平成 14 年度

学士学位論文

ワンタイムパスワード認証方式の高速化に関する検討

An examination of high-speed password authentication method

1030304　藤本 卓

指導教員　清水 明宏

2003 年 2 月 12 日

高知工科大学 情報システム工学科

要 旨

ワンタイムパスワード認証方式の
高速化に関する検討

藤本 卓

情報通信分野の発展，携帯電話の急速な普及に伴い，携帯電話を使用してのインターネット利用人口が増加している．現在様々なサービスが存在し，それらサービスにおいて資格認証が必要不可欠となっている．その認証技術のひとつである，ワンタイムパスワード認証方式は，ネットワーク上における盗聴や，盗難，紛失時のなりすましを防ぐ．その中でもSAS-2 (Simple And Secure password authentication protocol, ver.2) は，認証時の負荷，通信コストも少なく，モバイル端末には最適な認証方式である．SAS-2 ではハッシュ処理を用いているが，処理能力の低い端末においては，処理負荷の大きいハッシュ関数の適用回数はできるだけ少ないほうがよい．

本研究では，モバイル端末上での認証に視野を絞り，SAS-2 をベースに，ユーザ認証時の負荷を減少，高速化を行う新方式を提案する．そして，従来の SAS-2 の処理時間，CPU 負荷との比較, 評価を行った．

キーワード　ワンタイムパスワード，SAS-2，ハッシュ関数

Abstract

An examination of high-speed password authentication method

Suguru Fujimoto

People using portable phones and the Internet are increasing, because information communication's and portable phone's technologies advance. Then various services using such mobile phones are necessary to authenticate the user. One-time password authentication methods protect against tapping, steal, and impersonation in the Internet. SAS-2 (Simple And Secure password authentication protocol, ver.2) is most simple and secure one-time password authentication method, and the method is useful on mobile communications. However, SAS-2 use hash functions, which have calculation costs.

In this thesis, I propose a new method, which is SAS-2 based and removes the hash overhead. Therefore, this method is high speed. Moreover, I compare and evaluate that new method and SAS-2 about processing speed and calculation load.

key words one-time password, SAS-2, hash function

目次

図目次

表目次

第 1 章

はじめに

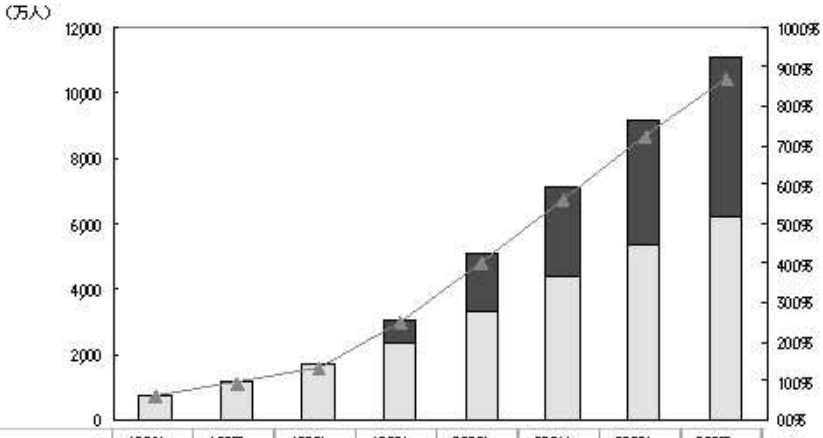
近年，携帯電話が急速に普及しており，通信手段として欠かせない存在となっている．ここ数年の間に性能，機能，共に急成長し，中でもインターネット接続システムは，企業にとって様々なサービスを生かすことのできる機能である．今現在においても，着信メロディや画像を有料で配信するなど，数多くのサービスが存在する．これらのサービスは，携帯電話の普及率からすれば，これからも増加していくのは間違いない．その中で，ユーザ，サービス提供者間には，セキュリティ確保の基本であるユーザ認証やセッション管理を，安全かつ容易に行える洗練された認証方式が必要不可欠である．その認証方式のひとつである，ワンタイムパスワード認証方式は，盗聴，盗難，紛失時のなりすましを防ぐことができる方式であり，その中でも，SAS-2 [5] , [6] は，認証時の通信量，コストも少ないため，携帯端末には最適の認証方式だといえる．そこで，ワンタイムパスワード認証方式，SAS-2 に着目し，さらなるユーザへの認証時の負担を減少，高速化の検討を行った．

第 2 章

研究背景

2.1　携帯電話利用によるインターネットの普及率

情報通信分野の発展に伴い，携帯電話が普及し，固定電話にはないその便利さや，基本料金の値下げ，サービスの充実等を理由に，その加入者数がここ数年で急激に増加している．また，携帯電話には，様々な機能が搭載されており，その中でもインターネット接続システムは，携帯電話の仕様からわかるように，いつどこでも好きなときに接続，情報を得ることができ，ユーザが楽しめる多種多様なサービスが存在し，その利用者は年々増加している．図 2.1 はインターネットの利用人口を表すグラフである．



	1996年	1997年	1998年	1999年	2000年	2001年	2002年	2003年
インターネット利用人口	783	1,155	1,694	3,132	5,126	7,177	9,192	11,075
(内訳)								
□固定網でのインターネット	783	1,155	1,694	2,442	3,359	4,374	5,378	6,268
■携帯電話インターネット				690	1,767	2,803	3,814	4,807
——人口普及率	6.2%	9.2%	13.4%	24.7%	40.4%	56.5%	72.2%	86.9%

(固定網) 1996~1998：郵政省通信白書平成11年版より（1998は推計）
1999~2003：情報通信総合研究所予測
(携帯電話) 情報通信総合研究所予測

図 2.1　固定網と携帯電話によるインターネット利用人口と人口普及率

図 2.1 からわかるように，固定網でのインターネット利用が普通であった 1996 年から 1998 年以降，携帯電話でのインターネット利用が加わり，その利用人口は年々増加，2003 年には固定網と並ぶであろうかという予測がたつ．携帯電話でのインターネットの利用が間違いなく増加しているのがわかるグラフである．　グラフから，携帯電話でのインターネット利用の著しい増加が伺えるが，それに伴い，サービスは，いっそう競争が激しくなっていくことは間違いない．その中で，ユーザとサービス提供者間には，不正を抑えるための洗練された認証方式が必要である．安全な認証方式であることはもちろん，ユーザのサーバへの集中等，考えられる事態も想定して，両者に負荷をできるだけかけないものでなければならない．そこで，モバイル端末においての認証方式として有効であると考えられるのがワンタイムパスワード認証方式である．

2.2　ワンタイムパスワード認証方式

ネットワークを経由したユーザ，サーバ間の操作では，盗聴されると危険な情報があふれている．盗聴を防ぐには，パスワードを盗めないようにするか，盗まれても使えないようにする必要がある．それが可能であり，ユーザ認証やセッション管理を，安全かつ容易に行える認証システムのひとつとして，ワンタイムパスワード認証方式があげられる．

ワンタイムパスワードとは，ユーザ認証の際にパスワードが毎回変わる，使い捨てパスワードであり，これを用いたワンタイムパスワード認証方式は，同じパスワードを使用しないので，盗聴でパスワードを知られた場合や，盗難，紛失時においてもパスワードは無効となり，なりすまし等を防ぐことが可能である．

このワンタイムパスワードを用いた代表的な認証方式として，Lamport [1] , [2] の方法，SAS [3] , [4] ，SAS-2 があげられる．これらを順に説明していく．

2.3 Lamport の方法

Lamport の方法とは，一方向性関数を複数回適用した認証情報を事前にサーバに登録しておき，認証時に登録している認証情報の一回前の認証情報を用いて認証を行い，ユーザが示した認証情報を次回の認証情報として置き換えるという方法である．以下に Lamport の処理手順を示す．

- ID：ユーザ ID
- S：ユーザのパスワード
- F^m：一方向性関数 (ハッシュ)

  例：F^1(a) は，a に 1 度ハッシュを適用．
- i：i 回目の認証
- $\oplus$：排他的論理和

2.3.1 登録フェーズ

1. ユーザは m を任意に設定し，$F^m(\mathrm{x})$ を生成．m を保存．

2. ユーザは，セキュアなルートを用いて $F^m(\mathrm{x})$ をサーバに送信．

3. サーバは，認証フェーズで使用するためのデータである $F^m(\mathrm{x})$ を登録．

2.3.2 認証フェーズ

1. ユーザは登録してある m を読み出し，$F^{m-i}(\mathrm{x})$ を計算．

2. ユーザは $F^{m-i}(\mathrm{x})$ を，インターネット等の一般的なネットワークを用いてサーバへ送信．

3. サーバは受信した $F^{m-i}(\mathrm{x})$ にハッシュをかけた $\mathrm{F}(F^{m-i}(\mathrm{x}))$ と，登録データである $F^{m-i+1}(\mathrm{x})$ との比較を行う．
   もし一致しなければ認証失敗，その時点で終了．
   一致した場合は認証成立，次のステップへと移行．

4. サーバは次の認証セッションのために，$F^{m-i}(\mathrm{x})$ を $F^{m-i+1}(\mathrm{x})$ の替わりに保存する．

以下に処理手順を図で簡単に説明する．これは例として m=1000 の場合を考えたものである．

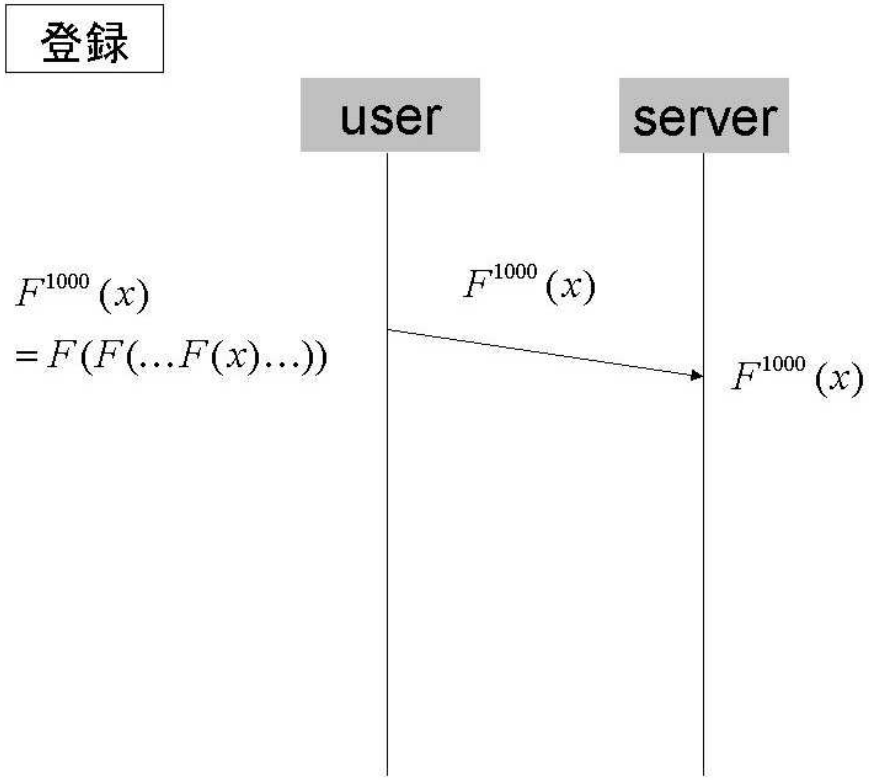


図 2.2 Lamport の方法における登録フェーズ

登録フェーズは図 2.2 のようになる．ユーザは m=1000 を用いて，認証情報 F^{1000}(x) を生成する．そして，F^{1000}(x) を，セキュアなルートでユーザからサーバへ送信し，サーバはこれを登録する．

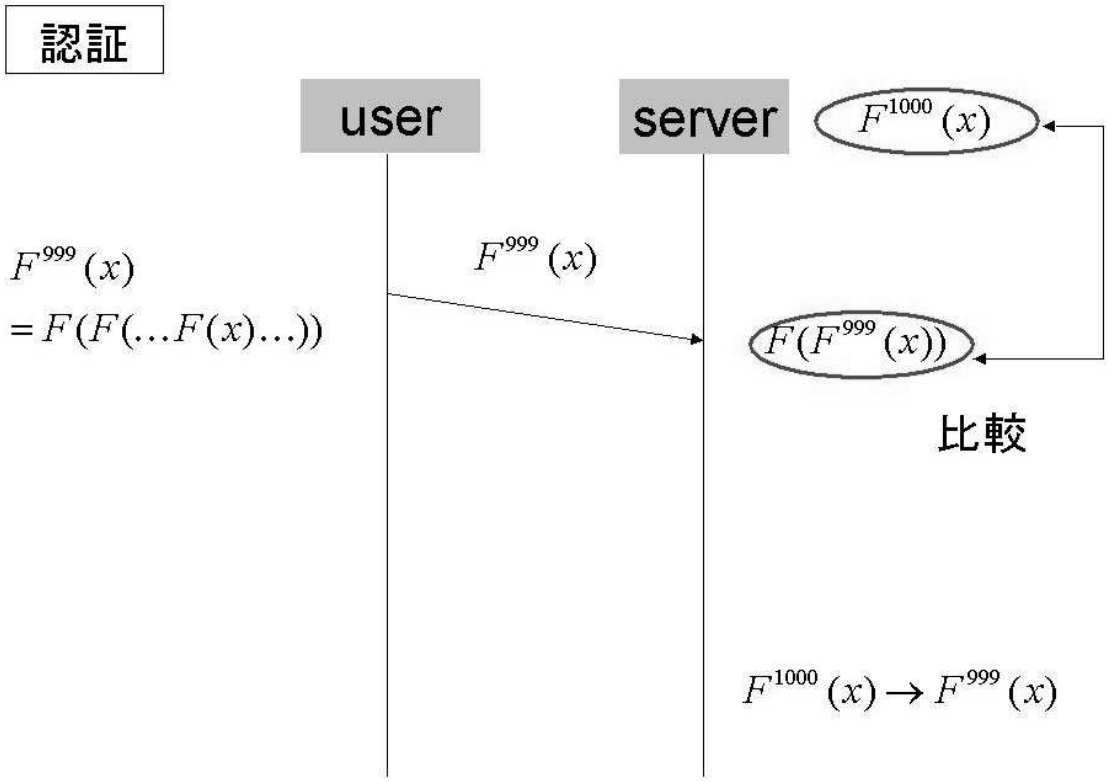


図 2.3 Lamport の方法における認証フェーズ

認証フェーズは図 2.3 のようになる．1 回目の認証と考える．まず m-i，つまり 1000-1=999 となり，ユーザは認証情報 F^{999}(x) を生成する．この認証情報をインターネット等の一般的なネットワークを用いて, ユーザからサーバへ送信する．

サーバは受信した認証情報 F^{999}(x) にハッシュをかけた F(F^{999}(x))=F^{1000}(x) と，登録時の認証情報 F^{1000}(x) とを比較し，一致すれば認証成立となる．また F^{999}(x) は，次回の認証情報として使用される．

この例では，ユーザの認証フェーズにおけるハッシュ回数は 999 回である．ハッシュ処理は負荷が大きく，例においてもハッシュ回数が非常に多いため，Lamport の方法では，大きな負荷がかかり，処理時間が非常にかかることが容易に予想できる．よって携帯電話等の低スペックマシンでの応用は望ましくない．そこで，この問題点を大きく改善した認証方式と

してあげられるのが SAS である.

2.4 SAS

SAS は，Lamport の方法における計算量とプログラムを比較しても，非常に小さく，通信上の盗聴による攻撃に強い認証方式である．この方式は，前回の認証フェーズで登録しておいた一方向性関数を二回適用した認証情報の元の情報と，次回認証用情報を今回認証用情報と次回認証用情報に一方向性関数を適用したものでマスクして送信し，認証と同時に次回認証用情報の正当性の検証と登録を行う.

以下に SAS の処理手順を示す．この手順は初期登録から最初の認証時のものである.

- ID : ユーザ ID
- S : ユーザのパスワード
- E : 一方向性関数 (ハッシュ)

  例：E^1(a) は，a に 1 度ハッシュを適用.
- N_1 : 1 回目の認証で用いられる乱数.
- $\oplus$: 排他的論理和

2.4.1 登録フェーズ

1. ユーザは ID 及び S を入力．同時に，乱数 N_1 を生成し，登録．その後，入力データ S 及び乱数を用いて $E_1^1 = \mathrm{E}(\mathrm{S} \oplus N_1)$ を計算．またその値にさらにハッシュをかけ $E_1^2 = \mathrm{E}(E_1^1)$ を計算．

2. ユーザは，セキュアなルートを用いて ID 及び E_1^2 をサーバに送信．

3. サーバは，認証フェーズで使用するためのデータである ID 及び E_1^2 を登録．

2.4.2 認証フェーズ

1. ユーザは ID 及び S を入力．入力データ S 及び登録している N_1 を用いて
   $E_1^1 = \mathrm{E}(\mathrm{S} \oplus N_1)$ を計算．またその値にさらにハッシュをかけ
   $E_1^2 = \mathrm{E}(E_1^1)$ を計算．次にユーザは乱数 N_2 を生成，登録．
   その後，入力データ S 及び乱数 N_2 を用いて
   $E_2^1 = \mathrm{E}(\mathrm{S} \oplus N_2)$, $E_2^2 = \mathrm{E}(E_2^1)$, $E_2^3 = \mathrm{E}(E_2^2)$ を計算．
   ここでの E_2^2 とは，次回の認証情報を意味する．
   さらに，$\alpha = E_2^2 \oplus E_1^2$ 及び $\beta = E_1^1 \oplus E_2^3$ の計算を行う．

2. ユーザは ID 及び α，β を，インターネット等の一般的なネットワークを用いて
   サーバへ送信．

3. サーバは，登録データである E_1^2 を用いて $\alpha \oplus E_1^2$ を行い，E_2^2 を取り出す．
   そして，その E_2^2 にハッシュをかけた $\mathrm{E}(E_2^2)$ を用いて $\beta \oplus \mathrm{E}(E_2^2)$ から E_1^1 を得る．
   次に，サーバは $\mathrm{E}(E_1^1)$ を計算して，登録データである E_1^2 との比較を行う．
   もし一致しなければ認証失敗，その時点で終了．
   一致した場合は認証成立．次のステップへと移行．

4. サーバは次の認証セッションのために，E_2^2 を E_1^2 の替わりに保存する．

以下に処理手順を図で簡単に説明する．

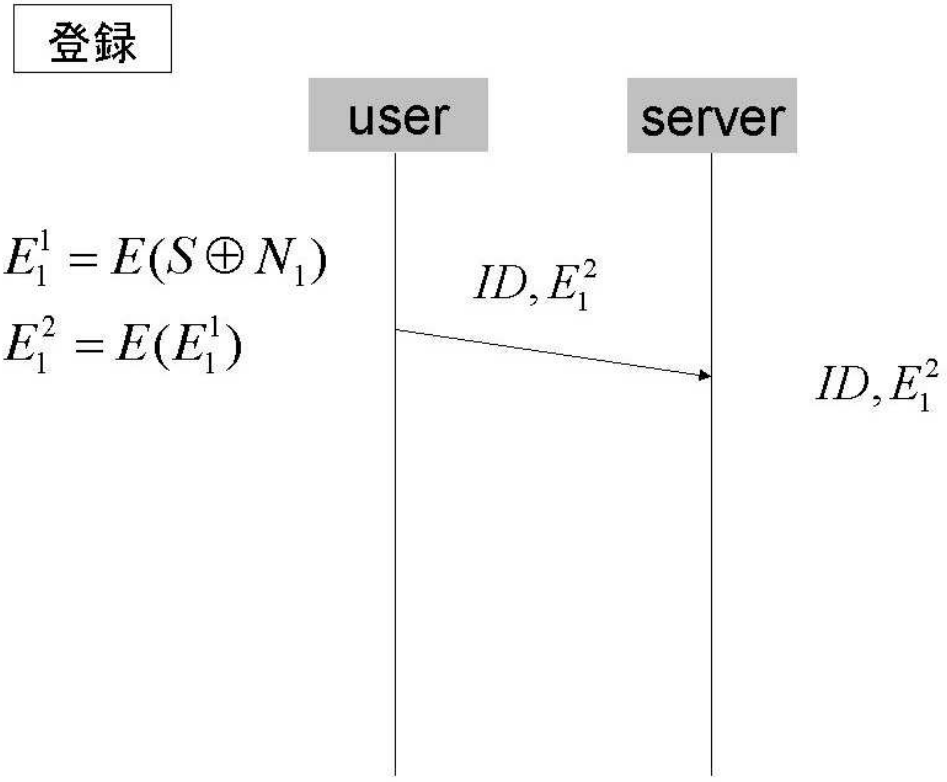


図 2.4 SAS における登録フェーズ

登録フェーズは図 2.4 のようになる．ユーザはパスワード S と乱数 N_1 の排他的論理和をとった情報にハッシュをかけ，認証情報 E_1^1 を生成し，その認証情報にさらにハッシュをかけ E_1^2 を生成する．ユーザ ID，E_1^2 を，セキュアなルートでユーザからサーバへ送信し，サーバはこの 2 つを登録しておく．

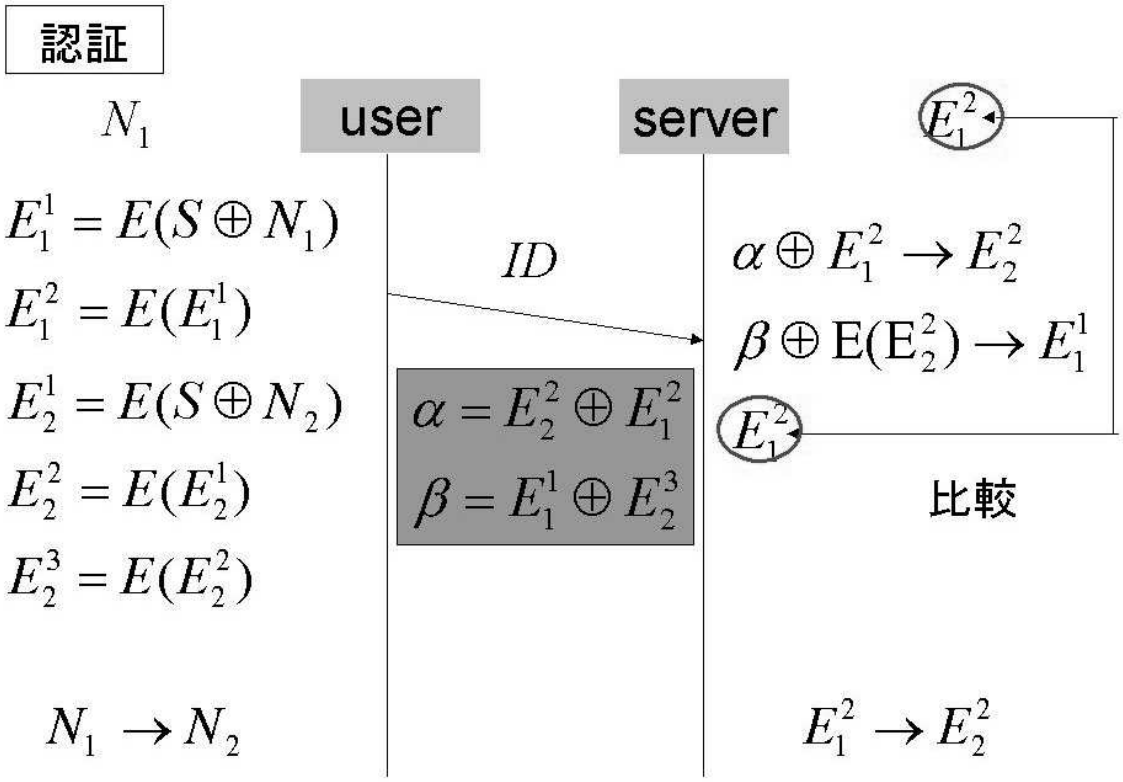


図 2.5　SAS における認証フェーズ

認証フェーズは図 2.5 のようになる．ユーザは登録時と同じ乱数を用いた認証情報 E_1^1，E_1^2，異なる乱数を用いた認証情報 E_2^1，E_2^1 に，さらにハッシュをかけた E_2^2，同様に E_2^3 を生成する．これらの認証情報から鍵 α，β を生成し，その鍵と ID をインターネット等の一般的なネットワークを用いて, ユーザからサーバへ送信する．

サーバは登録時の認証情報 E_1^2 と鍵を元に解読する．α，β，登録データから得られた E_1^1 にハッシュをかけた認証情報 $\mathrm{E}(E_1^1)$=E_1^2 と，登録している認証情報 E_1^2 とを比較し，一致すれば認証成立となる．また E_2^2 は，次回の認証情報として使用される．

ユーザの認証フェーズにおけるハッシュ回数は 5 回である．この回数を 3 回に削減したのが SAS-2 である．

2.5 SAS-2

SAS-2 は，ワンタイムパスワード認証方式の中でも，負荷，通信コストが少ないのが長所である．また，相互認証が可能であり，セキュリティ面においても安全性が高い．よって，SAS-2 は携帯端末には非常に相性の良い認証方式であるといえる．

SAS-2 の処理を説明する．SAS-2 の処理は，登録フェーズと認証フェーズからなる．登録フェーズでは，ユーザ情報をサーバに登録し，認証フェーズでは，認証情報を用いて，相互認証を行う．以下に SAS-2 の処理手順を示す．この手順は初期登録から最初の認証時のものである．

- ID : ユーザ ID
- S : ユーザのパスワード
- X : 一方向性関数 (ハッシュ)

  例 : X(a) は，a に 1 度ハッシュを適用．
- N_1 : 1 回目の認証で用いられる乱数．
- A_1 : 乱数 N_1 を用いた認証情報．
- $\oplus$: 排他的論理和
- $+$: 加算

2.5.1 登録フェーズ

1. ユーザは ID 及び S を入力．同時に，乱数 N_1 を生成し，登録．その後，入力データ及び乱数を用いて $A_1 = \mathrm{X}(\mathrm{ID},\mathrm{S} \oplus N_1)$ を計算．

2. ユーザは，セキュアなルートを用いて ID 及び A_1 をサーバに送信．

3. サーバは，認証フェーズで使用するためのデータである ID 及び A_1 を登録．

2.5.2 認証フェーズ

1. ユーザは ID 及び S を入力．入力データ及び登録している N_1 を用いて
   $A_1 = \mathrm{X}(\mathrm{ID},\mathrm{S} \oplus N_1)$ を計算．次にユーザは乱数 N_2 を生成，登録．
   その後，入力データ及び乱数 N_2 を用いて
   $A_2 = \mathrm{X}(\mathrm{ID},\mathrm{S} \oplus N_2)$, $\mathrm{X}(A_2) = \mathrm{X}(\mathrm{ID},A_2)$ を計算．
   ここでの A_2 とは，次回の認証情報を意味する．
   さらに，$\alpha = A_2 \oplus (\mathrm{X}(A_2) + A_1)$ 及び $\beta = \mathrm{X}(A_2) \oplus A_1$ の計算を行う．

2. ユーザは ID 及び α，β を，インターネット等の一般的なネットワークを用いてサーバへ送信．

3. サーバは，登録データである A_1 を用いて $\beta \oplus A_1$ を行い，$\mathrm{X}(A_2)$ を取り出す．
   そして，その $\mathrm{X}(A_2)$ を用いて $\alpha \oplus (\mathrm{X}(A_2) + A_1)$ から A_2 を得る．
   次に，サーバは $\mathrm{X}(\mathrm{ID},A_2)$ を計算して，取り出した $\mathrm{X}(A_2)$ との比較を行う．
   もし一致しなければ認証失敗，その時点で終了．
   一致した場合は認証成立，次のステップへと移行．

4. サーバは次の認証セッションのために，A_2 を A_1 の替わりに保存し
   $\gamma = \mathrm{X}(\mathrm{ID},\mathrm{X}(A_2))$ を計算．

5. サーバは，γ をインターネット等の一般的なネットワークを用いてユーザへ送信．

6. ユーザは X(ID,X(A_2)) を計算，受信データである γ との比較を行う．

   一致すれば，サーバ，ユーザ間の相互認証が成立．

以下に処理手順を図で簡単に説明する.

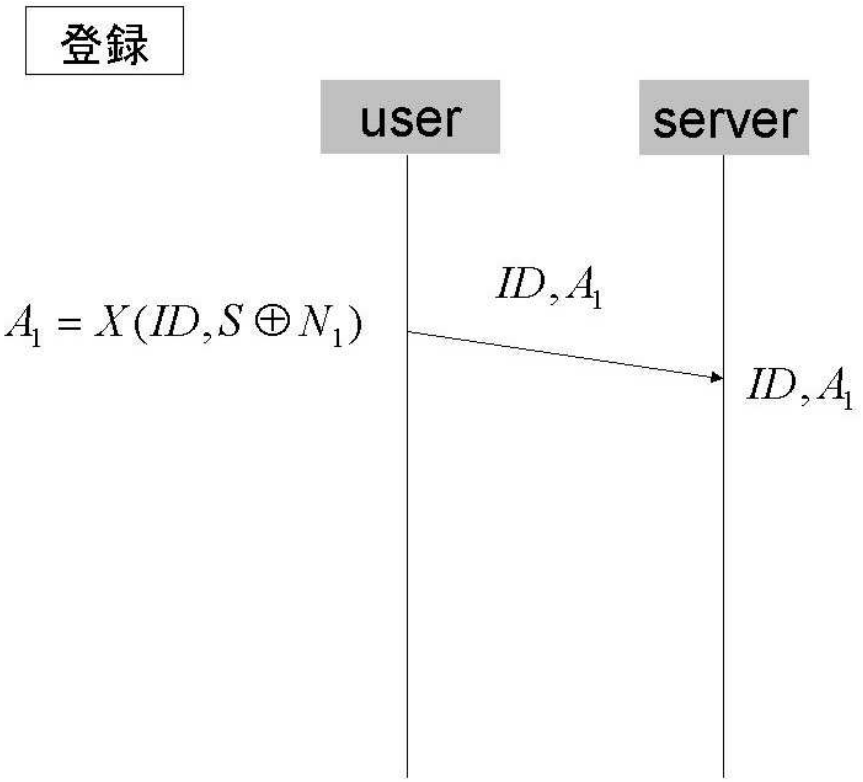


図 2.6 SAS-2 における登録フェーズ

登録フェーズは図 2.6 のようになる．ユーザは ID, パスワード S と乱数 N_1 の排他的論理和をとった情報にハッシュをかけ，認証情報 A_1 を生成する．ユーザ ID，A_1 を，セキュアなルートでユーザからサーバへ送信し，サーバはこの 2 つを登録しておく.

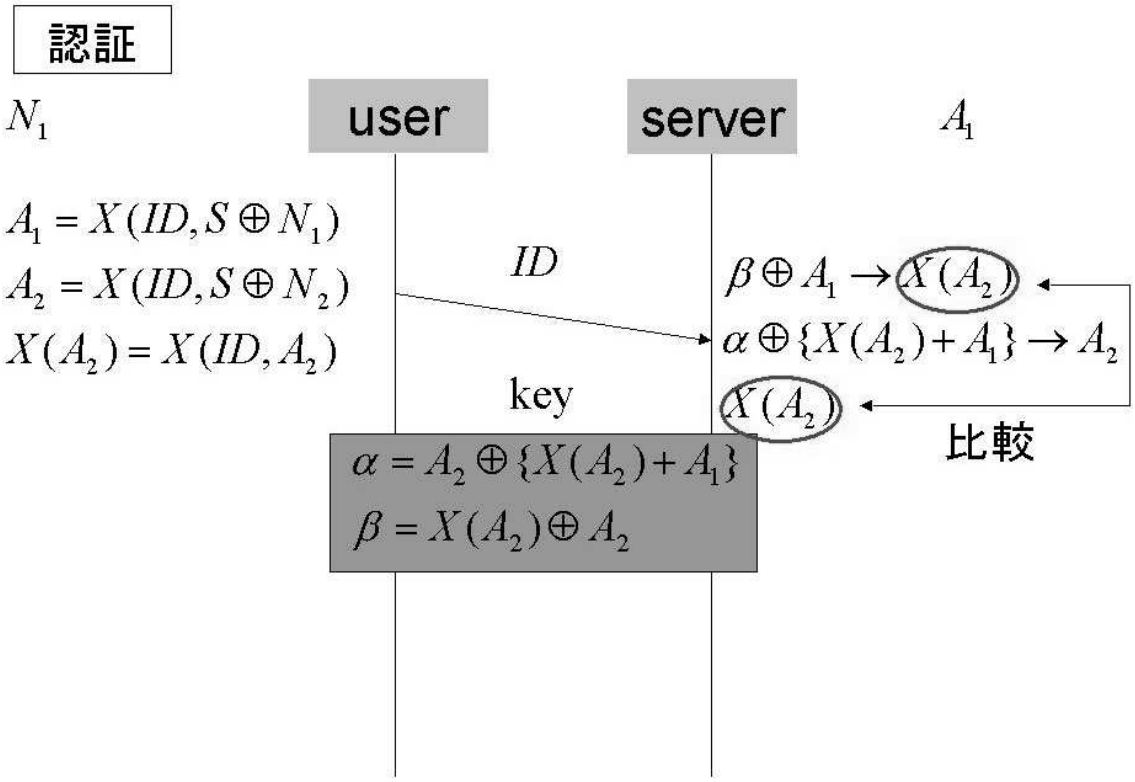


図 2.7 SAS-2 における認証フェーズ

認証フェーズは図 2.7 のようになる．ユーザは登録時と同じ乱数を用いた認証情報 A_1，異なる乱数を用いた認証情報 A_2，A_2 に，さらにハッシュをかけた認証情報 A_3 から鍵 α，β を生成し，その鍵と ID をインターネット等の一般的なネットワークを用いて, ユーザからサーバへ送信する．

サーバは登録時の認証情報 A_1 と鍵を元に解読する．$\beta \oplus A_1$ から取り出した，X(A_2) と，解読から得られた A_2 にハッシュをかけた X(A_2) とを比較し，一致すれば認証成立となる．また A_2 は，次回の認証情報として使用される．

注目する点は，ユーザの認証フェーズにおけるハッシュ回数であり，その回数は 3 回である．

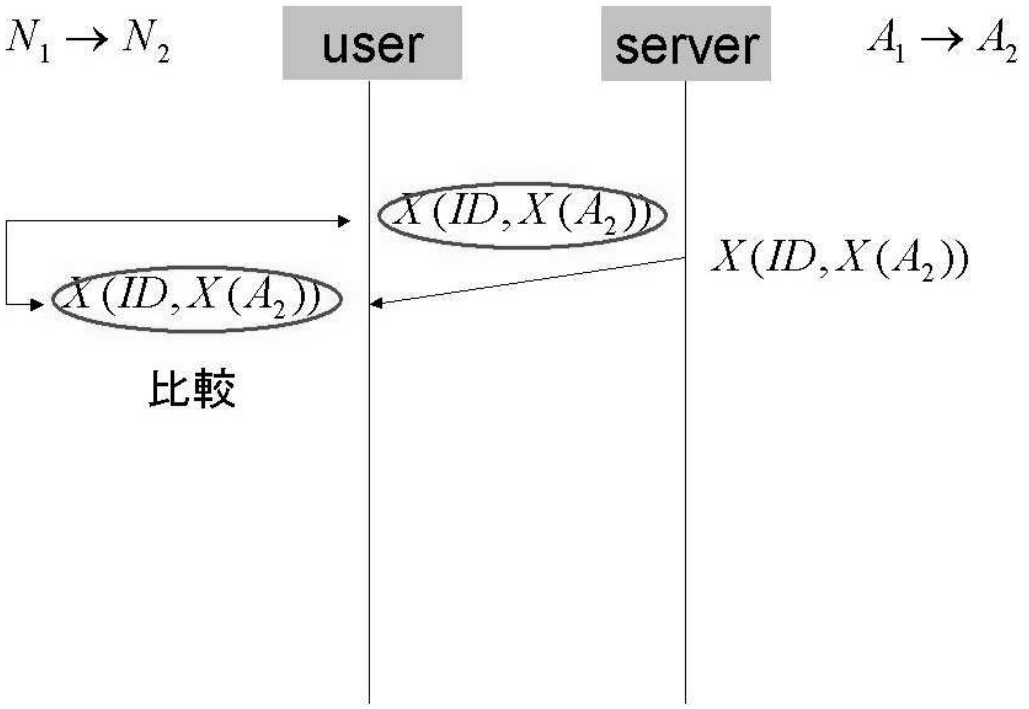


図 2.8 相互認証

相互認証部分は図 2.8 のようになる．次回の認証のために，サーバは A_2 を A_1 の替わりに登録する．

サーバは ID と比較情報 X(A_2) にハッシュをかけた X(ID,X(A_2)) を生成し，サーバからユーザへインターネット等の一般的なネットワークを用いて送信する．

ユーザはユーザ情報から X(ID,X(A_2)) を生成し，サーバから受信した情報とそれとを比較し，一致すれば認証成立となる．

第 3 章

提案方式

3.1 目的

ここまで Lamport の方法，SAS，SAS-2，三つのワンタイムパスワード認証方式について説明してきたが，全ての方式において，認証情報生成に一方向性関数，ハッシュを用いている．ハッシュ処理は，ユーザ，サーバに与える負荷が大きく，処理回数が増加すると，それに伴って負荷，処理時間が増加する．Lamport の方法を例にあげる．任意に設定する m の値，これはハッシュの回数となるわけだが，この値はシステム上，設定値から減少していき，データを公開し尽くした時点で，パスワードの再設定が必要となる．再設定の回数を減少させるには，その設定の間隔を大きくとる，つまり m=1000 というように大きく設定することが考えられるが，それにより，ハッシュ回数が増加し，負荷，処理時間が増大する．この場合ではハッシュ回数最大 1000 回と，計算量が膨大なのは確実である．

SAS，SAS-2 は，Lamport の方法と比較して，計算量を大幅に削減している．SAS-2 関していえば，認証時におけるハッシュ回数は 3 回であり，比較対照とならないほどである．これらのことから，処理負荷の大きいハッシュ回数を抑え，負荷，処理時間を軽減している，洗練された認証方式であるといえるが，今回，ハッシュ回数をさらに削減し，負荷，処理時間を抑える方法を提案した．負荷，処理時間の軽減により，ユーザがサーバへの集中した時の全体的な負荷の軽減や，携帯電話や IC カードの低スペックマシンへの利用など，応用の視野が広がる．

本研究の目的は，SAS-2 をベースとして，認証時におけるハッシュ回数を削減し，更なる高速化を行うことである．

3.2　余白時間の利用

認証時の負荷，処理時間を軽減することが，高速化につながる．上記のとおり，SAS-2 は，認証情報を得るためにハッシュを用いるが，ハッシュ処理は負荷が大きく，ハッシュの回数分，負荷，処理時間が増加する．SAS-2 は，認証フェーズにハッシュ処理が集中しており，高速化にはハッシュ処理の削減を考える．登録フェーズ，認証フェーズ，それぞれの中身に着目し，改良を加え，その結果，高速化を行うといった一連の流れが簡単に想像できる．だが，本研究では，登録フェーズ，認証フェーズ以外の部分に着目した．

最初の登録から認証までの間には，サービスを利用していない，またはサービスに対して通信，処理を行っていない時間が存在する．同様に，認証から次回の認証までの間においてもそのような時間が存在する．今回は，登録から認証，認証から認証といった処理の間の余白時間に着目した．

余白時間における処理量は，認証時と比較して少ない．これは間違いないと考えられる．そこで，この時間を利用し，ハッシュ処理を分散することを考えた．

3.3　内容

ワンタイムパスワード認証方式の中でも高速かつ通信コストの少ない SAS-2 をベースに，更なる高速化を行う方式を提案する．

高速化として考えられるのは，認証時のハッシュ回数を削減することである．SAS-2 の認証フェーズのハッシュ処理は 3 回であるが，提案方式は，この処理を 2 回にすることで認証時の負荷，処理時間を軽減し，高速化を図るというものである．

提案方式について説明する．提案方式は，SAS-2 ベースであり，登録フェーズ，認証フェーズが存在するが，余白時間の利用により，新しいフェーズを追加した．認証フェーズで行う，ハッシュ処理のひとつを，この追加フェーズで生成，登録し，登録した認証情報を認証フェーズで利用することにより，認証時におけるハッシュ回数を 2 回とした．この操作により，認証時の高速化が可能である．以下に提案方式の処理の手順を示す．

- ID : ユーザ ID
- S : ユーザのパスワード
- X : 一方向性関数 (ハッシュ)

  例 : X(a) は，a に 1 度ハッシュを適用.
- N_1 : 1 回目の認証で用いられる乱数.
- A_1 : 乱数 N_1 を用いた認証情報.
- $\oplus$: 排他的論理和
- $+$: 加算

3.3.1 登録フェーズ

1. ユーザは ID 及び S を入力．同時に，乱数 N_1 を生成し，登録．その後，入力データ及び乱数を用いて $A_1 = \mathrm{X}(\mathrm{ID},\mathrm{S} \oplus N_1)$ を計算．

2. ユーザは，セキュアなルートを用いて ID 及び A_1 をサーバに送信．

3. サーバは，認証フェーズで使用するためのデータである ID 及び A_1 を登録．

3.3.2 追加フェーズ 1

1. ユーザ，サーバ，それぞれにおいて，登録フェーズで得た A_1 にハッシュをかけ $\mathrm{X}(A_1)$ を生成，A_1 とは別に登録．
   $\mathrm{X}(A_1)$ は認証フェーズで使用．

3.3.3 認証フェーズ

1. ユーザは ID 及び S を入力．入力データ及び登録している N_1 を用いて
   $A_1 = \mathrm{X}(\mathrm{ID},\mathrm{S} \oplus N_1)$ を計算．次にユーザは乱数 N_2 を生成，登録．
   その後，入力データ及び乱数 N_2 を用いて
   $A_2 = \mathrm{X}(\mathrm{ID},\mathrm{S} \oplus N_2)$, $\mathrm{X}(A_2) = \mathrm{X}(\mathrm{ID},A_2)$ を計算．
   中間処理で得た $\mathrm{X}(A_1)$ を用い，
   $\alpha = A_2 \oplus [A_1 + (A_2 + \mathrm{X}(A_1))]$ 及び $\beta = (A_2 + \mathrm{X}(A_1)) \oplus A_1$ の計算を行う．

2. ユーザは ID 及び α，β を，インターネット等の一般的なネットワークを用いてサーバへ送信．

3. サーバは，登録データである A_1 を用いて $\beta \oplus A_1$ を行い，A_2+X(A_1) を取り出す．
   そして，その A_2+X(A_1) を用いて $\alpha \oplus$[A_1+(A_2+X(A_1)] から A_2 を得る．
   次に，サーバは A_2 と，中間処理で得た X(ID,A_1) の和をとり，
   取り出した A_2+X(A_1) との比較を行う．
   もし一致しなければ認証失敗，その時点で終了．
   一致した場合は認証成立，次のステップへと移行．

4. サーバは次の認証セッションのために，A_2 を A_1 の替わりに保存し
   γ = X(ID,X(A_2)) を計算．

5. サーバは，γ をインターネット等の一般的なネットワークを用いてユーザへ送信．

6. ユーザは X(ID,X(A_2)) を計算，受信データである γ との比較を行う．
   一致すれば，サーバ，ユーザ間の相互認証が成立．

3.3.4 追加フェーズ 2

1. ユーザ，サーバ，それぞれにおいて，登録された A_2 にハッシュをかけ
   X(A_2) を生成，A_2 とは別に登録．
   X(A_2) は次回の認証フェーズで使用．

以下に処理手順を図で簡単に説明する．

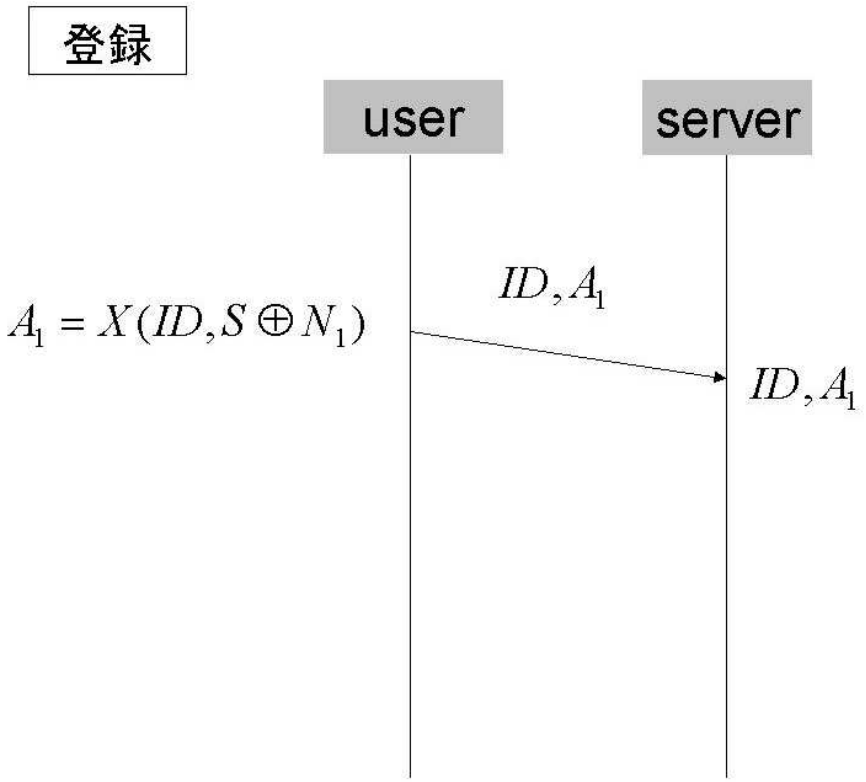


図 3.1　提案方式における登録フェーズ

登録フェーズは図 3.1 のようになる．SAS-2 と同様，ユーザは ID, パスワード S と乱数 N_1 の排他的論理和をとった情報にハッシュをかけ，認証情報 A_1 を生成する．ユーザ ID, A_1 を，セキュアなルートでユーザからサーバへ送信し，サーバはこの 2 つを登録しておく．

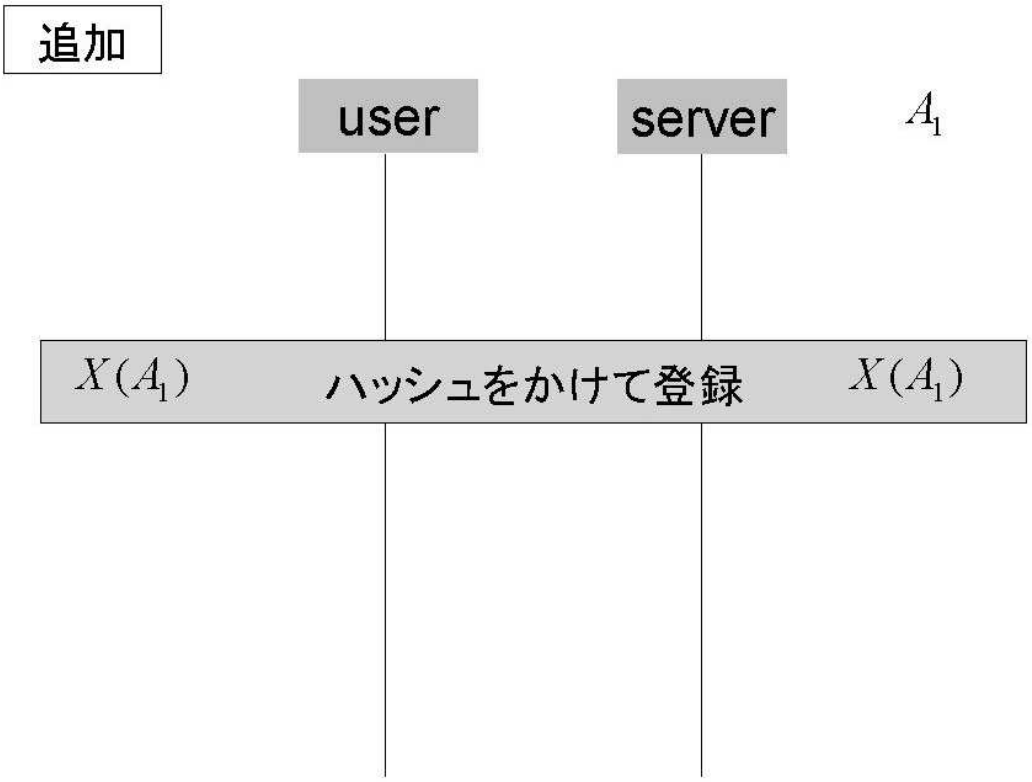


図 3.2 提案方式における追加フェーズ 1

追加フェーズは図 3.2 のようになり，以下の処理を行う．ユーザ，サーバ，それぞれが登録フェーズで得た認証情報 A_1 にハッシュをかけ，$\mathrm{X}(A_1)$ を生成し，その値を A_1 とは別に登録する．

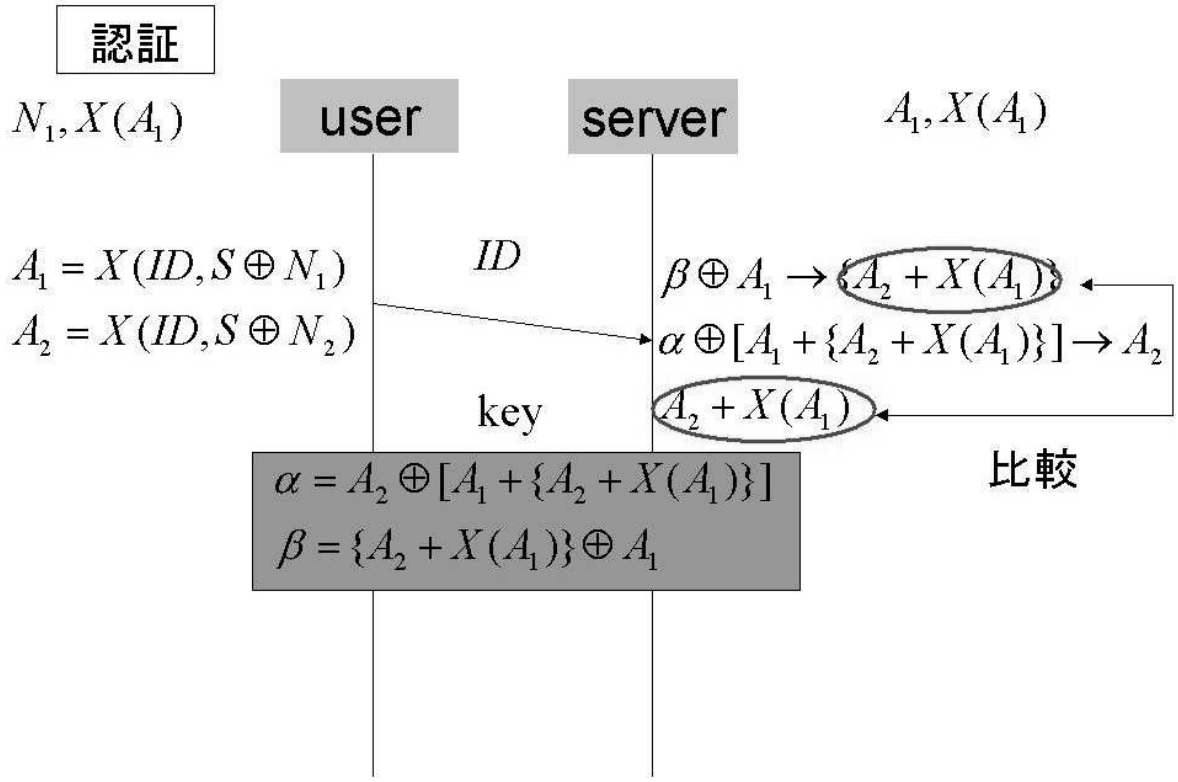


図 3.3 提案方式における認証フェーズ

追加フェーズは図 3.3 のようになる．登録時と同じ乱数を用いた認証情報 A_1，異なる乱数を用いた認証情報 A_2，そして，別保存しておいた $\mathrm{X}(A_1)$ から鍵 α，β を生成し，その鍵と ID をインターネット等の一般的なネットワークを用いて, ユーザからサーバへ送信する．

サーバは登録時の認証情報 A_1 と鍵を元に解読する．$\beta \oplus A_1$ から取り出した, A_2+$\mathrm{X}(A_1)$ と，解読から得られた A_2 に，別登録しておいた $\mathrm{X}(A_1)$ を加算した A_2+$\mathrm{X}(A_1)$ とを比較し，一致すれば認証成立となる．また A_2 は，次回の認証情報として使用される．

追加フェーズを用いて処理を分散した結果，ユーザの認証フェーズにおけるハッシュ回数は 2 回となった．

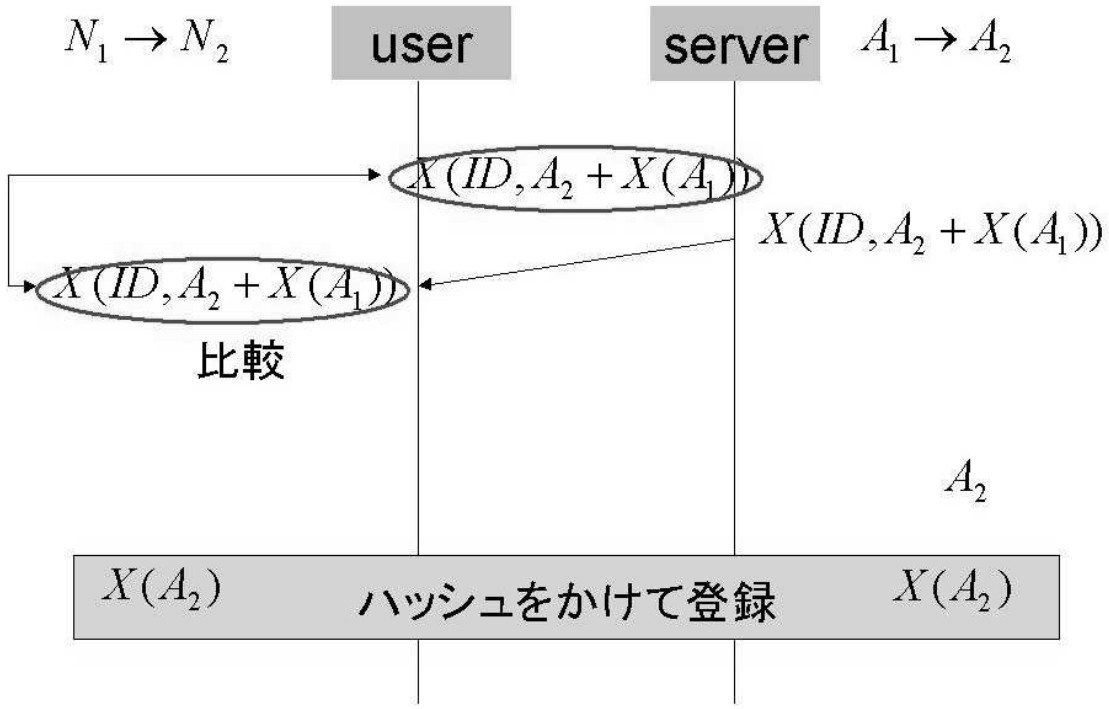


図 3.4 提案方式における認証・追加フェーズ 2

相互認証部分は図 3.4 のようになる．次回の認証のために，サーバは A_2 を A_1 の替わりに登録する．

サーバは ID と比較情報 A_2+X(A_1) にハッシュをかけた X(ID,A_2+X(A_1)) を生成し，サーバからユーザへインターネット等の一般的なネットワークを用いて送信する．

ユーザはユーザ情報から X(ID,A_2+X(A_1)) を生成し，サーバから受信した情報とそれとを比較し，一致すれば認証成立となる．

また，認証とは別の時間軸に追加フェーズを設け，そこでユーザ，サーバ，それぞれが次回の認証情報 A_2 にハッシュをかけ，X(A_2) を生成し，その値を A_2 とは別に登録する．

3.4 考察

提案方式は，携帯端末における認証方式として最適である SAS-2 をベースに作成した．登録認証間，または認証間における余白時間を利用し，認証時のハッシュ処理を分散することを考えた．分散させる三つのの処理のうち一つは情報をマスクするための処理で，残りの二つは今回と次回のための認証情報を生成する処理であるので，この二つは安全性のため切り離せないと考え，マスクするための処理を追加フェーズで行った．これにより安全性は従来の SAS-2 に等しいと考えられる．

ハッシュ処理の分散により，SAS-2 の認証時におけるハッシュ処理 3 回を，提案方式では 2 回に削減した．結果，ハッシュ 1 回分の負荷を抑え，SAS-2 と比較しても，その分の負荷軽減が見られ，認証時におけるユーザ，サーバ間の高速化を行うことが可能であると考えられる．

第 4 章

評価実験

4.1 比較内容

今回は SAS-2 と提案方式の認証時における処理時間と CPU 負荷率の評価を行った．実験を行ったマシンスペックを表 4.1 に示す．

表 4.1 使用マシンスペック

OS	Windows2000
メモリ	256MB
CPU	Celeron 500MHz

4.2 処理時間

認証時の処理時間を比較するために，SAS-2，提案方式の認証部分のプログラムを作成し，マシン上で処理時間の比較を行った．結果を図 4.1 に示す．

4.3 CPU 負荷率

SAS-2, 提案方式，それぞれが，CPU にどれだけ負荷をかけているか比較，調査した．プログラム実行から終了までの時間と，プログラム実行中の CPU 負荷率を，数回にわたってデータにとり，平均値をグラフにした．グラフを図 4.1 に示す．

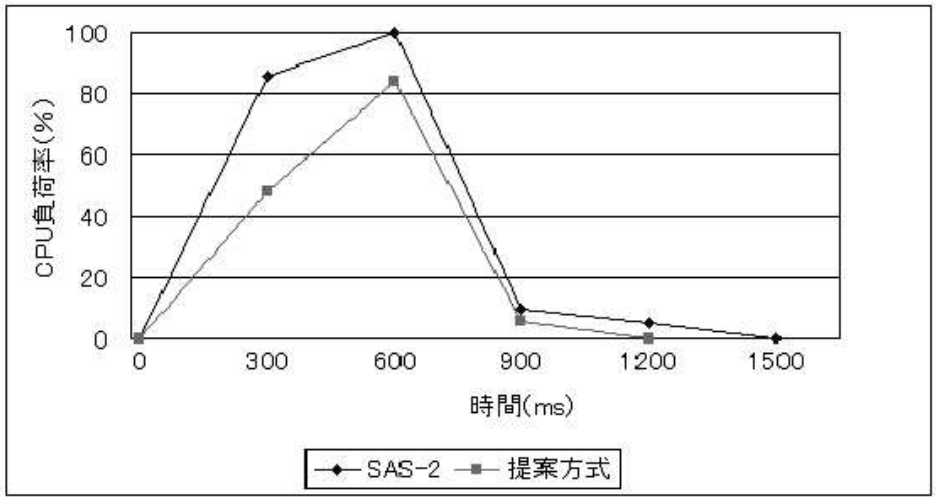


図 4.1 処理時間と CPU 負荷率

図 4.1 は二つの比較結果をグラフにしたものである．このグラフからわかるように，提案方式はハッシュの削減により処理時間が軽減されており，先にプログラムを終了させていることがわかる．提案方式=1200ms，SAS-2=1500ms と，300ms の差が見られた．ハッシュ回数を 3 回から 2 回に削減したことで，普通に考えれば 500ms あたりの差が見られると考えられる．しかし，処理に共通部分が多く存在するため，このような結果になったと考えられる．

CPU 負荷率についても，SAS-2 の CPU 負荷率は提案方式のものを全体的に上回る値をとり，提案方式が負荷を完全に抑えている．

提案方式における処理時間，CPU 負荷率を，SAS-2 のものと比較すると，約 20 %の軽減を確認できた．よってワンタイムパスワード認証方式に関する高速化に成功した．今後，実装部分等さらに検証していくことで，携帯電話はもちろん，他の低スペックマシンへの応用など，幅広く利用できるであろう．

第 5 章

おわりに

本研究では，携帯電話における認証方式の中で，最も適しているワンタイムパスワード認証方式，SAS-2 に着目し，プログラムや通信コストが少ない等，良い点をさらに伸ばす方向で改良を加え，ハッシュ処理を分散することで認証時における高速化を検討してきた．

評価実験では，提案方式と SAS-2 の処理時間，CPU 負荷率に関して比較，調査した．提案方式は，認証フェーズにおけるハッシュの削減を行い，認証時の負荷を軽減したものであり，実験結果から，提案方式は，負荷の軽減，処理の高速化が見られ，高速化に成功した．

今後の課題として，今回は認証部分の高速化をメインにしていたので，登録部分の実装，その部分を含めた全体的な評価，また，複数のユーザが一度にサーバに接続する場合等，考えられる事象に対しての負荷，処理時間はどうか，これらを詳しく検証する必要がある．

謝辞

高知工科大学工学部情報システム工学科 清水明宏教授には，卒業研究を含め，多岐に渡って懇切に御指導，御教示を賜った．ここに深謝申し上げる．

また，本学大学院生の辻貴介氏，上岡隆氏，河村智氏，大垣文誉氏ならびに清水研究室学部生 奈良裕介氏，永井慎太郎氏をはじめ研究室の方々には，研究途上において有益な御議論を頂いた．諸氏に心より感謝する．

参考文献

[1] L. Lamport, "Password authentication with insecure communication," Communications of the ACM, vol.24, no.11, pp.770-772, 1981.

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

[3] M. Sandirigama, A. Shimizu, and M.T. Noda, "Simple and secure password authentication protocol (SAS)," IEICE Trans. Commun., vol.E83-B, no.6, pp.1363-1365, June 2000.

[4] T. Kamioka, and A. Shimizu, "The examination of the security of SAS one-time password authentication," IEICE Technical Report, OFS2001-48, no.435, pp.53-58, 2001.

[5] T. Tsuji, T. Kamioka, and A. Shimizu, "Simple and secure password authentication protocol, ver.2 (SAS-2)," IEICE Technical Report, OIS2002-30, vol.102, no.314, pp.7-11, September 2002.

[6] T. Tsuji, and A. Shimizu, "Algorithm variations of SAS-2," IEICE Technical Report, IN2002-149, vol.102, no.498, pp.25-30, December 2002.