平成 29 年度

修士学位論文

IoT に適したワンタイムパスワード認証方式に関する研究

A one time password authentication method applicable to secure IoT for communications

1205079　　太田　愛里

指導教員　　清水　明宏

2018 年 2 月 2 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

IoT に適したワンタイムパスワード認証方式に関する研究

太田　愛里

近年, パソコン以外の物をインターネットに接続する IoT(Internet of THings) が発展してきている.IoT 機器には Web カメラや IC タグなどがある. 例えば Web カメラの通信が暗号化されずやりとりされた場合, 他人に見られたくない画像が流出してしまうという事が考えられる. よって IoT 機器にも暗号化が必要であるが, 暗号化には鍵配送方式が必要である. 暗号化の為に使う鍵配送方式としてワンタイムパスワード認証方式 SAS-2(Simple And Secure password authentication protocol, ver.2) を使う方法が存在する. しかし IC タグなどの処理能力の低い IoT 機器で利用する為には, 更に負荷の低い方法を考案する必要がある. 本稿では, 低処理能力の IoT 機器でも利用可能な鍵配送方式の提案を行う.

キーワード　IoT, ワンタイムパスワード認証

Abstract

A one time password authentication method applicable to secure IoT for communications

Eri Ota

In recent years, IoT (Internet of THings) to connect things other than personal computers to the Internet has been developed.IoT devices include Web cameras and IC tags.For example, when the communication of the Web camera is exchanged without being encrypted, it is conceivable that the image which is not wanted to be seen by others is leaked.Therefore, encryption is also required for IoT devices, but key distribution method is necessary for encryption.There is a method of using the one-time password authentication method SAS-2 (Simple And Secure password authentication protocol, ver. 2) as a key delivery method used for encryption.However, in order to use it on IoT equipment with low processing capability such as IC tags, it is necessary to devise a method with lower load.

In this paper, we propose a key distribution method that can be used for low throughput IoT devices.

key words IoT, One-time password authentication

目次

第 1 章　はじめに **1**

第 2 章　ワンタイムパスワード認証方式を用いた鍵配送方式 **3**

2.1　SAS-2 3

2.1.1　定義と記法 4

2.1.2　登録フェイズ 4

2.1.3　認証フェイズ 5

第 3 章　提案方式 **8**

3.1　既存方式の問題点 8

3.2　提案方式の概要 8

3.3　提案方式の原案 9

3.3.1　認証フェイズ 9

3.3.2　原案の問題点 10

3.4　提案方式 11

3.4.1　登録フェイズ 11

3.4.2　認証フェイズ 13

第 4 章　評価・考察 **20**

4.1　一方向性関数の適用回数 20

4.2　リプレイアタック 21

4.3　マスク値 M_i 21

4.4　IoT 機器での利用 22

第 5 章　まとめ **23**

謝辞 **24**

参考文献 **25**

図目次

2.1 SAS-2 の登録フェイズ 5
2.2 SAS-2 の認証フェイズ 6

3.1 提案方式の原案の認証フェイズ 10
3.2 図の表記法 11
3.3 提案方式の登録フェイズのフロー図 12
3.4 提案方式の登録フェイズの流れ 12
3.5 提案方式の認証フェイズ 14
3.6 ユーザ側がαとβを作成しサーバ側に送る 15
3.7 認証情報とマスク値とαで排他的論理和を取る 16
3.8 αから次回認証情報が出てくる 16
3.9 次回認証情報を認証情報の更新用に保存 16
3.10 取り出した次回認証情報と認証情報を足してβと比較しユーザを認証 17
3.11 M_{i+1}とγを作成 17
3.12 M_iと認証情報を M_{i+1}と次回認証情報に更新 17
3.13 γをユーザ側に送信 18
3.14 ユーザ側は M_{i+1}と E を計算 18
3.15 E とγを比較しサーバ側を認証 18
3.16 マスク値 M_iを M_{i+1}, 乱数 N_iを N_{i+1}に更新 19

表目次

4.1　認証情報を保存していた場合の一方向性関数の適用回数 20

第 1 章

はじめに

近年, モノのインターネット (IoT:Internet of Things) と呼ばれる物が普及し始めている.IoT とは, 様々な「物」がインターネットを利用した通信を行う事を意味する [1].IoT 機器の例として, 温度センサやテレビ,IC タグや Web カメラなどがある. これら IoT 機器の通信において, 秘匿すべき情報を取り扱っている場合が存在する. 例えば Web カメラの通信を盗聴された場合, 家の中の映像が見られてしまうなどの事が考えられる [2]. この事から,IoT の通信において暗号化が必要であると考えられる.

暗号化方式には公開鍵暗号方式と共通鍵暗号方式がある.IoT 以外の一般的なインターネット通信において, 通信の暗号化は公開鍵暗号方式を用いた TLS(Transport Layer Security) が利用されるが,IoT 機器への適用には問題がある.IoT 機器に利用される物には IC タグなどの小型で安価である事が求められる物も多く, その為に計算能力の低い CPU が使われる. また,IoT 機器にはバッテリー駆動の物も多い. 将来多くの物が IoT 機器に置き換わって行った場合, 個々のメンテナンスが困難になる. その事から電力消費を少なくしライフスパンを長くする必要性があり, 処理負荷が増えると電力消費が増える為, 処理負荷は低い方が望ましいと考えられる. よって, 公開鍵暗号のような負荷の高い計算を求められる暗号化方式は適していない. その為計算量の少ない共通鍵暗号方式の利用が適していると考えられる. しかしこの場合暗号化と復号に同じ鍵を利用するので, 鍵の共有方法が問題となる. 鍵配送方式としては, ワンタイムパスワード認証方式である SAS-2(Simple And Secure password authentication protocol，ver.2) の認証情報を鍵として利用する事で鍵配送方式として利用する方法が存在する [3]. しかしそれでも一度の通信に対し, 加算などの普通の演算数百回分である一方向性関数を一度は行う事となる.

そこで本稿では, 処理能力の低い IoT 機器でも利用できるよう,IoT 機器側で一方向性関数を使わない鍵配送方式として使えるワンタイムパスワード認証方式を提案する. まずは既存のワンタイムパスワード認証方式である SAS-2 を用いた鍵配送方式について説明する. 次にその問題点を改善した提案方式を説明し, 評価を行う.

第 2 章

ワンタイムパスワード認証方式を用いた鍵配送方式

本章では, 鍵配送方式として利用可能なワンタイムパスワードについて述べる.
ワンタイムパスワード認証方式は, 認証を行うたびにパスワードが変更される認証方式である. そのワンタイムパスワード認証方式の中でも, 鍵配配送方式として利用でき, 処理の軽量な SAS-2 について説明していく.SAS-2 では認証情報が毎回変更される為, その認証情報を鍵として用いる事で通信の度に鍵が更新される鍵配送方式としても利用可能である. 鍵が更新されない固定鍵を使うと, 一度鍵が漏洩してしまった場合その後の通信内容が盗まれ続けてしまうので, 鍵配送方式は通信の度に鍵が更新される物が望ましいと考えられる. 特に IoT 機器で利用するに当たっては, 機器が多くなる事が想定される. その場合メンテナンスが困難になる為, 自動的に鍵が更新される鍵配送方式を導入する事が望ましいと考えられる.
よって,SAS-2 は IoT 機器での通信に適した性質を持っていると考えられるので,SAS-2 を IoT での通信に利用する場合の検討を行う.

2.1 SAS-2

SAS-2 は, 一方向性関数と排他的論理和を使った回数制限無く認証可能なワンタイムパスワード認証方式である.SAS-2 には初回の通信でのみ行われる登録フェイズと次回以降の通信で行われる認証フェイズがある. 登録フェイズでは初回認証情報を安全なルートを使って通信を行いたい双方に共有する. 認証フェイズでは安全でないインターネットなどのルート

を用いても良く, 通信を行う双方が互いに認証を行い, 認証が成立した場合認証情報を新しい物に更新する.

2.1.1 定義と記法

本稿で用いる定義と記法は以下の通りである.

- User は, 認証されるユーザである
- Sever は,User を認証する認証者である
- ID は, ユーザの識別子である
- S は, ユーザのパスワードを示す
- F(s) は,s に対して一方向性関数を適用して得た出力値を示す
- N_iは, i 回目の認証時に生成される乱数を示す
- $+$は, 加算演算子を示す
- $m \mid n$ は, m と n の連結を示す
- $\oplus$は, 排他的論理和を示す

2.1.2 登録フェイズ

SAS-2 における登録フェイズでは, ユーザは初期情報を生成し, 安全なルートでサーバへ送信する. 図 2.1 に SAS-2 の登録フェイズを示す.

1. ユーザは自身の識別子 ID, パスワード S を入力する.
2. ユーザは 乱数 N_1 を生成し保存する.
3. ユーザは入力された ID, S, 生成された N_1を用いて $A = F(ID \mid S \oplus N_1)$ を算出する.
4. ユーザは ID,A を安全なルートを用いてサーバへ送信する.
5. サーバは受け取った ID,A を保存する.

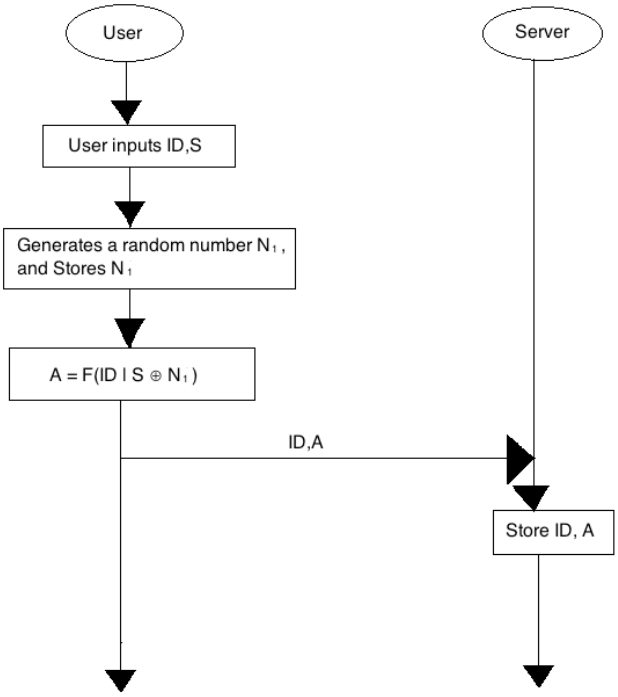

図 2.1　SAS-2 の登録フェイズ

2.1.3　認証フェイズ

SAS-2 における認証フェイズでは, 最初にユーザが生成した認証情報の正当性をサーバが検証, ユーザを認証する. 次にサーバが生成した認証情報の正当性をユーザが検証しサーバを認証する. これによってユーザとサーバ間での相互認証が行われる. 図 2.2 に i 回目の認証における SAS-2 認証フェイズを示す.

1. ユーザは自身の識別子 ID, パスワード S を入力する.
2. ユーザは入力されたデータと保存された乱数 N_iから $A = F(ID \mid S \oplus N_i)$ を算出する.
3. ユーザは乱数 N_{i+1}を生成し保存する.
4. ユーザは乱数 N_{i+1}から $C = F(ID \mid S \oplus N_{i+1}), F(C) = F(ID \mid C), \alpha = C \oplus (F(C) + A), \beta = F(C) \oplus A$ を求める.
5. ユーザは ID, α, βをサーバへ送信する.この時使用するネットワークはインターネットなどの安全で無いルートで良い.

2.1 SAS-2

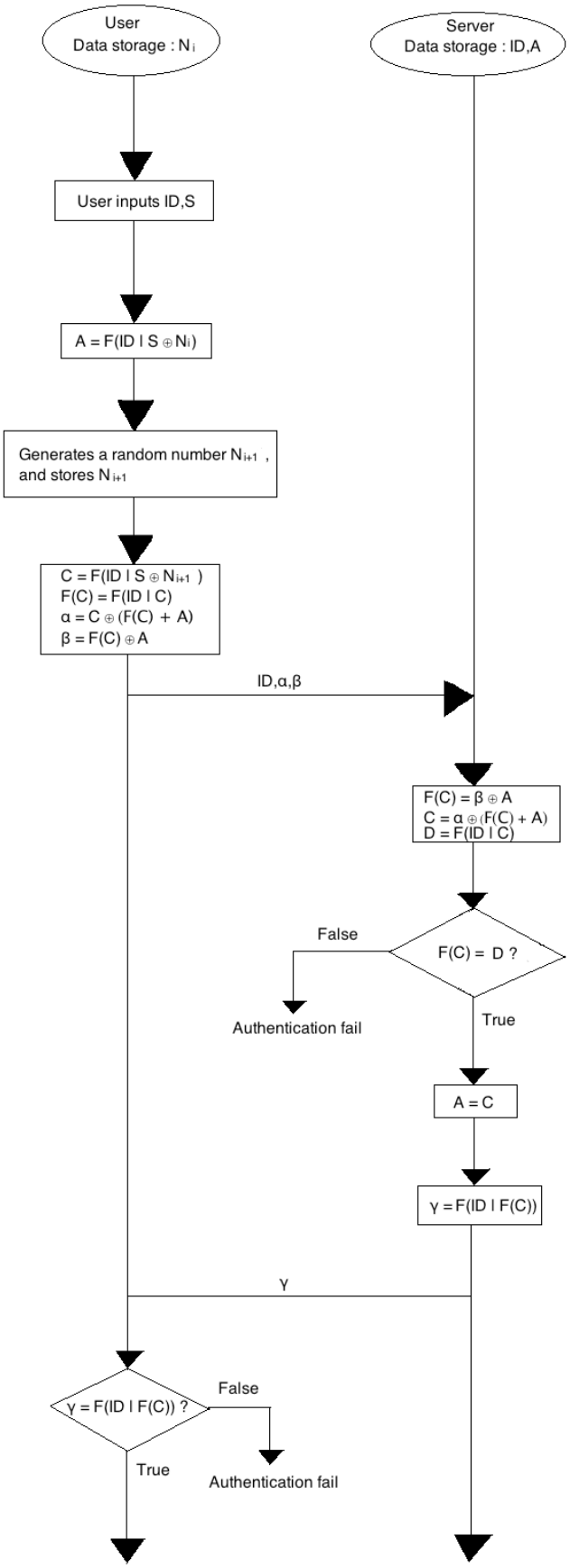

図 2.2 SAS-2 の認証フェイズ

6. サーバは受信したα, βと保存された A を用いて $F(C) = \beta \oplus A, C = \alpha \oplus (F(C) + A), D = F(ID \mid C)$ を算出する.
7. サーバは先程計算した F(C) と D を比較し, 一致すれば認証が成功, 以下の処理が実行される. 不一致ならば認証は不成立となり以下の処理は実行されない.
8. サーバは保存されている A の代わりに C を新しい認証情報として保存する
9. サーバは$\gamma = F(ID \mid F(C))$ を算出する.
10. サーバはγをインターネットなどを通してユーザへ送信する.
11. ユーザは, $F(ID \mid F(C))$ を算出し, 受信したγと比較する.不一致ならば認証は不成立となり, 一致すればサーバはユーザに認証される.

第 3 章

提案方式

本章では既存方式における問題点を整理し, 解決するための提案手法について記述する.

3.1 既存方式の問題点

低処理能力の IoT 機器で鍵配送方式として利用した場合の SAS-2 における問題点は, 一方向性関数を利用する事である.

$SAS-2$ では, ユーザ側は前回の認証情報である $A = F(ID \mid S \oplus Ni)$ を保存する事で一方向性関数の適用回数を減らす事が可能である.しかしその場合でもユーザ側は通信のたびに $C = F(ID \mid S \oplus N_{i+1})$, 及び $D = F(ID \mid C)$ の式を実行する必要があり, 一方向性関数の適用回数は二回となる.サーバ側は通信のたびに $F(C) = \beta \oplus A, \gamma = F(ID \mid F(C))$ が必要であり, 一方向性関数の適用回数は二回となる.更にサーバ側は, 相互認証を行わない場合 $\gamma = F(ID \mid F(C))$ が不要となり一方向性関数の適用回数を一回に減らす事が可能である.

一方向性関数は足し算や引き算などの普通の演算を数百回分利用しているのと同程度の処理を必要とする [4]. よって, 適用する一方向性関数が不要となればより高速な通信が行え, また, より低処理能力の機器での利用が可能になると考えられる.

3.2 提案方式の概要

既存方式の問題点を解決する為に, 一方向性関数が不要で, 鍵配送方式としても利用出来るワンタイムパスワード認証方式を提案する.

提案方式は初回のみ行う登録フェイズと次回以降の通信で実行される認証フェイズに分か

れている. 登録フェイズでは安全なルートで通信して必要な初回認証情報を登録する. 認証フェイズではインターネットなどの安全で無いルートで通信可能で, 通信を行う双方で認証を行い, 成立すれば認証情報の更新を行う.
定義と記法は SAS-2 の物と同じであるが, それに加えて新しく定義した物がある. 以下にそれを示す.

- M_iは, i 回目の認証時に生成される値を示す.これは鍵の暗号化の強度を上げる為に利用されるマスク値である

3.3　提案方式の原案

提案方式に入る前に, 提案方式より保存するパラメータが少ない為保存容量を削減可能な提案方式の原案を述べる.これはマスク値 M_iを使用しない場合の提案方式である. 登録フェイズは SAS-2 と同様である為省略し, 認証フェイズについて説明していく.

3.3.1　認証フェイズ

認証フェイズではユーザが生成した認証情報の正当性をサーバが検証, ユーザを認証する. 次にサーバが生成した認証情報の正当性をユーザが検証, サーバを認証する事でユーザとサーバ間の相互認証を行う. 図**??**に i 回目の認証における提案方式の認証フェイズのフロー図を示す. その後文章で処理の流れを説明する.

1. ユーザは自身の識別子 ID, パスワード S を入力する.
2. ユーザは入力されたデータと保存された乱数 N_iから認証情報 $A = F(ID \mid S \oplus N_i)$ を算出する.
3. ユーザは乱数 N_{i+1}を生成し保存する.
4. ユーザは入力された $ID, S,$ 生成された乱数 N_{i+1}を使って次回認証情報 $B = F(ID \mid S \oplus N_{i+1})$ を計算し, サーバへ送信する値$\alpha = A \oplus B, \beta = A + B$ を算出する.βの計算

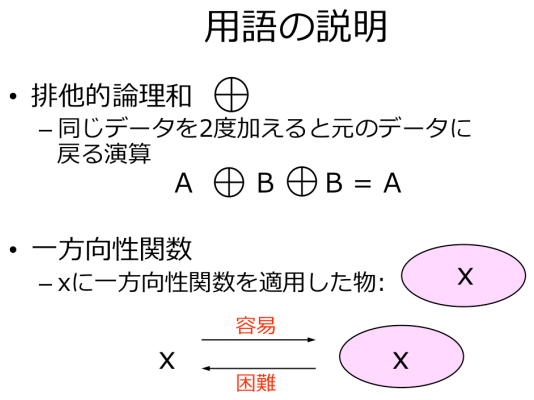

図 3.1　提案方式の原案の認証フェイズ

を行う際に桁溢れを起こした場合は, その値を切り捨てる.

5. ユーザは ID, α, βをサーバへ送信する.この時使用するネットワークはインターネットなどの安全で無いルートで良い.
6. サーバは受け取ったαと保存された A から $C = \alpha \oplus A$ を計算して次回認証情報 B を出し, $D = C + A$ を算出する.D の計算を行う際に桁溢れを起こした場合は, その値を切り捨てる.
7. サーバは受信した$\beta = B + A$ と算出した $D = C + A$ を比較し, 一致すれば認証が成立, 以下の処理が実行される.不一致ならば認証は不成立となり, 以下の処理は実行されない..
8. サーバは保存された A の代わりに C を新しい認証情報として次回認証用に保存する.

3.3.2　原案の問題点

マスク値 M_iが存在しない場合は i 回目の通信データα, βを $i+1$ 回目に認証情報として利用する事でリプレイアタックが可能である.ただし, $i+1$ 回目に i 回目に使った認証情報が利用されている場合を弾けば利用可能である.

また, α, βから認証情報を逆算する場合の計算は総当たりとなる.計算に使用する bit 数を $64bit$ と仮定する.マスク値 M_iが入っていない場合はαから認証情報と次回認証情報を計

算する.この時計算量は 2^{64} となる.マスク値 M_i が入っている場合は $\alpha = A \oplus B \oplus M_i$ とする為, M_i の計算が増え計算量は 2^{128} となる.

3.4　提案方式

上記の問題を解決した提案方式を述べていく. フロー図の他に図 3.4 から図 3.16 で提案方式の流れを示す. その際の表記方法を図 3.2 で示す.

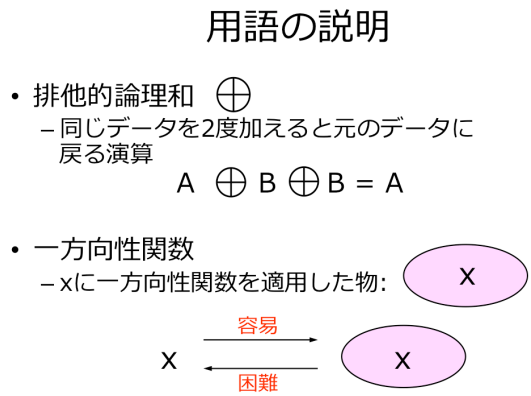

図 3.2　図の表記法

3.4.1　登録フェイズ

登録フェイズでは初回認証情報を安全なルートで共有する. 図 3.3 に提案方式の登録フェイズのフロー図を示す. また, それに加えて処理の流れを文章で説明する. 図 3.4 では登録フェイズで実際に行なっている処理を分かりやすく図式で示す.ただし, 図 3.4では $F(ID \mid S \oplus N_1)$ は図式の簡易化の為 $F(S \oplus N_1)$ で示している.

1. ユーザは自身の識別子 ID, パスワード S を入力する.
2. ユーザは乱数 N_1, M_1 の生成と保存を行う.
3. ユーザは入力された $ID, S,$ 生成された乱数 N_1 を使って $A = F(ID \mid S \oplus N_1)$ を算出する.

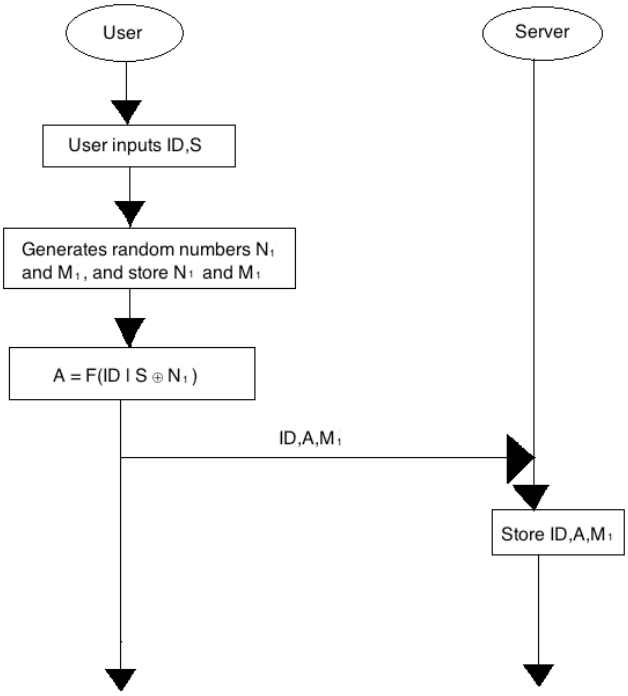

図 3.3　提案方式の登録フェイズのフロー図

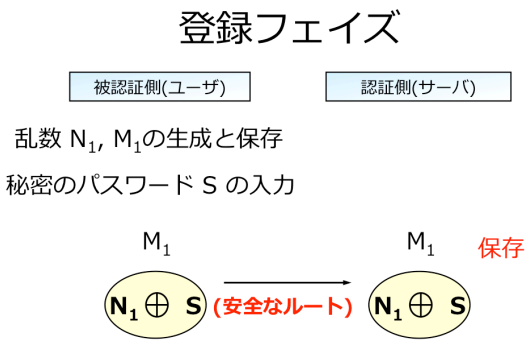

図 3.4　提案方式の登録フェイズの流れ

4. ユーザは ID, A, M_1 を安全なルートを用いてサーバへ送信する.
5. サーバは受け取った ID, A, M_1 を保存する.

3.4.2　認証フェイズ

認証フェイズではユーザが生成した認証情報の正当性をサーバが検証, ユーザを認証する. 次にサーバが生成した認証情報の正当性をユーザが検証, サーバを認証する事でユーザとサーバ間の相互認証を行う. 図 3.5 に i 回目の認証における提案方式の認証フェイズのフロー図を示す. また, 文章で処理の流れを説明する. 図 3.6 から図 3.16 では図式で処理の流れを示す.ただし図 3.6から図 3.16では $F(ID \mid S \oplus N_i), F(ID \mid S \oplus N_{i+1})$ は図式の簡易化の為 $F(S \oplus N_i), , F(ID \mid S \oplus N_{i+1})$ で示している.

1. ユーザは自身の識別子 ID, パスワード S を入力する.
2. ユーザは入力されたデータと保存された乱数 N_i から認証情報 $A = F(ID \mid S \oplus N_i)$ を算出する.
3. ユーザは乱数 N_{i+1} を生成し保存する.
4. ユーザは入力された ID, S, 保存されたマスク値 M_i, 生成された乱数 N_{i+1} を使って次回認証情報 $B = F(ID \mid S \oplus N_{i+1})$ を計算し, サーバへ送信する値 $\alpha = A \oplus B \oplus M_i, \beta = A + B$ を算出する.β の計算を行う際に桁溢れを起こした場合は, その値を切り捨てる.
5. ユーザは ID, α, β をサーバへ送信する.この時使用するネットワークはインターネットなどの安全で無いルートで良い.
6. サーバは受け取った α と保存された A, M_i から $C = \alpha \oplus A \oplus M_i$ を計算して次回認証情報 B を出し, $D = C + A$ を算出する.D の計算を行う際に桁溢れを起こした場合は, その値を切り捨てる.
7. サーバは受信した $\beta = B + A$ と算出した $D = C + A$ を比較し, 一致すれば認証が成立, 以下の処理が実行される.不一致ならば認証は不成立となり, 以下の処理は実行されない.

3.4　提案方式

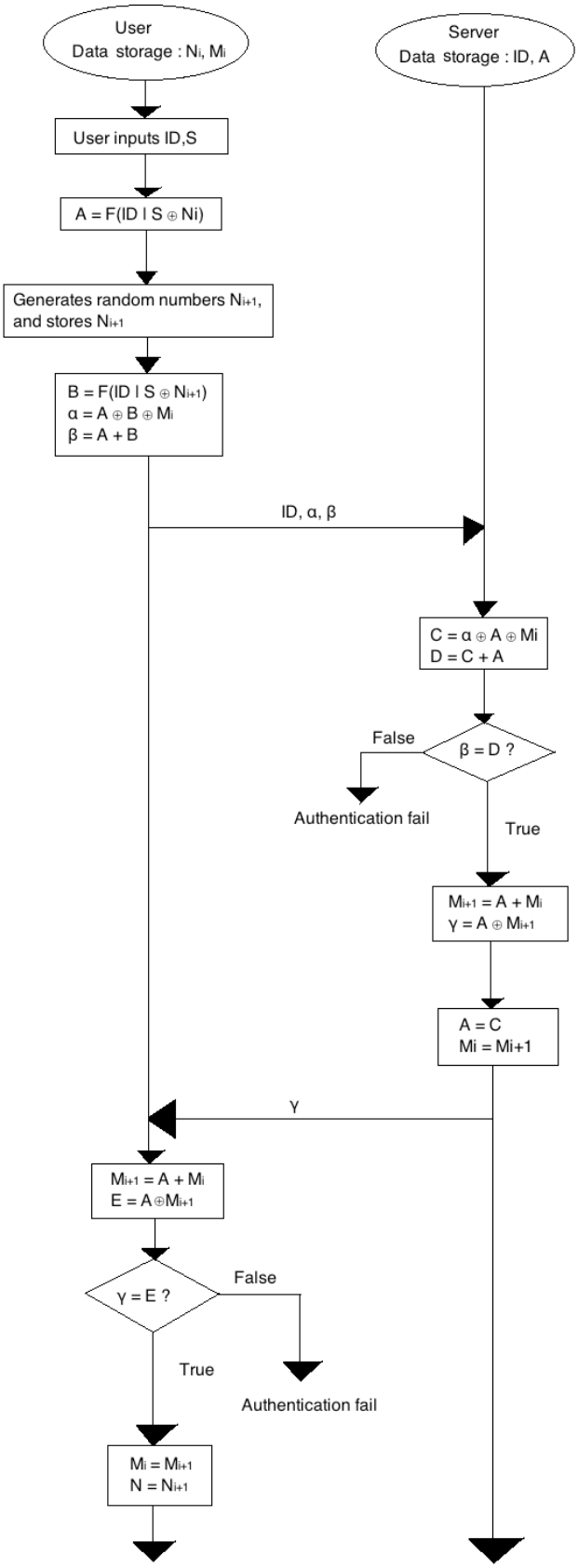

図 3.5　提案方式の認証フェイズ

8. サーバは保存された A, M_iから $M_{i+1} = A + M_i, \gamma = A \oplus M_{i+1}$を算出する.$M_{i+1}$の計算を行う際に桁溢れを起こした場合は, その値を切り捨てる.
9. サーバは保存された A, M_iの代わりに C, M_{i+1}を新しい認証情報として次回認証用に保存する.
10. サーバはγをインターネットなどを通してユーザへ送信する.
11. ユーザは, $M_{i+1} = A + M_i, E = A \oplus M_{i+1}$を算出する.$M_{i+1}$の計算を行う際に桁溢れを起こした場合は, その値を切り捨てる.
12. ユーザは, E と受信したγを比較する.不一致ならば認証は不成立となり, 一致すればサーバはユーザに認証される.
13. ユーザは次回認証用に, 保存されたマスク値 M_iの代わりに M_{i+1}, 乱数 N_iの代わりに N_{i+1}を保存する.

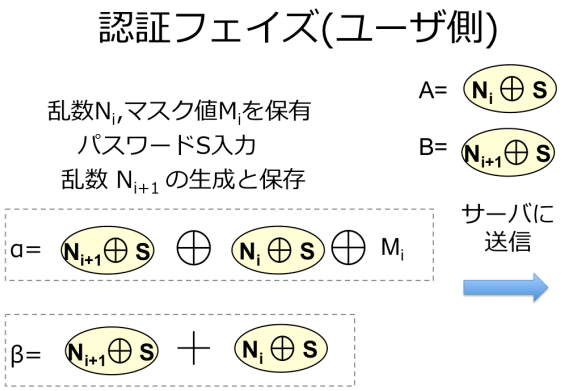

図 3.6　ユーザ側がαとβを作成しサーバ側に送る

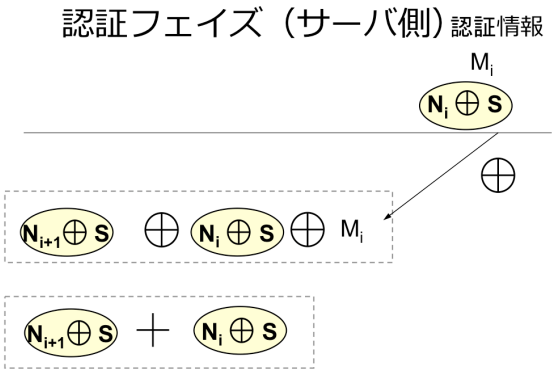

図 3.7　認証情報とマスク値とαで排他的論理和を取る

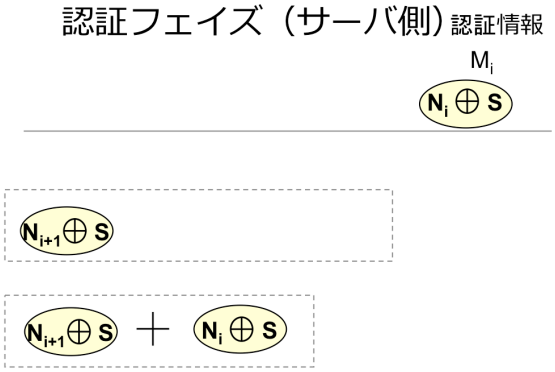

図 3.8　αから次回認証情報が出てくる

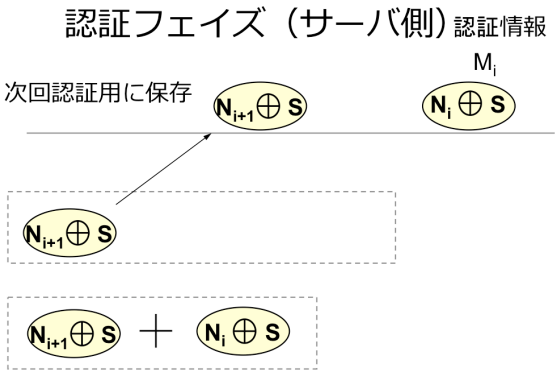

図 3.9　次回認証情報を認証情報の更新用に保存

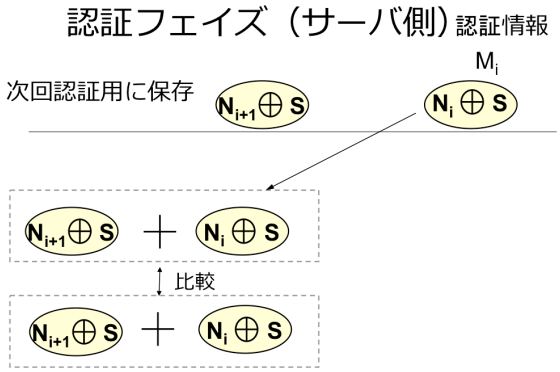

図 3.10　取り出した次回認証情報と認証情報を足してβと比較しユーザを認証

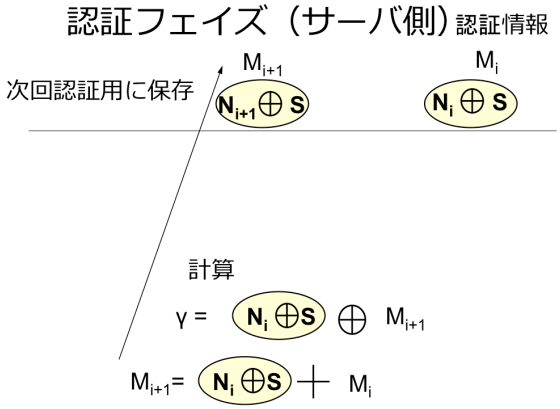

図 3.11　M_{i+1}とγを作成

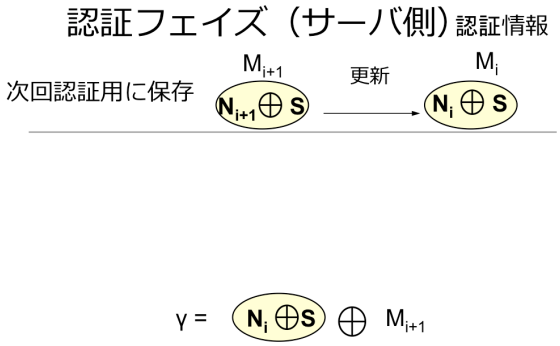

図 3.12　M_iと認証情報を M_{i+1}と次回認証情報に更新

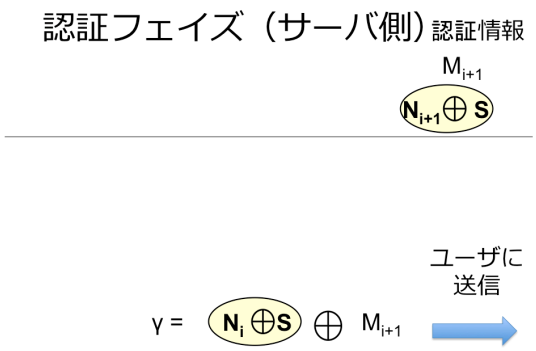

図 3.13　γをユーザ側に送信

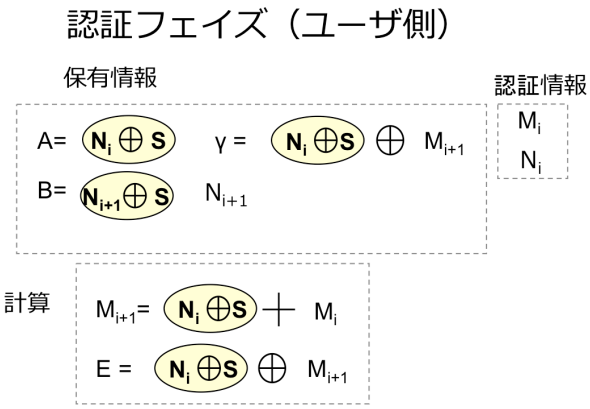

図 3.14　ユーザ側は M_{i+1}と E を計算

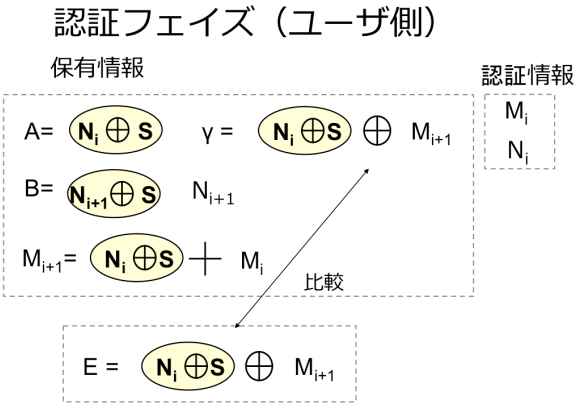

図 3.15　E とγを比較しサーバ側を認証

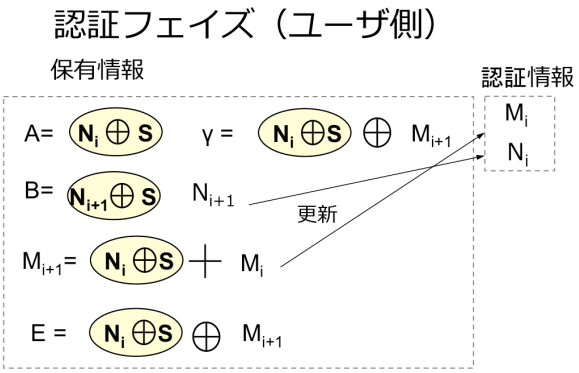

図 3.16　マスク値 M_iを M_{i+1}, 乱数 N_iを N_{i+1}に更新

第 4 章

評価・考察

4.1 一方向性関数の適用回数

提案方式の一方向性関数の適用回数は, $SAS-2$ と比べてユーザ側, サーバ側共に 1 回ずつ減少している.ユーザ側は一方向性関数を認証情報 $A = F(ID \mid S \oplus N_i)$ を作る際と次回認証情報 $B = F(ID \mid S \oplus N_{i+1})$ を作る際の二回使用している.上記の方法では乱数 N_iを保存しておいて認証情報 A を作っているが, N_iでは無く A を保存しておいても良い.その場合 B の計算のみとなり一方向性関数の適用回数は一回となる.対してサーバ側は一方向性関数が不要であり, 足し算などの普通の演算数回のみで処理が可能となる.鍵配送方式として利用する場合ユーザ側とサーバ側を逆にしても問題ない為, IoT 機器側などにサーバ側を適用する事で処理能力の低い IoT 機器などでも高速な通信が可能となる.

表 4.1 に, 一方向性関数の適用回数を示す.

表 4.1　認証情報を保存していた場合の一方向性関数の適用回数

	一方向性関数適用回数	
	ユーザ	サーバ
SAS-2	2	1
提案方式	1	0

4.2　リプレイアタック

リプレイアタックとは, 今までに流れた通信データを使って攻撃する手法である.
$A = F(ID \mid S \oplus N_i), B = F(ID \mid S \oplus N_{i+1}), M_{i+1} = A + M_i$とおく.$i$ 回目に流れるデータは$\alpha = A \oplus B \oplus M_i, \beta = A + B$ である.i 回目の認証情報は A であり, $i+1$ 回目の認証情報として B がセットされる.$i+1$ 回目の認証の時, 正しい認証であると偽装して通信を行い新しい認証情報に X をセットしたい場合, $\alpha 2 = B \oplus X \oplus M_{i+1}, \beta 2 = B + X$ を作れば良い.
提案方式へのリプレイアタックについて, 今までに流れた通信データを傍受し, そのデータを使って先述の $i+1$ 回目の認証を作ることが出来れば攻撃可能であると言える.$Z = F(ID \mid S \oplus N_{i-1})$ とおいた場合, $\alpha 3 = A \oplus Z \oplus M_{i-1}, \beta 3 = A + Z$ となる.$i-1$ 回目と i 回目の通信を傍受していたとする.$\alpha 3 \oplus \alpha = (A \oplus Z \oplus M_{i-1}) \oplus (A \oplus B \oplus M_i) = Z \oplus B \oplus M_i \oplus M_{i-1}$となる. $\oplus$ を使う式を幾ら $\oplus$ で足し合わせても, M_i, M_{i+1}, M_{i+2}と M_i の項が増えていくだけなので M_iが取り出せない限りβと対応する式を作ることは出来ない.
また, $\beta - \beta 3 = A + B - (A + Z) = B - Z$ となり X の値にマイナスが入るのでαと対応する式は作ることが出来ない.$\beta + \beta 3$ をした場合は項が減らない.
また, 相互認証を行う場合$\gamma = A \oplus M_{i+1}$も通信データ上を流れる.$\alpha \oplus \gamma = B \oplus M_i \oplus M_{i+1}$ となり, X が M_iとなる.他の式から M_iのみを取り出す事は出来ないので$\beta 2 = B + X$ は作れない.
よってリプレイアタックによる攻撃は効かないと考えられる.

4.3　マスク値 M_i

使用するマスク値の数を増やすと計算量を増やす事が可能である.$\alpha = A \oplus B \oplus M_i \oplus M_{i2}$ とすると計算量は 2^{128}から 2^{192}となる.

4.4　IoT 機器での利用

IoT 機器で利用する場合, 相互認証である為サーバ側とユーザ側を逆に適用しても問題ない. その場合,IoT 機器側の一方向性関数の適用回数は 0 回となり, 乱数生成も無い為, 加算と排他的論理和が可能ならば処理する事が可能となる. よって, 非常に処理能力の低い IoT 機器でも利用可能になると考えられる.

また, 自身の ID, パスワード S は,IoT 機器で利用する場合は IoT 機器側にあらかじめ出荷時に入れておくなどの方法が考えられる. その場合通信するゲートウェイなどに認証情報を手動で入力したり,QR コードで読み込ませるなどと言った方法が必要である.

第 5 章

まとめ

本稿では, 低処理能力の IoT 機器に適した, 一方向性関数を利用する事なく処理可能な鍵配送方式として使えるワンタイムパスワード認証方式について述べた. 提案方式では, ユーザ側とサーバ側を逆に適用する事で処理能力の低い IoT 機器側などで一方向性関数を適用する事なく処理が可能となる. 今後の課題として, 実際の IoT 機器への実装などが考えられる.

謝辞

本研究の遂行と論文作成に当たり, 非常に多くの御指導, 御助言を頂きました高知工科大学情報学群 清水明宏教授に心より感謝致します. また, 本研究の副査を担当して頂いた高知工科大学情報学群 村昌則教授, 吉田真一准教授に深く御礼申しあげます.
最後に, 有益な議論を交わして頂いた高知工科大学 清水研究室の関係者各位の深く感謝致します.

参考文献

[1] 総務省,“第 3 節 IoT 化する情報通信産業”,http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h29/pdf/n3300000.pdf,(2018/01/22 アクセス).

[2] トレンドマイクロ,“「IoT」にも「セキュリティ」が必要？”,https://www.trendmicro.com/jp/iot-security/special/26,(2018/01/22 アクセス).

[3] T.Tsuji，A.Shimizu,“A one-time password authentication method for low spec machines and on internet protocols”IEICE Trans.Commun., vol.E87-B no.6, pp.1594-1600，2004.

[4] 岡本龍明, 山本博資, “シリーズ/情報科学の数学　現代暗号”, pp.192-195, 産業図書,1997