用する際にも有用である. 大規模な選挙や投票等では, 技術的な経過措置として, あるいはデジタルデバイドに対する配慮として, 電子投票と紙ベース投票の両者を実行し, その集計結果を合算して最終的な投票結果とするような運用が現実的であると考えられる [2]. その際には, 電子的に投票を行った投票者が紙ベースの投票には参加しないよう, 電子投票を行ったか否かの識別が必要となる. 提案方式を利用すれば, 投票の匿名性を確保しつつ, 上記識別が可能となる. ただし, 二重投票を行おうとするユーザは, 受領書を

提出しない可能性があるため, 提案法そのままでは安全であるとはいえない. ハンドル登録フェーズの実行をもって投票を行ったと解釈することも考えられるが, 受領書の取り扱いについて, 提案方式にはまだ改善の余地があると考えられる.

電子投票・アンケート方式についてはこれまでも多くの研究が行われてきた. しかし, 実際にそれらの方式を実現するためには, たとえば法律的な問題や道徳的な問題などを解決する必要があり, 実証実験すらままならないというのが現状であった. 今回, 大学という比較的小さな組織においてではあるが, いわゆる電子アンケート方式を現実の用に供するべく試みた. アンケート回収率の観察を続行する等, 今後も長期的な評価が必要ではあるが, 本研究で行った試みは, 今後の電子投票方式の実用化研究に些少ながらも貢献するものと考えられる.

第5章　あとがき

本論文では, コンピュータネットワークを利用したサービスの安全な実現のために必要な技術に関する研究として, 電子透かし, ソフトウェアの従量制課金法, 匿名電子アンケートプロトコルに関する研究について述べた.

まず, ネットワーク上での著作権保護のために利用可能な技術である電子透かしについて, コンピュータプログラムに対する電子透かし法を提案した. 提案方式は Java, C, C++ などの言語で書かれたプログラムに対して利用可能であり, コンピュータプログラムの不正配布抑止に役立つと考えられる. 次に, 電子透かしで利用される結託耐性符号に対する新しい攻撃法を提案した. 提案攻撃は, 結託耐性符号とそのトレースアルゴリズムの性質を利用した攻撃法である. この研究によって, 今後結託耐性符号を設計する場合には, 符号の性質を利用した攻撃も考慮して設計する必要があることが示された.

次に, ソフトウェアの回数券型従量制課金法を提案した. 従来のように, 無期限でソフトウェアを利用できるライセンスを購入してからソフトウェアを利用する方式とは異なり, 利用した分だけ料金を支払うことができる. 従来は, 個人で高価なソフトウェアを利用したい場合など, そのライセンス代金の高さから購入するのはユーザにとって負担となり, 不正コピーの原因となっていた. 提案方式を用いることで, 比較的低コストでソフトウェアを利用できることになり, 不正コピーの減少にも役立つと考えられる. また, ソフトウェア開発者にとっても, 不正コピー減少は大きな利点であり, 提案手法を利用する価値は大きいと考えられる.

最後に, 大学での講義評価などで利用可能な匿名アンケートプロトコルを提案した. 提案方式は, アンケートを匿名で行いつつ未回答者の特定ができるという機能を持つ. 実際に, プロトコルを WWW 上に実装し, 評価実験を行った. その結果, 提案手法を用いる前と比較して回答率が上昇することが確認できた. 一方, 提案手法は, 3 つのフェーズからなり, 各フェーズにおいてユーザは自分でデータを移行しなければならないため, 利便性に欠ける面もある. 今後の課題として, ユーザの利便性を考慮した方式の提案が考えられる. 提案方式は, 大学での講義評価だけではなく, 学会での役員選挙や会社内での評価など,

匿名で行う必要がありかつ高い回答率が望まれるようなさまざまな投票に利用できる.

謝辞

本研究を進めるにあたり，主指導教官である関 浩之教授には，終始懇切なご指導とご鞭撻を賜りました, ここに心から感謝の意を表します.

副指導教官である楫 勇一助教授には, 大学院入学時より様々なご指導をいただきました, ここに謹んで感謝いたします.

ご多忙のなか審査委員をお引き受けいただきました渡邉 勝正教授, 伊藤 実教授に感謝いたします.

著者は現在, 独立行政法人 産業技術総合研究所で特別研究員を務めておりますが, 上司の渡辺 創博士には, 職務と研究に対し暖かいご支持をいただきました. ここに深く感謝いたします. また, 産業技術総合研究所 情報処理研究部門のみなさまには, 職務と研究に関してさまざまなサポートをいただき, 深く感謝いたします.

大阪大学の藤原 融教授には, シミュレーション結果の検定法について懇切丁寧なご教示をいただきました. 心からお礼申し上げます.

NRI セキュアテクノロジーズの岡 博文氏には在学中, 共同で研究するに当たり大変有用な議論をさせていただきました. ここに感謝いたします.

在学中は, 研究や息抜きなど生活全般にわたり, 情報基礎学講座のみなさまには良い環境をご用意いだたきました, たいへん感謝しております.

この論文をまとめるにあたり, この他にも多くの方のお世話になりました, この場をお借りして心より感謝申し上げます.

参考文献

[1] 朝日新聞, “ベストティーチャー選び,” 1 月 5 日 (夕刊), 2002.

[2] 朝日新聞, “ネット投票で行使率が上昇,” 5 月 22 日 (朝刊), 2002.

[3] D. Boneh and J. Shaw, “Collusion-Secure Fingerprinting for Digital Data,” Proc. of Crypto'95, LNCS 963, pp.452–465, 1995.

[4] D. Boneh and J. Shaw, “Collusion-Secure Fingerprinting for Digital Data,” IEEE Transactions on Information Theory, vol.44, no.5, pp.1897–1905, 1998.

[5] D. Chaum, “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,” Communications of the ACM, vol.24, no.2, pp.84–88, 1981.

[6] D. Chaum, “Blind Signatures for Untraceable Payments,” Proc. of Crypto'82, Plenum Press, pp.199–203, Santa Barbara, California, 1982.

[7] D. Chaum, “Security without Identification: Transaction Systems to Make Big Brother Obsolete,” Communications of the ACM, vol.28, no.10, pp.1030–1044, 1985.

[8] Decafe Pro - The Java Decompiler, http://decafe.hypermart.net.

[9] A. Fujioka, T. Okamoto and K. Ohta, “A Practical Secret Voting Scheme for Large Scale Elections,” Proc. of AUSCRYPT'92, LNCS 718, pp.244–251, Gold Coast, Queensland, Australia, Dec. 1992.

[10] H-J. Guth and B. Pfitzmann, “Error- and Collusion-Secure Fingerprinting for Digital Data,” Proc. of the 3rd International Workshop on Information Hiding (IH '99), LNCS 1768, pp.134–145, 2000.

[11] 井上 彰, “電子透かし,” 光芒社, 1997.

[12] Java Optimize and Decompile Environment (JODE), http://jode.sourceforge.net.

[13] 松井 甲子雄, “電子透かしの基礎,” 森北出版, 1998.

[14] H. Muratani, “A Collusion-Secure Fingerprinting Code Reduced by Chinese Remaindering and Its Random-Error Resilience,” Proc. of the 4th International Workshop on Information Hiding (IH ’01), LNCS 2137, pp.303–315, 2001.

[15] 村山 隆徳, 満保 雅浩, 岡本 栄司, 植松 友彦, “ソフトウェアの難読化について,” 電子情報通信学会技術研究報告, ISEC95–25, vol.95, no.353, pp.9–14, 1995.

[16] 村山 隆徳, 満保 雅浩, 岡本 栄司, 植松 友彦, “プログラムコードの難読化について,” 1996 年暗号と情報セキュリティシンポジウム (SCIS ’96), SCIS96–8D, 1996.

[17] Mocha - the Java Decompiler, http://www.brouhaha.com/%7Eeric/computers/mocha.html.

[18] 森 亮一, 河原 正治, 大瀧 保広, “超流通 : 知的財産権処理のための電子技術,” 情報処理, vol.37, no.2, pp.155–161, 1996.

[19] 森 亮一, 田代 秀一, “ソフトウェアサービスシステム（SSS）の提案,” 電子情報通信学会論文誌, vol.J70-D, no.1, pp.70–81, 1987.

[20] 門田 暁人, 高田 義広, 鳥居 宏次, “ループを含むプログラムを難読化する方法の提案,” 電子情報通信学会論文誌, vol.J80-D-I, no.7, pp.644–652, 1997.

[21] H. Nurmi, A. Salomaa and L. Santean, “Secret Ballot Elections in Computer Networks,” Computers & Security, vol.10, pp.553–560, 1991.

[22] M. Ohkubo, F. Miura, M. Abe, A. Fujioka and T. Okamoto, “An Improvement on a Practical Secret Voting Scheme,” Proc. of the 1999 Information Survivability Workshop (ISW’99), LNCS 1729, pp.225–234, Kuala Lumpur, Malaysia, Nov. 1999.

[23] OpenSSL, “The Open Source Toolkit for SSL/TLS,” http://www.openssl.org/.

[24] C. Park, K. Itoh and K. Kurosawa, “Efficient Anonymous Channel and All/Nothing Election Scheme,” Proc. of EUROCRYPT’93, LNCS 1334, pp.248–259, Lofthus, Norway, May 1993.

[25] R. Rivest and S. Dusse, "The MD5 Message-Digest Algorithm," Network Working Group Internet Draft, RSA Data Security Inc., 1991.

[26] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol.21, no.2, pp.120–126, 1978.

[27] K. Sako and J. Kilian, "Receipt-free Mixtype Voting Scheme - A Practical Solution to the Implementaion of a Voting Booth," Proc. of EUROCRYPT'95, LNCS 921, pp.393–403, Saint-Malo, France, May 1995.

[28] 吉岡克成, 松本勉, "結託耐性符号のランダム誤りに強い追跡アルゴリズム," 電子情報通信学会技術研究報告, ISEC2001-52, pp.247–254, 2001.

[29] 吉岡克成, 松本勉, "結託耐性符号のランダム誤りに強い追跡アルゴリズム（その2）," 2002年暗号と情報セキュリティシンポジウム (SCIS '02), pp.1021–1026, 2002.

[30] K. Yoshioka and T. Matsumoto, "Random-Error-Resilient Tracing Algorithm for a Collusion-Secure Fingerprinting Code," 情報処理学会論文誌, vol.43, no.8, pp.2502–2510, 2002.

研究業績

I. 本論文に関するもの

論文

1. 北川 隆, 岡 博文, 楫 勇一, “大学における講義評価のための匿名アンケートプロトコルとその試作,” 情報処理学会論文誌, vol.44, no.9, pp.2353–2362, Sept. 2003.

2. Hajime Watanabe and Takashi Kitagawa, “A Random-Error-Resilient Collusion-Secure Fingerprinting Code, Randomized c-Secure CRT Code,” IEICE Transactions on Fundamentals, vol.E86-A, no.10, pp.2589–2595, Oct. 2003.

国際会議（査読有）

1. Takashi Kitagawa and Yuichi Kaji, “A Protocol for the Pay-per-Use Pricing System of Software through Partially Off-Line Network,” Proceedings of the Fourth International Symposium on Wireless Personal Multimedia Communications (WPMC ’01), pp.1147–1152, Aalborg, Denmark, Sept. 2001.

2. Hajime Watanabe and Takashi Kitagawa, “An Attack for a Fingerprinting Code and its Success Probability,” Proceedings of 2002 International Symposium on Information Theory and Its Applications (ISITA ’02), pp.555–558, Xi’an, P.R.China, Oct. 2002.

3. Takashi Kitagawa, Hirofumi Oka and Yuichi Kaji, “An Anonymous Questionnaire System for Rating Faculty Courses in Universities,” Proceedings of 2002 International Symposium on Information Theory and Its Applications (ISITA ’02), pp.559–562, Xi’an, P.R.China, Oct. 2002.

4. Hajime Watanabe and Takashi Kitagawa, “An ID Coding Scheme for Fingerprinting, Randomized c-Secure CRT Code,” Proceedings of the Fourth International Conference on Information and Communications Security (ICICS ’02), LNCS 2513, pp.173–183, Singapore, Dec. 2002.

特許出願

1. “回答収集システム、回答収集方法、サーバ装置、コンピュータをサーバ装置として機能させるためのプログラム及びそのプログラムを記録するコンピュータで読み取り可能な記録媒体,” 北川 隆, 岡 博文, 楫 勇一, 特許出願 2002-91392 (出願日 2002 年 3 月 28 日).

国内シンポジウム発表

1. 北川 隆, 楫 勇一, 嵩 忠雄, “Java で記述されたプログラムに対する電子透かし法,” 1998 年暗号と情報セキュリティシンポジウム (SCIS ’98), SCIS98-9.2.D, 1998.

2. 北川 隆, 楫 勇一, 関 浩之, “難読化後も検出可能な Java プログラムに対する電子透かし法,” コンピュータセキュリティシンポジウム 1998 論文集 (CSS ’98), pp.63–68, 1998.

3. 北川 隆, 楫 勇一, “モバイル端末に適したソフトウェア課金方式,” 2001 年暗号と情報セキュリティシンポジウム (SCIS ’01), 9C-1, pp.489–494, 2001.

4. 岡 博文, 北川 隆, 楫 勇一, “大学における講義評価のための匿名アンケートシステムについて,” コンピュータセキュリティシンポジウム 2001 論文集 (CSS ’01), pp.361–366, 2001.

5. 岡 博文, 北川 隆, 楫 勇一, “大学における講義評価のための匿名アンケートシステムの設計と試作,” 2002 年暗号と情報セキュリティシンポジウム (SCIS ’02), 14A-4, pp.1045–1050, 2002.

II. その他

国際会議（査読有）

1. Takashi Kitagawa and Hajime Watanabe, “An Auction Protocol which Hides Losers and Their Prices,” Proceedings of the International Symposium on Information Theory and Its Applications (ISITA2000), T-A-3, pp.509–512, Hawaii, U.S.A., Nov. 2000.