平成 12 年度

学士学位論文

モバイルコンテンツ課金プロトコル

Billing Protocols in Mobile Communication Environments

1010374　　大石 恭裕

指導教員　　清水 明宏

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

モバイルコンテンツ課金プロトコル

大石 恭裕

本研究室では，コンテンツ流通支援方式の研究を行っている．近年，i-mode に代表される携帯電話でのコンテンツ流通が注目を浴びており，課金システムの開発が急務となっている．コンテンツを円滑に流通させるために，課金システムは重要であり，その中核として認証方式が位置付けられる．

本検討の目的は，本研究室で提案している SAS(Simple And Secure) と呼ぶワンタイムパスワード認証プロトコル，SAS をベースとした小額課金システムである SAS コインを，モバイルコミュニケーション，特に携帯電話等の処理能力の低い端末へ適用する際の最適な実装方式を確立することである．

本検討では，SAS に対する問題点を指摘し，それらを解決する SAS-K を提案し，評価を行った．さらに，SAS-K を用いた小額課金システムである SAS コインのプロトコルについて考察した．実装フィールドとして，ユーザ，コンテンツプロバイダ，課金センタの三者から構成される小額課金システムのプラットフォームを作成し，その環境において SAS-K および SAS コインの実装を行った．

提案方式 SAS-K を利用することにより，モバイル環境を含むあらゆる環境において，極めて安全な認証を行うことを可能とした．

キーワード　ワンタイムパスワード，パスワード認証方式，SAS，盗聴，なりすまし

Abstract

Billing Protocols in Mobile Communication Environments

Yasuhiro OISHI

We are studying on contents communications systems. Contents communication using "browser phones", particularly i-mode, have been getting popular. Billing systems are located as many functions in contents communications systems.

The research is aiming to install security functions, which are a one-time password authentication protocol, called SAS, and billing system, called SAS-Coin, to mobile communication environments with lower processing abilities.

First, SAS-K, a new type of one-time password authentication method, is proposed to cope with some attacks on SAS. Next, the SAS-Coin is studied, which is a simple billing method applying SAS-K. The SAS-K and SAS-Coin protocols are installed on a platform which has three players, Customer, Vendor, and Broker.

SAS-K, and an SAS-K based billing method, SAS-Coin have brought secure functions to mobile communications environments.

key words one-time password, password authentication, SAS, wiretapping, inpersonation

目次

図目次

第 1 章

はじめに

本研究室では，コンテンツ流通支援方式の研究を行っている．近年，i-mode に代表される携帯電話でのコンテンツ流通が注目を浴びており，課金システムの開発が急務となっている．コンテンツを円滑に流通させるために，課金システムは重要であり，その中核として認証方式が位置付けられる．

資格認証方法として，(1) 指紋や顔，声等の身体情報を利用したもの，(2) IC カードのような本人の所有物を利用したもの，(3) パスワードや暗証番号等の本人の知識によるものが考えられる．しかし，(1)，(2) は指紋，顔，声あるいは IC カード等を読み込む装置が必要であり，インターネットやモバイルコミュニケーション環境で利用するにはコストが大きく，現実的でない．

そこで，これらの環境でよく用いられるのが (3) のパスワード等を用いた資格認証方法である．パスワード等の認証情報を用いて通信相手やユーザの資格を認証する方法は，公開鍵暗号方式を応用したものと，共通鍵暗号方式を応用したものの二つに大別することができるが，インターネット関連の通信プロトコル等への組み込みにおいては，公開鍵暗号方式より格段の高速処理が可能な共通鍵暗号方式を応用した方法，特に，パスワード認証方式 [1, 2, 3, 4, 5] がよく用いられる．基本的なパスワード認証の手順は以下の通りである．

まず，被認証者（装置を含む）が認証者（サーバなどの装置を含む）にパスワードを送信する．認証者は，受信したパスワードと登録されているパスワードを比較する．

しかし，この方法には，次のような問題点がある．

(a) 認証側にあるパスワードファイルの盗見によりパスワードが盗まれる．

(b) 通信中，回線盗聴によってパスワードが盗まれる．

(c) 被認証者は認証者に，自分の秘密情報であるパスワードを公開する必要がある．

最初の問題 (a) を解決する方法として，例えば，被認証者が認証者に，パスワードに一方向性関数を施したデータを登録しておき，認証時に，認証者が受信したパスワードに同じ一方向性関数を施し，結果を比較するという方法がある．

一方向性関数とは，入力の総当たり以外に，出力から入力を得る効率的な手段が存在しない関数であり，総当たりの計算量を十分大きくしておけば，無資格者が入力データを算出して，被認証者になりすますことを防止できる．一般に，一方向性関数は，ハッシュ関数あるいは DES[6] や FEAL[7] などの共通鍵暗号方式によって得ることができる．共通鍵暗号方式は，共通秘密鍵を用いて入力される平文を処理して暗号文を出力として得るもので，平文と暗号文が与えられても共通秘密鍵が算出できない．特に FEAL では，平文や共通秘密鍵の入力が 1 ビット変化しただけでも，その入力変化の痕跡を全くとどめない出力を得ることができるという特徴を有している．

以上説明した通り，一方向性関数を用いた方法によって，基本的なパスワード認証方式の問題 (a) は解決できる．しかし，これを回線盗聴が簡単なインターネットに適用する場合，問題 (b) を解決することはできない．また，問題 (c) に関しては，この基本的なパスワード認証方式は，銀行の顧客認証などには適用できても，同一レベルのユーザ同士の資格認証には適していない．

このような問題を解決する方法として，パスワードなどの認証情報を可変にする資格認証方法 [3, 4, 5] が広く用いられている．しかし，それらにもいくつかの問題点があった．Lamport の方法 [3] には，被認証者がパスワードを定期的に再設定する必要性や，認証者側に比較して処理能力の小さい被認証者側での処理負担が大きいという問題がある．また，CINON 法 [4] では，Lamport の方法の欠点であるパスワードの更新の必要性をなくすことができたが，被認証者および認証者における計算量が大きいという問題は依然残った．さらに，通信途中の認証情報を第三者により二回続けて盗聴された場合，「なりすまし」の危険性もあった．

そこで我々は，セキュリティが十分でないネットワーク上でユーザ等の資格認証を行う

際，被認証者側および認証者側共に，実行する計算量が極めて少なく，被認証者側にも認証者側にも簡易で小さいプログラムサイズで実現可能，さらに，通信路上での盗聴に強い安全な認証方式，SAS パスワード認証方法 [8] を提案した．そして，SAS を応用した小額課金システムである SAS コイン [9] も併せて提案した．しかし，これらの提案方式にもいくつかの問題点が残されている．SAS については，同期の問題，さらに通信路上での盗聴や不正操作により，「なりすまし」や「いたずら」の危険性がある．SAS コインについては，計算量の問題や，コイン追加方法などの問題が残されている．

本論文では，SAS にほとんど追加コストを必要とすることなく，それらの問題点を克服できる方法，SAS-K を提案する．さらに，SAS-K を用いた小額課金システムである SAS コインについて考察し，これらの提案方式をモバイルコミュニケーション，特に携帯電話等の処理能力の低い端末に実装する際の最適な実装方式を確立することを目指す．実装フィールドとして，Customer(ユーザ)，Vendor(コンテンツプロバイダ)，Broker(課金センタ) の三者から構成される小額課金システムのプラットフォームを作成し，その環境に SAS-K および SAS コインを実装する．

本論文の以下の部分では，まず SAS に対する脅威および同期問題，SAS コインの問題点について**第 2 章**で述べる．そしてそれらの問題点を解決する方式である SAS-K の提案を**第 3 章**で行い，SAS-K に対する評価を**第 4 章**で行う．さらに，**第 5 章**で記憶装置を持たない端末での実装方法を述べる．続いて**第 6 章**で SAS-K，SAS コインを実装するフィールドとして小額課金プラットフォームについて考察し，実装内容を記す．**第 7 章**では残された課題について述べ，最後に**第 8 章**で本論文のまとめとする．

なお，付録として，従来の提案方式である SAS，SAS コインについての概要を添付している．

第 2 章

SAS，SAS コインの問題点

現在，清水研究室では SAS および SAS コインを提案しているが，これらにはいくつかの問題点が残されている．本章では，これらの問題点について指摘する．

なお，従来の SAS，SAS コインについては**付録 A**，**B** としている．

2.1 SAS の問題点

現状の SAS では，以下のような問題点が考えられる．

1. すり替え攻撃：この攻撃により，次の認証フェーズから，盗聴者は「なりすまし」を行うことが可能になる．
2. でたらめ攻撃：この攻撃によって，User は再登録を行わない限り，次回から正常な認証を行われなくなる．
3. 同期問題：通信が途切れた場合，同期がとれなくなり，以降の認証が正常に行われなくなる．

以下では，これらの問題点について考察する．

2.1.1 すり替え攻撃

たとえば，第 k 回目の認証フェーズで，User が Host に対して送信した認証用データ $E^1_{k-1} \oplus E^2_{k-1}$, $E^2_k \oplus E^2_{k-1}$ を第三者が盗聴したとする．そして，$(k+1)$ 回目の認証データをも盗聴し，そのデータ

$A, E_k^1 \oplus E_k^2, E_{k+1}^2 \oplus E_k^2$

を,

$A, E_k^1 \oplus E_k^2, E_k^2 \oplus E_{k-1}^2$

に置き換える.

Host では，今回認証用のデータを用いて認証を行い，次回認証用データとして,

$E_{k-1}^2 = (E_k^2 \oplus E_{k-1}^2) \oplus E_k^2$

を登録してしまう．次回から，第三者は

$A, E_{k-1}^1 \oplus E_{k-1}^2, E_k^2 \oplus E_{k-1}^2$

と,

$A, E_k^1 \oplus E_k^2, E_k^2 \oplus E_{k-1}^2$

を交互に送信することにより，ユーザ A になりすますことができる.

2.1.2　でたらめ攻撃

たとえば，第 k 回目の認証フェーズで，User が Host に対して送信した認証用データ

$A, E_{k-1}^1 \oplus E_{k-1}^2, E_k^2 \oplus E_{k-1}^2$

を第三者により

$A, E_{k-1}^1 \oplus E_{k-1}^2, X$

にすり替えられたとする．すると Host は，はじめの方のデータ $E_k^1 \oplus E_k^2$ を用いて正常に認証を行う．当然，認証はパスでき，後の方のデータ X を用いて $X \oplus E_{k-1}^2$ を計算し，そのデータに対する何のチェックも行わず登録する．そして，次の認証フェーズで，正規のユーザは今回認証用データとして,

$E_k^1 \oplus E_k^2$

を送信するわけだが,

$E(E_k^1 \oplus E_k^2 \oplus (X \oplus E_{k-1}^2)) \neq X \oplus E_{k-1}^2$

なので，再登録を行わない限り認証をパスすることができなくなってしまう.

2.1.3　同期問題

User は認証フェーズ毎に乱数を生成し，それを元に認証用データを作り出しているが，通信が途切れた場合，同期が取れなくなり，その認証フェーズで使用しなければならない乱数ではなく，一回前の乱数を使ってしまい，正常な認証が行えなくなる．

2.2　SAS コインの問題点と改良方針

2.2.1　SAS コインの問題点

現状の SAS コインでは，小額支払システムとして，以下のような問題点が考えられる．

- コイン追加方法 (利用済み金額の取り扱い等) が不明な点．
- Broker，Vendor 認証情報が更新されない，あるいは両者間での認証の必要性有無．
- 支払いフェーズのロジックが不十分 (セキュアルートの利用必須)．
- 利用金額に対して確認は残金額 (利用金額総計) でしかできない問題．また，アンマッチ時の対応が考慮されていない．
- 登録金額が大きくなると処理時間が必要である問題．

2.2.2　改良方針

最初に挙げた問題であるコイン追加方法については図 6.5 の手順により，一つの解決策を提示しているが，さらに検討が必要である．また，最後に挙げた，「登録金額が大きくなると処理時間が必要になる」という問題の解決方法として，単位毎の計算を行うことにより計算量を削減するという方法が考えられる．つまり，我々が日常生活で行う両替と同じ概念を利用するということである．現状の SAS コインにおいて，一円単位でコインを作成するとすると，一万円のコインを生成するには一万回の計算が必要である．ここで，両替の概念を利用し，例えば，一万円のコインを生成するには，一度に五千円を生成する計算を 2 回，五千円のコインを生成する場合は一度に千円を生成する計算を 5 回，というように，単位毎の

2.2　SAS コインの問題点と改良方針

計算を行う．この際，単位毎のコインを作成するために，単位毎の乱数を生成する必要がある．(図 2.1 参照)

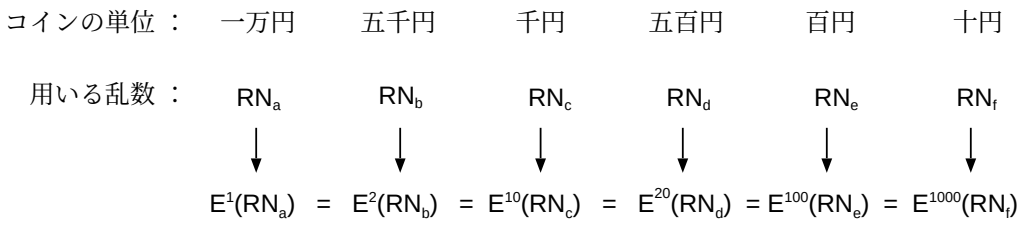

図 2.1　単位毎の計算

第 3 章

SAS-K の提案

本章では **2.1** で挙げた問題点を，わずかな変更によって解決することができる方式，SAS-K の提案を行う．

以下に今回の提案方式 SAS-K を記述し，その過程を図 3.1 に示している．さらに，同期問題の解決策については **3.2** で提案し，その過程を図 3.2 に示している．

なお，提案方式 SAS-K は，従来の SAS と同様に登録および認証フェーズから成り，登録フェーズについては全く同じ手順である．

3.1 攻撃対策

2.1.1 のすり替え攻撃，**2.1.2** のでたらめ攻撃の問題を解決するために必要なのは，次回認証用データの正当性を検証することである．従来の SAS と同じく User は，ユーザ ID と認証用データを 2 つ，認証フェーズ毎に Host 側に送信する．具体的には

$A,\ E_{k-1}^1 \oplus E_{k-1}^2,\ E_k^2 \oplus E_{k-1}^2$

を，

$A,\ E_{k-1}^1 \oplus E_{k-1}^2 \oplus E_k^3,\ E_k^2 \oplus E_{k-1}^2$

に変更して送信する．その後，Host 側では，

$E_k^2 = (E_k^2 \oplus E_{k-1}^2) \oplus E_{k-1}^2$

により，次回認証用データを取得する．そして，

$E_k^3 = E(E_k^2)$

という認証確認用データを生成する．次に，その E_k^3 を用いて，

3.1 攻撃対策

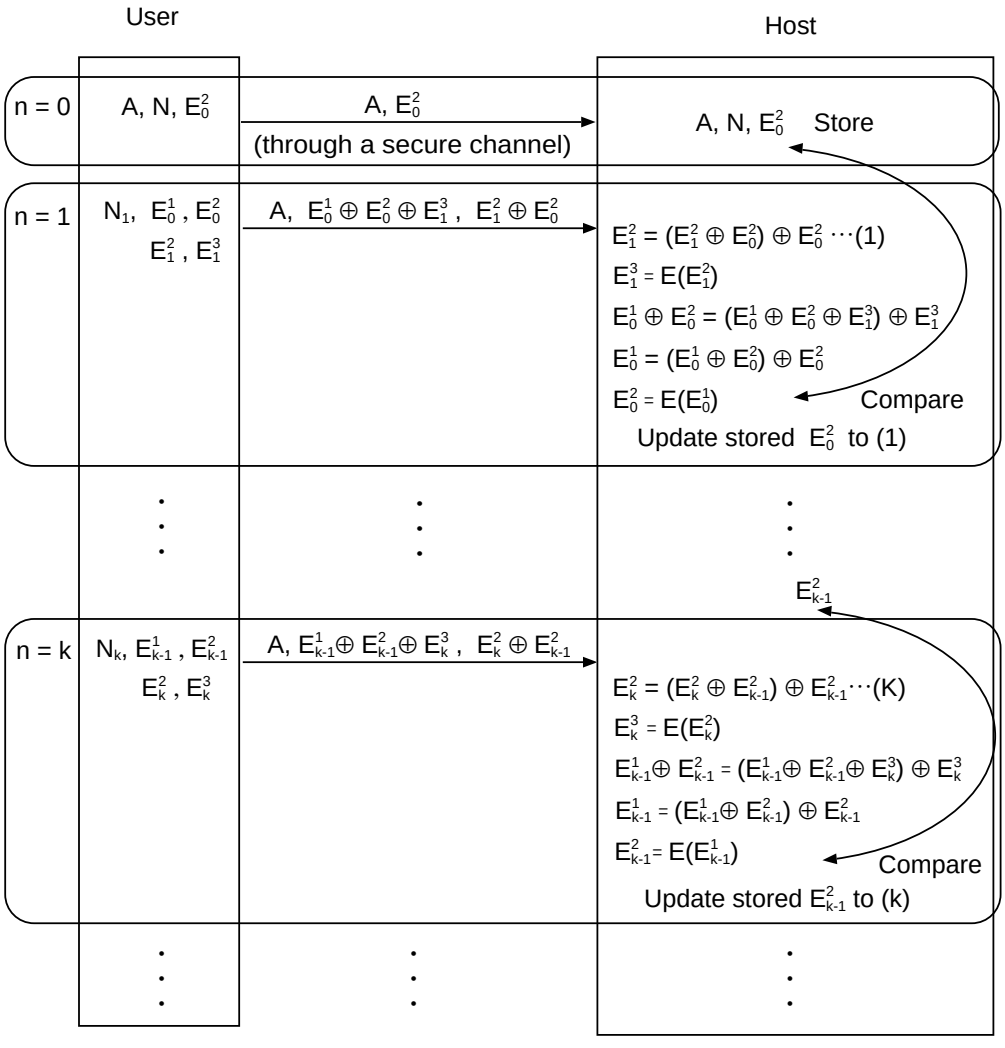

図 3.1　SAS-K の登録，および認証フェーズ

$E_{k-1}^1 \oplus E_{k-1}^2 = (E_{k-1}^1 \oplus E_{k-1}^2 \oplus E_k^3) \oplus E_k^3$

を計算し，さらに，登録されてある E_{k-1}^2 を用いて，

$E_{k-1}^1 = (E_{k-1}^1 \oplus E_{k-1}^2) \oplus E_{k-1}^2$

を導出する．そして，そのデータにもう一度ハッシュ関数を適用し，E_{k-1}^2 を算出し，そのデータと，登録されている E_{k-1}^2 を比較することによって認証を行う．こうすることによって，認証用データの正当性が検証された認証を可能とする．

3.2　同期対策

以前より，SAS では，同期の問題があった．ここでは今回提案する SAS-K における通信切断時の回復方法について検討する．通信が切断される場合として，User から Host への認証用データが未達の時，Host から User への認証 OK のメッセージが未達の時の 2 つが考えられる．これらの問題に対し，User，Host 共にカウンタ (0，もしくは 1 の値を取る) を持ち，カウンタにより同期を取る方式を提案する．(図 3.2 参照)

登録フェーズで，User，Host のカウンタをそろえておく (例えば 0 に)．認証フェーズで User は，ユーザ ID，認証用データに加え，カウンタの値も Host に送信する．Host 側では受け取ったカウンタの値によって，同期が取れているかどうかを検知．同期が取れていない場合は User に対してエラーを返す．User はエラーメッセージを受け取り，前回の認証フェーズで認証 OK のメッセージが受け取れなかったことを検知し，新しい乱数を生成することによって正常な認証を行うことができる．

3.2　同期対策

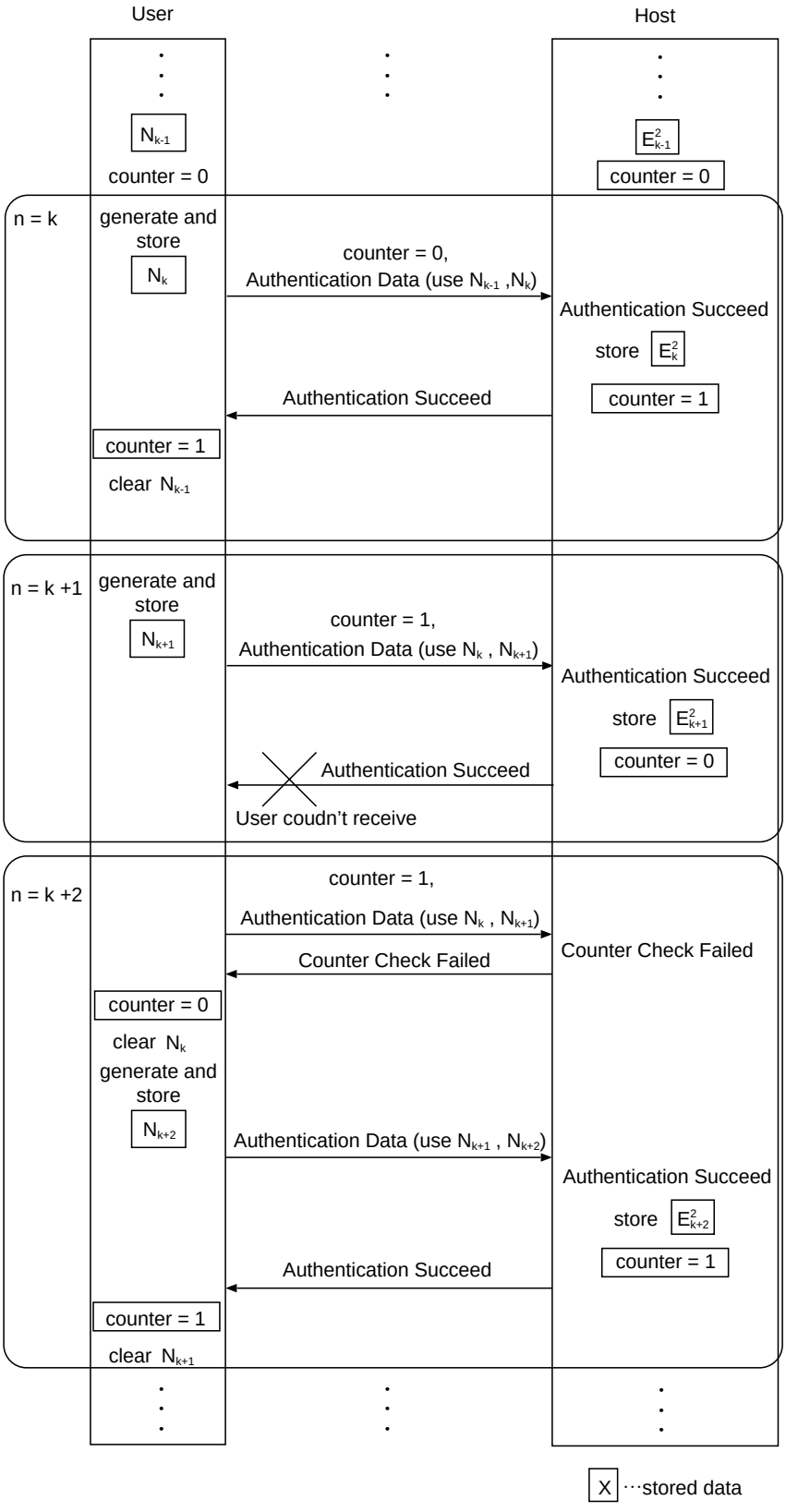

図 3.2　カウンタを利用した同期方法

第 4 章

SAS-K の評価

4.1 安全性

以下では，考え得る様々な攻撃方法に対しての SAS-K の安全性を評価する．

4.1.1 認証情報の盗聴

たとえば Host にある認証用データ E_k^2 を第三者によって盗まれたとする．ハッシュ関数の衝突回避性 (collision free property) と，一方向性 (one-way property) により，第三者は E_k^2 から E_k^1 をつくり出すことは不可能である．ゆえに，本当のパスワードを知っている User のみが E_k^1 をつくり出すことができ，認証をパスできる．

4.1.2 すり替え攻撃

従来の SAS で，この攻撃が問題になるのは次回認証用データの正当性が全く検証されていないことであった．第三者からすると，今回認証用データの正当性を Host に確認させるためには，このデータの本質的な部分である E_{k-1}^1 を変更することはできない．つまり，なりすましを成功させるためには，すり替えた今回認証用データの正当性を Host に保証させることである．

今回の提案方式において，もし第 $(k+1)$ 回目の認証フェーズで，次回認証用データである $E_{k+1}^2 \oplus E_k^2$ を第 k 回目認証フェーズで送信された次回認証用データである $E_k^2 \oplus E_{k-1}^2$ にすり替えられたとする．すると，Host では $(E_k^2 \oplus E_{k-1}^2) \oplus E_k^2$ を計算し，E_{k-1}^2 を取り

出すことになる．しかし，次回認証用データの正当性の検証に，今回認証用データを用いることにより，そのデータが不当であることが判明し，認証は失敗に終る．

ここで第三者が，なりすましを成功させるために必要なデータについて考えてみる．(図 4.1 参照)

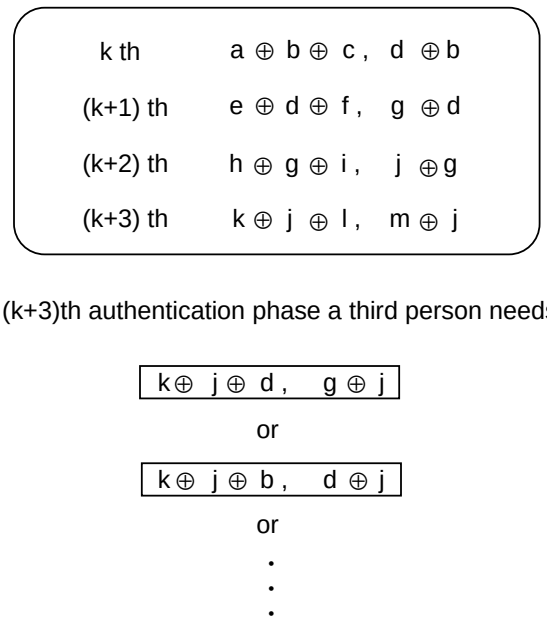

図 4.1　アタック方法

第 k 回目の認証フェーズにおいて，User から送信される認証用データ

$E^1_{k-1} \oplus E^2_{k-1} \oplus E^3_k$, $E^2_k \oplus E^2_{k-1}$

は,

$a \oplus b \oplus c$, $d \oplus b$

と表すことができる．同様に第 $(k+1)$ 回目から $(k+3)$ 回目の認証用データをそれぞれ

$e \oplus d \oplus f$, $g \oplus d$,　…第 $(k+1)$ 回目

$h \oplus g \oplus i$, $j \oplus g$,　…第 $(k+2)$ 回目

$k \oplus j \oplus l$, $m \oplus j$,　…第 $(k+3)$ 回目

と表すことができる．ここで，第三者はこれまでの全ての通信を盗聴したとし，例えば第 $(k+3)$ 回目の認証フェーズでユーザからの認証データを改ざんし，Host に g を登録するこ

とによって第 $(k+2)$ 回目に戻して，なりすましを行おうとした場合，

$k \oplus j \oplus d,\ g \oplus j$

というデータが必要になる．ここで，第 $(k+2)$ 回目の認証フェーズで送信された $j \oplus g$ を利用したとしても，$k \oplus j \oplus d$ を作り出すことは不可能である．なぜなら，そのデータを導出するには今回はじめて送信された k というデータが必要になるが，k を取り出すには l というデータが必要となり，このデータもまた，今回はじめて現れるため，$k \oplus j \oplus d$ が導出できないからである．つまり，Host に g を登録することは不可能であるため，第 $(k+3)$ 回目に戻すことはできない．

また，Host に d を登録することによって第 $(k+2)$ 回目に戻そうとした場合，

$k \oplus j \oplus b,\ d \oplus j$

というデータが必要になる．しかし，この場合も上述したように l を導出できないため，そのデータを作り出すことができない．同様にして，いずれの段階に戻すにしても，なりすましを行うために必要なデータを過去の認証データから作り出すことが不可能であるため，すり替え攻撃に対して安全だと言える．

4.1.3　でたらめ攻撃

Host 側では，次回認証用データの正当性を検証し，正当であることが言えた場合のみ次回認証用データを登録するという手順を取るため，通信路上で次回認証用データをでたらめなものにすり替えたとしても，認証をパスすることができないので，全く意味のない攻撃になる．

4.2　エラーリカバリ (同期問題)

以下では，提案方式 SAS-K を実装する際，通信が切断された場合の安全性について評価を行う．

4.2.1 User から Host へ

User から Host への認証用データが未達の場合，タイムアウト処理等ののち，User は次回の認証フェーズで，今回と全く同じデータを送信することになるので，この方式 SAS-K を利用することにより，エラーとして処理できる．つまり，これは，User から Host への認証用データが未達でも，同期がずれることがないことを示している．

さらに，たとえ送信中に第三者によってカウンタの値を変えられたとしても，Host では認証を拒否するために，認証用データを更新することがないので，全く影響を受けない．

4.2.2 Host から User へ

従来の方式では，Host から User への認証用データが未達の場合，Host では次回認証用の情報が更新されているが，User はそのことを検知できない．このことにより，次の認証フェーズで，一回前の乱数を用いて認証用データを作ってしまい，認証をパスすることができなくなる．しかし，上記の方式によりカウンタを用いると，両者間で認証情報を授受する際，併せて同期チェックを行うことができるため，確実に同期を取ることができる．

4.3 処理量

従来の SAS と比べ (**A.3** 参照)，ハッシュ関数の適用回数が User 側，Host 側で 1 度だけ増えるが，これは次回認証用データの正当性を検証する上で避けては通れない最小の過程である．そして，同期問題を解決するためのコストは，カウンタを用意すること，送信データ量がわずか 1bit 増えること，User 側では第 k 回目の認証フェーズで，認証 OK のメッセージが届くまで N_k だけでなく N_{k-1} も併せて記憶しておく必要があることである．しかし，N_{k-1} に代えて E^1_{k-1}，E^2_{k-1} を記憶しておくことにより，User 側で実行しなければならないハッシュ関数の適用回数は，わずか 3 回に削減することができる．

第 5 章

SAS-K を記憶領域のない端末に実装する場合

今回の提案方式 SAS-K では，従来の SAS と同様に乱数を用いて認証データを生成している．この方式ではユーザ端末に乱数を発生する機能およびそれを記憶する領域が必要となる．しかし実用の際，ユーザ側に乱数生成機能および記憶領域がない場合も考えられる．たとえば本研究室で検討中である「Thin Client を用いた Server Based Computing」にセキュリティ機能を盛り込む場合である．そこでこのような場合にも SAS-K の適用を可能とする方式を提案する．(図 5.1 参照)

5.1 手順

Host 側に User の認証回数を記憶しておく．そして User は認証フェーズ毎にサービスリクエストを行い，その都度 Host は User に対して認証回数 (カウンタの値) を送信する．そして User 側では乱数ではなく，認証回数を用いて認証用データを生成する．そして，ユーザ ID，認証用データを Host に送信し，Host 側で認証を行う．認証が成立した場合，カウンタを 1 進める．

5.2 評価

この方式を用いた場合，第三者によって送信中のカウンタの値を盗聴，もしくは改ざんされたとしても，認証は失敗に終り，なりすまし等の危険性は全くない．さらに，認証フェー

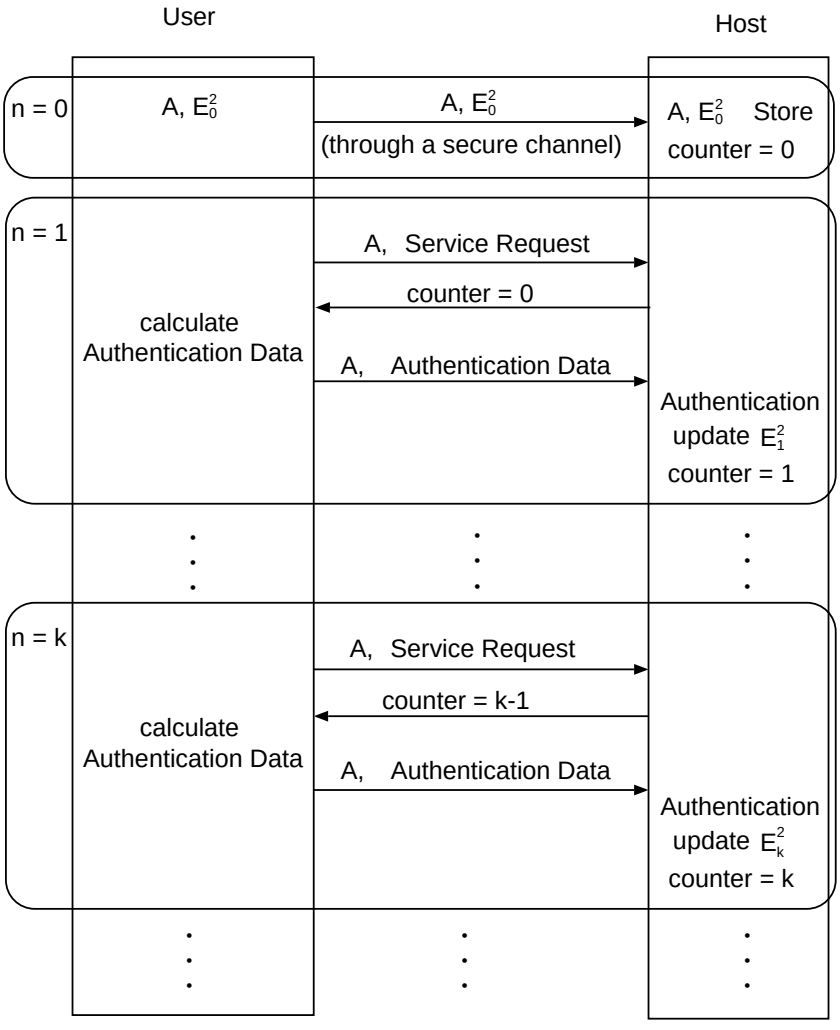

図 5.1　記憶領域がない場合

ズ毎に認証回数を送信するために，同期がずれることはない．また，乱数に代えて認証回数を用いることによる SAS-K の安全性は何ら変化ない．なぜならば，SAS-K の安全性は認証データを生成する際のハッシュ関数の強度に依存しているからである．

第 6 章

実装

6.1 プラットフォームについての考察

以下では，SAS-K および SAS コインを実装するにあたり，そのプラットフォームにおけるシステム構成，システム用件について考察する．

6.1.1 システム構成

本システムは，Broker(課金センタ)，Vendor(コンテンツプロバイダ)，Customer(ユーザ) の 3 者で構成される．デジタルコンテンツを円滑に流通させるために，最低限必要である条件として，3 者の定義を以下に記述する．なお，今回，処理の内容を簡易化する目的で，SAS コインの実装にあたり，それぞれ一つの Broker，一つの Vendor，一人の Customer でシステムを構成している．よって，**B.2.5** に示すマルチベンダシステムは利用していない．

Broker は，Customer，Vendor に対するアカウント (口座) の開設，Customer に対する残高の照会機能等を持つ課金情報管理機関である．今回，Broker は中立な立場であり，極めてセキュアな機関であることを前提としている．Broker において管理するデータは，各 Customer 毎のユーザ ID，認証情報，および使用可能額である．Vendor に対する管理データは，各 Vendor 毎のベンダ ID と認証情報，利用者の使用額分を移すためのアカウントである．

Vendor は，Customer に対して情報 (デジタルコンテンツ) を提供する．コンテンツには無料コンテンツと有料コンテンツがあり，Customer が有料コンテンツの購入を希望した場合のみ直接 Customer，Vendor 間で認証処理を行い，認証が成立した場合，コンテンツ配信

および課金を行う．

Customer は，Vendor からインターネットを介してコンテンツを閲覧し，購入する．コンテンツの購入を希望した場合のみ，Vendor に認証を依頼する．認証が成立し，課金処理が行われ，コンテンツを受け取る．

6.1.2　システム用件

課金システムを構築する際の用件として，次の 2 点が挙げられる．

- 資格を有する Customer のみがサービスを利用できる．
- Customer の秘密情報であるパスワードおよびコインが悪意ある第 3 者に知られない．

また，この小額課金システムでは，小額のデジタルコンテンツの販売を目的としているため，以下の要求条件を満たす必要がある．

- 決済にかかる手数料等のコンテンツ料金以外に必要となるコストが全くない，あるいは極めて小さいこと．
- 十分なセキュリティを確保すること．

さらに，今回，携帯電話等の処理能力の低い端末に実装することを想定しているため，以下の要求条件を満たす必要がある．

- 計算量が小さく，高速な処理が可能であること．
- 小さなプログラムで実装可能であること．

上記 6 つの要求条件を満たすシステムを構築する．Broker に登録されたユーザ ID と認証情報を有する Customer のみ利用を可能とする．また，認証処理で用いる SAS-K 認証方式では，ユーザ認証に必要な認証パラメータが毎回異なるため，ユーザの秘密情報が知られることはない．また，通信路上の情報の不正操作に強いため，極めて安全である．

また，今回，認証時に使用するハッシュ関数の部分には，FEAL[7] を利用することにより，極めて高速な処理および小さいプログラムでの記述が可能となっている．

6.2　実装内容

以下に SAS-K，SAS コインを実装するにあたり，今回作成した小額課金プラットフォーム，SAS-K の実装内容，SAS コインのフローチャートについて記述する．

なお，実装にあたり使用した言語は C 言語である．また，上述したように実装する際のハッシュ関数の部分には，FEAL を適用した．(**付録 C** は，そのソースコード)

6.2.1　小額課金プラットフォーム

以下に，今回作成した小額課金プラットフォームについて記述する．(図 6.1 参照)

本システムは 6.1 に従い，Broker(課金センタ)，Vendor(コンテンツプロバイダ)，Customer(ユーザ) の 3 者で構成した．

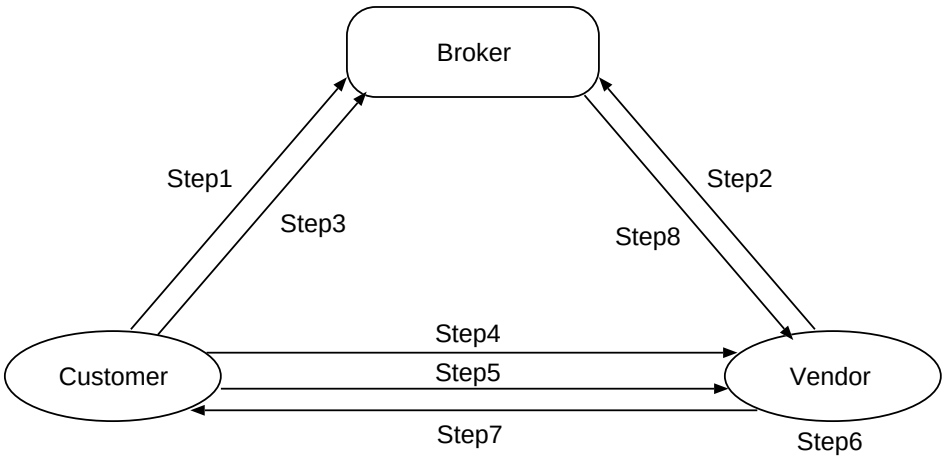

図 6.1　小額課金プラットフォーム

- Step1：Customer がユーザ登録を行う．(図 6.2)
- Step2：Vendor がベンダ登録を行う．(図 6.3)
- Step3：Customer が金額登録を行う．(図 6.4)

- Step4：インターネットを通じて Customer が Vendor に接続し，コンテンツを閲覧.
- Step5：有料コンテンツの購入依頼 (認証情報およびコインの提示).(図 6.6)
- Step6：認証処理および課金.(図 6.6)
- Step7：デジタルコンテンツの送信.
- Step8：月一回程度で清算.(図 6.7)

図 6.1 の流れに従い，3 者間での TCP/IP を用いた通信を可能とした.

6.2.2　SAS-K，SAS コイン

以下に認証処理の詳細なフローチャートを示す.(図 6.2〜図 6.7 参照)

従来の SAS コインに対して上記 SAS-K プロトコルを適用することにより，より安全な小額支払システムが提案できる．コイン生成時の計算量の問題については **2.2** に示すような，単位毎の計算を行うことによって，計算量を削減できると考えている．また，コイン追加方法については一つの解決策として図 6.5 に提示した．その他の問題点については，今後の課題としたい.

・図 6.2 は Customer 登録のフローチャートである.(SAS-K の登録手順)

・図 6.3 は Vendor 登録のフローチャートである．Customer の登録と同じ手順で登録を行う.(SAS-K の登録手順)

・図 6.4 は金額登録 (初回) の流れ図である.

Customer は，Broker，Vendor，金額用にそれぞれ乱数を用意し，必要なデータを生成する必要がある．これは，Broker に認証してもらい，金額を登録し，使用フェーズでベンダに認証してもらう必要があるからである.

Broker は認証 (SAS-K の認証手順) を行い, 認証が成立した場合, 金額を登録し, Customer の Vendor に対する認証情報とコインを Vendor の Customer に対する認証情報でマスクして，送信する.

・図 6.5 は金額登録 (k 回目) のフローチャートである.

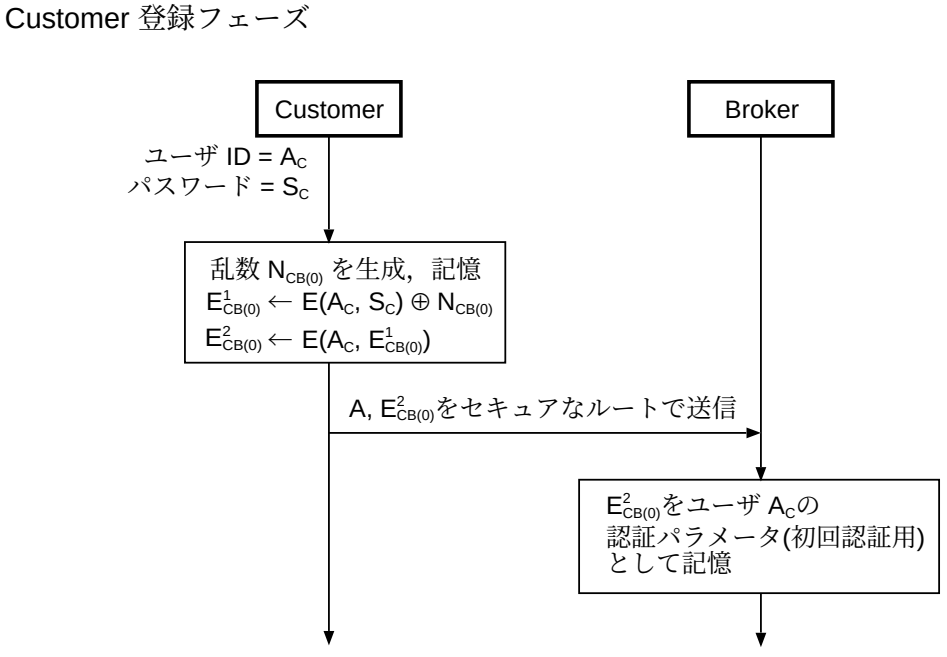

図 6.2　Customer 登録のフロー

今回，残っていた金額を，新しく登録する金額に加算し，新しい乱数を生成することにより，新コインを生成する方法を提案している．

・図 6.6 は使用フェーズのフローチャートである．

Customer は Vendor に認証情報およびマスクしたコイン (今回を含む過去の利用金額総計) を送信する．

Vendor 側では認証情報，コイン，今回の使用金額 (コンテンツ料) に対して認証を行い，認証が成立した場合，新しい認証情報，金額，コインを記憶し，デジタルコンテンツを送信する．

・図 6.7 は支払フェーズのフローチャートである．

今回，定期的に (たとえば月一回程度)Vendor は Customer の利用金額総額およびコインを，セキュアルートで Broker に送信するものとしている．

Broker は認証を行ったのち，Customer のアカウントから Vendor のアカウントに振込を行う．

従来の SAS コインに SAS-K を適用することにより，よりセキュリティの高い小額支払

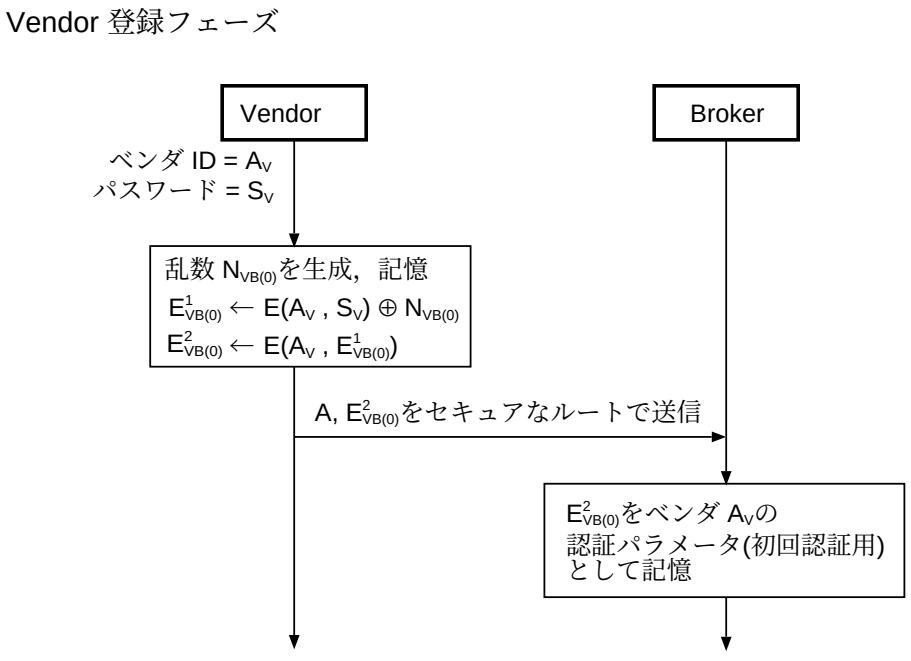

図 6.3　Vendor 登録のフロー

システムの提案を行った．また，コイン生成時の計算量の問題は，**2.2** に示したように，単位毎の計算を行うことによって解決できる見込みである．さらに，コイン追加方法の問題も図 6.5 の手順により，一つの解決策を提示できた．その他の問題については，今後の研究課題とする．

6.2 実装内容

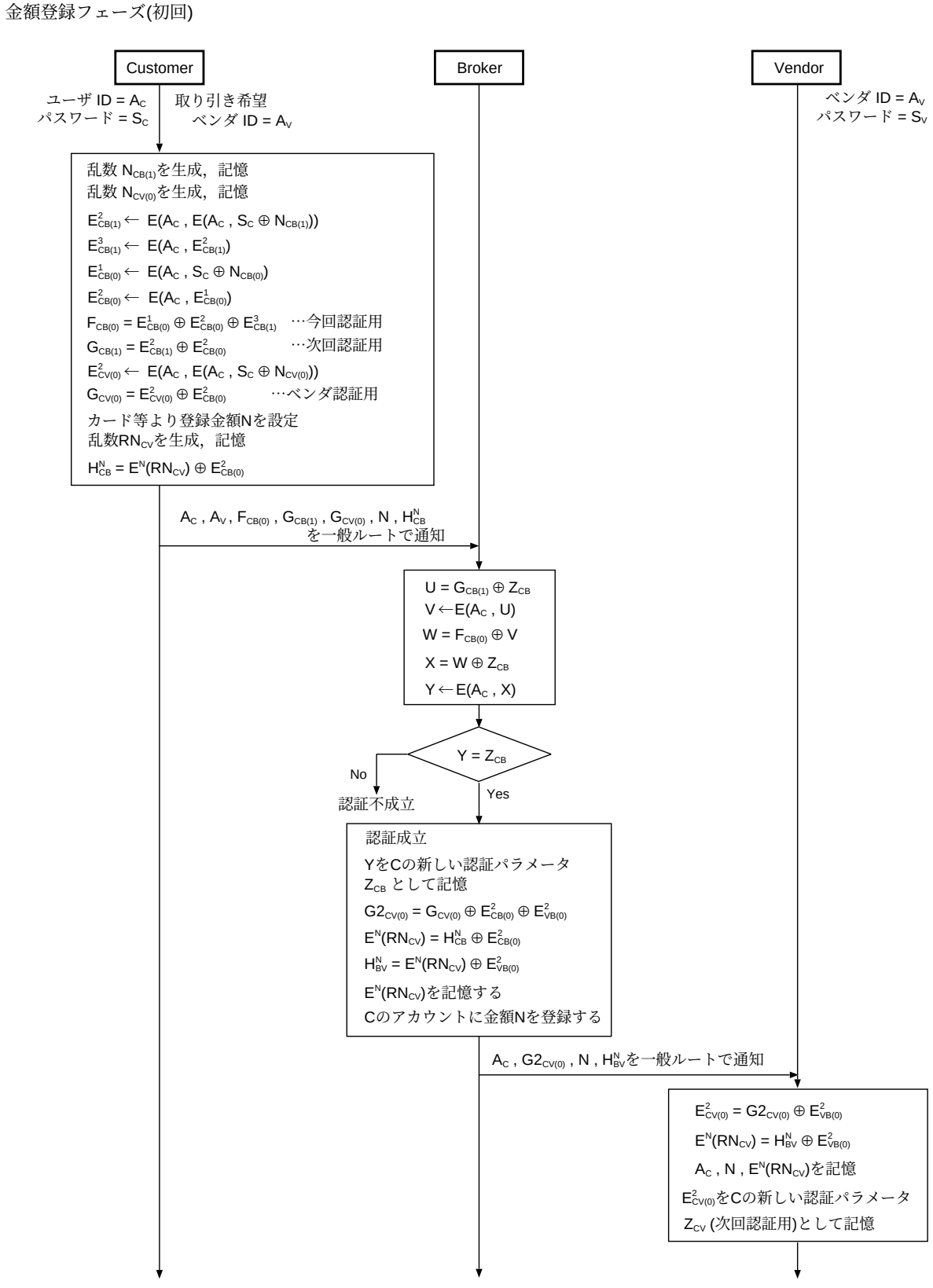

図 6.4　金額登録 (初回) のフロー

6.2 実装内容

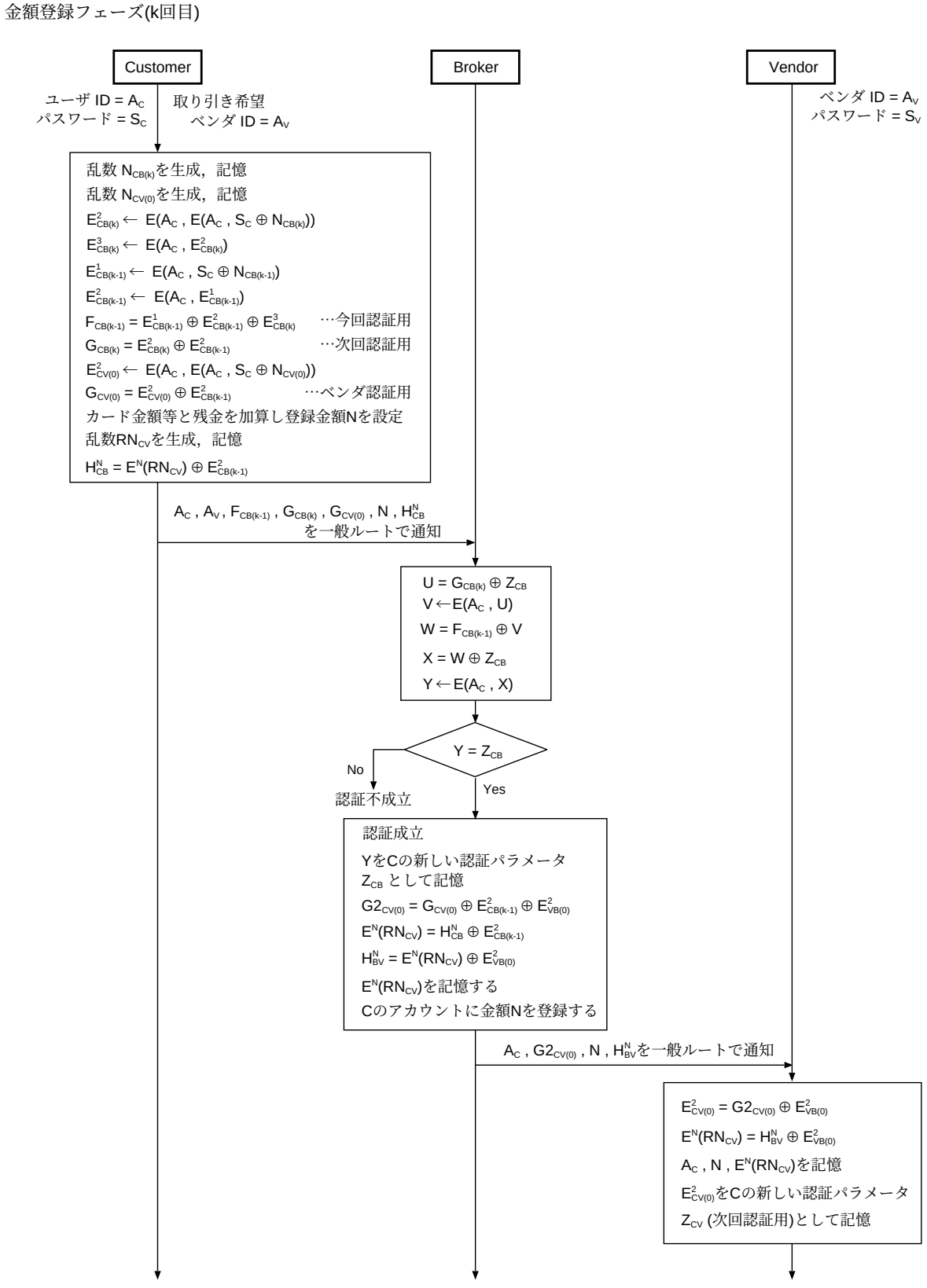

図 6.5　金額登録 (k 回目) のフロー

6.2　実装内容

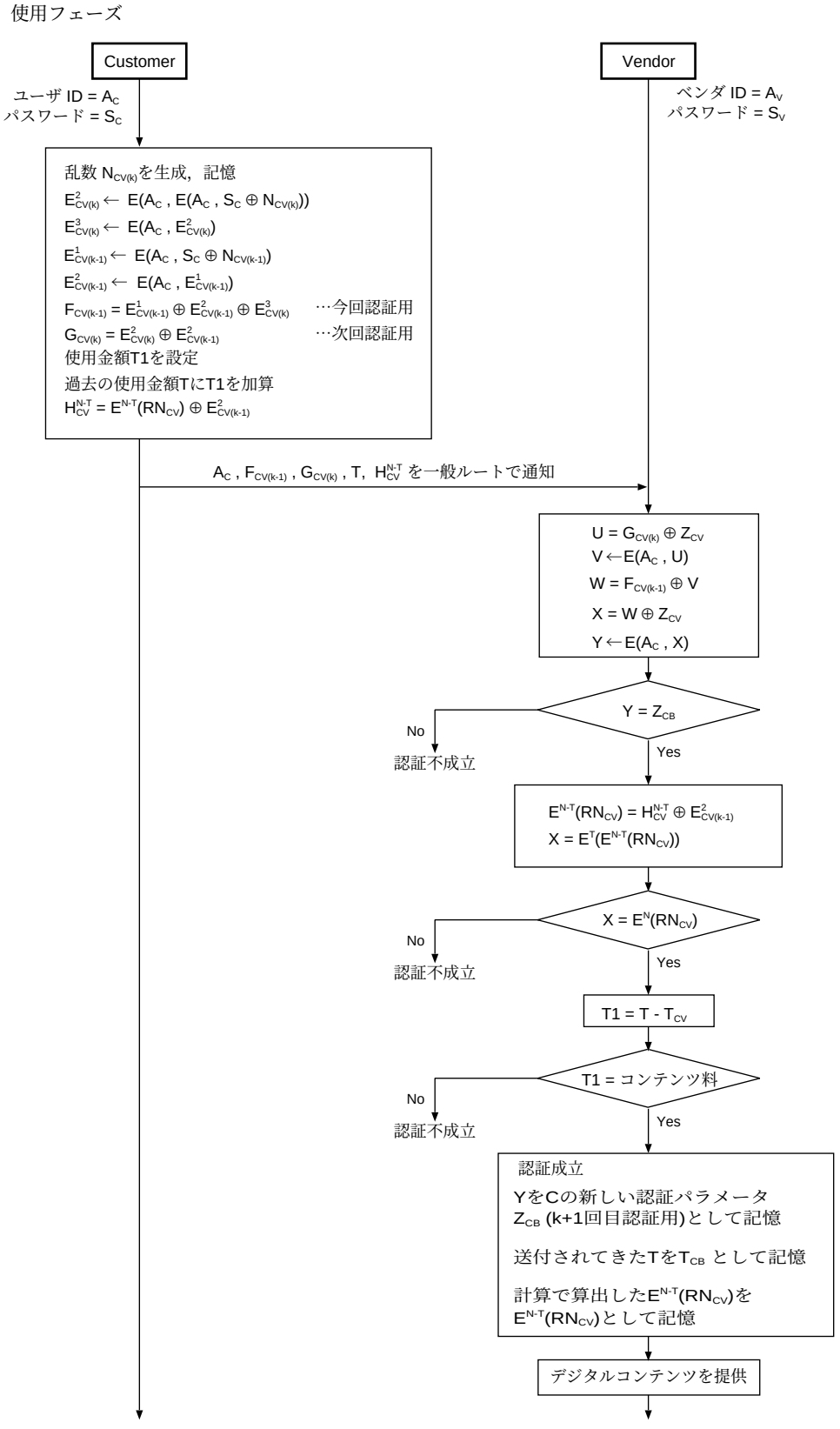

図 6.6　使用フェーズのフロー

6.2　実装内容

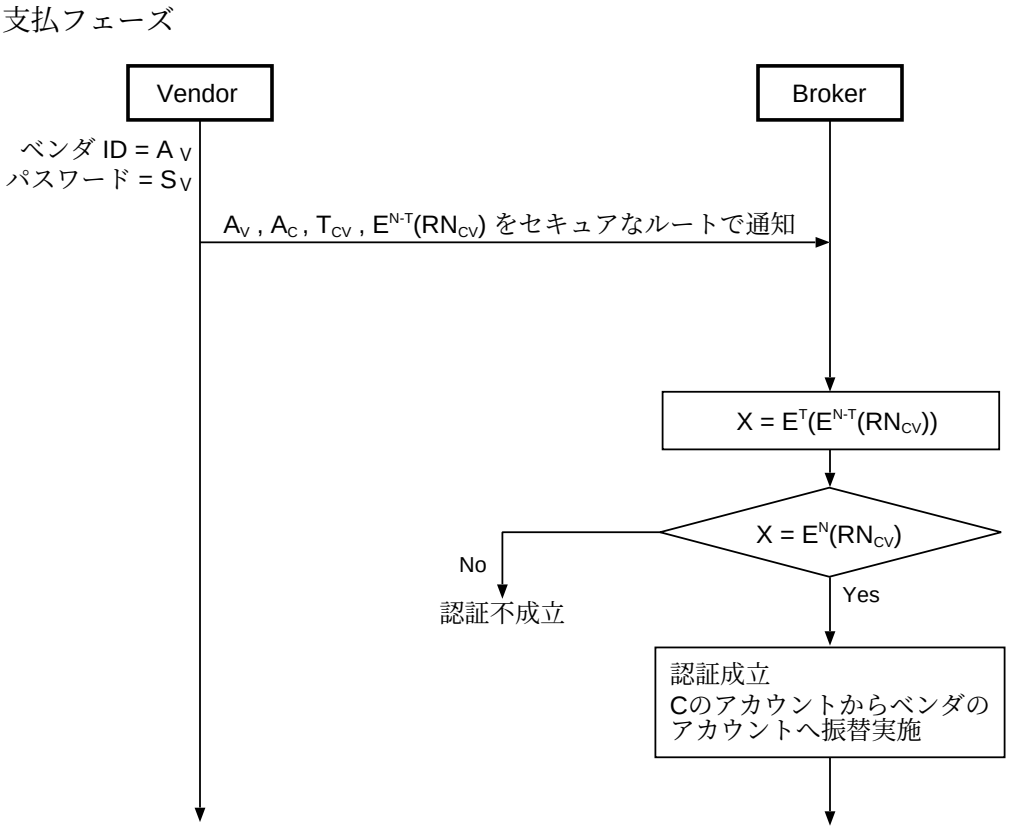

図 6.7　支払フェーズのフロー

第 7 章

今後の課題

本研究の目的は，清水研究室で提案している SAS パスワード認証方式，SAS をベースとした小額課金システムである SAS コインをモバイルコミュニケーションに適用する際の種々の問題点について考察し，最適な実装方式を確立することであった．

今回 SAS-K を提案することによって，セキュリティが十分保証された認証システムの実現が可能となった．SAS コインについても SAS-K を利用することにより，よりセキュリティの高い小額支払システムの提案ができた．そして，詳細な処理のフローを提示できた．従来の SAS コインの問題点に対しては，単位毎の計算を行うことによって，計算量の問題は解決できる見込みである．また，コイン追加の問題も図 6.5 の手順により，一つの解決案を提示できた．その他の問題については，さらにシステム全体を通して検討して行く必要がある．

以下に今後検討が必要であろう問題点を指摘する．

- 提案したコイン追加方法 (図 6.5) についての考察.
- Broker，Vendor 認証情報が更新されない，あるいは両者間での認証の必要性有無の問題.
- 支払いフェーズのロジックが不十分 (セキュアルートの利用必須).
- 利用金額に対して確認は残金額 (利用金額総計) でしかできない問題．また，アンマッチ時の対応がない.
- コイン生成時に，単位毎の計算を行うための，詳細な手順の提示.
- コイン送信の際の安全性についての考察.

提案方式，SAS-K，SAS コインをモバイルコミュニケーションに適用する際の最適な実装方式を確立するために，今後，これらの問題点について検討を進め，さらに改良を重ねていく必要がある．

第 8 章

むすび

本論文では，SAS に残されている問題点について考察し，SAS-K を提案した．この方式では，今回認証用データと次回認証用データを関連付けて認証を行うことにより，様々なアタックを防ぐことを可能とした．また同期問題についてはカウンタを利用することにより，確実に同期を取ることを可能とした．さらに，User 側に乱数生成装置や，記憶領域がない場合でも対応できる方法についても併せて提案した．この提案方式 SAS-K は従来の SAS と比べ，ほとんど追加コストを必要とせず，また，非常に実用的かつ堅牢な認証システムである．

さらに，SAS-K をベースとした小額課金システムである SAS コインについて考察し，これらの提案方式をモバイルコミュニケーション，特に携帯電話等の処理能力の低い端末に実装する際の最適な実装方式の確立を目指した．実装フィールドとして，Customer，Vendor，Broker の三者から構成される小額課金システムのプラットフォームを作成し，その環境に SAS-K を実装した．従来の SAS コインに SAS-K を適用することで，よりセキュリティの高い小額課金システムを提案できた．いくつかの問題点のうち，計算量の問題は単位毎の計算を行うことによって解決できる見込みである．また，コイン追加の問題も一つの解決案を提示できた．その他の問題点については，システム全体を通して検討して行く必要があり，今後の検討課題とする．

謝辞

高知工科大学工学部情報システム工学科 清水明宏助教授には，研究室配属以来，就職活動，卒業研究を含め，学生生活全般に渡って懇切なる御指導，貴重な御教示を賜わった．ここに深謝申し上げる．

また，NTT アドバンステクノロジの渋谷充喜氏には，SAS-K の検討から情報通信学会投稿予定，” 強力なパスワード認証方式の提案 (SAS-K)” の作成まで，有効な御助言を頂き，御指導賜わった．

本学大学院修士課程の井上富幸氏，情報システム工学科の林竜也氏には，SAS-K の検討にあたり，有益な御議論を頂いた．ここに記して謝意を表する．

参考文献

[1] A. Evans, W. Kantrowitz and E. Weiss, “A user authentication scheme not requiring secrecy in the computer,” Commun. ACM, 17, 8, pp. 437–442, 1974.

[2] R.Morris and K. Thompson, “Password security : A case history, “Unix Programmer’s Manual, Seventh Edition, 2B, 1979.

[3] L. Lamport, “Password authentication with insecure communications,” Commun.ACM, vol. 24, no. 11, pp. 770–772, 1981.

[4] A. Shimizu, “A dynamic password authentication method by one-way function,” IEICE Trans., vol. J73-D-I, no. 7, pp. 630–636, July 1990.

[5] A. Shimizu, T. Horioka, and H. Inagaki, “A password authentication method for contents communication on the internet,” IEICE Trans. Commun., vol. E81-B, no. 8, pp. 1666–1673, Aug. 1998.

[6] “Data Encryption Standard”, FIPS Publication, 46, NBS (1977)

[7] A. Shimizu and S. Miyaguchi, “Fast data encipherment algorithm FEAL,” IEICE Trans., vol. 170-D, no. 7, pp. 1413–1423, July. 1987.

[8] M. Sandirigama, A. Shimizu, M. Noda, “Simple and Secure Password Authentication Protocol (SAS),” IEICE Trans. Commun., vol. E83-B, no.6, pp. 1363–1365, June 2000.

[9] M. Sandirigama, A. Shimizu, M. Noda, “Simple and Secure Coin (SAS-Coin)–A Practical Micropayment System,” IEICE Trans. Fundamentals, vol. E83-A, no.12, pp. 2679–2688, Des. 2000.

[10] G.B.Purdy, “A high-security log-in procedure,” Commun. ACM, 17, 8, pp. 442–445, 1974.

[11] M. Matsui, “New block encryption algorithm MISTY,” Lecture Notes in Computer

Science, FSE 1977, pp. 54–68, 1977.

[12] 清水明宏/パスワード方式への FEAL の適用について, 情報処理学会全国大会講演論文集, Vol. 第 37 回昭和 63 年後期 Num. 1 pp. 211-212 (1988.09)

[13] N. Haller, "The S/Key(TM) one-time password system," Proc. of the Internet Society Symposium on Network and Distributed System Security, pp. 151–158, 1994.

[14] M. Manasse, "The Millicent Protocol for Inexpensive Electronic Commerce," http://www.w3.org/Conferences/WWW4/Papers/246, 1955.

[15] B. Cox, J.D. Tygar, and M. Sirbu, "NetBill Security and Transaction Protocol," http://www.ini.cmu.edu/netbill/pubs/Usenix.html, 1955.

[16] R. Anderson, C. Manifavas, and C. Sutherland, Netcard - A Practical Electronic Cash System, Computer Laboratory, Cambridge, 1955.

[17] R. Hauser, M. Steiner, and M. Waidner, Micro-Payments based on iKP, IBM Zurich, 1996.

[18] R. Rivest and A. Shamir, Payword and Micromint, MIT Laboratory for Computer Science, May 1996.

[19] A. Shimizu, "Public E-mail Messages Forwarding Services," IEICE Technical Report, OFS96-39, No. 380, pp. 19–24, 1996.

[20] T. Horioka, M. Toda and A. Shimizu, "E-mail Messages Forwarding Services," IEICE Technical Report, OFS97-39, No. 280, pp. 37–42, 1997.

[21] 大石，林，井上，清水，"強力なパスワード認証方式の提案 (SAS-K)" 電子情報通信学会投稿予定.

付録 A

SAS について

SAS(Simple And Secure) パスワード認証方式とは，被認証者から認証者への認証依頼毎にパスワード等の認証情報を変更して認証を行う方法である．そして SAS は，通信プロトコルに組み込める程，簡易なプロセスで認証を行うことができる方式で，携帯電話上を含めた，コンテンツ流通のプリペイド課金やマイクロペイメントシステムへの応用等が可能である．

SAS の手順は以下の通りである．

A.1 定義と記法

本稿で用いる SAS についての定義と記法は以下の通りである．

1. User を，認証用プロトコルを用いるコンピュータユーザとする．
2. Host を，ユーザを認証するサーバとする．
3. A を，ユーザの ID とする．
4. S を，ユーザのパスワードとする．
5. n を，認証セッション回数を表す 0 以上の整数とする．
6. N_n を，n 回目の認証に対応する乱数とする．
7. E を，暗号用のハッシュ関数とする．また E_n^m とは，N_n を使い，$(S \oplus N_n)$ が m 回ハッシュされたことを示す．
8. $\oplus$ は，ビットごとの排他的論理和を表す．

A.2　SAS の手順

このプロトコルは登録フェーズと認証フェーズからなる．登録は一度だけ行われ，認証は User がログインするたびに毎回行われる．その過程を図 A.1 に示している．

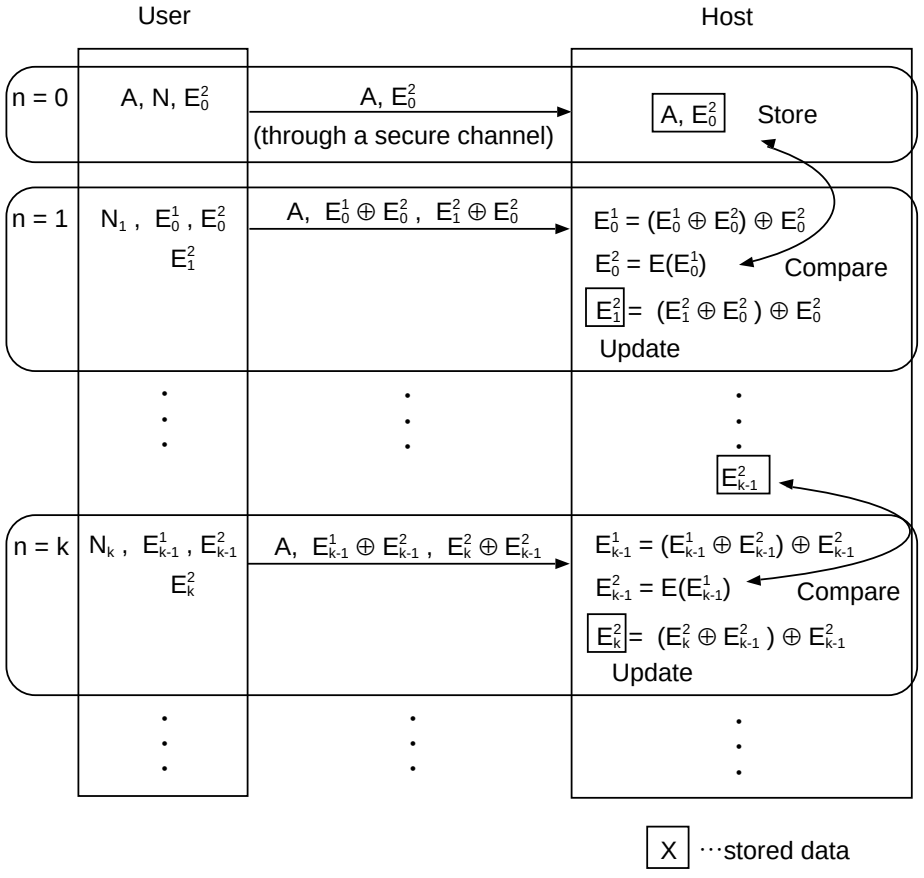

図 A.1　SAS の登録，および認証フェーズ

A.2.1　登録フェーズ $(n = 0)$

User: $E_0^2 = E^2(S \oplus N_0)$ を計算する．

User: A, E_0^2 を Host に安全なチャネルを用いて送信する．

Host: A, E_0^2 を 保存する．

A.2.2 認証フェーズ $(n = k)$

User: 以下のデータを計算し，ユーザ ID A と共に Host に送信する．

$E_k^1 \oplus E_k^2$, $E_{k+1}^2 \oplus E_k^2$

Host: 受信したデータと，保存されている E_k^2 との排他的論理和を取り，以下を取得する．

E_k^1, E_{k+1}^2

Host: E_k^1 にハッシュ関数を適用する．$E(E_k^1) = E_k^2$

保存されている E_k^2 と比較する．

マッチすれば，User は認証され，次回認証フェーズのために，E_k^2 を E_{k+1}^2 に更新する．

アンマッチならば認証は拒否される．

A.3 SAS の評価

Host 側では，User 毎に記憶しておかなければならないデータは，E_k^2 のただ 1 つのみであり，認証フェーズ毎に Host で実行しなくてはならないハッシュ関数の適用は，1 回のみである．これは従来の方式 [3, 4, 5] と比べ，極めて軽い処理負荷を実現している．また，User 側では，ハッシュ関数の適用回数は，4 回 (今回の認証用中間データ E_{k-1}^1, 今回の認証データ E_{k-1}^2, 次回の認証用中間データ E_k^1, 次回の認証データ E_k^2) であり，処理は十分に軽くてすむ．さらに，User と Host の間で行われる情報の授受は，認証フェーズ毎に User から Host への送信が 1 回のみであるため，通信セッション (コネクション) の状態が不安定なネットワークにおいても確実に認証処理を行うことができる．

付録 B

SAS コインについて

本節では，SAS をベースとした支払システム SAS コインを紹介する．

このモデルには 3 種の関与者が存在する．Broker(B)，Vendor(V)，Customer(C) である．Broker は Vendor と Customer のアカウントを立ち上げ，維持し，金銭取り引きを行う．

B.1　定義と記法

本稿で用いる SAS コインについての定義と記法は以下の通りである．

1. X ： プリペイドカードに書かれているシリアル番号 (異なるユーザに対応するサブスクリプトをつける)
2. Y ： プリペイドカードに書かれているマスク処理された乱数 (異なるユーザに対応するサブスクリプトをつける)
3. $E^2_{nCB} = E^2(S_C \oplus N_{nCB})$ ： Customer の Broker に対する n 番目のデータ
4. $E^2_{nCV} = E^2(S_C \oplus N_{nCV})$ ： Customer の Vendor に対する n 番目のデータ
5. $E^2_{nVB} = E^2(S_V \oplus N_{nVB})$ ： Vendor の Broker に対する n 番目のデータ

B.2　SAS コインの手順

B.2.1　初期データ配分

初期データの配分には，プリペイドカードを用いた安全な方法を使用する．Broker は初期データ配分のためにカードを販売する．カードのシリアル番号 X をユーザ ID として使用する．(図 B.1 参照)

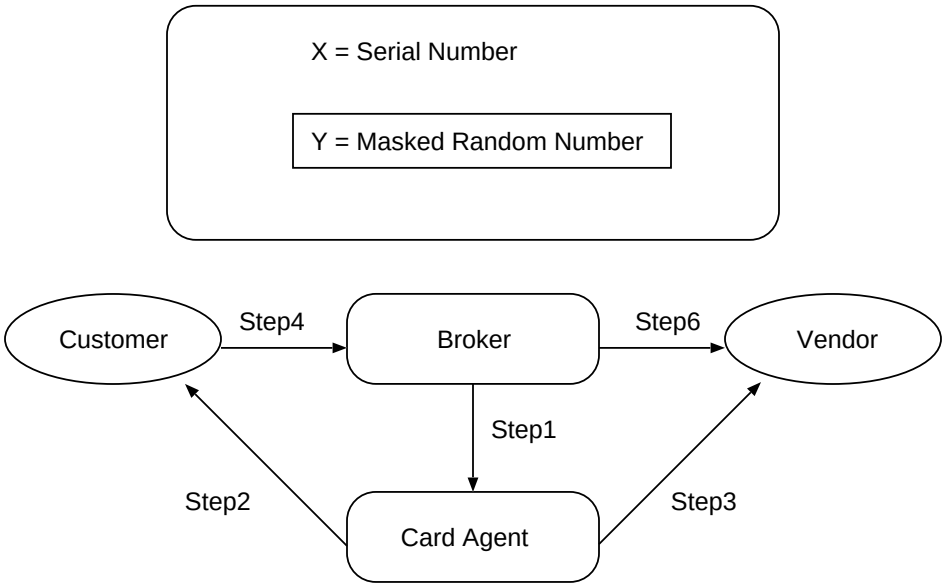

図 B.1　初期データ配分

1. Broker ： カードを製造し，コンビニエンスストア等に分配する．
2. Customer ： カードを購入する．
3. Vendor ： カードを購入する．
4. Customer ： $X_C, E^2_{0CB} \oplus Y_C$ を Broker に送信する．
5. Broker ： E^2_{0CB} を取得する．
6. Vendor ： $X_V, E^2_{0VB} \oplus Y_V$ を Broker に送信する．
7. Broker ： E^2_{0VB} を取得する．

B.2.2　コイン生成

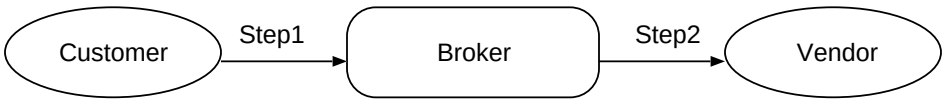

図 B.2　コイン生成

1. Customer ： 乱数 (RN) を生成，$E^n(RN)$ を計算し，以下のデータを Broker に送信する．
   - X_C, X_V : カスタマ ID と ベンダ ID
   - $E^2_{0CB} \oplus E^n(RN), n$: コインとコイン内の単位数 n
   - $E^2_{0CB} \oplus E^1_{0CB}$: 今回認証用
   - $E^2_{0CB} \oplus E^2_{1CB}$: 次回認証用
   - $E^2_{0CB} \oplus E^2_{0CV}$: ベンダ用認証データ
2. Broker ： SAS を用いて認証し，次回認証用に，E^2_{0CB} を E^2_{1CB} に置き換える．そしてカスタマのアカウントに n 単位振り込み，以下をベンダに送信する．
   - $E^2_{0VB} \oplus E^n(RN)$
   - $E^2_{0VB} \oplus E^2_{0CV}, n, X_c$
3. Vendor ： $E^n(RN)$ と E^2_{0CV} を取得し，n, X_C と共に保存する．

B.2.3　支払フェーズ

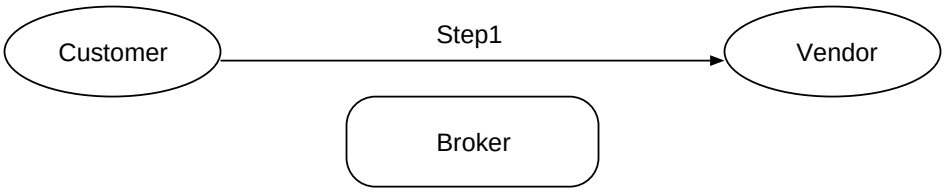

図 B.3　支払フェーズ

1. Customer：以下をベンダに送信する．
   - $X_C, E^2_{0CV} \oplus E^1_{0CV}$：今回認証用
   - $E^2_{0CV} \oplus E^2_{1CV}$：次回認証用
   - $E^{n-t}(RN)$：t 単位支払
2. Vendor：E^2_{0CV} を取得し，SAS を用いて認証し，次回認証用に E^2_{0CV} を E^2_{1CV} に置き換える．そして，$E^{n-t}(RN)$ に t 回ハッシュをかけ，$E^n(RN)$ でチェックする．正しければ情報を提供する．

   注) 必要であれば情報を暗号化するためのセッション鍵として E^1_{0CV} を使うことができる．

B.2.4　コインの回収

1. Vendor：最終支払 ($E^{n-t}(RN)$ または RN) 受領後ブローカに提示する．
2. Broker：最終支払 ($n-t$ 回または n 回) にハッシュをかけ，ベンダのアカウントを検証し，そこから引き落とす．

   カスタマのアカウントが空きの場合，カスタマは金を預ける際の ID として X を使い，匿名性を確保できる．

B.2.5　マルチベンダシステム

本システムではベンダ毎にコインが必要である．これまでの方式の多く ([14, 15, 16, 17, 18]) にこの特性がある．これは，銀行，新聞，天気予報会社のように定期的にアクセスされるプロバイダについては問題でない．定期的に利用しないベンダに対して，マルチベンダシステムを提案する．このシステムでは，ブローカ–ベンダ間にコイン認証エージェント (CAA) を採用している．システムの概要は図 B.4 の通りである．

このシステムでは，ベンダ (A, B, C 等) が一つの CAA グループを形成し，そのグループ (A, B, C 等) 固有のコインが CAA に送られる．このコインを使えばカスタマはグループ内

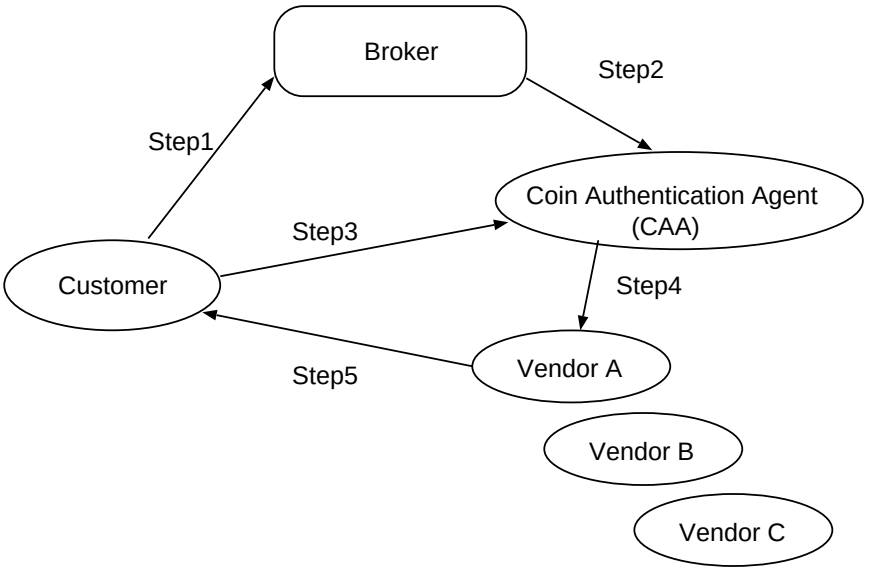

図 B.4　マルチベンダシステム

のどのベンダからでも買物をすることができる．

1. Customer：認証情報と $E^n(RN)$ をブローカに送る．
2. Broker：カスタマのアカウントの貸方に記入した後，認証情報と $E^{n-t}(RN)$ をベンダ自身 (A, B, C 等) ではなく CAA に送る．
3. Customer：カスタマがあるベンダに支払いたいと思えば，利用したいベンダ名，認証情報および $E^{n-t}(RN)$ をベンダ自身ではなく CAA に送る．
4. CAA：コインをチェックし，情報送出命令を特定のベンダに送る
5. Vendor：情報をカスタマに送る．

B.3　SAS コインの評価

B.3.1　信用

従来のいずれの支払システムでも，Broker は長期投資しており，信用を失いたくないため，信用できる．Broker は Customer と Vendor の両方と長い付き合いを持つ．次に信用のおけるのは Vendor である．Vendor も長期投資し，Broker と長い付き合いがある．

Vendor は Customer とは長期の付き合いを持たない場合がある．しかしながら，Vendor は Broker ほど信用が高くない．一番信用が低いのは，システムに出入りする Customer である．Customer は Broker とは多少の期間付き合う可能性はあるが，大抵の場合 Vendor とは一時的な関係しか持たない．それゆえ，従来のどのシステムでも Customer を一番に用心し，次に Vendor，最後に Broker の順に注意を払うべきである．

以下では SAS コインがこの状況下でどのような動きを取るか見てみる．実際，SAS コインは最低限の信用を必要とするか，あるいは全く必要としない．

- Customer：　Customer はコインを作成し，それを Broker へ送る．すると，Broker が Customer のアカウントの貸方に記入する．そのため，Customer は支払いを逃れることができない．Customer が Broker と良好な関係である場合は，コインをクレジットベースで作ることさえできる．Customer は n よりも低く Broker に申告することができるが，この n は Broker から Vendor へ送られるため，Customer は虚偽の申告をすることができない．

  Vendor が $E^{n-t}(RN)$ の最終値を持っているため，Customer は複数回使用することによって Vendor をごまかすことはできない．さらに，各 Customer が持つ $E^{n-t}(RN)$ は独自のパスワード S に基づく認証データと共に保存されているため，他の Customer になりすますこともできない．Vendor が最終支払いの $E^{n-t}(RN)$ を知っているということは，すなわち Customer がそれを Vendor に送ったことになるため，Customer は後日いかなる支払いも否定することはできない．
- Vendor：　Vendor は Customer がそれを使用するまで最終支払い $E^{n-t}(RN)$ が分からないので，Customer に不当な支払いを請求できない．
- Broker：　Broker は最初の支払いを実現するまでに最終支払 $E^{n-t}(RN)$ を受け取らなければならないため，Customer の口座から必要以上の金を引き出せない．さらに，Customer だけが乱数を知っているために Broker は Vendor と一緒になって Customer のアカウントを操作することもできない．

Vendor と Broker ができることと言えば，外部に Customer の認証データを流出させることである．そうすることで外部者が自身のパスワードに基づいて自身の認証データを投入し，通信回線を傍受して一回だけ支払いまで受けることができる．このためのコストを考えると，わずかな支払いを盗んでも割が合わない．それ以前に，Broker は信用ができ，Vendor も争いにより，小銭を失いたくないであろう．

B.3.2　安全性

Customer と Broker 間の取り引きの場合，すべての通信は，SAS を利用するため，極めて高いセキュリティが保証される．また，コインがデータ送信中に重複して使われたり，盗まれたりすることはない．Customer は支払いを逃れることはできず，いかなる者も他人をだますことはできない．

付録 C

プログラム

以下に示すソースコードは，今回作成した小額課金プラットフォームに SAS-K を適用したものである．

```
/*/*/*/*/*/*/* 以下は，関数群のヘッダ (sas.h)*/*/*/*/*/*/*/*/*/*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define BUFSIZE      1000
#define DATALEN      9
#define DEFAULT_PORT_center_user 8000
#define DEFAULT_PORT_provider_center 8001
#define DEFAULT_PORT_provider_user 8002
#define DEFAULT_PORT 5320
#define MEMBER 30

typedef struct userdata
{
  char uID[BUFSIZE];
  char E2x[BUFSIZE];
  int zannkinn;
}udata;

struct sockaddr_in settei(int port, unsigned long address);
int serverstart(int port);
int clientstart(int port, unsigned long address);
int makekey(unsigned char *key);
int cipher(unsigned char *block);
int sbox(unsigned char data);
void int_to_char(int rn, char *a);
void exor(unsigned char *data1, unsigned char *data2);
void makeE1x(unsigned char *uID, unsigned char *S,
unsigned char *N0);
void loadfile(udata *ap, int *m, char *fn);
void savefile(udata *ap, int *m, char *fn);
int searchmember(char n[20], udata *ap, int m);
```

```
/*/*/*/*/*/*/*/*/*/*/* 以下は，関数群 (sas.c)*/*/*/*/*/*/*/*/*/*/
```

```
#include "sas.h"

unsigned char KS[32];

/*-------------makekey-------------*/
makekey(key)
     unsigned char *key;
{
  unsigned char t[4],b[4],d[4];
  register i,j;

  for ( i = 0 ; i < 4 ; i++ )
    d[i] = 0;

  for ( i = 0 ; i < 8 ; i++ ) {
    for ( j = 0 ; j < 4 ; j++ ){
      t[j] = key[j+4];
      b[j] = key[j+4] ^ d[j];
      d[j] = key[j];
    }
    key[5] = key[1] ^ key[0];
    key[6] = key[2] ^ key[3];
    key[5] = sbox(key[5] + (key[6] ^ b[0]) + 1);
    key[6] = sbox(key[6] + (key[5] ^ b[1]));
    key[4] = sbox(key[0] + (key[5] ^ b[2]));
    key[7] = sbox(key[3] + (key[6] ^ b[3]) + 1);
    for ( j = 0 ; j < 4 ; j++ )
      key[j] = t[j];
    for ( j = 0 ; j < 4 ; j++ )
      KS[4*i+j] = key[j+4];
  }
}

/*-------------cipher-------------*/
cipher(block)
     unsigned char *block;
{
  unsigned char t[4];
  register i,j;
  int k;

  k = 16;

  for ( i = 0 ; i < 8 ; i++ )
    block[i] = block[i] ^ KS[k++];

  for ( i = 0 ; i < 4 ; i++ )
    block[i+4] = block[i+4] ^ block[i];

  k = 0;

  for ( i = 0 ; i < 8 ; i++ ) {
    for ( j = 0 ; j < 4 ; j++ )
      t[j] = block[j+4];
```

```
    block[5] = block[5] ^ block[4] ^ KS[k];
    block[6] = block[6] ^ block[7] ^ KS[k+1];
    block[5] = sbox(block[5] + block[6] + 1);
    block[6] = sbox(block[6] + block[5]);
    block[4] = sbox(block[4] + block[5]);
    block[7] = sbox(block[7] + block[6] + 1);
    for ( j = 0 ; j < 4 ; j++ )
      block[j+4] = block[j+4] ^ block[j];
    for ( j = 0 ; j < 4 ; j++ )
      block[j] = t[j];
    k += 2;
  }

  for ( i = 0 ; i < 4 ; i++ )
    t[i] = block[i+4];
  for ( i = 0 ; i < 4 ; i++ )
    block[i+4] = block[i] ^ block[i+4];
  for ( i = 0 ; i < 4 ; i++ )
    block[i] = t[i];

  k = 24;

  for ( i = 0 ; i < 8 ; i++ )
    block[i] = block[i] ^ KS[k++ ];
}

/*-------------sbox-------------*/
sbox(data)
     unsigned char data;
{
  return ( data << 2 | data >> 6 );
}

/*-------------exor-------------*/
void exor(unsigned char *data1, unsigned char *data2)
{
  int i, small, big;

  if (strlen(data1) <= strlen(data2)){
    small = strlen(data1);
    big = strlen(data2);
    for (i = 0; i < small; i++)
      data1[i] = data1[i] ^ data2[i];
    for (i = small; i < big; i++)
      data1[i] = data2[i];
  }
  else{
    small = strlen(data2);
    big = strlen(data1);
    for (i = 0; i < small; i++)
      data1[i] = data1[i] ^ data2[i];
    for (i = small; i < big; i++)
      data1[i] = data1[i];
  }
}

/*-------------int_to_char-------------*/
void int_to_char(int rn, char *N)
{
  int n[10];

  n[0] = rn / 1000000000;
  rn = rn % 1000000000;
  n[1] = rn / 100000000;
  rn = rn % 100000000;
  n[2] = rn / 10000000;
  rn = rn % 10000000;
  n[3] = rn / 1000000;
  rn = rn % 1000000;
  n[4] = rn / 100000;
  rn = rn % 100000;
  n[5] = rn / 10000;
  rn = rn % 10000;
  n[6] = rn / 1000;
  rn = rn % 1000;
  n[7] = rn / 100;
  rn = rn % 100;
  n[8] = rn / 10;
  rn = rn % 10;
  n[9] = rn;

  N[0] = n[0] + n[9];
  N[1] = n[1] + n[8];
  N[2] = n[2] + n[7];
  N[3] = n[3] + n[6];
  N[4] = n[4] + n[5];
}

/*-------------settei--------------*/
struct sockaddr_in settei(int port, unsigned long address){

  struct sockaddr_in server;

  /* server no settei */
  memset((char *) &server, 0, sizeof(server));
  server.sin_family      = AF_INET;
  server.sin_addr.s_addr = address;
  server.sin_port        = htons(port);

  return server;
}

/*-------------serverstart-------------*/
int serverstart(int port)
{
  struct sockaddr_in server, client;
  int s, rs, len;

  /*socket open*/
  if (( s = socket(AF_INET, SOCK_STREAM, 0)) < 0){
    perror("socket");
    exit(1);
  }

  /*provider-center bind*/
  if (bind(s, (struct sockaddr *) &server,
sizeof(server)) < 0){
    perror("bind");
    exit(1);
  }

  listen(s, 1);
  len = sizeof(client);

  /*provider-center TCP connection no uketuke*/
  if ((rs = accept(s, (struct sockaddr *) &client, &len)) < 0){
    perror("accept");
    exit(1);
  }

  close(s);  /* socket close */
  return rs;
}

/*-------------clientstart-------------*/
int clientstart(int port, unsigned long address){
```

```
  struct sockaddr_in server;
  int s, len;

  /* socket open */
  if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror( "socket");
    exit(1);
  }

  if (connect(s, (struct sockaddr *) &server,
sizeof(server)) < 0) {
    perror("connect");
    exit(1);
  }

  return s;
}

/*-------------makeE1x-------------*/
void makeE1x(unsigned char *uID, unsigned char *S,
unsigned char *N0)
{
 makekey(uID);
 exor(S, N0);
 cipher(S);
}
```

/*/*/*/*/* 以下は，Broker プログラムのヘッダ (center.h)*/*/*/*/*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define BUFSIZE      1000
#define DATALEN      9
#define DEFAULT_PORT_center_user 8000
#define DEFAULT_PORT_provider_center 8001
#define DEFAULT_PORT_provider_user 8002
#define DEFAULT_PORT 5320
#define MEMBER 30

typedef struct userdata
{
  char uID[BUFSIZE];
  char E2x[BUFSIZE];
  int zannkinn;
}udata;

struct sockaddr_in settei(int port, unsigned long address);
int serverstart(int port);
int clientstart(int port, unsigned long address);
int makekey(unsigned char *key);
int cipher(unsigned char *block);
int sbox(unsigned char data);
void int_to_char(int rn, char *a);
void exor(unsigned char *data1, unsigned char *data2);
void makeE1x(unsigned char *uID, unsigned char *S,
unsigned char *N0);
void loadfile(udata *ap, int *m, char *fn);
void savefile(udata *ap, int *m, char *fn);
int searchmember(char n[20], udata *ap, int m);
void registration_c_u(void);
void registration_c_p(void);
void comunication_c(void);
void start_c(void);

extern int rs1, rs2, n;
extern char buf[BUFSIZE];
extern struct sockaddr_in user, provider;
extern int pn1, pn2, n, port_p_c, port_c_u;
extern char ip[20];
extern unsigned long myaddress, address;
extern int argc2;
extern char *argv2[];
extern unsigned char uID[DATALEN], uID2[DATALEN];
extern unsigned char pID[DATALEN], pID2[DATALEN];
extern unsigned char E2xuc[DATALEN], E2xpc[DATALEN];
extern unsigned char konnkai[BUFSIZE], jikai[BUFSIZE],
kennsyou[BUFSIZE];
extern unsigned char ninnsyoudata[BUFSIZE], kekka[BUFSIZE];
extern int member, zannkinn, hit;
```

/*/*/*/*/*/*/* 以下は，Broker の main 関数 (center.c)*/*/*/*/*/*/*/

```
#include "center.h"

int rs1, rs2, n;
char buf[BUFSIZE];
struct sockaddr_in user, provider;
int pn1, pn2, n, port_p_c, port_c_u;
char ip[20];
unsigned long myaddress, address;
int argc2, i;
char *argv2[5];
unsigned char uID[DATALEN], uID2[DATALEN];
unsigned char pID[DATALEN], pID2[DATALEN];
unsigned char E2xuc[DATALEN], E2xpc[DATALEN];
unsigned char konnkai[BUFSIZE], jikai[BUFSIZE],
kennsyou[BUFSIZE];
unsigned char ninnsyoudata[BUFSIZE], kekka[BUFSIZE];
int member, zannkinn, hit;

int main(int argc, char *argv[])
{
  int menu;

  argc2 = argc;
  for (i = 0; i < argc2; i++)
    argv2[i] = argv[i];
  start_c();
  registration_c_p();

  do
    {
      read(rs2, buf, BUFSIZE);
      menu = atoi(buf);
      printf("menu = %d\n", menu);
      switch(menu)
{
case 1:
  registration_c_u();
  break;

case 2:
  comunication_c();
  break;

case 0:
  break;
```

```
default:
  printf("You inputed illegal menu number!\nAHO~!!\n");
  break;
}
    }
  while(menu != 0);

  close(rs1); /*connection no setudann*/
  close(rs2); /*connection no setudann*/
  exit(0);
}
```

/*/*/*/*/*/*/* 以下は，Broker 起動関数 (start_c.c)*/*/*/*/*/*/*/

```
#include "center.h"

void start_c(void)
{
  /*hikisuu no kennsa*/
  if (argc2 != 1) {
    fprintf(stderr, "Usage: %s \n", argv2[0]);
    exit(1);
  }

  /* port banngou no settei */
  port_p_c = DEFAULT_PORT_provider_center;
  myaddress = inet_addr("172.21.32.92");
  provider = settei(port_p_c, myaddress);
  rs1 = serverstart(port_p_c);

  strcpy(ip, inet_ntoa(provider.sin_addr));
  pn1 = (int)ntohs(provider.sin_port);
  printf("-------------------------------------------------\n");
  printf("Provider IP Address = %s Port Number =
%d\n", ip, pn1);
  printf("-------------------------------------------------\n");

  /* port banngou no settei */
  port_c_u = DEFAULT_PORT_center_user;
  user = settei(port_c_u, myaddress);
  rs2 = serverstart(port_c_u);

  strcpy(ip, inet_ntoa(user.sin_addr));
  pn2 = (int)ntohs(user.sin_port);
  printf("-------------------------------------------------\n");
  printf("User IP Address = %s Port Number =
%d\n", ip, pn2);
  printf("-------------------------------------------------\n");
}
```

/*/* 以下は，Broker への Vendor 登録関数 (registration_c_p.c)*/*/*/

```
#include "center.h"

void registration_c_p(void)
{
  int i;

  /*------------registration---------------*/
  printf("\n****Registration of Provider.****\n\n");
  read(rs1, ninnsyoudata, BUFSIZE); /*receive touroku you data*/
  printf("resieve data from provider\n");

  for (i = 0; i < 8; i++)
    pID[i] = ninnsyoudata[i];

  for (i = 0; i < 8; i++)
    E2xpc[i] = ninnsyoudata[i+8];

  /*strcpy(tourokudata[member].uID, uID);
  strcpy(tourokudata[member].E2xpc, E2xpc);
  tourokudata[member].zannkinn = zannkinn;
  member++;
  savefile(ap, &member, FILENAME);*/

  printf("\nProvider Registration was finished!! \n");
}
```

/*/*/* 以下は，Broker への Vendor 登録関数 (registration_c_u.c)*/*/*/

```
#include "center.h"

void registration_c_u(void)
{
  int i;

  /*------------registration---------------*/
  printf("\n****Registration of User.****\n\n");
  read(rs2, ninnsyoudata, BUFSIZE);  /*receive touroku you data*/
  printf("resieve data from user\n");

  for (i = 0; i < 8; i++)
    uID[i] = ninnsyoudata[i];

  for (i = 0; i < 8; i++)
    E2xuc[i] = ninnsyoudata[i+8];

  /*strcpy(tourokudata[member].uID, uID);
  strcpy(tourokudata[member].E2xuc, E2xuc);
  tourokudata[member].zannkinn = zannkinn;
  member++;
  savefile(ap, &member, FILENAME);*/

  printf("\nUser Registration was finished!! \n");
}
```

/*/*/*/*/* 以下は，Broker 認証関数 (comunication_c.c)*/*/*/*/*/*/*/

```
#include "center.h"

void comunication_c(void)
{
  int i;

  /*------------comunication---------------*/
  printf("\n****ninnsyou****\n\n");
  read(rs2, ninnsyoudata, BUFSIZE); /*receive ninnsyou you data*/
  printf("receive data\n");

  for (i = 0; i < 8; i++)
    uID[i] = ninnsyoudata[i];
  strcpy(uID2, uID);

  for (i = 0; i < 8; i++)
    konnkai[i] = ninnsyoudata[i+8];

  for (i = 0; i < 8; i++)
    jikai[i] = ninnsyoudata[i+16];

  printf("uID = %s\n", uID);

  makekey(uID);
  exor(jikai, E2xuc);
```

```
  cipher(jikai);
  exor(konnkai, jikai);
  exor(konnkai, E2xuc);
  cipher(konnkai);

  if (strcmp(E2xuc, konnkai) == 0){
    kekka[0] = 1;
    write(rs2, kekka, 1);     /*send kekka to user*/
    write(rs1, kekka, 1);
    printf("send OK\n");
    strcpy(E2xuc, jikai);
    printf("\n****comunication****\n\n");

    /*n = read(rs1, buf, BUFSIZE);
      printf("(2)  recieve \"%s\" from user\n", buf);

      n = read(0, buf, BUFSIZE);
      write(s2, buf, n);
      printf("(3)  send to provider \"%s\"\n", buf);

      n = read(s2, buf, BUFSIZE);
      printf("(6)  recieve \"%s\" from provider\n", buf);

      n = read(0, buf, BUFSIZE);
      write(rs1, buf, n);
      printf("(7)  send to user \"%s\"\n", buf);
    */
  }
  else{
    kekka[0] = 0;
    write(rs2, kekka, 1);     /*send kekka to user*/
    write(rs1, kekka, 1);
    printf("send NG\n");
  }
}
```

/*/*/*/*/*/* 以下は，Vendor プログラムのヘッダ (provider.h)*/*/*/*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define BUFSIZE      1000
#define DATALEN      9
#define DEFAULT_PORT_center_user 8000
#define DEFAULT_PORT_provider_center 8001
#define DEFAULT_PORT_provider_user 8002
#define DEFAULT_PORT 5320
#define MEMBER 30

typedef struct userdata
{
  char uID[BUFSIZE];
  char E2x[BUFSIZE];
  int zannkinn;
}udata;

struct sockaddr_in settei(int port, unsigned long address);
int serverstart(int port);
int clientstart(int port, unsigned long address);
int makekey(unsigned char *key);
int cipher(unsigned char *block);
int sbox(unsigned char data);
void int_to_char(int rn, char *a);
void exor(unsigned char *data1, unsigned char *data2);
void makeE1x(unsigned char *uID, unsigned char *S,
unsigned char *N0);
void loadfile(udata *ap, int *m, char *fn);
void savefile(udata *ap, int *m, char *fn);
int searchmember(char n[20], udata *ap, int m);
void registration_p(void);
void comunication_p(void);
void start_p(void);

extern int s1, rs2, n;
extern char buf[BUFSIZE];
extern struct sockaddr_in center, user;
extern int port_p_c, port_p_u, pn1, pn2;
extern char ip1[20], ip2[20];
extern unsigned long myaddress, address;
extern int argc2, i;
extern char *argv2[5];
extern unsigned char pID[DATALEN], pID2[BUFSIZE];
extern unsigned char S[DATALEN], S2[DATALEN];
extern unsigned char N1pc[5], N0pc[5];
extern unsigned char E10pc[DATALEN], E20pc[DATALEN];
extern unsigned char E11pc[DATALEN], E21pc[DATALEN];
extern int rn, content_no;
```

/*/*/*/*/* 以下は，Vendor の main プログラム (provider.c)*/*/*/*/*/

```
#include "provider.h"

int s1, rs2, n;
char buf[BUFSIZE];
struct sockaddr_in center, user;
int port_p_c, port_p_u, pn1, pn2;
char ip1[20], ip2[20];
unsigned long myaddress, address;
int argc2, i;
char *argv2[5];
unsigned char pID[DATALEN], pID2[BUFSIZE];
unsigned char S[DATALEN], S2[DATALEN];
unsigned char N1pc[5], N0pc[5];
unsigned char E10pc[DATALEN], E20pc[DATALEN];
unsigned char E11pc[DATALEN], E21pc[DATALEN];
int rn, content_no;

int main(int argc, char *argv[])
{
  argc2 = argc;
  for (i = 0; i < argc2; i++)
    argv2[i] = argv[i];

  start_p();

  registration_p();

  comunication_p();

  close(s1);  /*socket close*/
  close(rs2); /*connection no setudann*/
  exit(0);
}
```

/*/*/*/*/*/*/* 以下は，Vendor 起動関数 (start_p.c)*/*/*/*/*/*/*/*/

```
#include "provider.h"
```

```
void start_p(void)
{
  /*hikisuu no kennsa*/
  if (argc2 != 2) {
    fprintf(stderr, "Usage: %s center_hostname \n", argv2[0]);
    exit(1);
  }

  /* port banngou no settei */
  port_p_c = DEFAULT_PORT_provider_center;

  /* center IP addres no settei */
  if ((address = inet_addr(argv2[1])) == INADDR_NONE) {
    struct hostent *he;

    if ((he = gethostbyname(argv2[1])) == NULL) {
      herror("gethostbyname");
      exit(1);
    }
    memcpy((char *) &address, (char *)he->h_addr, he->h_length);
  }

  center = settei(port_p_c, address);
  s1 = clientstart(port_p_c, address);

  strcpy(ip1, inet_ntoa(center.sin_addr));
  pn1 = (int)ntohs(center.sin_port);
  printf("-------------------------------------------------\n");
  printf("Center IP Address = %s Port Number = %d\n", ip1, pn1);
  printf("-------------------------------------------------\n");

  /* port banngou no settei */
  port_p_u = DEFAULT_PORT_provider_user;
  myaddress = inet_addr("172.21.32.92");
  user = settei(port_p_u, myaddress);
  rs2 = serverstart(port_p_u);

  strcpy(ip2, inet_ntoa(user.sin_addr));
  pn2 = (int)ntohs(user.sin_port);
  printf("-------------------------------------------------\n");
  printf("User IP Address = %s Port Number = %d\n", ip2, pn2);
  printf("-------------------------------------------------\n");
}
```

/*/*/*/*/*/* 以下は，Vendor 登録関数 (registration_p.c)*/*/*/*/*/*/

```
#include "provider.h"

void registration_p(void)
{

  int i;

  /*------------registration---------------*/
  printf("\n****Provider Registration.****\n\n");

  printf("Input your Provider ID !\n");
  scanf("%s", pID);
  strcpy(pID2, pID);
  printf("Input your Password !\n");
  scanf("%s", S);

  rn = rand();

  /*fp = fopen(FILENAME, "w");
  fprintf(fp, "%d", rn);
  fclose(fp);*/

  int_to_char(rn, NOpc);

  for (i = 0; i < strlen(NOpc); i++)
    printf("NOpc[%d] = %d\n", i, NOpc[i]);

  makeE1x(pID, S, NOpc);                /*make E10*/
  strcpy(E10pc, S);

  cipher(S);
  strcpy(E20pc, S);                     /*make E20*/

  printf("\n");

  strcat(pID2, E20pc);

  write(s1, pID2, strlen(pID2));       /*send touroku you data*/
  printf("send data\n");
}
```

/*/*/*/*/*/*/* 以下は，Vendor 認証関数 (comunication_p.c)*/*/*/*/*/

```
#include "provider.h"

void comunication_p(void)
{
  int i;
  char content[BUFSIZE];

  do{
    /*------------comunication---------------*/
    printf("\n****comunication****\n\n");
    n = read(rs2, buf, BUFSIZE);
    printf("recieve content No. from user\n");
    printf("content No. = %s\n", buf);

    content_no = atoi(buf);

    n = read(s1, buf, BUFSIZE);
    for (i = 0; i < strlen(buf); i++)
      printf("kekka[%d] = %d\n", i, buf[i]);

    if (buf[0] = 1){
      switch(content_no)
{
case 1:(content, "\n\n\nA\nB\nC\nD\n");
  break;
case 2:
  strcpy(content, "\n\n\n\nLove\nLove\nLove\n\n");
  break;
case 3:
  strcpy(content, "\n\n\n\nYuzu ou!\n\n");
  break;
case 4:
  strcpy(content, "\n\n\n\nahhhaaaaann\n");
  break;
  }

      write(rs2, content, BUFSIZE); /*send centent to user*/
    }
  }while(1);

  /*n = read(rs1, buf, BUFSIZE);
    printf("(4)  recieve \"%s\" from center\n", buf);

    n = read(0, buf, BUFSIZE);
```

```
    write(rs1, buf, n);
    printf("(5)  send to center\"%s\"\n", buf);

    n = read(rs2, buf, BUFSIZE);
    printf("(10)  recieve \"%s\" from center\n", buf);

    n = read(0, buf, BUFSIZE);
    write(rs2, buf, n);
    printf("(11)  send to center\"%s\"\n", buf);
  */
}
```

/*/*/*/*/* 以下は，Customer プログラムのヘッダ (user.h)*/*/*/*/*/*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define BUFSIZE      1000
#define DATALEN      9
#define DEFAULT_PORT_center_user 8000
#define DEFAULT_PORT_provider_center 8001
#define DEFAULT_PORT_provider_user 8002
#define DEFAULT_PORT 5320
#define MEMBER 30

typedef struct userdata
{
  char uID[BUFSIZE];
  char E2x[BUFSIZE];
  int zannkinn;
}udata;

struct sockaddr_in settei(int port, unsigned long address);
int serverstart(int port);
int clientstart(int port, unsigned long address);
int makekey(unsigned char *key);
int cipher(unsigned char *block);
int sbox(unsigned char data);
void int_to_char(int rn, char *a);
void exor(unsigned char *data1, unsigned char *data2);
void makeE1x(unsigned char *uID, unsigned char *S,
unsigned char *N0);
void loadfile(udata *ap, int *m, char *fn);
void savefile(udata *ap, int *m, char *fn);
int searchmember(char n[20], udata *ap, int m);
void registration_u(void);
void comunication_u(void);
void start_u(void);

extern int s1, s2, n;
extern char buf[BUFSIZE];
extern struct sockaddr_in center, provider;
extern int port_c_u, port_p_u, pn1, pn2;
extern char ip1[20], ip2[20];
extern unsigned long address1, address2;
extern int argc2;
extern char *argv2[5];
extern unsigned char uID[DATALEN], uID2[BUFSIZE];
extern unsigned char S[DATALEN], S2[DATALEN];
extern unsigned char N1uc[5], NOuc[5];
extern unsigned char E10uc[DATALEN], E20uc[DATALEN];
extern unsigned char E11uc[DATALEN], E21uc[DATALEN],
E31uc[DATALEN];
extern unsigned char data[BUFSIZE];
extern unsigned char kekka[BUFSIZE];
extern int rn, member;
```

/*/*/*/*/*/*/* 以下は，Customer の main 関数 (user.c)*/*/*/*/*/*/*/

```
#include "user.h"

int s1, s2, n;
char buf[BUFSIZE];
struct sockaddr_in center, provider;
int port_c_u, port_p_u, pn1, pn2;
char ip1[20], ip2[20];
unsigned long address1, address2;
int argc2, i;
char *argv2[5];
unsigned char uID[DATALEN], uID2[BUFSIZE];
unsigned char S[DATALEN], S2[DATALEN];
unsigned char N1uc[5], NOuc[5];
unsigned char E10uc[DATALEN], E20uc[DATALEN];
unsigned char E11uc[DATALEN], E21uc[DATALEN], E31uc[DATALEN];
unsigned char data[BUFSIZE];
unsigned char kekka[BUFSIZE];
int rn, member;

int main(int argc, char *argv[])
{
  int menu, content_no;

  argc2 = argc;
  for (i = 0; i < argc2; i++)
    argv2[i] = argv[i];
  start_u();
  do
    {
      printf("\n#------This is SAS_SYSTEM program.------#\n\n");
      printf("#1 : Registration.\n");
      printf("#2 : Comunication.\n");
      printf("#0 : Exit.\n");
      scanf("%d",&menu);

      switch(menu)
{
case 1:
  write(s1, "1", BUFSIZE);   /*send menu no to center.*/
  registration_u();
  break;
case 2:
  printf("****Conect to Provider****\n");
  printf("****and Browsing****\n\n");
  printf("****Which contents do you want to get?****\n");
  printf("#1 : One day of Ryuya\n");
  printf("#2 : Chikotto Love\n");
  printf("#3 : Mr. Colonel Bandana\n");
  printf("#4 : eroero\n");
  scanf("%d",&content_no);

  switch(content_no)
    {
    case 1:
      write(s2, "1", BUFSIZE);
      printf("send content No. 1 to provider");
      break;
    case 2:
      write(s2, "2", BUFSIZE);
```

```
      printf("send content No. 2 to provider");
      break;
    case 3:
      write(s2, "3", BUFSIZE);
      printf("send content No. 3 to provider");
      break;
    case 4:
      write(s2, "4", BUFSIZE);
      printf("send content No. 4 to provider");
      break;
    case 0:
      write(s1, "0", BUFSIZE);
      break;
    default:
      printf("You inputed illegal menu number!\nAHO~!!\n");
      break;
    }
    write(s1, "2", BUFSIZE);  /*send menu no to center.*/
  comunication_u();
  break;
case 0:
  write(s1, "0", BUFSIZE);    /*send menu no to center.*/
  break;
default:
  write(s1, "9", BUFSIZE);    /*send menu no to center.*/
  printf("You inputed illegal menu number!\nAHO~!!\n");
  break;
}
    }
  while(menu != 0);

  close(s1);  /* socket close */
  close(s2);  /* socket close */
  exit(0);
}
```

/*/*/*/*/*/*/* 以下は，Customer 起動関数 (start_u.c)*/*/*/*/*/*/*/

```
#include "user.h"

void start_u(void)
{
  /*hikisuu no kennsa*/
  if (argc2 != 3) {
    fprintf(stderr, "Usage: %s sever_hostname
provider_hostname \n",
argv2[0]);
    exit(1);
  }

  /* port banngou no settei */
  port_c_u = DEFAULT_PORT_center_user;

  /* center IP addres no settei */
  if ((address1 = inet_addr(argv2[1])) == INADDR_NONE) {
    struct hostent *he;
    if ((he = gethostbyname(argv2[1])) == NULL) {
      herror("gethostbyname");
      exit(1);
    }
    memcpy((char *) &address1, (char *)he->h_addr, he->h_length);
  }

  center = settei(port_c_u, address1);
  s1 = clientstart(port_c_u, address1);

  strcpy(ip1, inet_ntoa(center.sin_addr));
  pn1 = (int)ntohs(center.sin_port);
  printf("-------------------------------------------------\n");
  printf("Center IP Address = %s Port Number = %d\n", ip1, pn1);
  printf("-------------------------------------------------\n");

  /* port banngou no settei */
  port_p_u = DEFAULT_PORT_provider_user;

  /* provider IP addres no settei */
  if ((address2 = inet_addr(argv2[2])) == INADDR_NONE) {
    struct hostent *he;
    if ((he = gethostbyname(argv2[2])) == NULL) {
      herror("gethostbyname");
      exit(1);
    }
    memcpy((char *) &address2, (char *)he->h_addr, he->h_length);
  }

  provider = settei(port_p_u, address2);
  s2 = clientstart(port_p_u, address2);

  strcpy(ip2, inet_ntoa(provider.sin_addr));
  pn2 = (int)ntohs(provider.sin_port);
  printf("-------------------------------------------------\n");
  printf("Provider IP Address = %s Port Number =
%d\n", ip2, pn2);
  printf("-------------------------------------------------\n");
}
```

/*/*/*/*/* 以下は，Customer 登録関数 (registration_u.c)/*/*/*/*/*/

```
#include "user.h"

void registration_u(void)
{
  int i;

  /*------------registration---------------*/
  printf("\n****User Registration****\n\n");
  printf("Input your User ID !\n");
  scanf("%s", uID);
  strcpy(uID2, uID);
  printf("Input your Passward !\n");
  scanf("%s", S);

  rn = rand();

  /*fp = fopen(FILENAME, "w");
  fprintf(fp, "%d", rn);
  fclose(fp);*/

  int_to_char(rn, NOuc);

  for (i = 0; i < strlen(NOuc); i++)
    printf("NOuc[%d] = %d\n", i, NOuc[i]);

  makeE1x(uID, S, NOuc);                /*make E10*/
  strcpy(E10uc, S);

  cipher(S);
  strcpy(E20uc, S);                     /*make E20*/

  printf("\n");

  strcat(uID2, E20uc);

  write(s1, uID2, strlen(uID2));       /*send touroku you data*/
```

```
  printf("send data\n");
}


/*/*/*/*/* 以下は，Customer 認証関数 (comunication_u.c)*/*/*/*/*/

#include "user.h"

void comunication_u(void)
{
  int i;

  /*------------ninnsyou---------------*/
  printf("\n****ninnsyou****\n\n");
  printf("Input your User ID !\n");
  scanf("%s", uID);
  strcpy(uID2, uID);
  printf("Input your Password !\n");
  scanf("%s", S);
  strcpy(S2, S);

  /*fp = fopen(FILENAME, "r");
  fscanf(fp, "%d", &rn);
  fclose(fp);*/

  int_to_char(rn, N0uc);
  printf("rn = %d\n", rn);

  for (i = 0; i < strlen(N0uc); i++)
    printf("N0uc[%d] = %d\n", i, N0uc[i]);

  makeE1x(uID, S, N0uc);              /*make E10*/
  strcpy(E10uc, S);

  cipher(S);                          /*make E20*/
  strcpy(E20uc, S);

  rn = rand();
  int_to_char(rn, N1uc);
  printf("rn = %d\n", rn);

  for (i = 0; i < strlen(N1uc); i++)
    printf("N1uc[%d] = %d\n", i, N1uc[i]);

  exor(S2, N1uc);
  cipher(S2);                         /*make E11*/
  strcpy(E11uc, S2);
  cipher(S2);                         /*make E21*/
  strcpy(E21uc, S2);
  cipher(S2);                         /*make E31*/
  strcpy(E31uc, S2);
  exor(E10uc, E20uc);
  exor(E10uc, E31uc);        /*konnkai no ninnsyou data*/
  exor(E21uc, E20uc);        /*jikai no ninnsyou data*/
  strcat(uID2, E10uc);
  strcat(uID2, E21uc);

  write(s1, uID2, strlen(uID2));     /*send ninnsyou you data*/
  printf("send data\n");

  read(s1, kekka, BUFSIZE);          /*receive kekka*/
  for (i = 0; i < strlen(kekka); i++)
    printf("kekka[%d] = %d\n", i, kekka[i]);

  printf("recieve %s\n", kekka);
  printf("strlen(kekka) =  %d\n", strlen(kekka));
  for (i = 0; i < strlen(kekka); i++)
    printf("kekka[%d] = %d\n", i, kekka[i]);

  if (kekka[0] == 1){
    /*fp = fopen(FILENAME, "w");
    fprintf(fp, "%d", rn);
    fclose(fp);*/

    printf("rn was over writed!\n");
    printf("\n****comunication****\n\n");

    read(s2, kekka, BUFSIZE);  /*receive content from provider*/
    printf("content = %s\n", kekka);

    /*n = read(0, buf, BUFSIZE);
    write(s1, buf, n);
    printf("(1)  send to center \"%s\"\n", buf);

    n = read(s1, buf, n);
    printf("(8)  recieve \"%s\" from center\n", buf);

    n = read(0, buf, BUFSIZE);
    write(s2, buf, n);
    printf("(9)  send to provider \"%s\"\n", buf);

    n = read(s2, buf, BUFSIZE);

    printf("(12)  recieve \"%s\" from center\n", buf);
    */
  }
}
```