卒業制作 2003 年度 (平成 15 年度)

RFID のプライバシ保護機構に関する研究

指導教員

徳田 英幸

村井 純

楠本 博之

中村 修

南 政樹

慶應義塾大学 環境情報学部

成瀬大亮

t00743dn@sfc.keio.ac.jp

平成 16 年 1 月 15 日

概 要

近年、Suica や Edy などの RFID 技術が一般的に普及し、本や洋服などの非計算機を個体識別する技術の開発が試みられている。RFID を用いることで実空間の物体の存在を認識し、その情報を用いた様々なアプリケーションを展開できる。しかし、RFID システムは、タグとリーダ間で非接触交信を行うため、リーダを持っていれば誰でも RFID タグの中に格納されている ID を取得できてしまう。さらに、Auto-ID システムのように、ID が構造化されている場合、ID を取得するだけで RFID が貼付されている物体の素性が憶測できてしまう危険性がある。

本研究では、オブジェクトの所有者が家庭などで、私的なオブジェクトを管理することを想定し、第三者へのプライバシの漏洩を防止することを目的とする。RFID タグを貼付することにより個体識別可能なものをオブジェクトと定義し、RFID タグに保存されている ID、および ID に関連づけられた情報をプライバシと定義する。

RFID のプライバシ保護の既存の手法として、ID を暗号化・隠蔽する手法が存在する。しかし、現実社会では、RFID が貼付されたオブジェクトには所有者が存在し、オブジェクトが譲渡される際に、その所有権は別の所有者に委譲される。既存のプライバシ保護の手法では所有権の概念がないため、所有権の委譲が発生した際に、以前の所有者からプライバシが漏洩する危険性がある。

この問題を解決するため、本研究では RFID に、オーナという所有権を表す属性を設定し、オーナの認証を行うことを提案する。本研究では、オーナの認証と ID の暗号化を用いたプライバシ漏洩防止のシステムモデルを提案する。さらに、プロトタイプの設計・実装を行う。そして、既存研究との機能比較・システムの機能評価を行うことにより、オブジェクトの所有者が変わった際に、ID を取得できる権限を次の所有者に委譲できる点で、本システムの優位性を示すことができた。

キーワード

1, RFID　2, プライバシ　3, 暗号化　4, 認証　5, Auto-ID

慶應義塾大学 環境情報学部
成瀬 大亮

abstract

Recent popularization in RFID technology, as represented by SUICA and Edy, has lead to further researches in developing technologies to identify non-computers such as books and clothes. The use of RFID allows computers to identify objects in real-world and allows various applications which use obtained information to be developed. However, information stored in RFID tags can be read easily with RFID readers since RFID system is a system which RFID tags and readers communicate using radio waves. Furthermore, systems which uses structured IDs, as in Auto-ID system, has risks of allowing users with RFID readers to speculate the object from the information obtained from the RFID tags.

This research aims to protect private information from outsiders in the situation where RFID systems are used in families to manage intimate properties. A matter which can be identified by RFID tag attached to it is called object and ID and information associated with the ID is called privacy in this research.

Existing method to protect privacy is to encrypt/conceal ID. However, in the real-world situation, objects belong to the owners and the rights of ownership change as the objects belong to other owners. Such situation is especially common when commercial products are sold to customers. Existing method does not have the notion of ownership and the risk of privacy leakage can not be avoided as the objects are handed to new owners.

This research approaches such problem by proposing the attribute which represent ownership and by providing the authentication mechanism by owners. Privacy is secured by authentication using ownership and encryption of IDs. Prototypes of this mechanism are designed and implemented in this thesis. Developed system is tested and comparisons with existing systems are conducted to show the superiority of proposed mechanism. The system was proven to be effective in the real-world situation by providing mechanisms to transfer rights to obtain privacy from tags from previous user to new user.

Keywords

1, RFID　2, Privacy　3, Encryption　4, Authentication　5, Auto-ID

Faculty of Environmental Information, Keio University
Daisuke Naruse

目 次

第 1 章　序論 **1**
1.1 本研究の背景 1
1.2 研究目的 2
1.3 本論文の構成 3

第 2 章　Auto-ID システムの問題点と既存研究 **4**
2.1 Auto-ID システム 4
2.1.1 EPC Global の概要 4
2.1.2 Auto-ID システム概要 4
2.2 Auto-ID システムの問題点 7
2.3 既存研究 8
2.3.1 RFID タグとリーダ間の認証 9
2.3.2 EPC の隠蔽・暗号化・認証 9
2.3.3 既存研究の考察 13

第 3 章　利用モデル **15**
3.1 本研究で想定する利用モデル 15
3.2 要求事項 17
3.3 考えられるシステムモデル 17
3.3.1 EPC の取得を防止する方法 17
3.3.2 EPC からアプリケーションに至る間で情報の取得を防止する方法 . . . 18
3.3.3 考えられるシステムモデルの考察 19

第 4 章　SARU の設計 **20**
4.1 本研究で提案するシステム 20
4.1.1 システム概要 20
4.1.2 機能要件 20
4.1.3 暗号方式 22
4.2 SARU のシステム構成 22
4.3 Auto ID システムとの差分 24
4.4 EPCADS 25
4.4.1 EPCADS の機能要件 25
4.4.2 EPCADS の動作概要 25
4.5 EPCAC 27
4.5.1 EPCAC の機能要件 27
4.5.2 EPCAC の動作概要 28

4.6 Owner Management Client 29
4.6.1 Owner Management Client の機能要件 29
4.6.2 Owner Management Client の動作概要 29
4.7 Key Registry Client 30
4.7.1 Key Registry Client の機能要件 30
4.7.2 Key Registry Client の動作概要 30

第 5 章 SARU の実装 31
5.1 実装環境 31
5.2 実装概要 33
5.3 ID の暗号化プログラム 33
5.4 EPCAC 34
5.4.1 RF リーダ・ライタ制御 35
5.4.2 EPCADS への認証復号化要求 36
5.5 EPCADS 37
5.5.1 オーナ認証 37
5.5.2 ID 復号化 38
5.5.3 EPCADS 上に構築するデータベース 39

第 6 章 評価 40
6.1 評価方針 40
6.2 機能評価 40
6.2.1 実験環境 40
6.2.2 ID を暗号化・復号化できたか 40
6.2.3 RFID タグに所有権を設定できたか 42
6.2.4 所有権の委譲はできたか 43
6.2.5 Long-term Tracking 対策できたか 43
6.2.6 既存研究との機能比較 44
6.2.7 まとめ 44
6.3 性能評価 45
6.3.1 評価目的 45
6.3.2 評価方法 45
6.3.3 実験環境 45
6.3.4 実験結果 46
6.3.5 まとめ 46

第 7 章 結論 48
7.1 まとめ 48
7.2 今後の課題 48
7.2.1 規模性の確保 48
7.2.2 暗号化アルゴリズム 49

付 録 A RSA 暗号 **53**
A.1 RSA 暗号のアルゴリズム 53
A.2 RSA 暗号の暗号化・復号化プログラム 53

図 目 次

1.1 様々な RFID の例 (左から Suica、Edy、Speedpass) 1

2.1 Auto-ID システムのアーキテクチャ概要図 5
2.2 Auto-ID システムにおける EPC の構造 5
2.3 Alien Technology 社の EPC チップ 6
2.4 Nano Block の様子 (チップ一辺 0.17mm) 6
2.5 鞄の中身の情報が第三者に漏洩 8
2.6 Anonymous EPC の処理フロー 10
2.7 Anonymous EPC 11
2.8 Long-term Tracking に対処した Anonymous EPC の処理フロー 11
2.9 Encrypted EPC 12
2.10 E-signed EPC 12
2.11 Encrypted E-signed EPC 13

3.1 本研究で想定するオブジェクトの管理モデルの例 15
3.2 本研究における利用モデル 16

4.1 本研究で提案するシステムモデル概要図 21
4.2 暗号化されいているため鞄の中身は判別不能 21
4.3 SARU のシステムアーキテクチャ概要図 23
4.4 EPCADS 上のモジュール相関図 26
4.5 EPCAC 上のモジュール相関図 28
4.6 Owner Management Client 上のモジュール相関図 29
4.7 Key Registry Client 上のモジュール相関図 30

5.1 本研究で使用した RF リーダ・ライタ 32
5.2 本研究で使用した RFID タグ 32
5.3 EPCAC のフローチャート 34
5.4 put_serial_string() 35
5.5 get_serial_char() 35
5.6 EPCADS への ID 送信・受信モジュール 36
5.7 EPCADS のフローチャート 37
5.8 復号化関数 38
5.9 テーブル作成と確認 39
5.10 データ入力と確認 39

6.1 暗号化アプリケーションの動作画面 41
6.2 復号化アプリケーションの動作画面 42
6.3 EPCAC の動作画面 43
6.4 データベース上のオーナ情報 43
6.5 データベースのエントリー数に応じた EPCADS の性能変化 46

表 目 次

2.1 既存研究の考察 13

5.1 サーバ環境 31
5.2 クライアント環境 31
5.3 本研究で使用した RF リーダ・ライタ 31
5.4 本研究で使用した RFID タグ 32

6.1 既存研究との機能比較 44
6.2 EPCADS の実験環境 45
6.3 EPCAC の実験環境 45

第1章　序論

本章では、本研究の背景、および研究の目的について述べ、本論文の構成について述べる。

1.1　本研究の背景

近年、JR 東日本の非接触 IC カードを用いた出改札システム Suica(Super Urban Intelligent CArd) [1] や、電子マネーの Edy [2]・Speedpass [3]、回転寿司での自動精算システム [4] など (図 1.1[1])、比較的身近に RFID(Radio Frequency IDentification) 技術が用いられるようになってきた。

図 1.1: 様々な RFID の例 (左から Suica、Edy、Speedpass)

RFID とは、IC タグを使った無線通信による識別技術であり、数 cm 程度の大きさのタグ (RFID タグ) にデータを記録し、電波や電磁波で読み取り機 (リーダ) と交信をする。RFID の特徴として、非接触でのデータ読出しや書換えが可能、汚れやほこり等の影響を受けない、交信領域内の IC タグへの順次アクセスや複数の IC タグへの一斉アクセスが可能なことなどが挙げられる。このように RFID ををを用いることで、コンピュータだけでなく非コンピュータを含めた全てのオブジェクトを、ネットワークを介して個体識別する技術が実現されようとしている。RFID を用いることにより、ネットワークを介して、オブジェクトの位置情報などの状態が取得可能となる。本研究では、RFID タグを貼付することにより、個体識別可能な物体をオブジェクトと定義する。

オブジェクトの位置情報や属性を収集・蓄積・利用することで、流通において、個々のオブジェクトの位置情報を把握でき、在庫管理・貨物配送を効率化できる。しかし、現在のコンピュータは「自分のまわりに何があるのか、それはどのような状態にあるのか」など、周囲の物の存在や状態を「感知する」ことはできない。RFID とセンサを用いれば、コンピュータは自分の周囲の情報を収集・管理・処理できるようになる。非計算機に RFID タグを貼付することにより、端末が計算機であることを前提とするインターネット上で非計算機を扱えるようになる。

[1] それぞれ Suica、Edy、Speedpass の各 web サイトより

RFID をはじめとした自動認識技術を用いて CRM(Customer Relationship Management) や FA(Factory Automation)、SCM(Supply Chain Management) を管理する手法として、AIDC (Automatic Identification and Data Capture) がある。AIDC には、RFID タグ自体がオブジェクトの情報を保持するもの (データキャリア型 RFID) と、RFID タグは ID のみを保持し、オブジェクトの情報は ID と関連づけられ、ネットワーク上に保持するもの (メタデータ型 RFID) に分類できる。RFID の ID 情報の漏洩防止をする際、データキャリア型 RFID の場合は暗号化や認証機能付きの高機能な RFID タグを用いることにより実現できる。しかし、メタデータ型 RFID はネットワーク上にある情報のアクセス制御を伴うため、データキャリア型 RFID に比べ、RFID の ID の漏洩防止をすることは技術的に困難である。具体的には、タグの読み取りの認証だけでなく、ネットワーク上のサーバの認証や安全な通信路の確保など、考慮すべき点が多くなる。

メタデータ型 RFID の例として、MIT(マサチューセッツ工科大学) や慶應義塾大学が中心となって研究開発を行っている EPC Global [5]、Auto-ID Laboratory [6](旧 Auto-ID Center [7]) が注目を集めている。EPC Global では、RFID をバーコードに代わるオブジェクト認識技術として位置づけ、製品の製造・流通・在庫管理・販売・決済に至るまでを一貫して管理することを目指している。

EPC Global のアーキテクチャでは、全てのオブジェクトに固有の ID を割り振った RFID タグを取り付けて識別するため、製造から流通、小売に至るまでだけでなく、家庭内などさまざまな状況でオブジェクトを一元的に管理できる。EPC Global の提供するシステムにより、自分の買った商品が、「いつどこで作られ、どのような流通経路をたどってきたのか」などといったオブジェクトに関連する情報を容易に取得できる。また、私的な空間において、RFID 技術を用いて所有物を管理することにより、所有物の情報をいつどこにいても管理できる。例えば、自分の部屋に存在する本の情報を参照することにより、所有者は書店において「どの本を購入するか」の判断の指標にできる。

1.2 研究目的

本研究では、「RFID タグに保存されている ID」、および「ID に関連づけられた情報」をプライバシ情報と定義し、オブジェクトの所有者が家庭などで、SCM 後に自分のオブジェクトを管理することを想定し、プライバシの漏洩を防止することを目的とする。

現在 EPC Global で提案されているシステム (Auto-ID システムと呼ぶ) をはじめ、既存の RFID 技術の多くは、リーダを持っていれば誰でも RFID タグの情報を取得できてしまう。EPC Global で提唱されている ID は階層化された名前空間を持つため、RFID タグの ID が漏洩することにより、RFID タグの取り付けられたものが何であるのかを容易に推測できてしまう。さらに、ID を基に、オブジェクトの情報までもが第三者に容易に取得されてしまう可能性がある。

しかし、現在の Auto-ID システムでは SCM での利用を中心にシステムが設計されており、小売店以降 (家庭) での利用をあまり考慮していない。現実社会では、RFID が貼付されたオブジェクトには所有者が存在し、オブジェクトが譲渡される際に、その所有権は別の所有者に委譲される。既存のプライバシ保護の手法では所有権の概念がないため、所有権の委譲が発生した際に、以前の所有者からプライバシが漏洩する危険性がある。

この問題を解決するため、本研究では RFID に、オーナという所有権を表す属性を設定し、オーナの認証を行うことを提案する。本研究では、オーナの認証と ID の暗号化を用いたプラ

イバシ漏洩防止のシステムモデルを提案する。さらに、プロトタイプの設計・実装を行う。

1.3 本論文の構成

本論文は以下のように構成される。第 2章で、EPC Global とそのシステム、本研究で想定する利用環境について述べ、それを踏まえた上で本研究における要求事項を洗い出し、Auto-ID システムの問題点について述べる。さらに、RFID においてプライバシの漏洩を防止する既存技術について述べ、その問題点を整理する。第 3章では、問題点解決をする際に考えられるアプローチについて述べ、本研究で用いる問題点解決のためのアプローチを提示する。第 4章では、本研究で提案するシステムの機能用件を整理し、プロトタイプの設計を行う。第 5章でシステムのプロトタイプ具体的な実装について述べる。第 6章では、実装された SARU の評価を行い、他のシステムに対する優位性の検証および、性能評価を行うことにより本研究の有意性を検証する。第 7章において、まとめと今後の課題を挙げ、本論文の結論とする。

第2章　Auto-IDシステムの問題点と既存研究

本章では、RFIDシステムの代表例としてEPC Global、およびAuto-IDシステムとその問題点について述べ、本研究で想定する利用環境について述べる。それを踏まえた上で、本研究における要求事項を洗い出し、Auto-IDシステムを私的なオブジェクトの管理に利用する際の問題点について述べる。

2.1　Auto-IDシステム

本節ではEPC Globalの概要を説明し、Auto-IDシステムの概要について述べる。

2.1.1　EPC Globalの概要

EPC Globalは、最先端の自動認識技術の研究開発と、その利用および標準化を目指す国際的な非営利研究組織で、1999年にAuto-ID CenterとしてMITに設立された。バーコードの標準化団体であるUCC(Uniform Code Council) [8]および、EAN(European Article Number) International [9]が参加し、世界各国からさまざまな分野の研究者や技術者が集い、100社を超える企業が活動を支援している。2003年には、組織としての目的を研究開発から実用化に切り替え、EPC Globalに改組し、現在に至る。

EPC Globalは、次世代バーコードとしてRFIDを用い、世の中にあるすべての製品に固有のIDを持たせ、製品に関するさまざまな情報をインターネットを通じで取得し、活用できる世界の構築を目指している。Auto-ID Laboratoryでは、RFIDを用いた新しいAIDCシステムの研究開発と、それらを産業界で活用するための実証実験を行い、国際的なインフラの構築と標準化への提案を行っている。

2.1.2　Auto-IDシステム概要

本節では、現在Auto-IDシステムのシステムモデルについて述べる。

現在提案されているAuto-IDシステムの概要を図2.1に示す。Auto-IDシステムは以下のコンポーネントによって構成される。

- EPC(Electronic Product Code) [10]
- EPCタグ(RFIDタグ)、リーダ [11]
- PML(Physical Markup Language) [12]
- Savant [13]

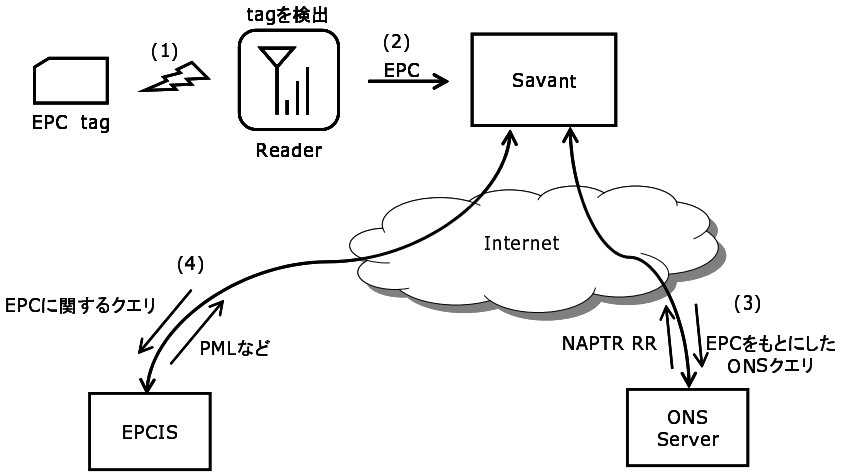

図 2.1: Auto-ID システムのアーキテクチャ概要図

- ONS(Object Name Service) Server [14]
- EPCIS(EPC Information Service) Server [15]

それぞれについて、以下に詳述する。

EPC

Auto-ID システムではオブジェクトの識別子として、EPC を用いる。EPC は EPC Global によって提案されている ID 体系で、世の中のすべてのオブジェクトに対し、固有の ID を提供する。EPC には、64bit・96bit・256bit の 3 種類のコード長が提案されており、3 種類とも 4 つの部分に分けられる。以下の図 2.2は 96bit EPC の構造を表している。先頭から順番に、Header(ヘッダ)、EPC Manager(ベンダコード)、Object Class(製品コード)、Serial Number(製造番号) となっている。

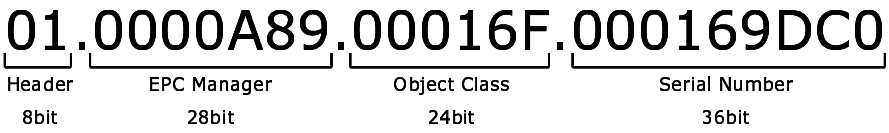

図 2.2: Auto-ID システムにおける EPC の構造

EPC タグ、リーダ

Auto-ID システムでは、全てのオブジェクトに EPC タグという RFID タグを取り付け、それをリーダを通じて自動的に検出する。この RFID は IC チップ部分とアンテナ部分から構成

され、リーダと無線を用いて通信する。EPC タグの中には ID として EPC が格納されている。

以下の図 2.3[2]は Alien Technology 社 [16] の EPC チップで、図 2.4[2]はその拡大図である。このチップの一辺は $0.17mm$ であるが、このように微小なチップを用いることで全てのオブジェクトに RFID タグを貼付することが可能となる。

図 2.3: Alien Technology 社の EPC チップ

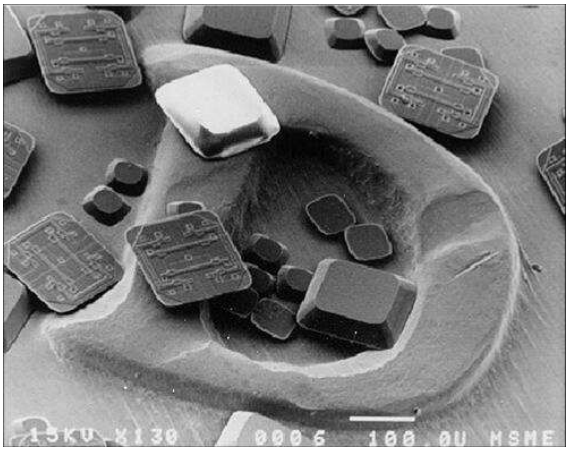

図 2.4: Nano Block の様子 (チップ一辺 $0.17mm$)

PML

Auto-ID システムでは、EPC はオブジェクトの ID のみを表現する。そこで、EPC と関連づけて、オブジェクトの属性を管理する機構が必要になる。オブジェクトの属性とは、製造年月日・流通経路・位置情報など、様々なサービスに必要になる情報である。このようなオブジェク

[2]Alien Technology 社資料より

トの属性を記述するための言語が PML である。PML は XML(eXtensible Markup Language) を基にして記述され、オブジェクトの名前、サイズ、状態などを記述する世界標準の言語となるように設計されている。PML を用いることで、オブジェクトの状態に応じた柔軟な属性の記述が可能となる。

Savant

Savant は、リーダで検出された EPC および、その EPC の PML をもとにしたサービスを提供する。Savant は複数のリーダからの EPC の情報を収集、蓄積、処理する。Savant は以下の主要な 3 つのコンポーネントにより構成される。

- EMS(Event Management System)
  様々なベンダーのリーダを扱う Java の API を提供する。共通フォーマットで EPC データの収集や、収集した EPC データのフィルタリングを行う。
- RIDS(Real-time In-memory Data Structure)
  データのやりとりの簡略化やデータベースの処理の高速化のため、EMS からのリクエストとデータベースの間を仲介をする。また、任意の時点におけるスナップショット保存も行う。
- TMS(Task Management System)
  タスク管理のためのインターフェースを提供する

ONS Server

Savant が、EPC をもとにオブジェクトの属性を取得する際に、その EPC の PML を保持する EPCIS サーバのアドレスを知る必要がある。この時に用いられるのが ONS Server で、Savant から EPC を受け取り、その EPC の PML 情報を保持する EPCIS の NAPTR(Naming Authority Pointer) [17] [18] [19] [20] [21] を Savant に返答する。

EPCIS Server

EPCIS Server は、EPC の属性を保持するコンテンツサーバである。Savant からの要求に対し、EPC に対応する PML を送信する。それらの属性情報は、SOAP(Simple Object Access Protocol) [22] や XML-RPC(XML Remote Procedure Call) [23]、SQL(Structured Query Language) などのプロトコルを用いて格納される。

2.2 Auto-ID システムの問題点

現在の Auto-ID システムには以下の 4 点が問題点として挙げられる。

第三者による EPC の読み取りによる個人情報の漏洩

現在の Auto-ID システムには、現実社会における所有権を実現する概念がなく、また使用される RFID にもアクセスコントロールの機能がないため、リーダを持つだけで誰で

もオブジェクトの ID を取得できるようになる。EPC は個々のオブジェクトを識別可能なため、第三者に EPC を読み取られると、そのオブジェクトの情報が漏洩する可能性は高い。例えば、図 2.5のように、鞄をリーダで読み取ると、鞄の ID だけでなく、鞄の中に入っている全てのオブジェクトの ID を一意に識別できる。その結果、鞄の中に入っている製品情報が、第三者に漏洩してしまう可能性がある。

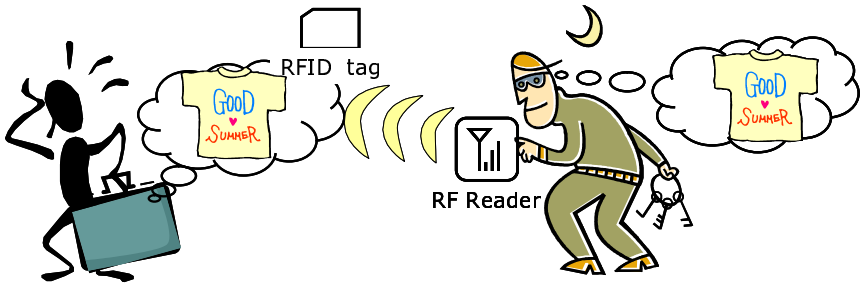

図 2.5: 鞄の中身の情報が第三者に漏洩

EPC を容易に推測可能

EPC は、EPC Manager、Object Class、Serial Number で構成されているため、特定のメーカの製品の EPC がわかると、その EPC をもとに他のオブジェクトの製品名やメーカ名を推測できる。

EPC をもとにオブジェクトの情報を取得可能

現在の Auto-ID システムでは、EPCIS サーバにユーザの認証機構について言及されていないため、取得した EPC を基に EPCIS サーバからオブジェクトの情報を取得できてしまう可能性がある。

ID の追跡が可能

現在の Auto-ID システムでは、EPC により全てのオブジェクトを一意に識別できるが、EPC は書き換わることがないため、EPC 自体を追跡することにより持ち主の特定が可能となる可能性がある (Long-term Tracking)。例えば、いつも電車の中で読み取れる EPC を他の場所でも読み取った場合、そのオブジェクトのオーナの行動が第三者に漏洩してしまうと共に、オーナを推定できてしまう可能性がある。

2.3 既存研究

本節では、RFID 技術においてプライバシの漏洩を防止する既存研究について述べ、それら既存研究の問題点について整理する。

RFID 技術においてプライバシの漏洩を防止する既存研究では、以下の 2 通りの方法に大別できる。

- RFID タグとリーダ間の認証
- EPC の隠蔽・暗号化・認証

これらについて、次節以降で詳説する。

2.3.1 RFID タグとリーダ間の認証

Hashed Lock Algorithm

Auto-ID Center が提案する技術に、RFID のセキュリティに関するものがある [24]。それが Hashed Lock Algorithm で、RFID においてセキュリティを守るための技術として、タグ自身が単一方向ハッシュ関数を用いて、リーダからの読み取りを制限するというものである。これは、ID の書き換えおよび、RFID タグがタグ自身をロック可能で、ハッシュ演算処理可能な RFID タグを用いるということを前提としている。

RFID タグをロックすることにより、認証されていないリーダから RFID タグの読み取りを制限するというものであり、ロックされた RFID タグは認証されたリーダにのみ ID を返答する。RFID タグがロックされていない状態では、誰でもその RFID タグの情報を読み取り可能である。

具体的には以下の手順で RFID タグがロックされる。

1. 任意の鍵 (パスフレーズ) を用意する
2. オーナは鍵のハッシュ値を計算し、ハッシュキーとして RFID タグに書き込む
3. RFID タグはハッシュキーが書き込まれると、自身がロックしてるという状態を保持する

RFID タグがロックされた状態になると、RFID タグはリーダからのいかなるコマンドに対しても、自信がロックしているという状態情報以外は返答しなくなる。これにより、RFID のセキュリティおよび、プライバシを確保する。

RFID タグのロックを解除するためには以下の処理が必要となる。

1. オーナはリーダから RFID タグに鍵を送信する
2. RFID タグは受信した鍵をハッシュする
3. ハッシュされた値と RFID タグ内に保存されているデータと比較し、合致した場合のみロックを解除する。

RFID タグのロックを解除する際に電力が足りなかった場合や、通信が途中で途絶えた場合には、RFID タグは自動的にロックされた状態に戻る。

2.3.2 EPC の隠蔽・暗号化・認証

大日本印刷と NTT、サン・マイクロシステムズの 3 社より、以下の 5 つの EPC の隠蔽・暗号化に関する技術が提案されている [25]。そのいずれもが、Auto-ID アーキテクチャを拡張し、隠蔽や暗号化、認証によりユーザのプライバシを保護することを目指している。

- Anonymous EPC [26]
- Reencrypt Anonymous EPC [26]
- Encrypted EPC

- E-signed EPC
- E-signed + Encrypted EPC

それぞれについて以下に詳述し、それぞれが抱える問題についてまとめる。

Anonymous EPC

Anonymous EPC は、EPC を取得されても EPC Manager や Object Class、Serial Number が判別できなくなるようにすることにより、ユーザのプライバシを保護する方法であり、Auto-ID Center アーキテクチャをそのまま使用できる。EPC タグに格納される EPC をランダムコードにすることにより、EPC からその EPC タグが取り付けられた製品が何であるのかといった情報を隠蔽することを目的としている。Anonymous EPC は、オリジナルの EPC の Object Class と Serial Number の代わりに、信頼された証明書会社によって提供されるランダムコードを使用する。その際、EPC Manager には証明書会社の ID がつけられる。

以下の図 2.6は、リーダが Anonymous EPC を読み取った際の処理のフローを表している。証明書会社のサーバには、Anonymous EPC とオリジナルの EPC の関係を管理するデータベースをもち、認証済みクライアントからの要求に対しては、オリジナルの EPC を返答する。しかし、認証されていないリーダに対しては、オリジナルの EPC を返答しないので、第三者からのオリジナルの EPC の取得を防ぐことができる。

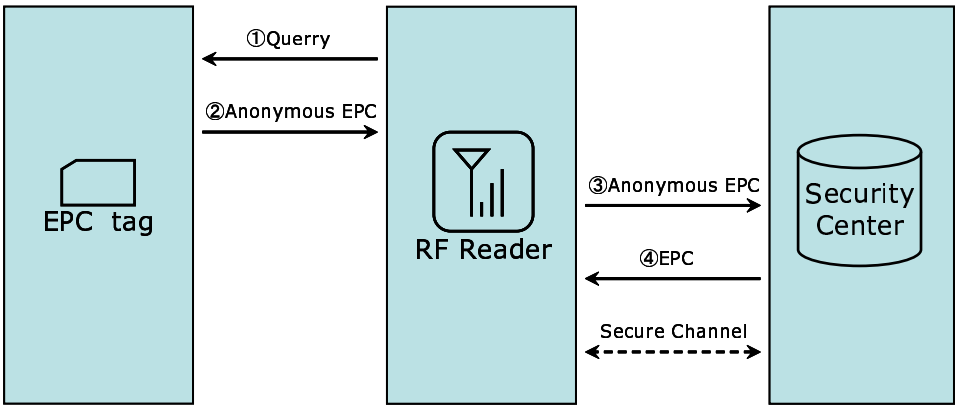

図 2.6: Anonymous EPC の処理フロー

以下の図 2.7は、オリジナルの EPC と Anonymous EPC の違いを表している。認証されていないクライアントからは製品コード以下の EPC は構造化されなくなっており、オブジェクトを特定することができない。

Reencrypt Anonymous EPC

しかし、EPC を隠蔽しても、隠蔽した EPC が恒久的に変化しなければ Long-term Tracking に対処できない。そこで、以下の図 2.8のように、Anonymous を拡張し、リーダで読み取るたびに別の Anonymous EPC に隠蔽し直す、というものも提案されている。これは認証された

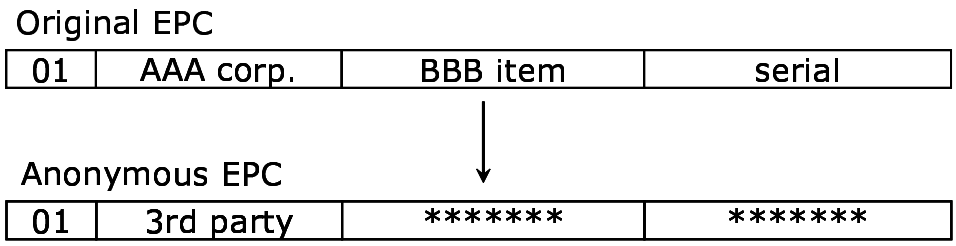

図 2.7: Anonymous EPC

リーダが Anonymous EPC を読み取る度に、証明書会社のサーバで新しく Anonymous EPC を生成し、EPC タグに書き込むというものである。

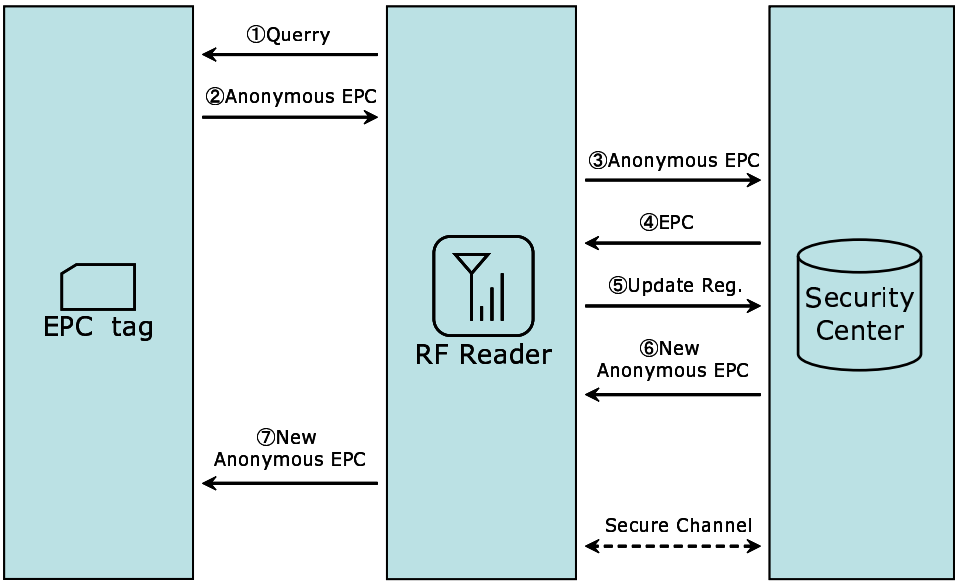

図 2.8: Long-term Tracking に対処した Anonymous EPC の処理フロー

Encrypted EPC

Encrypted EPC は、EPC を暗号化することによりユーザのプライバシを保護することを目的とする。Encrypted EPC では、オリジナルの EPC は暗号化され、EPC Manager に信頼された証明書会社の ID がつけられる。Encrypted EPC は、Proxy EPC とオリジナルの EPC を暗号化したデータから構成され、Proxy EPC の中には暗号化のアルゴリズムと暗号化の鍵が埋め込まれている。Proxy EPC により、その EPC が Encrypted EPC であることが判別可能となるが、Encrypted EPC の構造自体はオリジナルの EPC と変わらない。そのため、Proxy EPC は Auto-ID システムに変更を加えることなく暗号化を実現するために必要なものである。

認証済みのリーダから要求があると、証明書会社の EPCIS サーバは暗号化された EPC をオ

リジナルの EPC に復号化する。一見 Anonymous EPC と同じような方法であるが、Encrypted EPC では暗号化のアルゴリズムと暗号化の鍵を Proxy EPC に内包するため、EPCIS サーバに翻訳用のデータベースを持つ必要がない。

以下の図 2.9は Encrypted EPC を利用した場合の、EPC の構造を示している。EPC の製品コード以下は暗号化され、第三者からはモノを特定することができないようになっている。

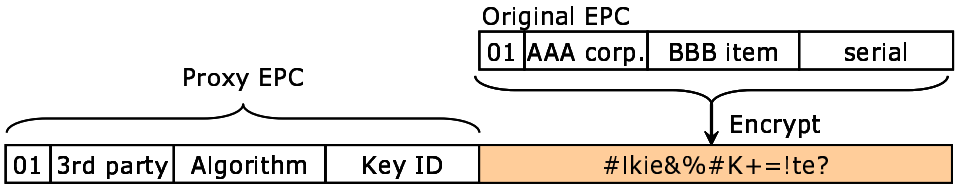

図 2.9: Encrypted EPC

E-signed EPC

EPC の構造上 (図 2.2)、あるベンダのある商品の EPC を元に、同一商品の EPC を容易に推測できてしまう。そのため、EPC タグに ID を書き込める RF ライタさえあれば、容易に EPC の詐称ができてしまう。E-signed EPC は、Auto-ID システムに変更を加えることなく、詐称した ID を検出するための技術である。

E-signed EPC では、オリジナルの EPC は電子署名され、そのシグネチャは信頼された証明書会社によって E-signed EPC に埋め込まれる。これは、公開暗号鍵を持っていれば誰でも、電子署名の確認により EPC の確実性をチェックすることができる。利用の際には、電子署名を簡単にコピーできることに注意しなければならない。

以下の図 2.10は E-signed EPC を利用した場合の、EPC の構造を示している。Encrypted EPC と同様に、Proxy EPC を用いることにより、その EPC が E-signed EPC であることが判別可能となる。Proxy EPC を用いることで、E-signed EPC の構造自体はオリジナルの EPC と変わないため、Proxy EPC は Auto-ID システムに変更を加えることなく暗号化を実現するために必要なものである。

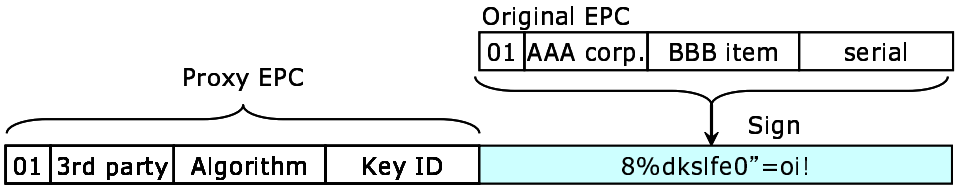

図 2.10: E-signed EPC

E-signed Encrypted EPC

これは Encrypted EPC と E-signed EPC の 2 つを同時に利用する方法で、EPC の暗号化と確実性を保証することを目的としている。オリジナルの EPC を暗号化した後に、暗号化されたデータを電子署名し、暗号化されたデータの確実性を保証する。

以下の図 2.11は E-signed Encrypted EPC の EPC の構造を示している。

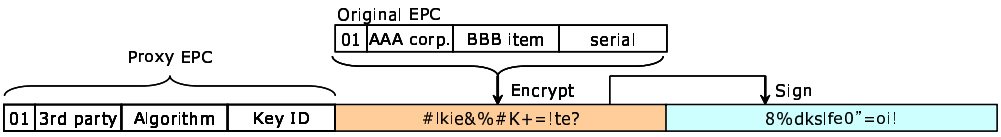

図 2.11: Encrypted E-signed EPC

2.3.3 既存研究の考察

本項では、これまで述べた既存研究について、機能要件毎にまとめる。また、それぞれの研究が各要件を実現できているか、検証を行う。以下の表 6.1に、各研究手法と、その手法と機能要件の関係を示す。

表中で用いられる記号は、以下の意味に示す通りである。○は、その研究を利用することにより実現することが可能である機能である。△は、その研究で想定されている機能ではあるが、不完全、もしくはどのようにして実現するのか不明瞭な機能である。×は実現不可能な機能である。

表 2.1: 既存研究の考察

	隠蔽・暗号化	リーダの認証	所有権の設定	所有権の委譲	Long-term Tracking 対策
Hashed Lock Algorithm	×	○	×	×	○
Anonymous EPC	○	△	×	×	×
Reencrypt Anonymous EPC	○	△	×	×	○
Encrypted EPC	○	△	×	×	×
E-signed EPC	○	△	×	×	×
E-signed + Encrypted EPC	○	△	×	×	×

表 6.1からわかるように、既存の研究では ID の暗号化や ID 読み取りの際の認証はされているものの、所有権の概念や所有権の委譲まで実現しているものはない。所有権のまた、ID の隠蔽や暗号化をしているものでも、Long-term Tracking に対処しているものは少ない。

次に、個別に問題点について検証する。RFID タグとリーダ間の認証を用いた手法の問題点して以下の 3 点が挙げられる。

- リーダが鍵を管理するため、オブジェクトの所有者が変わっても、同じリーダを用いることにより、第三者に RFID タグの情報を取得される恐れがある。
- 無線通信を用いて、RFID タグとリーダが認証を行うため、認証されたリーダと RFID タグとの通信を傍受することにより、第三者の持つ別のリーダに RFID タグの情報を取得される恐れがある。
- RFID タグに演算機能や書き換え機能などを必要とし、RFID タグが比較的高価なものになるため、全てのオブジェクトに RFID タグを貼付することがコスト的に困難になる可能性がある。

また、Anonymous EPC や Encrypted EPC をはじめとした EPC の隠蔽・暗号化・認証に関する 5 つの手法は、いずれも隠蔽・暗号化・認証手法のみの提案であり、これらを実現するためのアーキテクチャは提案されていない。また、Anonymous EPC を以外の手法では EPC のコード長が長くなってしまうため、EPC タグの仕様変更などを必要とする可能性がある。さらに、リーダの認証をどのようにして実現するのか不明瞭である。

以上の問題点を踏まえ、本研究では所有権の委譲を含めた所有権の管理が実現可能な RFID のプライバシ保護システムを提案する。所有権の管理についてや、想定する利用環境については次章以降で詳しく説明する。

第3章　利用モデル

本章では、本研究における利用モデルを定義し、第 2章で挙げた問題点解決のために考えられるいくつかのシステムモデルを挙げ、考察する。

3.1　本研究で想定する利用モデル

本研究では、全てのオブジェクトに工場出荷の状態で EPC タグが取り付けられていること (Source Tagging) を前提とする。そして商店や家庭において、リーダを用いて RFID タグが貼付されているオブジェクトを検出し、オブジェクトの管理に EPC を用いることを想定する。このようなオブジェクトの管理モデルの例が図 3.1で示されている。

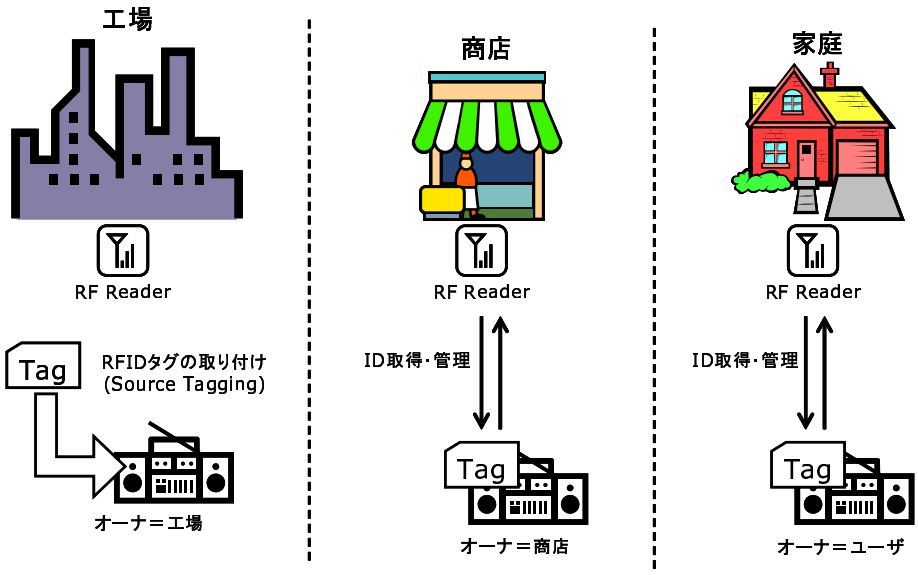

図 3.1: 本研究で想定するオブジェクトの管理モデルの例

商店、家庭において、リーダを用いて EPC から個体 ID を取得可能であり、在庫管理やオブジェクトの管理に EPC を用いている。具体的には以下のようなアプリケーションを想定できる。

捜し物検索

家の中などでオブジェクトを探しているときに、リーダで検出される範囲に探しているオブジェクトがあれば、リーダの位置を元にそのオブジェクトがどこにあるのかがわかる。

物品管理

冷蔵庫や書架、物置などにリーダを設置することにより、扉を開けずにその中に何が入っているのかという情報および、入っているものの情報を取得できる。

ショッピングアシスタント

食料品を買うときに、冷蔵庫にリーダを設置しておくことにより、買い物先 (商店) などで、冷蔵庫の中になにが入っているのかという情報や、中に入っているものの賞味期限などを調べることにより、買い物を手助けする。このシステムは同様に、洋服や書籍にも適用できる。

そして、本研究で想定する利用モデルは、図 3.2で示されているように、図 3.1で示されているような管理モデルが複数存在するモデルである。複数のオブジェクト管理モデル間を、オブジェクトが移動することを想定している。この移動の際には、所有者が変わるため、オブジェクトの所有権の委譲が発生する。この時、オブジェクトの所有権の委譲後に、以前のオーナがオブジェクトの情報を取得できないような利用モデルを想定している。

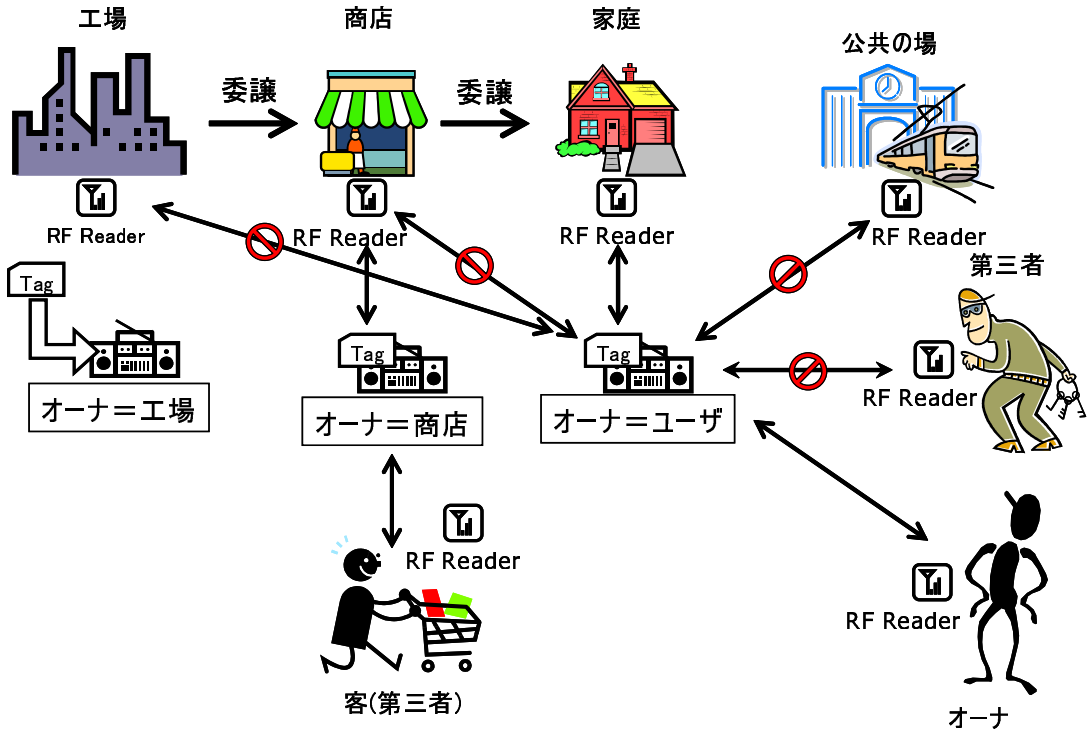

図 3.2: 本研究における利用モデル

以上を踏まえ、購入後のオブジェクトの管理には家庭内のリーダだけでなく、例えば駅や店頭など、公共の場にあるリーダの利用も想定する。これにより、ユーザが店に行くと、商店は客 (ユーザ) がどのようなものをその店で購入したのかという情報が取得でき、商品の購入に際し適切なアドバイスをすることが可能となる。また、アフターサポートなどのシチュエーションにおいても、その商品を本当にその商店で購入したかどうかの判断が容易になる。

3.2 要求事項

前節の利用モデルを踏まえ、以下の視点から Auto-ID システムを家庭で利用する際のオブジェクトのプライバシ漏洩防止システムの要求事項を洗い出す。

所有権を設定

オブジェクトにはオーナを設定する。オーナはリーダを用いて ID を取得できるが、オーナ以外の第三者は、オブジェクトの ID を取得できてはならない。オブジェクトが店頭に並んでいるときや、商店の倉庫にあるときは、商店がオブジェクトの管理のためにオブジェクトの ID を取得できる必要がある。しかし、客が商店でオブジェクトを購入すると、オブジェクトのオーナが商店側から客側に移る。一度オブジェクトが客のものになれば、そのオブジェクトの ID が商店側に取得できる必要はなく、オブジェクトの ID は客だけが取得できれば良い。

所有権の委譲

実世界において、オブジェクトのオーナは移り変わる。オブジェクトの所有権が変わった際に、所有権の委譲をネットワーク上の情報にも反映し、以前のオーナがオブジェクトの情報を取得できないようにする必要がある。

Long-term Tracking 対策

長期間の ID 追跡により、オブジェクトのオーナをある程度推測可能になる。また ID とオーナが関連付けられている状態で、ID の検出を行うことにより、オブジェクトのオーナの行動範囲が第三者に取得されてしまう。そのため、定期的もしくは、任意の時にオーナが ID を書き換えられる必要がある。

3.3 考えられるシステムモデル

EPC の読み取りによる第三者へのプライバシの漏洩を防止する方法として、以下の 2 通りに大別できる。

- EPC の取得を防止する方法
- EPC からアプリケーションに至る間で情報の取得を防止する方法

それぞれについて、以下に詳述する。

3.3.1 EPC の取得を防止する方法

EPC の取得を制御することによりプライバシを保護する方法で、EPC の暗号化や隠蔽、EPC タグ自体に認証機能を持たせることが考えられる。

EPC の暗号化・隠蔽

EPC の暗号化・隠蔽とは、EPC を暗号化・隠蔽し、認証されたオーナ以外は復号化・復元できなくする方法である。暗号化・隠蔽された EPC は、ONS サーバに問い合わせてもエントリが存在しないため、EPCIS サーバの IP アドレスを取得できず、オブジェクトの情報を取得することができない。認証されたオーナのみ、復号化・復元された EPC を取得でき、オブジェクトの情報を取得できる。これにより、第三者の EPC の読み取りによるプライバシの漏洩を防止できる。

EPC タグによる認証

EPC タグによる認証というのは、EPC タグとリーダの間で認証を EPC タグが行う方法である。前章の関連研究で述べた Hashed Lock Algorithm などがこれにあたる。この EPC タグから ID を読み出すためには、正当な鍵 (パスフレーズ) を EPC タグに入力する必要がある。この方法では、EPC タグが自分自身で、鍵を持たない第三者からの不正な ID の読み取りを防止する。

3.3.2 EPC からアプリケーションに至る間で情報の取得を防止する方法

EPC タグに保存される ID にはそのままに、リーダが EPC を読み取った後に、プライバシに関わる情報のアクセス制御をすることによりプライバシを保護する方法である。この方法では、以下に挙げるような方法に細分できるが、そのいずれも EPC の構造は変わらないため、EPC からオブジェクトの EPC Manager、や Object Class を判別・推測できてしまう恐れがある。

EPCIS でのアクセス制御

EPCIS サーバでのアクセス制御とは、EPCIS サーバが Savant に対してオブジェクトの情報を提供する際にユーザの認証を行い、オーナ以外にはオブジェクトの情報を提供しない、というものである。Auto-ID システムではオブジェクトの情報は全て EPCIS サーバで管理されるため、オブジェクト情報の漏洩を防止したい場合には有効である。

Savant でのアクセス制御

Savant でのアクセス制御とは、Savant がリーダから EPC を受け取った際に、その EPC のオブジェクト情報の取得権限の認証を行い、認証されていないリーダに対してはオブジェクトの情報を提供しない、というものである。Auto-ID システムではリーダが読み取った EPC は必ず Savant に受け渡されるので、オブジェクト情報の漏洩を防止したい場合には有効である。また、Savant はオブジェクトの情報をアプリケーションに渡す機能を持っているので、アプリケーションごとに提供する情報を変えたいときには有効である。

3.3.3 考えられるシステムモデルの考察

以上をまとめると、EPC からアプリケーションに至る間で情報の取得を防止する方法では、EPC Manager、Object Class、Serial Number といったリーダで読み取った際の EPC Manager の構成は変わらないため、読み取った EPC Manager からそのオブジェクトが何であるのかという情報の漏洩を防止できない。すなわち、EPCIS サーバおよび、Savant でのアクセス制御のみでは、基本的な EPC の構造が変わらないため、本研究で定義するプライバシを守る上では機能が不十分である。また、Hashed Lock Algorithm では所有権の委譲に対応できないことや、EPC タグとリーダの通信電波を盗聴することによりプライバシが漏洩する危険性がある。

第4章　SARUの設計

本章では、前章で述べた機能要件を基に、本研究での問題点解決のアプローチについて述べる。さらに、本研究で提案するSARUのシステムの設計について述べ、本システムを構成する要素について述べた後、それぞれの要素内で処理される情報、および動作概要について述べる。

4.1　本研究で提案するシステム

本節では今研究で提案するシステム、およびその機能要件について述べる。

4.1.1　システム概要

前節までの内容を踏まえ、本研究では、第三者へのプライバシ漏洩問題解決へのアプローチとして、**SARU(Secure Authenticated RFID strUcture)**を提案する。SARUではEPCを暗号化し、EPCタグにそれぞれオーナを設定することを提案する。さらに、全ての暗号化や認証の機能を管理するために、**EPC Authentication and Decryption Server(EPCADS)**という認証・復号化サーバを用意する。EPCADSではオーナの認証、EPCの復号化、オーナの管理、鍵の管理を行う。EPCADSでオーナ認証されたオーナのみが復号化されたEPCを取得できる。また、リーダではEPCADSに対応させるため、**EPC Authentication Client(EPCAC)**というクライアントプログラムを用意する。図4.1は、以上のアプローチ概要を表したものである。

これにより、図4.2のように、悪意を持った第三者が鞄をリーダで読み取っても、EPCが暗号化されているため、鞄の中に何が入っているのかわからない。しかし、オーナであれば、公共の場所にあるリーダなど、一時的にRFIDを読み取る必要がある際にも、リーダに鍵を入力することによりオーナは自分の持ち物のは認識できる

4.1.2　機能要件

3.2章で挙げた要求事項、および2.2章で述べたAuto-IDシステムの問題点をふまえ、以下の機能が機能要件として挙げられる。それぞれについて以下に詳述する。

EPCの暗号化・隠蔽

第三者からのEPCの読み取りによるプライバシの漏洩を防止するため、EPCを暗号化する。オブジェクトに貼付されたEPCは工場出荷時には暗号化されておらず、オーナが任意の時点で自分の秘密鍵を用いて暗号化できるようにする。2.3.3章で述べた様に、既存のRFIDタグの暗号化に関する研究は、IDの隠蔽・暗号化手法のみの提案であり、これらを実現するためのリー

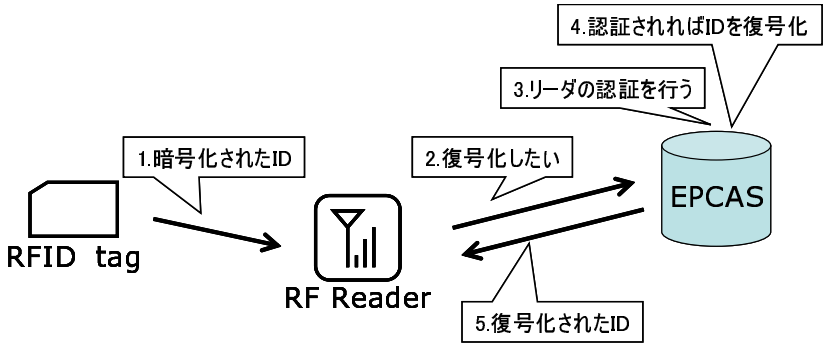

図 4.1: 本研究で提案するシステムモデル概要図

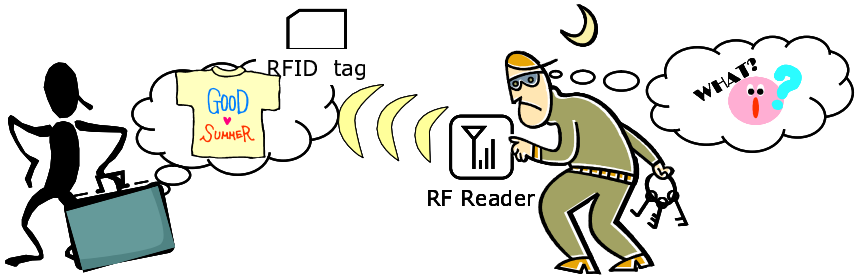

図 4.2: 暗号化されいているため鞄の中身は判別不能

ダの認証などのアーキテクチャは提案されていない。本研究で提案するシステムを利用することにより、これらの隠蔽、暗号化手法が実現可能となる。

安全な通信

復号化後の EPC を傍受されることを防止するため、SSL や IPsec などを用いて安全な通信路を確保する。

鍵管理

リーダで鍵の管理を行った場合、オブジェクトのオーナが更新しても、リーダの中に鍵が残存する。そのため、オーナ変更後に第三者が同じリーダを用いることにより、第三者に EPC の情報を取得されてしまう可能性がある。これを防止するため、認証を行うサーバ上で鍵を管理する必要がある。

また、Encrypted EPC を使用する際、秘密鍵をネットワーク上に流すことは安全性の点から望ましくない。そのため、EPCADS 上で復号化を行うことが必要である。

オーナ管理・認証

オブジェクトは商品として市場に流通するため、そのオーナは変遷する。オーナが変わる際に、以前のオーナがそのオブジェクトの情報にアクセスすることを制限する必要がある。そのため、オブジェクトに、現在のオーナを示す付加属性を設定する。オーナ属性を 1 つだけに限定し、オーナの認証を行うことにより、オーナ以外の人からのアクセスを制限する。

オブジェクトのオーナが複数に存在する場合は、オーナ属性に特定の個人ではなくグループを指定することにより、対応可能である。また、プライバシを保護する必要のないオブジェクトや情報を公開したいオブジェクトは、オーナを設定しないことにより、第三者からのアクセスを可能とする。

Long-term Tracking 対策

EPC の暗号化や隠蔽を行っても、暗号化・隠蔽後の ID が永久不変であった場合、Long-term Tracking が可能となる。そのため、任意の時点 (もしくは ID 読み取り毎に) 再び暗号化・隠蔽し直せる必要がある。

4.1.3 暗号方式

暗号には、データの暗号化と複号化に同じ鍵を使う共通鍵暗号方式と、別々の鍵を使う公開鍵暗号方式とがある。共通鍵暗号方式は、アルゴリズムが単純なため暗号化・復号化の処理が速いという利点がある一方、暗号化データを相手に送る際、別途安全な方法で鍵を相手に渡さなくてはならないという問題がある。

これに対し、公開鍵暗号方式では、本人のみが持つ秘密鍵と一般に公開して使う公開鍵の 2 つの鍵を使う。公開鍵で暗号化されたデータは対となる秘密鍵のみで復号化が可能なため、送信先の公開鍵で暗号化すれば、送信先のみが復号化できることになる。また、送信元の秘密鍵による電子署名が付けられていれば、受取側は送信元の署名の検証を行ない、検証されれば間違いなく本人が送信してきたと証明できる。この機能をオーナの認証に用いることが可能であると考える。公開鍵暗号方式を用いることで、オーナの認証と ID の暗号化に、同じ暗号化の鍵を用いることが可能となるため、本研究では、公開鍵暗号方式を用いて EPC を暗号化する。ただし、本研究では 2 つの鍵を公開鍵・秘密鍵としては使用せず、鍵対として、EPCAC と EPCADS の双方で 1 つずつ管理する。双方の鍵の区別をつけるため、便宜上 EPCAC で管理する鍵を「鍵 A」、EPCADS で管理する鍵を「鍵 B」とする。

4.2 SARU のシステム構成

本システムは以下の 5 つから構成される。

- EPCADS
- EPCAC
- Owner Management Client
- Key Registry Client

- EPC の暗号化

図 4.3に、本研究で提案するシステムアーキテクチャ概要図を示す。図中に示されている各構

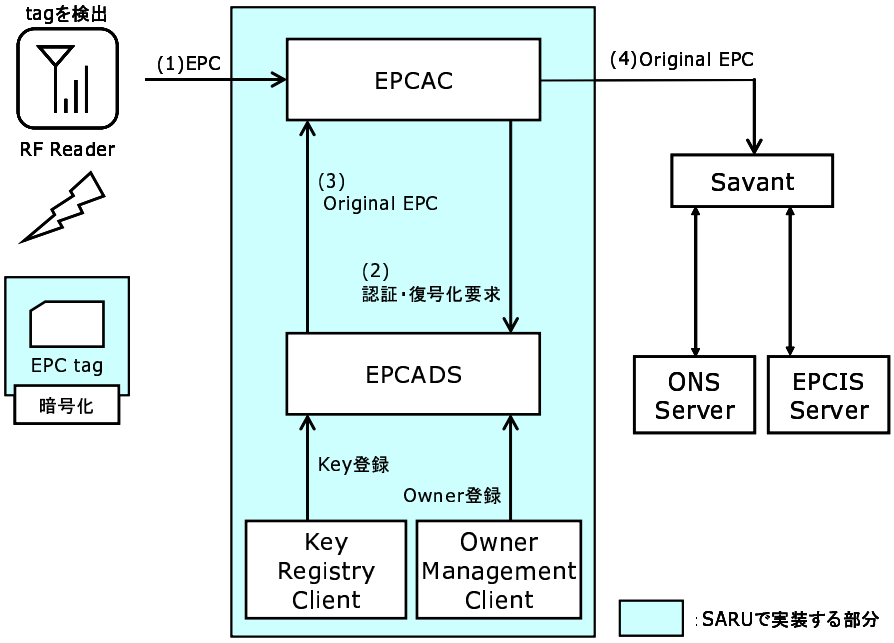

図 4.3: SARU のシステムアーキテクチャ概要図

成要素について以下にまとめ、次節以降で各々の設計について詳述する。

EPCADS

EPCADS は Savant から要求があった際に、オーナ認証と EPC の復号化を行う。オーナの認証がされると、EPCADS は EPC の復号化を行い、暗号化された伝送路を用いて、復号化された EPC を Savant に送信する。EPCADS では以下の情報を扱う。

- 暗号化された EPC
- EPC のオーナを示す Owner ID
- 暗号化・復号化・オーナ認証に用いる鍵 B・鍵 A
- オーナ認証に使用するチャレンジワード
- 復号化された EPC

EPCAC

既存の Savant は、暗号化された EPC を想定していない。本研究では、暗号化された EPC を取得した際に、EPCADS に対してオーナ認証、および EPC の復号化を要求する機能を Savant

に対して追加し、EPCAC として実現する。また、Savant に接続するリーダは事前に認識できるため、Savant に使用者を特定する情報 (Owner ID) を入力することにより、ユーザの認証に用いる。EPCAC では以下の情報を扱う。

- 暗号化された EPC
- EPC のオーナを示す Owner ID
- 暗号化・復号化・オーナ認証に用いる鍵 A
- オーナ認証に使用するチャレンジワード
- 復号化された EPC

Owner Management Client

Owner Management Client はオーナの変更が生じた際に、オーナが EPC のオーナ情報を登録・更新するためのものである。第三者によるオーナ情報改竄を防止するため、オーナ情報の更新の際にはオーナ認証を行う。Owner Management Client では以下の情報を扱う。

- 暗号化された EPC
- EPC のオーナを示す Owner ID

Key Registry Client

Key Registry Client は ID を暗号化する際に、オーナが鍵 B を登録するためのものである。オーナが鍵を変更し ID を暗号化しなおす際や、オーナの変更が生じた際にも、オーナによる Key Registry Client による鍵の登録が必要となる。Key Registry Client では以下の情報を扱う。

- 暗号化された EPC
- 暗号化・復号化・オーナ認証に用いる鍵 B

EPC の暗号化

本システムでは、オーナが任意の時点で EPC を暗号化できる。先述の理由から公開鍵暗号方式を用いて暗号化され、暗号化に用いた鍵 B は Key Registry Client によって、EPCADS に登録される。

4.3 Auto ID システムとの差分

本研究では、全ての EPC を隠蔽または暗号化し、Auto ID システムに、認証サーバとして EPCADS を追加する。また、EPCADS に対応するように Savant に変更を加える必要がある。さらに、EPCADS に対し、オーナの登録、更新を行う Owner Management Cleint と、鍵 B を登録する Key Registry Client を追加する。ONS Server、EPCIS Server、リーダには全く変更を加えることなく、本研究で提案するモデルを実現できる。

4.4 EPCADS

本節では EPCADS の機能要件を整理し、システムアーキテクチャを設計する。

4.4.1 EPCADS の機能要件

EPCADS の機能要件として、以下の 4 点が挙げられる。

- **オーナ認証**
  復号化を行う前に、復号化要求をしてきたオーナが、復号化しようとしている EPC のオーナであるのかどうかを認証する必要がある。本研究においては、暗号化に使った鍵対を用いて EPCAC を認証することで、オーナの認証とする。

- **EPC 復号化**
  EPCADS の持つデータベース上にある鍵 B を用いて、EPCAC から送信された EPC を復号化する。

- **オーナ登録・更新**
  復号化の前に行うオーナ認証に用いられる EPC タグのオーナ属性を、EPCADS 上のデータベースに登録・更新する。

- **鍵登録・更新**
  復号化、およびオーナ認証に用いられる鍵 B を、EPCADS 上のデータベースに登録・更新する。

以上の機能要件を踏まえ、次項で EPCAC の設計について述べる。

4.4.2 EPCADS の動作概要

EPCADS 上のモジュールの動作について、処理の流れを詳述する。図 4.4は、本研究で提案する EPCADS のモジュール相関図である。

オーナ認証・EPC 復号化

オーナ認証モジュールと EPC 復号化モジュールは、Savant から EPCADS へのオーナ認証・EPC 復号化要求を処理する。オーナ認証モジュールと EPC 復号化モジュールの処理の流れを以下に示す。

(1) オーナ認証モジュールから EPCAC に対し、オーナの認証に用いる「チャレンジワード」を送信する。このチャレンジワードは毎回異なるものを用いる。

(2) オーナ認証モジュールが EPCAC からの認証要求を受け取る。この認証要求は、「チャレンジワードを鍵 A で暗号化したもの」と、「Owner ID」、「暗号化された EPC」の 3 つから構成される。

(3) 認証要求を受け取ったオーナ認証モジュールは、データベースに対して「EPC のオーナ (Owner ID)」と「鍵 B」を要求する。この要求には「暗号化された EPC」を用いる。

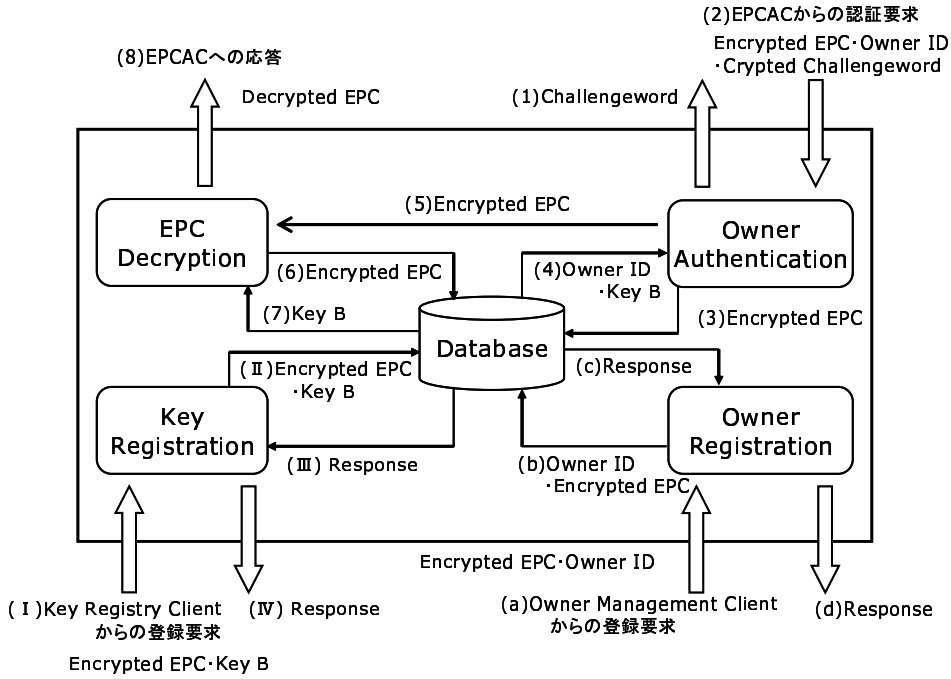

図 4.4: EPCADS 上のモジュール相関図

(4) データベースから「Owner ID」と「鍵 B」を受け取る。「データベースから受け取った Owner ID」が、「EPCAC から送信された Owner ID」と一致した場合、(1) で受け取った「鍵 A で暗号化されたチャレンジワード」を、鍵 B で復号化する。

(5) チャレンジワードが正しく復号化された場合、オーナの認証が成立し、EPC 復号化モジュールに処理が移る。

(6) EPC 復号化モジュールは、Savant から受け取った EPC の鍵 B をデータベースに要求する。

(7) データベースから受けとった鍵 B で EPC を復号化する。

(8) 復号化した EPC を EPCAC に送信する。

オーナ登録・更新

EPC のオーナ情報に変更が生じた際、Owner Management Client は EPCADS に、登録・更新を要求する。オーナ登録モジュールの処理の流れを以下に示す。

(a) オーナ登録モジュールが、Owner Management Client からの登録・更新要求をオーナ登録モジュールが受け取る。この登録・更新要求は、「Owner ID」と「暗号化された EPC」から構成される。

(b) 受け取った Owner ID と EPC を、オーナ情報管理データベースに登録・更新する。オーナ情報新規登録の際 (Source Tagging 時) には Owner の EPC リストに新規エントリーを

追加する。オーナ情報更新の際には、オーナ認証モジュールと同様に、要求をしている人がその EPC のオーナであるかの認証を行う。認証されれば、現在の EPC の Owner ID を、次のオーナの Owner ID に更新する要求をデータベースに送信する。

(c) 処理結果をオーナ登録モジュールが受け取る。データベースで正常に処理された時点で、所有権の委譲が完了する。

(d) 正常に処理が行われたかどうかの結果を Owner Management Client に返答する。

鍵登録・更新

EPC Manager が、工場出荷時にオブジェクトに EPC タグをつける際には ID は暗号化されていない。オーナが商品を購入後、暗号化した際に EPCADS に鍵 B を登録する必要があるが、その際に鍵登録モジュールを使用する。

また、オーナが鍵対を変更し暗号化する際や、オーナの変更時に新しいオーナが暗号化する際にも鍵 B の登録が必要となり、同様に鍵登録モジュールを使用する。鍵登録モジュールの処理の流れを以下に示す。

(I) 鍵登録モジュールが、Key Registry Client からの鍵の登録・更新要求を受け取る。この鍵登録要求は、「暗号化された EPC」(暗号化される前であればオリジナルの EPC) と「鍵 B」から構成される。

(II) 受け取った EPC と鍵 B を、データベースに登録・更新する要求をデータベースに送信する。鍵の登録・更新の際には、オーナ認証モジュールと同様に、要求をしている人がその EPC のオーナであるかの認証を行う。認証されれば、鍵の登録更新要求がデータベースに送信される。

(III) 処理結果を鍵登録モジュールが受け取るデータベースで正常に処理された時点で、鍵の登録・更新が完了する。

(IV) 正常に処理が行われたかどうかの結果を Key Registry Client に返答する

4.5 EPCAC

本節では EPCAC の機能要件を整理し、システムアーキテクチャを設計し、動作概要について述べる。

4.5.1 EPCAC の機能要件

EPCAC の機能要件として、以下の 2 つが挙げられる。

- **リーダ制御**
  リーダを制御し、EPC タグから EPC を読み出す。常にリーダを監視し、リーダの検出範囲に EPC タグが現れるまで待機する。

- **認証・復号化要求**
  「EPCADS から送られてきたチャレンジワードを鍵 A で暗号化したもの」と、「Owner ID」から EPCADS に対し認証復号化要求を行う。

以上の機能要件を踏まえ、次項で EPCAC の設計について述べる。

4.5.2 EPCAC の動作概要

リーダから暗号化された EPC を受け取り、Owner ID と合わせて EPCADS に認証・復号化要求を行い、復号化された EPC を受け取るのが EPCAC の目的である。復号化された EPC を受け取った EPCAC は、その EPC を用いて通常の Savant の処理を行う。図 5.3は、本研究で提案する EPCAC のモジュール相関図である。それぞれのモジュールの動作について、処理の流れを詳述する。

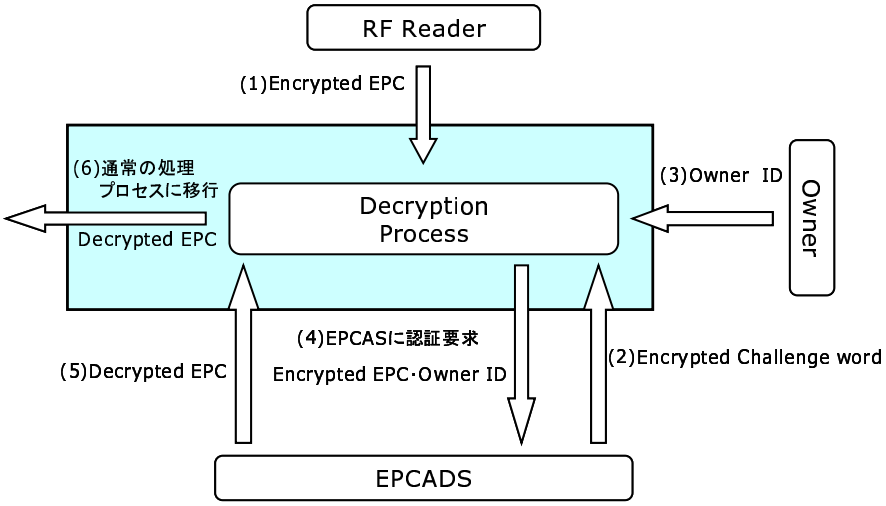

図 4.5: EPCAC 上のモジュール相関図

(1) リーダから暗号化された EPC を受け取る。

(2) EPCADS からチャレンジワードを受け取る。

(3) オーナが入力した Owner ID を受け取る。

(4) 受け取った Owner ID と暗号化された EPC と、鍵 A で暗号化したチャレンジワードを EPCADS に送信し、認証・復号化要求を行う。

(5) 復号化された EPC を受け取る。

(6) 受け取った EPC を用いて通常の処理に移行する (通常の Savant に送信する)。

4.6 Owner Management Client

本節では Owner Management Client の機能要件を整理し、システムアーキテクチャを設計する。

4.6.1 Owner Management Client の機能要件

Owner Management Cleint の機能要件として挙げられるのは、オーナ情報を EPCADS 上のデータベースに登録・更新することである。このオーナ情報はオーナの認証に用いられ、暗号化された EPC と対応する Owner ID のことを指す。

4.6.2 Owner Management Client の動作概要

Owner Management Cleint はオーナから「暗号化された EPC」と「Owner ID」を受け取り、EPCADS に対して、データベースへの登録を要求する。図 4.6は、本研究で提案する Owner Management Client のモジュール相関図である。それぞれのモジュールの動作について、処理の流れを詳述する。

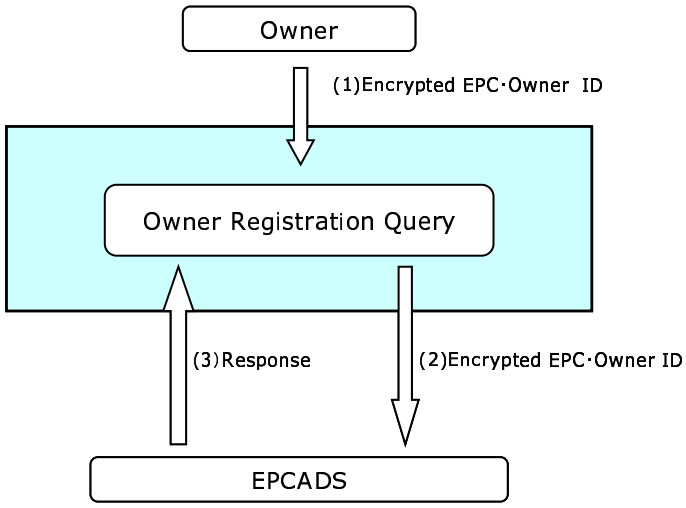

図 4.6: Owner Management Client 上のモジュール相関図

(1) オーナが入力した「Owner ID」と「暗号化された EPC」を受け取る。

(2) 受け取った「Owner ID」と「暗号化された EPC」を、登録・更新要求として EPCADS に送信する。

(3) 処理の結果が EPCADS から受け取る。

4.7 Key Registry Client

本節では Key Registry Client の機能要件を整理し、システムアーキテクチャを設計する。

4.7.1 Key Registry Client の機能要件

Key Registry Client の機能要件として挙げられるのは、鍵情報を EPCADS 上のデータベースに登録・更新することである。このオーナ情報は EPC の復号化とオーナの認証に用いられ、暗号化された EPC と対応する鍵 B のことを指す。

4.7.2 Key Registry Client の動作概要

Key Registry Client はオーナから「暗号化された EPC」と「鍵 B」を受け取り、EPCADS に対して、データベースへの登録を要求する。図 4.7は、本研究で提案する Key Registry Client のモジュール相関図である。それぞれのモジュールの動作について、処理の流れを詳述する。

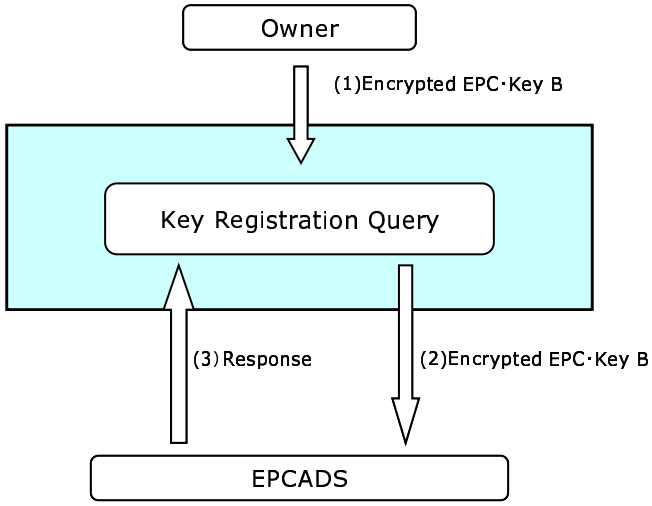

図 4.7: Key Registry Client 上のモジュール相関図

(1) オーナが入力した「鍵 B」と「暗号化された EPC」を受け取る。

(2) 受け取った「鍵 B」と「暗号化された EPC」を、登録・更新要求として EPCADS に送信する。

(3) 処理の結果が EPCADS から受け取る。

第5章　SARUの実装

本章では、プロトタイプとして実装したSARUの具体的な実装について述べる。

5.1　実装環境

本節では、実装したサーバの環境を以下の表5.1に、クライアントの環境を表5.2に示す。

表 5.1: サーバ環境

CPU	Pentium Ⅲ 600MHz
Memory	512Mbytes
OS	FreeBSD [27] 5.1-RELEASE
データベース	PostgreSQL7.4 [28]

表 5.2: クライアント環境

CPU	Pentium Ⅲ 1.13GHz
Memory	512Mbytes
OS	FreeBSD 4.8-RELEASE

次に、本システムで使用した機器を以下に示す。本研究で使用したRFリーダ・ライタ (図5.1 [3]) の仕様を、以下の表5.3に示す。 形V720S-HMF01 RFIDユニットは、RFリーダ・ラ

表 5.3: 本研究で使用したRFリーダ・ライタ

製品名	オムロン社 [29] 形V720S-HMF01 [30]
動作周波数	13.56MHz
インターフェース	コンパクトフラッシュ TypeⅡ

イタとCFカードを一体化したもので、PDAやノートブック型PCとの組み合わせで携帯型RFIDリーダ・ライタを実現できる。オムロン社の形V720シリーズのRFIDタグだけでなく、ISO/IEC15693完全準拠に対応する。

[3]オムロン社webサイトより

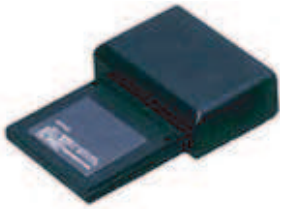

図 5.1: 本研究で使用した RF リーダ・ライタ

本研究で使用した RFID タグ (図 5.2[3]) の仕様を、以下の表 5.4に示す。 この RFID タグ (ス

表 5.4: 本研究で使用した RFID タグ

製品名	オムロン社 形 V720-D52P [31]
IC チップ	Philips 社 [32] I・CODE1 [33]
メモリ容量	44byte(ユーザエリア)
動作周波数	13.56MHz
データ書き換え回数	各アドレス毎 10 万回

図 5.2: 本研究で使用した RFID タグ

マートラベル) は、非接触型 IC カードの世界標準である ISO15693 規格対応の IC チップを用いた RFID タグでる。非常に薄く、柔軟性に富み、書き換え可能な RFID タグの中でもコストパフォーマンスの高い製品である。

5.2 実装概要

本研究で提案するプライバシ保護機構は、EPCAC と EPCADS から構成される。本研究の実装では、提案する全ての機能の実装は行わずに、プロトタイプとして以下に示す機能だけを実装した。

- ID の暗号化プログラム
- EPCAC
  - RF リーダ・ライタ制御
  - EPCADS への認証・復号化要求
- EPCADS
  - オーナ認証
  - ID 復号化
  - EPCADS 上に構築するデータベース

各々について次節以降で詳述する。

5.3 ID の暗号化プログラム

本研究の暗号化方式として用いる公開鍵暗号方式で最も有名なのが、Ronald Rivest・Adi Shamir・Leonard Adleman によって提唱された RSA 暗号 [34] [35] [36] (付録 A.1) である。RSA 暗号方式では、公開鍵で暗号化、秘密鍵で復号化を行うだけでなく、秘密鍵で暗号化、公開鍵で復号化できる仕組みになっているため、本実装では暗号化アルゴリズムとして RSA 暗号を用いた。

本研究ではプライバシ保護機構の設計と実装が目的であり、暗号強度は問わない。そのため、暗号化プログラムの実装では、暗号化したい文字列を与えると、それを 1 文字ずつ暗号化し“ . ” (ドット) で区切るという方法を用いた。

暗号化プログラムの実装は、付録 A.2に示したアプリケーションの、暗号化部分を用いて実装を行った。

5.4 EPCAC

EPCAC は RF リーダを接続した PC から RFID タグの ID を読み取り、そのデータと復号化のために鍵 A を EPCADS に送信する。その後、EPCADS での RF リーダの認証が成立した場合のみ、復号化された ID を受信しコンソールに表示する。RF リーダの認証については EPCADS の実装の節で詳述するが、EPCAC は鍵 A を EPCADS に送信する。以下の図 5.3は、今回実装した EPCAC のフローチャートを示したものである。各モジュールについて以下に詳述する。

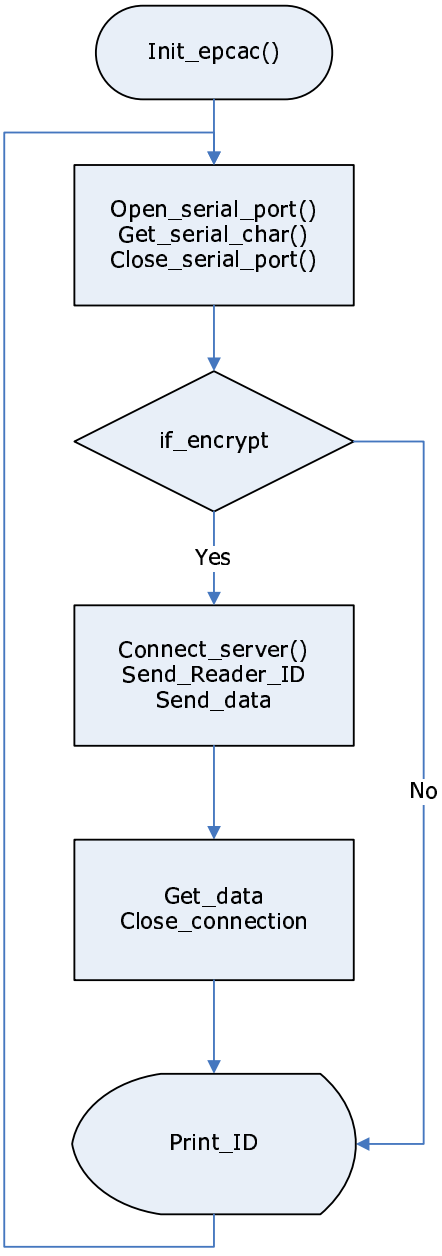

図 5.3: EPCAC のフローチャート

5.4.1 RF リーダ・ライタ制御

今回用いた RF リーダ・ライタは PC の CF スロットに刺すとシリアルモデムとして認識される。RF リーダ・ライタの制御はシリアルポートにコマンドを送信することによって行う。以下に示す関数はシリアルポートにコマンドを送信する関数 (図 5.4) と、シリアルポートからコマンドの実行結果を受信する関数 (図 5.5) である。

```
int put_serial_string(char *s){
  if(write(fd,s,strlen(s)) != 1)
    return -1;
  return 0;
}
```

図 5.4: put_serial_string()

```
int get_serial_char{
  char c;
  int res;
  res = read(fd,(char *)&c,1);
  return c;
}
```

図 5.5: get_serial_char()

5.4.2 EPCADS への認証復号化要求

RF リーダ・ライタから暗号化された ID を読み取ると、EPCAC は RF リーダ・ライタの RFID 読み取りを一時休止する。そして、RF リーダ・ライタのメッセージから暗号化された ID をパースし、EPCADS に送信する。EPCAC はそのまま待機し、EPCADS から ID が復号化されて返信されるのを待つ。EPCADS から受け取ったメッセージは、そのまま標準出力に出力される。

```
while(1){
    c=0;
    c=get_serial_char();
    if(i > 1){
        rs_rbuffer[j] = c;
        j ++;
    }
    i++;
    if( c == CR){
        printf("Crypted ID is %s\n",rs_rbuffer);
        write(fd1, rs_rbuffer, 64);
        memset(rs_rbuffer,'\0',64);
        ret = read(fd1, wideno, 16);
        wideno[ret] = '\0';
        printf("decrypting...\n");
        printf("\tGet Decrypted ID %s\n\n\n",wideno);
        memset(rs_rbuffer,'\0',64);
        i = 0;
        j = 0;
        break;
    }
    if(i>63){
        break;
}
```

図 5.6: EPCADS への ID 送信・受信モジュール

5.5 EPCADS

EPCADS は EPCAC からの認証・復号化要求を処理する。

以下の図 5.7は、今回実装した EPCADS のフローチャートを示したものである。各モジュールについて以下に詳述する。

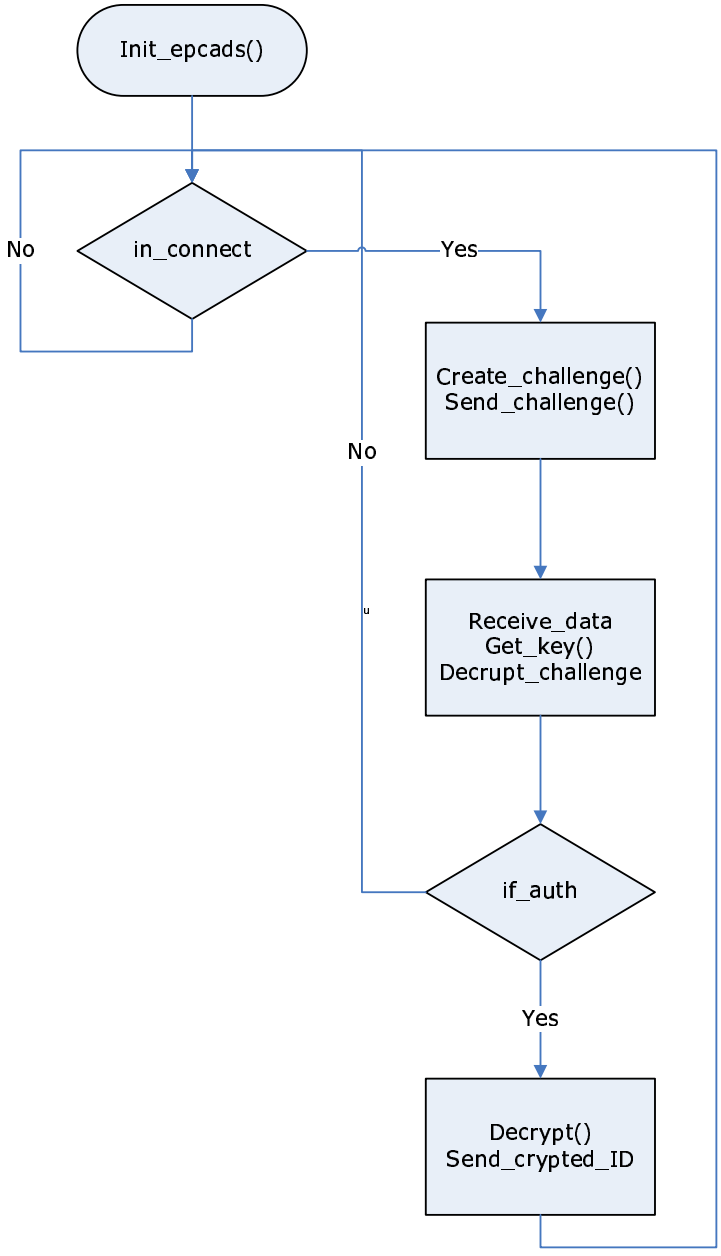

図 5.7: EPCADS のフローチャート

5.5.1 オーナ認証

5.5.2 ID 復号化

復号化モジュールの実装は、付録 A.2に示したアプリケーションの、復号化部分を用いて実装を行った。

```
char decrypt() {
    unsigned long c;      /*one cypher code in the cypher codes*/
    char p;               /*one plain charactor translated from c*/
    char crypt[4][8];     /* トークンをしまう２次元配列 */
    char *token = " .";   /* 区切り文字はスペースとカンマ */

    /*set keys*/
    setD();
    setM();

    /*translation*/
        strcpy(crypt[0], strtok(buf, token));
        c=strtoul(crypt[0],NULL,10);
        id[0]=c2p(c);
        strcpy(crypt[1], strtok(NULL, token));
        c=strtoul(crypt[1],NULL,10);
        id[1]=c2p(c);
        strcpy(crypt[2], strtok(NULL, token));
        c=strtoul(crypt[2],NULL,10);
        id[2]=c2p(c);
        strcpy(crypt[3], strtok(NULL, token));
        c=strtoul(crypt[3],NULL,10);
        id[3]=c2p(c);

    return((char)id);
}
```

図 5.8: 復号化関数

5.5.3 EPCADS 上に構築するデータベース

```
saru=> create table epcads (
saru(> id text not null CONSTRAINT bar PRIMARY KEY,
saru(> owner text not null,
saru(> modulo int not null,
saru(> public int not null,
saru(> secret int not null
saru(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "bar"
for table "epcads"
CREATE TABLE

saru=> \d
        List of relations
 Schema |  Name  | Type  | Owner
--------+--------+-------+-------
 public | epcads | table | don
(1 row)
```

図 5.9: テーブル作成と確認

```
saru=> insert into epcads (id, owner, modulo, public, secret)
saru=> values ('260.95.34.233', 't23', 323, 13, 133);
INSERT 17191 1
saru=> select * from epcads;
      id       | owner | modulo | public | secret
---------------+-------+--------+--------+--------
 260.95.34.233 | t23   |    323 |     13 |    133
(1 row)
```

図 5.10: データ入力と確認

第6章　評価

本章では、第5章で実装したSARUシステムのプロトタイプの評価を行う。

6.1　評価方針

本節では、本システムの評価方針について述べる。定性的な評価として、第3章で述べたプライバシ保護システムの要件を満たしているかどうかの機能評価を行う。さらに、プロトタイプの評価により、本システムの、既存研究に対する優位性を検証する。定量的な評価としては、性能評価を行うことにより、本システムを実際に運用する際の性能を評価する。

6.2　機能評価

本節では、前節の評価方針に基づき、本システムを定性的に評価する。本システムの機能評価における評価項目は、以下に挙げるとおりである。

- IDを暗号化・復号化できたか
- RFIDタグに所有権を設定できたか
- 所有権の委譲はできたか
- Long-term Tracking対策できたか

それぞれについて詳述する。

6.2.1　実験環境

本項では、上記の評価項目を評価するにあたって、構築した実験環境について述べる。実験に用いた機材は第5章で述べた実装環境 (表5.1、表5.2) と同一である。これらの機材を、それぞれスイッチングハブに接続し、プライベートネットワーク上で実験を行った。

6.2.2　IDを暗号化・復号化できたか

本項では、本システムで用いるRFIDタグの暗号化と復号化を評価する。以下に示す図6.1は、RFIDタグに暗号化されたIDを書き込むためにIDを暗号化するプログラムの動作画面である。本プログラムでは、plain.txtに記述されているIDを指定した鍵で暗号化し、cypher.txtに保存する。

```
[don@epcac] > cat plain.txt
0936
0253
```

```
[don@epcac] > ./rsa2

'e' encrypt / 'd' decrypt _e
Public key E [13] =13
00000000000000000000000000001101
Modulo M [323] =323

Input PLAIN text filename for input.
[plain.txt] _plain.txt
[plain.txt] was succesfully opened.

Input CYPHER codes filename for output.
[cypher.txt] _cypher.txt
[cypher.txt] was succesfully opened.

File succesfully transformed. :D
```

```
[don@epcac] > cat cypher.txt
260.95.34.233...
260.50.219.34...
```

図 6.1: 暗号化アプリケーションの動作画面

一方、以下に示す図 6.2は、前述のプログラムで暗号化された ID を復号化するプログラムである。本プログラムでは、cypher.txt に記述されている ID を指定した鍵で復号化し、decrypt.txt に保存する。

decrypt.txt の内容は、図 6.1中の plain.txt と同一であり、正常に復号化されたことが確認できた。以上により、本システムでも用いる RFID の ID を暗号化でき、さらに暗号化された ID を復号化できることが確認された。

次に、この暗号化・復号化モジュールを利用して作った本システムのプロトタイプの動作確認を行った。これは、上記のプログラムで暗号化された ID を書き込んだ RFID タグを、EPCAC で読み取らせ、EPCADS にて復号化を正常に行えているかどうかの評価である。以下の図 6.3は

```
[don@epcac] > ./rsa2

'e' encrypt / 'd' decrypt _d
Secret key D [133] =133
00000000000000000000000010000101
Modulo M [323] =323

Input CYPHER codes filename for input.
[cypher.txt] _cypher.txt
[cypher.txt] was succesfully opened.

Input PLAIN text filename for output.
[decrypt.txt] _decrypt.txt
[decrypt.txt] was succesfully opened.

File succesfully transformed. :D
```

```
[don@epcac] > cat decrypt.txt
0936
0253
```

図 6.2: 復号化アプリケーションの動作画面

EPCAC の動作画面である。

図 6.1で暗号化した ID を EPCADS に送信することにより、図 6.2で復号化したものと同じ ID を取得できている。これにより、EPCAC が送信した暗号化された ID を、EOCADS で正常に復号化できていることが確認された。

6.2.3 RFID タグに所有権を設定できたか

以下の図 6.4は、EPCADS 上のデータベース内のオーナ情報を閲覧している画面である。

図 6.4からわかるように ID 毎にオーナの属性を設定してある。EPCADS がこのデータベース上のオーナ情報を参照することにより、オーナの認証を行えることを確認した。

```
[don@epcac] > ./client /dev/ttyd4 192.168.100.2
Crypted ID is 260.95.34.233...
decrypting...
        Get Decrypted ID 0936

Crypted ID is 260.50.219.34...
decrypting...
        Get Decrypted ID 0253
```

図 6.3: EPCAC の動作画面

```
saru=>  select id, owner from epcads;
      id       | owner
---------------+--------
 260.95.34.233 | t23
 260.50.219.34 | test
(2 rows)
```

図 6.4: データベース上のオーナ情報

6.2.4 所有権の委譲はできたか

EPCADS が、前項のオーナ情報を参照し、オーナの認証を行い、オーナ以外に対しては ID を復号化しないことで、オーナ以外からの復号化された ID の取得を防止する。所有権の委譲が発生した際には、EPCADS 上のデータベース内のオーナ情報を更新することで、所有権の委譲を実現する。本システムのプロトタイプでは、Owner Management Client の実装を行っていないため、データベース上のオーナ情報を直接変更する必要があるが、オーナ以外からの復号化を防止することで所有権の委譲ができたことを確認した。

6.2.5 Long-term Tracking 対策できたか

本システムでは、オーナが任意の時点で ID を暗号化し直すことが可能である。新たな鍵を用意し、ID の暗号化プログラムを用いて ID を暗号化し、それをタグに書き込み、鍵を EPCADS 上のデータベースに登録することにより実現可能である。本システムのプロトタイプでは Key Registry Client の実装を行っていないため、EPCADS 上のデータベースを直接編集する必要があるが、これにより、RFID タグに格納される ID を、任意の時点で変更することが可能となるため、本システムでは Long-term Tracking に対処できていると言える。

6.2.6　既存研究との機能比較

本節では、本システムのプロトタイプと、既存研究の機能比較について述べる。以下の表6.1は、第2章の表6.1に本システムのプロトタイプの項目を追加したものである。

表6.1と同様に、表中で用いられる記号は、以下の意味に示す通りである。○は、その研究を利用することにより実現することが可能である機能である。△は、その研究で想定されている機能ではあるが、不完全、もしくはどのようにして実現するのか不明瞭な機能である。×は実現不可能な機能である。

表 6.1: 既存研究との機能比較

	隠蔽・暗号化	リーダの認証	所有権の設定	所有権の委譲	Long-term Tracking 対策
Hashed Lock Algorithm	×	○	×	×	○
Anonymous EPC	○	△	×	×	×
Reencrypt Anonymous EPC	○	△	×	×	○
Encrypted EPC	○	△	×	×	×
E-signed EPC	○	△	×	×	×
E-signed + Encrypted EPC	○	△	×	×	×
本システムのプロトタイプ	○	○	○	○	○

前節までで述べたように、本システムではIDの暗号化により、IDの非構造化を実現している。また、RFIDタグに所有権の概念を導入し、オーナの認証を行うことにより所有権の委譲にも対応している。Long-term Trackingへの対策についても、システムとしてIDの再暗号化を可能としていることにより、Long-term Trackingに対処可能であると言える。

一方、既存研究では所有権の概念が存在しないため、所有権の委譲も行えない。また、所有権を伴わないリーダの認証を行える既存研究はあるが、どのようにしてリーダの認証を行うかが不明瞭なものが多い。さらに、Long-term Trackingに対処可能な既存研究は少ない。

6.2.7　まとめ

本項で、前項までの内容をまとめる。本システムは、IDの暗号化や、所有権の委譲、Long-term Tracking対策など、第3章で述べたプライバシ保護システムの機能要件は全て満たした。また、既存研究と比較し、以下の点で既存研究に対して本システムが優位性を持っていると言える。

- IDを暗号化することにより、プライバシ情報の漏洩を防止できること
- 所有権の概念があり、所有権の委譲が行えること
- 認証方法を明確にした上で、リーダの認証を行えること
- Long-term Trackingに対処可能であること

6.3 性能評価

本節では、前述の評価方針に基づき、本システムを定量に評価する。

6.3.1 評価目的

EPCADS を実際に運用することを考慮し、以下の評価項目で EPCADS の性能を評価する。

- EPCADS が 1 分間に処理できる EPCAC からのクエリの数
- 1 クエリあたりの処理時間

これにより、EPCADSが1ホストあたりどの程度のクエリを処理できるのかを調査し、EPCADS の実際の運用上の問題点を整理する。

6.3.2 評価方法

データベースのエントリー数を段階的に増やしていき、EPCADS が 1 分間に処理できる EPCAC からのクエリの数と、1 クエリあたりの処理時間を計測する。データベースのエントリー数は 50 件から計測し、その次は 100 件、その後は 100 件ごとに 6500 件までデータベースのエントリー数を増やしていく。

6.3.3 実験環境

前項で挙げた評価項目を評価するため、以下の環境において評価実験を行った。以下図 6.2は評価に用いた EPCADS の環境を表している。また、EPCAC の実験環境は以下の表 6.3に示す

表 6.2: EPCADS の実験環境

CPU	Pentium Ⅲ 600MHz
Memory	512Mbytes
OS	FreeBSD 5.1-RELEASE
データベース	PostgreSQL7.4

通りである。これらの EPCADS と EPCAC を、インターネットと隔離されたプライベートネッ

表 6.3: EPCAC の実験環境

CPU	Pentium Ⅲ 1.13GHz
Memory	512Mbytes
OS	FreeBSD 4.8-RELEASE

トワークに接続し、実験を行った。計測データの誤差や、ばらつきを減らすため、このプライベートネットワーク上には、他の PC を接続していない。

6.3.4 実験結果

以上のような環境において実際に評価実験を行った結果を、以下の図 6.5に示す。

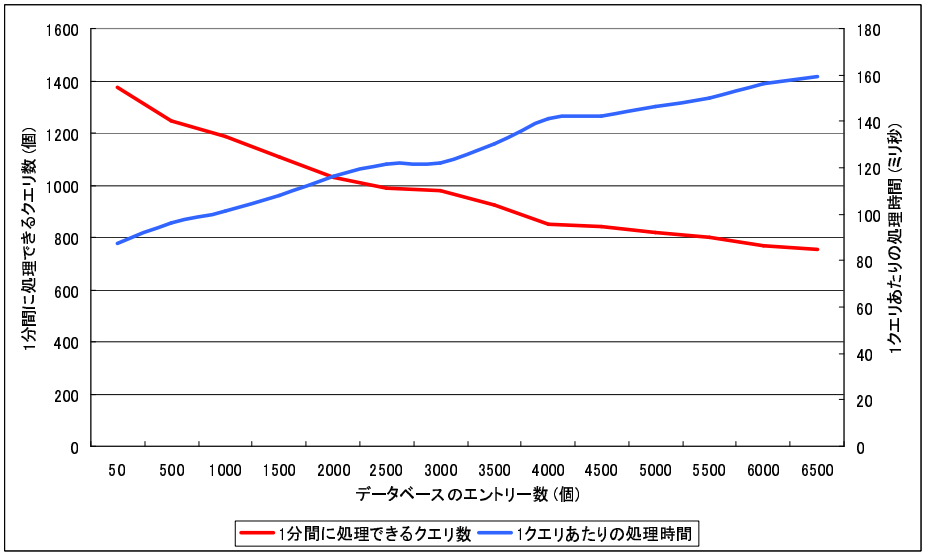

図 6.5: データベースのエントリー数に応じた EPCADS の性能変化

データベースのエントリー数が 50 件の時には 1 分間に約 1400 個のクエリを処理できている。しかし、データベースのエントリー数が増えるにつれて、処理できるクエリ数が減少していき、エントリー数が 6500 件の時にはおよそ半数の 700 個になっている。1 クエリあたりの処理時間を見てみると、データベースのエントリー数が 50 件の時にはおよそ 80$msec$ であった処理時間が、エントリー数が 6500 件の時には 160$msec$ 以上かかってしまっている。

6.3.5 まとめ

実験結果から、本システムはデータベースのエントリー数が増加するのに応じて、EPCADS の性能が低下することが確認された。これは、データベースのエントリー数が増加することにより、データの検索にかかる時間が長くなることが原因である。このことから、本システムの今後の課題として、データベースの応答性能の向上が挙げられる。今回の評価では、データベース内のエントリ数の増加に伴い、応答性能の低下が確認された。この応答性能の低下は PostgreSQL 特有の現象であるのか、また、他のデータベースを用いた場合にはどのような結果が出るのかについて、調査する必要がある。

また、データベースの応答性能を除外して考えた際にも、性能面で大きな課題が残る。実際の運用を考えた際には多数クライアントから、同時に多くのクエリが集中することが考えられる。データベースのエントリー数を最小にして計測した場合でも、1 分間に処理できるクエリ

の数が 1500 個程度であるため、世界中のオブジェクトにタグを貼付し、本システムでプライバシを保護することは不可能である。さらに、復号化にかかる時間も今後の課題としてあげられる。今回の評価では暗号強度は評価対象外であったため、非常に短時間で計算できる暗号化しかしていないが、実際の運用を考えた際には暗号強度を十分に確保する必要がある。このような場合には、復号化のための計算にかかる時間は飛躍的に増加するため、同一時間内に処理できるクエリの数はさらに減少することが予想される。

以上のことをまとめると、本システムを実際に運用する際には、データベースの応答性能と、復号化処理にかかる時間が問題点となり、規模性を確保できないと思われる。

第7章　結論

本章では、本研究についてまとめ、今後の課題を挙げ、本論文の結論とする。

7.1　まとめ

本研究では、RFID のプライバシ問題を解決するため、オーナという所有権を表す属性を設定し、オーナの認証を行い、ID の暗号化をすることでプライバシの漏洩を防止するシステムモデルを提案し、設計した。

さらに、プロトタイプの実装・評価を行い、既存研究との機能比較・システムの機能評価を行うことにより、オブジェクトの所有者が変わった際に、ID を取得できる権限を次の所有者に委譲できる点で、本システムの優位性を示すことができた。

本システムを用いることで、ユーザは RFID のプライバシ漏洩を未然に防ぐことが可能となり、RFID を用いた私的なオブジェクトの管理を安全に行うことができるようになる。

7.2　今後の課題

本節では、本システムを運用していく上での今後の課題について述べる。

7.2.1　規模性の確保

前章での評価の結果からもわかるように、本システムを運用するに当たり、1 台の EPCADS で世界中の RFID タグを処理するのは不可能である。実際に運用する際には、EPCADS を複数設置し、クライアントからのクエリの負荷分散をする必要がある。規模性を確保するに当たり、以下のような問題点が挙げられる。

- **負荷分散の方法**
  どのような技術を用いて負荷分散を実現するのかが問題になる。また、複数の EPCADS を設置して負荷分散を実現する場合、各 RFID タグと対応する EPCADS をどのようにして発見するのかについても、検討する必要がある。

- **運用形態**
  EPCADS ではオーナと鍵の情報を管理するため、EPCADS を複数設置して負荷分散する場合、EPCADS を誰が管理するのかが問題になる。公的な機関が管理するのか、各個人が自分の EPCADS を設置して使用するのか、製品を製造したベンダが EPCADS を設置するのか、よく検討する必要がある。

- **データベースの整合性確保**
  データベースでオーナの管理と鍵の管理を行うため、EPCADS を負荷分散した場合、データベースの整合性を保つ必要がある。どのように EPCADS に対する負荷を分散させるのかを考慮するのと同時に、データベースの分散の方法も考える必要がある。データベースが扱っている情報がオーナや鍵の情報など、本システムにおいて極めて重要な情報であるため、この問題はよく吟味する必要がある。また、前項の運用形態とも関わってくるが、分散させた EPCADS の管理者がそれぞれ異なる場合、問題はさらに複雑化する。

- **データベースの応答性能向上**
  プロトタイプの評価では、データベースに格納されたエントリ数の増加に伴い、応答性能の低下が確認された。この応答性能の低下は PostgreSQL 特有の現象であるのか、また、他のデータベースを用いた場合にはどのような結果が出るのかについて、調査する必要がある。

- **暗号化アルゴリズム**
  プロトタイプの評価では、暗号強度は評価対象外であったため、比較的単純な暗号化アルゴリズムを使用し、暗号化の鍵も短いものを用いて、処理時間が短くなるようにシステムを構築した。しかし、実際の運用を考えた際には、暗号強度が十分に確保されないと ID が容易に解読されてしまい、本システムの優位性を確保できなくなる。暗号強度を確保するには、暗号化の鍵を長くするほかに、複雑な暗号化方式を用いることが考えられが、一般的に、双方とも復号化にかかる処理時間は長くなる。復号化にかかる処理時間が短く、暗号強度を確保できる暗号化アルゴリズムを検討する必要がある。

7.2.2 暗号化アルゴリズム

本論文では、システムの提案が目的であったため、暗号強度や生成される暗号文のサイズは問わなかった。しかし実際の運用を考えた際には、使用される暗号化方式が脆弱だと、暗号化してあっても容易に解読されてしまう可能性が高い。また、暗号文のサイズは、暗号化以前の文字列よりも大幅にサイズが大きくなる。同時に、RFID タグに格納できるデータには限りがある。このため、精製される暗号文のサイズは、可能な限り小さい方が好ましい。

以上を踏まえ、実際に使用する暗号化方式は、暗号強度や精製される暗号文のサイズを含めて比較・検討する必要がある。

謝辞

本論文の作成にあたり、御指導いただきました慶應義塾大学環境情報学部教授 村井純博士、並びに同大学環境情報学部教授 徳田英幸博士、同大学環境情報学部助教授 楠本博之博士、同大学環境情報学部助教授 中村修博士、同大学環境情報学部専任講師 南政樹氏に感謝致します。

本研究を進めていく上で、絶えずご指導と御助言をいただきました慶應義塾大学院政策・メディア研究科博士課程 川喜田佑介氏同大学院政策・メディア研究科専任講師 羽田久一博士、株式会社インターネットイニシアチブ技術研究所 宇夫陽次朗博士に深く感謝致します。

また、日常的にご指導頂き、お世話になった慶応義塾大学政策・メディア研究科特別研究専任講師 植原啓介博士、同大学院特別研究助手 佐藤雅明氏、同大学大学院 SFC 研究所研究員 渡辺恭人博士、同大学院政策・メディア研究科博士課程 湧川隆次氏、同大学院政策・メディア研究科修士課程 三屋光史朗氏、日野哲志氏、西原サヤ子氏、岡田耕司氏、小柴晋氏、渡里雅史氏、久松慎一氏に深く感謝致します。

さらに、一つ屋根の下で共に暮らし、絶えず足を引っ張り合いながらも切磋琢磨しあった慶應義塾大学環境情報学部 橋本和樹氏、いつも隣の席で罵倒し合い、励まし合った同大学環境情報学部 廣瀬峻氏、3 年間の長きに渡って寝食を共にした同大学環境情報学部 清水崇史氏、谷岡洋平氏、犬山隆太郎氏、松谷宏紀氏、金子紘子氏、同大学総合政策学部 高橋宏明氏、堀岡大輔氏、ならびに同期の仲間たちに感謝の意を表します。

そして、本論文の作成にあたり御協力して下さった慶應義塾大学環境情報学部 塚田学氏、苧阪浩輔氏、佐藤泰介氏、Auto-ID Laboratory の皆様、並びに慶應義塾大学 徳田・村井・楠本・中村・南合同研究室の皆様、特に NACM 研究グループの皆様に感謝致します。

最後に、これまで私を支えて下さったすべての方々に最大限の感謝の意を表ます。

本当に皆様ありがとうございました。

参考文献

[1] Suica. http://www.jreast.co.jp/suica/.

[2] 電子マネー Edy. http://www.edy.jp/.

[3] Speedpass. http://www.speedpass.com/.

[4] 回転寿司の自動清算システム「OAISO」.
http://www.omron.co.jp/card/rfid/case/003f.htm.

[5] EPC Global. http://www.epcglobalinc.org/.

[6] Auto-ID Labs. http://www.autoidlabs.org/.

[7] Auto-ID Center. http://www.autoidcenter.org/.

[8] Uniform Code Council, Inc. http://www.uc-council.org/.

[9] EAN International. http://www.ean-int.org/.

[10] David L. Brock. The Electronic Product Code (EPC) A Naming Scheme For Physical Objects, January 2001.

[11] John Price, Ed Jones, Howard Kapustein, Ravi Pappu, Darrell Pinson, Richard Swan, Ken Traub. Auto-ID Reader Protocol 1.0, September 2003.

[12] Christian Floerkemeier, Dipan Anarkat, Ted Osinski, Mark Harrison. PML Core Specification 1.0, September 2003.

[13] Sean Clark, Ken Traub, Dipan Anarkat, Ted Osinski. Auto-ID Savant Specification 1.0, September 2003.

[14] Oat Systems, MIT AutoID Center. Auto-ID Object Name Service (ONS) 1.0, September 2003.

[15] Mark Harrison, Duncan McFarlane. Development of a Prototype PML Server for an Auto-ID Enabled Robotic Manufacturing Environment, February 2003.

[16] Alien Technology. http://www.alientechnology.com/.

[17] M. Mealling and R. Daniel. *The Naming Authority Pointer (NAPTR) DNS Resource Record*, September 2000. RFC 2915.

[18] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS*, October 2002. RFC 3401.

[19] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm*, October 2002. RFC 3402.

[20] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database*, October 2002. RFC 3403.

[21] M. Mealling. *Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI)*, October 2002. RFC 3404.

[22] Simple Object Access Protocol. http://www.w3.org/TR/SOAP/.

[23] XML-RPC. http://www.xmlrpc.com/.

[24] Sanjay E. Sarma, Stephen A. Weis, Daniel W. Engels. RFID Systems, Security & Privacy Implications, November 2002.

[25] 日経 BP IT Pro 記事, “ Anonymous EPC, Encrypted EPC ”, April 2003. http://itpro.nikkeibp.co.jp/free/NBY/NEWS/20030425/2/.

[26] Toshiharu ISHIKAWA, Yukiko YUMOTO, Michio KURATA, Makoto ENDO, Shingo KINOSHITA, Fumitaka HOSHINO, Satoshi YAGI, Masatoshi NOMACHI. Applying Auto-ID to the Japanese Publication Business, October 2003.

[27] The FreeBSD Project. http://www.freebsd.org/.

[28] PostgreSQL. http://www.postgresql.org/.

[29] オムロン株式会社. http://www.omron.co.jp/.

[30] 形 V720S-HMF01. http://www.omron.co.jp/card/rfid/prod/v720/v720cf.html.

[31] 形 V720-D52P. http://www.omron.co.jp/card/rfid/prod/v720/id.html.

[32] Royal Philips Electronics. http://www.philips.com/.

[33] I-CODE Smart Label Technology. http://www.semiconductors.philips.com/markets/identification/products/icode/.

[34] B. Kaliski. *PKCS #1: RSA Encryption Version 1.5*, March 1998. RFC 2313.

[35] B. Kaliski and J. Staddon. *PKCS #1: RSA Cryptography Specifications Version 2.0*, October 1998. RFC 2437.

[36] J. Jonsson and B. Kaliski. *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, February 2003. RFC 3447.

付 録A　RSA暗号

A.1　RSA暗号のアルゴリズム

本項ではRSA暗号のアルゴリズムについて述べる。

RSA暗号系では2つの鍵は次のようにして決める。ある2つの異なる大きな素数pとqを選び、

$$n = p \times q$$

$$\phi(n) = (p-1) \times (q-1)$$

を求める。このnを係数と呼ぶ。e(公開指数)を

$$\gcd(e, \phi(n)) = 1$$

とし、

$$e \times d \equiv 1 \bmod \phi(n)$$

を満たす整数d(秘密指数)を求める。すると(e, n)が公開鍵、(d, n)が秘密鍵となる。p、q、dは誰にも知られないようにする。

平文Mを暗号化するには、次のようにする。

$$C = M^e \bmod n$$

暗号文Cを復号化するには、次のようにする。

$$M = C^d \bmod n$$

RSAは、整数論であるオイラーの定理と2つの素数を使って公開鍵暗号の仕掛けを実現しており、大きい数の素因数分解の困難さを暗号化手法としている。

A.2　RSA暗号の暗号化・復号化プログラム

本研究の実装にあたり、RSA暗号の暗号化と複合化をするアプリケーションを実装した。以下にそのアプリケーションのソースコードを掲載する。

```
/* RSA Encryption and Decryption */

#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <math.h>

/*global variables*/

unsigned long keyE=13;          /*public key*/
unsigned long keyM=323;         /*modulo*/
unsigned long keyD=133;         /*secret key*/
char E[32],M[32],D[32];         /*their binaly values*/


/*functions*/

char encrypt(void);                                     /*file encrypting rootine*/
char decrypt(void);                                     /*file decrypting rootine*/
unsigned long p2c(char);                                /*charactor encrypting kernel*/
char c2p(unsigned long);                                /*code decrypting kernel*/
void setE(void);                                        /*let user input public key e*/
void setM(void);                                        /*let user input modulo m*/
void setD(void);                                        /*let user input secret key d*/
void bin(unsigned long,char *);                         /*translate decimal value to octal valu
unsigned long ulpow(unsigned long,unsigned long);       /*power*/
unsigned long modulo(unsigned long,unsigned long);      /*modulo*/
unsigned long modpow(unsigned long,unsigned long);      /*power with modulo*/


void main() {
    char line[10];    /*input line*/
    char ed;        /*encrypt or decrypt*/
    char flag;        /*error check flag*/

    printf("\n=======================================\n");
    printf("      RSA Enctyption and Decryption\n");
    printf("=======================================\n\n");

    do {
        printf("\n'e' encrypt / 'd' decrypt _");
        fgets(line,sizeof(line),stdin);
        sscanf(line,"%c",&ed);
        switch(ed) {
            case 'e':
                flag=encrypt();
                break;
```

```
            case 'd':
                flag=decrypt();
                break;
            default:
                flag = -1;
        }
        if (flag!=0) {
            printf("\nGood night. =D\n");
        } else {
            printf("\nFile succesfully transformed. :D\n");
        }
    } while(flag==0);
}

char encrypt() {
    char p;                 /*one charactor in the plain text*/
    unsigned long c;    /*one cypher code translated from p*/
    char counter;        /*encrypted charactors*/

    char p_name[128];    /*the name of plain text file*/
    char c_name[128];    /*the name of cypher text file*/
    FILE *p_file;        /*plain text file for input*/
    FILE *c_file;        /*cypher code file for output*/

    /*set keys*/
    setE();
    setM();

    /*set plain text filename*/
    printf("\nInput PLAIN text filename for input.\n");
    printf("[plain.txt] _");
    fgets(p_name,sizeof(p_name),stdin);
    if (strlen(p_name)==1) {        /*default*/
        strcpy(p_name,"plain.txt");
    } else {
        p_name[strlen(p_name)-1] = '\0';
    }

    /*opening plain text file*/
    p_file = fopen(p_name,"r");
    if (p_file == NULL) {            /*check the file error*/
        printf("Cannot open [%s] for input.\n",p_name);
        return(1);
```

```
}
printf("[%s] was succesfully opened.\n",p_name);

/*set cypher codes file name*/
printf("\nInput CYPHER codes filename for output.\n");
printf("[cypher.txt] _");
fgets(c_name,sizeof(c_name),stdin);
if (strlen(c_name)==1) {        /*default*/
    strcpy(c_name,"cypher.txt");
} else {
    c_name[strlen(c_name)-1] = '\0';
}

/*opening cypher codes file*/
c_file = fopen(c_name,"w");
if (c_file == NULL) {            /*check the file error*/
    printf("Cannot open [%s] for output.\n",c_name);
    return(1);
}
printf("[%s] was succesfully opened.\n",c_name);

/*translation*/
printf("\n[Encrypted Plain Text]\n");
counter=0;
while(1) {
    p=fgetc(p_file);
    if (p==EOF) break;
    c=p2c(p);
    printf("%ld ",c);
    fprintf(c_file,"%ld ",c);
    if (++counter==16) {
        counter=0;
        printf("\n");
        fprintf(c_file,"\n");
    }
}
printf("[EOF]\n");

/*closing files*/
fclose(p_file);
fclose(c_file);

return(0);
```

```
}

char decrypt() {
    unsigned long c;     /*one cypher code in the cypher codes*/
    char p;              /*one plain charactor translated from c*/
    char flag,counter;   /*decrypted codes*/

    char c_name[128];    /*the name of cypher text file*/
    char p_name[128];    /*the name of plain text file*/
    FILE *c_file;        /*cypher codes file for input*/
    FILE *p_file;        /*plain text file for output*/

    /*set keys*/
    setD();
    setM();

    /*set cypher codes filename*/
    printf("\nInput CYPHER codes filename for input.\n");
    printf("[cypher.txt] _");
    fgets(c_name,sizeof(c_name),stdin);
    if (strlen(c_name)==1) {        /*default*/
        strcpy(c_name,"cypher.txt");
    } else {
        c_name[strlen(c_name)-1] = '\0';
    }

    /*opening cypher codes file*/
    c_file = fopen(c_name,"r");
    if (c_file == NULL) {            /*check the file error*/
        printf("Cannot open [%s] for input.\n",c_name);
        return(1);
    }
    printf("[%s] was succesfully opened.\n",c_name);

    /*set plain text filename*/
    printf("\nInput PLAIN text filename for output.\n");
    printf("[decrypt.txt] _");
    fgets(p_name,sizeof(p_name),stdin);
    if (strlen(p_name)==1) {        /*default*/
        strcpy(p_name,"decrypt.txt");
    } else {
        p_name[strlen(p_name)-1] = '\0';
    }
```

```
    /*opening plain text file*/
    p_file = fopen(p_name,"w");
    if (p_file == NULL) { /*check the file error*/
        printf("Cannot open [%s] for output.\n",p_name);
        return(1);
    }
    printf("[%s] was succesfully opened.\n",p_name);

    /*translation*/
    printf("\n[Decrypted Plain Text]\n");
    counter=0;
    while(1) {
        flag=fscanf(c_file,"%d ",&c);
        if (flag!=1) break;
        if (++counter==16) {
            counter=0;
            fscanf(c_file,"\n");
        }
        if (flag!=1) break;
        p=c2p(c);
        printf("%c",p);
        fprintf(p_file,"%c",p);
    }
    printf("[EOF]\n");
    fclose(c_file);

    /*closing files*/
    fclose(p_file);

    return(0);
}

unsigned long p2c(char p) {
    unsigned long c=1;       /*cypher code*/
    unsigned long temp=p;    /*fat baloon*/
    int b;                   /*one bit*/

    for (b=31 ; b>=0 ;b--) {
        if (E[b]==1) {
            c *= temp;
            c = modulo(c,keyM);
        }
```

```
        temp=modpow(temp,2);
    }
    return(c);
}

char c2p(unsigned long c) {
    unsigned long p=1;    /*plain charactor*/
    int b;                /*one bit*/

    for (b=31 ; b>=0 ;b--) {
        if (D[b]==1) {
            p *= modpow(c,ulpow(2,31-b));
            p = modulo(p,keyM);
        }
    }
    return((char)p);
}

void setE(void) {
    char line[100];

    printf("Public key E [%ld] =",keyE);
    fgets(line,sizeof(line),stdin);
    if (strlen(line)==1) {
        keyE=keyE;            /*default*/
    } else {
        sscanf(line,"%ld",&keyE);
    }
    bin(keyE,&E[0]);
}

void setM(void) {
    char line[100];

    printf("Modulo M [%ld] =",keyM);
    fgets(line,sizeof(line),stdin);
    if (strlen(line)==1) {
        keyM=keyM;        /*default*/
    } else {
        sscanf(line,"%ld",&keyM);
    }
}
```

```
void setD(void) {
    char line[100];

    printf("Secret key D [%ld] =",keyD);
    fgets(line,sizeof(line),stdin);
    if (strlen(line)==1) {
        keyD=keyD;            /*default*/
    } else {
        sscanf(line,"%ld",&keyD);
    }
    bin(keyD,&D[0]);
}

void bin(unsigned long v,char *B) {
    unsigned long mask;    /*a value when only one bit is standing*/
    int b;                 /*one bit*/

    for (b=31 ; b>=0 ; b--) {
        mask=ulpow(2,b);
        if (v>=mask) {
            *(B+31-b)=1;
            v -= mask;
        } else {
            *(B+31-b)=0;
        }
        printf("%d",*(B+31-b));
    }
    printf("\n");
}

unsigned long ulpow(unsigned long x,unsigned long y) {
    unsigned long c,pow=1;

    if (y==0) return(1);
    for (c=1 ; c<=y ; c++) {
        pow*=x;
    }
    return(pow);
}

unsigned long modulo(unsigned long x,unsigned long y) {
    unsigned long ans;
    long i=x/y;
```

```
    ans=x-i*y;
    return (ans);
}

unsigned long modpow(unsigned long x,unsigned long y) {
    unsigned long ans=1,c;

    if (y==0) return(1);
    for (c=1 ; c<=y ; c++) {
        ans *= x;
        ans = modulo(ans,keyM);
    }
    return (ans);
}
```