[研究論文]　モバイルクラウド用プッシュ型通信における相互信頼プロトコルの研究

岡崎美蘭

情報ネットワーク・コミュニケーション学科

A Study of Mutual-Reliability Protocol on Push-Type Communication for Mobile Cloud

Mirang OKAZAKI

Abstract

Google Cloud Messaging for Android (GCM) is a service that allows us to send data from our server to our users' Android-powered device, and also to receive messages from devices on the same connection. The GCM service handles all aspects of queue of messages and delivery to the target Android application running on the target device. It is desirable that the communication port of the users' device is always open. This is a practical problem of power consumption of the mobile terminal. To solve this problem, there is a push-type communication for mobile cloud. In this paper, we study how to establish the safety and encrypts the message in push-based one-way communication. We propose mutual-reliability protocol that combines the mechanism of key sharing mechanism and user authentication.

Keywords: Mobile clouding, GCM, push-type communication, mutual-reliability, key sharing

1.　はじめに

現在，クラウドコンピューティングの普及により，従来はユーザ自身が管理・利用していたハードウェア，ソフトウェア，データを，ユーザが保有せずにインターネット等のネットワークを経由し，様々なサーバが提供しているサービスとして利用できるようになった．また，汎用的なコンピュータと同等な機能を持つスマートフォンやタブレット端末の登場により，利用者は移動中や外出先からネットワーク経由で社内システムやクラウドサービスに簡単に接続できるようになった．そのため，今後はこのようなモバイル端末を用いたモバイルクラウドサービスが急速に伸びていくことが予想される[1，2]．

一方，クラウドコンピューティングにおいては，ユーザが使用する端末の通信ポートは常に開いている状態が望ましいとされている．これは，サーバから提供されるサービスをユーザが適宜利用できる環境にしておくためである．しかし，これはモバイル端末の電力消費に繋がってしまい，常に充電できる環境での使用が期待できないモバイル端末では実用上の問題となってしまう．また，昨今のモバイル端末は，スマートフォンに象徴されるように，様々なアプリケーションが実装されており，電力消費量が多くなる傾向にある．

そこで，Google 社はモバイル端末の消費電力を抑えて運用できるように，最低限のコマンドメッセージのみを受け付けるプッシュ型通信ポートを用意し，必要時にサーバからモバイル端末にコマンドメッセージを送信することで，実際の処理を行うための正規の通信ポートによる接続を開始する GCM（Google Cloud Messaging for Android）というサービスを提供している[3]．

しかし，この GCM においては，そのメッセージが送信されるネットワークがオープンな環境であるにもかかわらず，そのメッセージの信頼性を確立する手段が取り入れられていない．また，Android 端末は端末の仕様がオープンであることから，サーバ側やモバイル端末側にとって不正な処理が行われる危険性がある．このような危険性に対し，通常はサーバ側・モバイル端末側相互の信頼性を確立するために相互認証を行うことが一般的であるが，一方向性のプッシュ型通信では既存の相互認証プロトコル[4,5]を適用できないといった問題点がある．そのため，新たな考えに基づき，相互の信頼性を確立する手段を検討する必要がある．

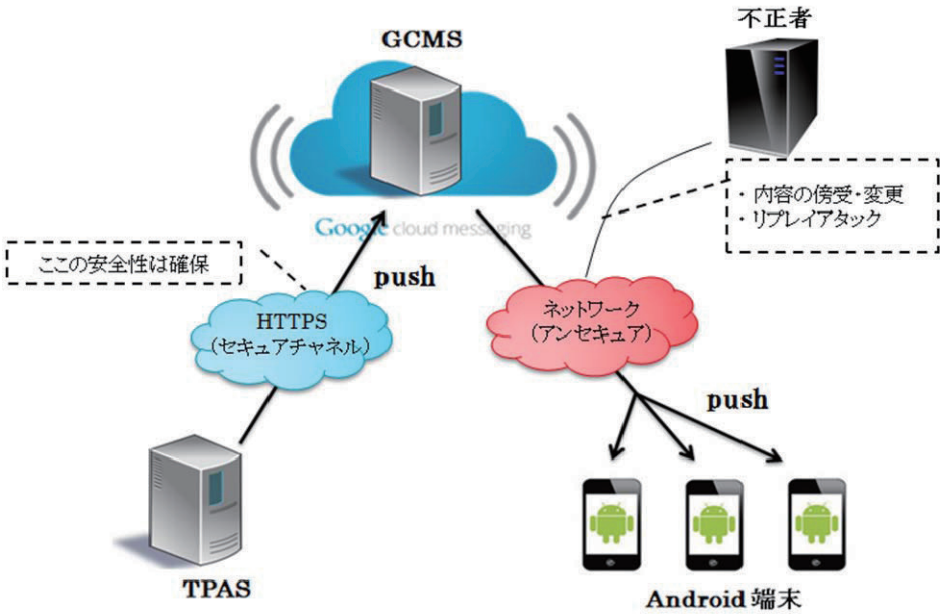

Fig.1 Structure of GCM

本研究では,モバイルクラウドサービス用プッシュ型通信での安全性を向上することを目的として,一方向性のプッシュ型通信においてもメッセージを暗号化し安全性を確立する方法について検討する.

2. モバイルクラウド用プッシュ型通信の仕組みと安全性の課題

2.1 GCM の仕組み

Google Cloud Messaging for Android (GCM) とは, アプリケーション開発者がサーバから Android 端末上の Android アプリケーションにデータを送信できるようにするサービスである. 例えば, サーバから Android アプリケーションへ新しいデータの存在を伝えるメッセージや, ペイロードデータのメッセージ送信を行えるサービスである.

図 1 に GCM の概要を示す. GCM では, クラウドサービスの提供者側である TPAS (Third-Party Application Server) と利用者側である Android 端末に加えて, TPAS からの依頼でプッシュ通信を用いたメッセージ送信を行う GCMS (GCM Server) で構成されている. また, Android 端末では GCMS からメッセージを受け取るためのモジュールが実装されている. このモジュールによって, 受信したメッセージの画面への表示や対象となるアプリケーションの起動など, TPAS が提供したいサービスを実行することができる. TPAS から Android 端末へサービスを提供する手順は次のようになる.

(1) *TPAS* → *GCMS* : *push* (*rID*, *S*, *App*, *M*)

TPAS は, サービスを提供したい Android 端末の登録 ID (*rID*), 対象となるサービス (*S*) またはアプリケーション (*App*), および送信したいメッセージ (*M*) (制御命令やショートメッセージなど) を GCMS へ送信する.

(2) *GCMS* → *AT* : *push* (*M*)

GCMS は, TPAS から受信した内容に従い, 指定された Android 端末 (*AT*) に対してメッセージ (*M*) を送信する.

(3) Android 端末は, GCMS から受信したメッセージに従い, 処理を実行する.

ここで, TPAS がデータを送信すると, GCMS が登録された端末にデータをプッシュする. 従って, Android 端末は GCM を使用することによって, メッセージを受信するためのポーリングをする必要がなくなる. こうしてポーリングをなくすことにより, Android 端末のバッテリー消費を節約することができる. また, GCMS からのメッセージを受信するために, 必ずしも Android アプリケーションが動いている必要もない. 必要な準備 (適切なブロードキャストレシーバとパーミッションのセットアップ) がしてあれば, メッセージが端末にプッシュされると, アプリケーションが起動される[3].

一方, GCM においては, GCMS と Android 端末間でセキュアな通信を行う仕組みを導入していないことや, 一方向性の通信であることから, 既存の相互認証プロトコルを適用できないといった安全性の問題が考えられる.

2.2 GCM における安全性の問題

GCM において, TPAS と GCMS との間では双方向通信が可能な環境下で HTTPS などのセキュアなチャネルを用いて通信が行われる (図 1). これにより, GCMS は TPAS を認証することができ, 成りすましなどにより不正な TPAS によるメッセージ送信を防御すると同時に, この 2 者間で行われる通信の内容に対する傍受・不正な変更やリプレイアタックなどの攻撃を防御できる.

一方, GCMS と Android 端末間では, 一方向性のプッシ

ュ型通信であり，Android 端末との認証が行えない．そのため，セキュアな通信を確立しておかない限り，GCMS から送信されるメッセージに対する傍受・不正な変更などの危険性がある．これは，対象となる Android 端末に関する情報漏洩につながると共に，不正者が他の Android 端末の識別子を何らかの手段で入手していた場合には，そのメッセージを他の Android 端末へ送信するといった不正行為に繋がる危険性を意味している．また，暗号化が行われたとしても，送信されたメッセージを不正者が傍受し，そのメッセージを用いてリプレイアタックを行う危険性が残される．

したがって，GCMS から Android 端末へのプッシュ通信の安全性を向上させるためには，上記の課題に対する対策が必要となる．

3. GCM における相互信頼プロトコル

本研究では，GCM における安全性の問題を解決するために一般的な認証の仕組みと鍵共有の仕組みを組み合わせた相互信頼プロトコルについて検討する．

3.1 相互信頼プロトコルの概要

2.2 節で述べた GCM での安全性の問題を解決するために，GCMS と Android 端末間で相互信頼を確立する手法が提案されている[6]．ここでは，プッシュ型通信の一方向性により，既存の相互認証を導入するのは難しいので，相互認証に代わるプッシュ型通信に適した安全性について以下に示す仕組みを導入した方式が提案されている．

(1) サーバ側からモバイル端末側に対しては，一般的な認証プロトコル(ID による認証やパスワード認証など)による仕組みを導入する．これにより，GCMS から Android 端末への通信に関しては認証が成立するが，反対方向については認証を行うことができない．

(2) モバイル端末からサーバ側に対しては，事前準備としてサーバとモバイル端末が共通鍵を保有し，メッセージを送信する際にその鍵で暗号化を行い送信する．これにより，Android 端末から GCMS サーバ側へのメッセージの傍受・変更を防ぐ．

(3) リプレイアタックを防御するため，送信されるメッセージが有効である期間を定める「リプレイアタック防止コード」を定義する．

以上のことから，正規のサーバから対象となるモバイル端末へメッセージが送信された時のみ有効となり，ユーザに関する情報漏洩，あるいはメッセージの改ざん，対象外のモバイル端末へのメッセージ送信，およびリプレイアタックによるモバイル端末への攻撃を防御できる．

しかし，本方式を実現するためにはいくつかの課題が残る．

- GCMS と Android 端末は，相互信頼を確立するための事前準備として，相互信頼を行うのに必要な情報の設定を行っている．しかし，この内容は第三者に傍受される危険性のある情報のため，セキュアな通信を用いて処理を行うなどの仕組みを考慮しなければならない．
- 共通鍵暗号方式を用いていることから，GCMS は Android 端末 1 台につき 1 つの共通鍵を持つこととなる．その結果，GCMS での鍵管理が複雑化してしまう可能性がある．
- 事前準備で送信された情報は，GCMS・Android 端末側双方で保存される．特に Android 端末においては端末を紛失しやすい等のことからこの情報を安全に保存する仕組みが必要となる．

本研究ではこれらの問題点を解決するためのセキュア相互信頼プロトコルを提案する．本提案方式を考えるにあたり，GCMS に関しては，Google 社が独自に保有しているサーバであるため，GCMS でメッセージを暗号化させる仕組みを提案したところで，Google 社がそれを検討しない限り実装は不可能と考えられる．そこで，本研究では，TPAS と Android 端末間の通信に着目し，TPAS がメッセージを送信する段階でそのメッセージを暗号化して GCMS へ送る．GCMS はこの暗号化されたメッセージを Android 端末へ送ることによって，GCMS と Android 端末間でセキュアな通信網が確立されていなくても，安全性を確保することができると考えられる．次に，これらの考えに基づく実現可能なプロトコルについて検討する．

3.2 公開鍵暗号を用いて ID 情報を共有する方式（提案方式 1）

TPAS と Android 端末が事前準備の段階で公開鍵暗号方式を用いてユーザ側の ID 情報を共有し，それを基にメッセージを暗号化する方式である．TPAS は，自身の秘密鍵とユーザ側の ID 情報を用いてメッセージを暗号化し，Android 端末へ送信する．Android 端末では，TPAS の公開鍵と自身の ID 情報を基にメッセージを復元する．これにより，この ID 情報を知らない第三者がメッセージを送信したり受信して，メッセージを解読したりすることを防御し，TPAS・Android 端末双方の信頼性を確立する．

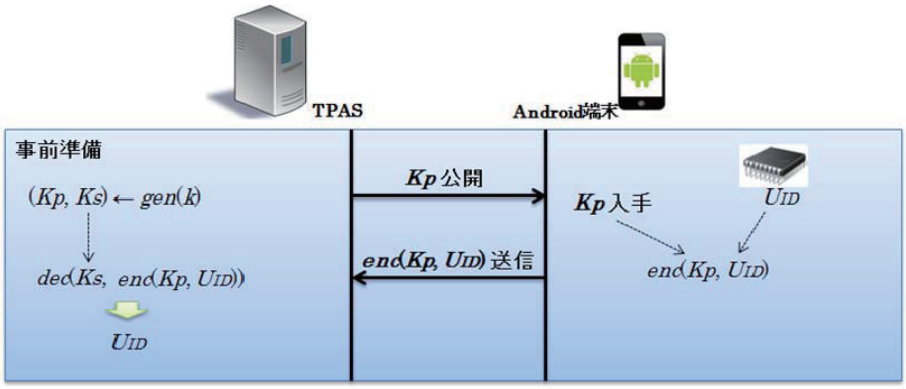

Fig.2 Pre-shared Procedure of proposed method 1

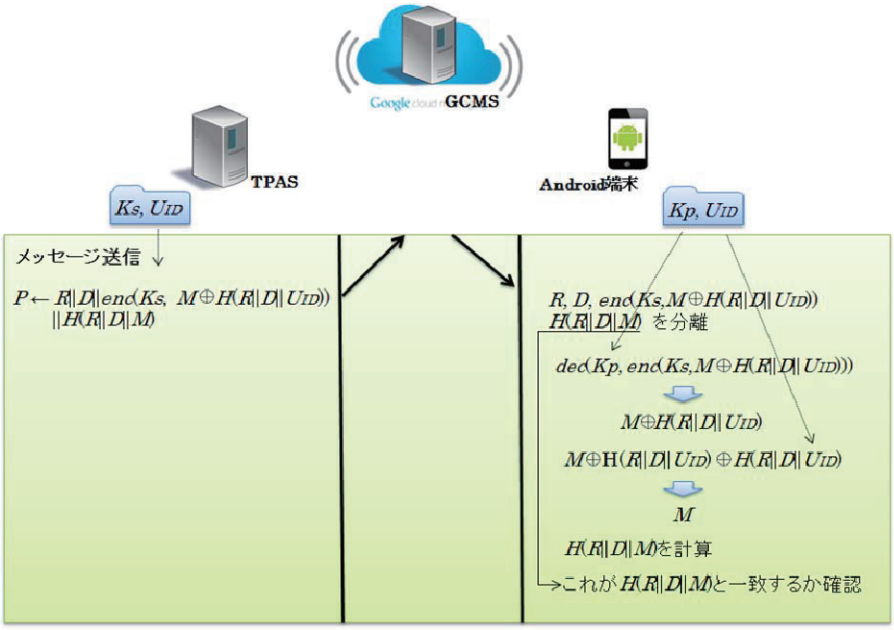

Fig.3 Procedure of proposed method 1

3.2.1 記号の定義

提案プロトコルで用いる記号の意味を以下に示す.

・P: 送信パケット
・M: 送信したいメッセージ
・R: 乱数
・D: リプレイアタック防止コード
・U_{ID}: ユーザ側の ID 情報（端末固有情報，ユーザ ID 情報など）
・Kp: TPAS の公開鍵（*public key*）
・Ks: TPAS の秘密鍵（*private key*）
・||: 前後の値を連接する二項演算子
・⊕: 前後の値の排他的論理和を計算する二項演算子
・$gen(\cdot)$: 鍵生成アルゴリズム
・k: セキュリティパラメータ（暗号の安全性の強度）
・$enc(a, b)$: 鍵 a で b を暗号化
・$dec(a, b')$: 鍵 a で b' を複号
・$H(\cdot)$: 一方向性ハッシュ関数

以下では，x が y にメッセージ M を送ることを “$x \rightarrow y$ M” と表す.

3.2.2 公開鍵暗号に基づく相互認証プロトコル

Step1：事前準備

以下の手順で事前準備を行う．図 2 にその概要を示す.

(1-1) TPAS: $(Kp, Ks) \leftarrow gen(k)$

TPAS は鍵生成アルゴリズム(gen)にセキュリティパラメータ(k)を入力し，公開鍵(Kp)と秘密鍵(Ks)のペアをランダムに生成し，この生成した公開鍵(Kp)を公開する.

(1-2) AT: $enc(Kp, U_{ID})$

Android 端末は，TPAS の公開鍵(Kp)を入手し，この Kp を用いて自身の ID 情報(U_{ID})を暗号化する.

(1-3) $AT \rightarrow TPAS$: $enc(Kp, U_{ID})$

Android 端末は，(1-2)で暗号化した ID 情報を TPAS へ送信する.

GKMT

グループ情報 (GID)	グループ鍵 (GK)	GKに対応するパスワード(P_W)	グループ内ユーザのID情報 UID(G)
G1	GK1	Pw1	UID1-1, UID1-2, UID1-3, …
G2	GK2	Pw2	UID2-1, UID2-2, UID2-3, …
Gi	GKi	Pwi	UIDi-1, UIDi-2, UIDi-3, …
…	…	…	…

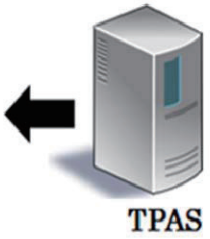

Fig.4 Group Key management table of TPAS

(1-4) TPAS: $dec(Ks,\ enc(Kp,\ U_{ID}))$

TPASはAndroid端末から送られてきた暗号化ID情報を自身が保有している秘密鍵(Ks)を用いて復元する.

(1-5) $TPAS \rightarrow GCMS: U_{ID}$

TPASは復元したユーザ側のID情報(U_{ID})を保存するとともに$GCMS$へ送信する.

Android端末はTPASの公開鍵(Kp)を保存する.

Step2：メッセージ送信

プッシュ型通信を用いた通信時には,以下の手順でメッセージ送信を行う．図3にその概要を示す.

(2-1) TPAS: $P(U_{ID}) \leftarrow R||D||enc(Ks,\ M \oplus H(R||D||U_{ID}))||H(R||D||M)$

TPASは送信したいメッセージ(M),乱数(R),リプレイアタック防止コード(D),ユーザ側のID情報(U_{ID}),自身の秘密鍵(Ks)の値から上記の計算を行い，送信パケット(P)を生成する.

(2-2) $TPAS \rightarrow GCMS: P(U_{ID})$

TPASは(2-1)で生成した送信パケット(P)をGCMSへ送信する.

(2-3) $GCMS \rightarrow AT: P(U_{ID})$

GCMSはTPASから送られてきたパケット(P)を受信し，指定されたAndroid端末(AT)へ送信する.

(2-4) Android端末は受信したパケット(P)からR, D, $enc(Ks,\ M \oplus H(R||D||U_{ID}))$, $H(R||D||M)$をそれぞれ分離する.

(2-5) AT: $dec(Kp,\ enc(Ks,\ M \oplus H(R||D||U_{ID})))$

Android端末は事前準備で保存したTPASの公開鍵(Kp)を用いて上記の計算を行い，$M \oplus H(R||D||U_{ID})$を復元する.

(2-6) AT: $(M \oplus H(R||D||U_{ID})) \oplus H(R||D||U_{ID})$

Android端末は(2-4)で分離した乱数(R),リプレイアタック防止コード(D)および，(2-5)で復元した$M \oplus H(R||D||U_{ID})$および，自身のID情報(U_{ID})を用いて上記の計算を行い，メッセージ(M)を復元する.

(2-7) Android端末は(2-4)で分離した乱数(R),リプレイアタック防止コード(D)および,(2-6)で復元したメッセージ(M)を用いて$H(R||D||M)$の値を求める．この値が(2-4)で分離したH($R||D||M$)と一致するかを確認し，一致したらメッセージ(M)は有効なものと判断する.

3.3　グループ鍵を用いた認証方式（提案方式2）

3.2で提案した方式を用いてTPASがプッシュ通信を行う場合,複数のAndroid端末へ同一のメッセージを送信する際もAndroid端末1台ずつにメッセージを暗号化して送信する必要がある.

例えば，TPASがAndroid端末10台へ同一のメッセージを送りたい場合，現在のGCMのサービスでは，TPASは送りたいメッセージと一緒にAndroid端末10台分の識別子をGCMSへ送ることによって，GCMSが指定された10台のAndroid端末へメッセージを届けるという仕組みになっている．そのため，Android端末が何台あろうとTPASとGCMS間の通信は一回の通信で完了する.

しかし，3.2で提案した方式の場合，TPASはメッセージを暗号化するのにユーザ側のID情報を使用する.そのため，TPASがAndroid端末10台に同一のメッセージを送りたい場合，TPASはAndroid端末1台ずつユーザ側のID情報を用いてメッセージを暗号化してGCMSへ送信するので，単純にTPASとGCMS間の通信は10回行う必要がある．その結果，TPAS側でメッセージを送信する際に手間がかかってしまう可能性がある．また，TPASとGCMS間の通信トラフィックが多くなり,その間の通信が混雑する可能性がある.

そこで,グループ鍵を用いてグループ単位でメッセージを暗号化する方式を提案する．ここでTPASは，図4に示すようなグループ鍵管理テーブル(GKMT: Group-Key Management Table)を持ち，グループごとのグループ鍵(GK)とそれに対応するパスワード(P_W)とグループ内ユーザのID情報(U_{ID})を，各グループのグループ情報(G_{ID})に紐づけて管理する.

TPASはグループ鍵とそれに対応するパスワードをユーザ側のID情報から生成したセッション鍵を用いて暗号化し，グループ内のAndroid端末へ送信する．Android端末では，自身のID情報から生成したセッション鍵を用いてグループ鍵とパスワードを復元する．TPASがメッセージを送る際はグループ鍵で暗号化して送信することによって,グループ内のAndroid端末全てに送りたい場合もTPAS

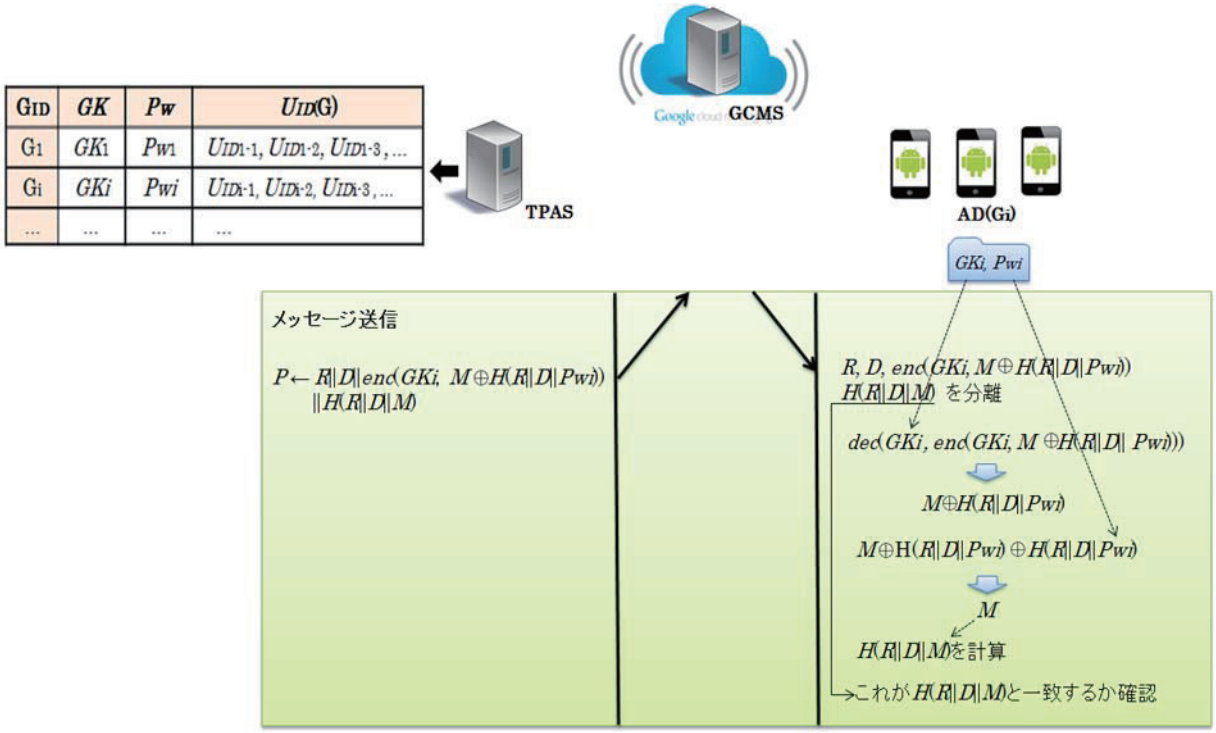

Fig.5 Procedure of proposed method 2

と GCMS 間の通信は 1 回の通信で完了する． Android 端末ではグループ鍵とパスワード認証を基にメッセージを復元する．これにより，このグループ鍵やパスワードを知らない第三者がメッセージを傍受して解読したりすることを防御し，TPAS・Android 端末双方の信頼性を確立することができる．

3.3.1 記号の定義

本提案プロトコルで改めて用いる記号を以下に示す．その他の記号は，3.2.1 での定義と同様である．

・GKi（group key）：グループ i において通信データを暗号化するためのグループ鍵

・SKi（session key）：Android 端末にグループ鍵 GKi を送る際に用いるセッション鍵

・Pwi：グループ鍵 GKi に対応するパスワード

・$AT(Gi)$：グループ i に属する Android 端末

3.3.2 グループ鍵を用いた相互認証プロトコル

STEP1：事前準備

グループ単位でのメッセージ送信を行う際には，以下の手順で事前準備を行う．

(1-1) 3.2.2 節で説明した提案方式 1 の事前準備（手順 1～5）と同等の操作を行う．

(1-2) TPAS: $SKi \leftarrow gen(U_{ID}i)$

TPAS は各 Android 端末の ID 情報（$U_{ID}i$）を用いて上記の計算を行い，各 Android 端末とのセッション鍵（SKi）を生成する．

(1-3) TPAS: $enc(SKi, GKi||Pwi)$

TPAS は各 Android 端末が属しているグループのグループ鍵（GKi）とそれに対応するパスワード（Pwi）を(1-2)で生成したセッション鍵（SKi）で暗号化する．

(1-4) $TPAS \rightarrow ATi$: $enc(SKi, GKi||Pwi)$

手順(1-3)で暗号化したものをグループ i に属している各 Android 端末（ATi）へ送信する．

(1-5) ATi : $SKi \leftarrow gen(U_{ID}i)$

Android 端末は自身の ID 情報（$U_{ID}i$）を用いて上記の計算を行い，セッション鍵（SKi）を生成する．

(1-6) ATi : $dec(SKi, enc(SKi, GKi||Pwi))$

Android 端末は(1-5)で生成したセッション鍵（SKi）を用いて上記の計算を行い，TPAS から送られてきた情報からグループ i のグループ鍵（GKi）とそれに対応するパスワード（Pwi）を復元する．

(1-7) Android 端末は(1-6)で復元した GKi と Pwi を保存する．

STEP2：メッセージ送信

プッシュ型通信を用いてグループ単位でメッセージを送りたい場合，以下の手順でメッセージ送信を行う．図 5 に TPAS がグループ i に向けメッセージを送り，グループ i に属している Android 端末でこのメッセージを復元する手順の概要を示す．

(2-1) TPAS: $P(Gi) \leftarrow R||D||enc(GKi, M \oplus H(R||D||Pwi))||H(R||D||M)$

TPAS は送信したいメッセージ（M），乱数（R），リプレイアタック防止コード（D），グループ i のグループ鍵（GKi）とそれに対応するパスワード（Pwi）の値から上記の計算を行い，送信パケット（P）を生成する．

(2-2) $TPAS \rightarrow GCMS$: $P(Gi)$

TPAS は(2-1)で生成した送信パケット（P）を GCMS へ送信する．

(2-3) $GCMS \rightarrow AT(Gi) : P(Gi)$
GCMS は TPAS から送られてきた送信パケット(P)を受信し，指定されたグループ i に属している Android 端末 $AT(Gi)$ へ送信する．
(2-4) グループ i に属している Android 端末は，受信したパケット(P)から R，D，$enc(GKi, M \oplus H(R||D||Pwi))$，$H(R||D||M)$ をそれぞれ分離する．
(2-5) AT: $dec(GKi, enc(GKi, M \oplus H(R||D||Pwi)))$
グループ i に属している Android 端末は，(2-4)で分離した $enc(GKi, M \oplus H(R||D||Pwi))$ を事前準備で保存したグループ鍵(GKi)を用いて上記の計算を行い，$M \oplus H(R||D||Pwi)$ を復元する．
(2-6) AT: $M \oplus H(R||D||Pwi) \oplus H(R||D||Pwi)$
グループ i に属している Android 端末は，(2-4)で分離した乱数(R)，リプレイアタック防止コード(D)および，(2-5)で復元した $M \oplus H(R||D||Pwi)$ および，事前準備で保存したパスワード(Pwi)を用いて上記の計算を行い，メッセージ(M)を復元する．
(2-7) グループ i に属している Android 端末は，(2-4)で分離した乱数(R)，リプレイアタック防止コード(D)および，(2-6)で復元したメッセージ(M)を用いて $H(R||D||M)$ の値を求める．この値が(2-4)で分離した $H(R||D||M)$ と一致するか確認し，一致したらメッセージ(M)は有効なものと判断する．

4. 評価と考察

ここでは，提案方式の評価を行うために，GCM の危険性について，いくつかの項目に分けて考察を行うとともに，3.1 節で示した従来方式の問題点と本提案方式の比較を行う．

4.1 安全性

3.2.2 節，および 3.3.2 節において，提案方式の手順が示されている．この手順に関する安全性について，いくつかの項目に分けて考察を行う．

(1) Android 端末によるサーバの認証
・提案方式 1： Android 端末は，事前準備により TPAS と自身の ID 情報を共有している．したがって，Android 端末は GCMS から受信したデータにその ID が提示されていれば正規の TPAS からのメッセージであると判断する．
・提案方式 2： Android 端末は，事前準備により TPAS からグループ鍵とパスワードを受信し，保存している．したがって，Android 端末はグループ鍵で暗号化されているデータを復元し，さらにパスワード認証の仕組みを利用して，正規の TPAS から送信されたメッセージであることを判断する．

(2) サーバによる Android 端末の信頼
・提案方式 1： TPAS は事前準備により Android 端末との間でユーザ側の ID 情報を共有している．したがって，その ID 情報を送信するデータに含めることによって，対象となる Android 端末のみでメッセージが有効となる．
・提案方式 2： TPAS は，事前準備により Android 端末との間でグループ鍵とそれに対応するパスワードを共有している．したがって，このグループ鍵で送信するデータを暗号化し，さらにそのデータにパスワードを含めることによって，このグループ鍵やパスワードを知らない第三者でメッセージが有効となることを防御している．

Table 1 Security verification of proposed methods

GCMにおいて考えられる危険性	提案方式1	提案方式2
Android端末によるサーバの認証	○	○
サーバによるAndroid端末の信頼	○	○
不正者による通信データの傍受・変更	△	○
リプレイアタック	○	○

(3) 不正者による通信データの傍受・変更
・提案方式 1： 通信データは TPAS の秘密鍵で暗号化されており，さらに送信データに対するハッシュ値が送信されている．したがって，送信されたデータを不正者によって変更されてもそのハッシュ値により検出できる．しかし，メッセージの復元に用いる TPAS の公開鍵は誰でも入手可能な情報のため，事実上送信データに対するハッシュ値のみで安全性を確保していることになる．そのため，安全性が高いとは言い難い．
・提案方式 2： 通信データはグループ鍵によって暗号化されており，さらに送信するメッセージやパスワードに対するハッシュ値が送信されている．したがって，そのグループ鍵やパスワードを知らない限り通信データの傍受は行えず，メッセージやパスワードを変更してもそのハッシュ値により検出できる．

(4) リプレイアタック
提案方式 1，2 とも通信データにリプレイアタック防止コードが含まれており，メッセージの有効期限が定められている．有効期限を定めるコードの内容については，以下のような例が上げられる．
・シリアル番号： 通信データにシリアル番号を付与し，その番号をモバイル端末で記録しておく．モバイル端末は同じ番号の通信データを受信したら，そのメッセージを破棄する．

・時刻情報： 通信データにメッセージが有効である期限を示す時刻情報を付与する．モバイル端末はその時刻情報を超過したタイミングで通信データを受信したら，そのメ

ッセージを破棄する.

厳密にメッセージのリプレイアタックを防御するのであれば, シリアル番号を用いる方式が望ましいが, モバイル端末で今まで受信したメッセージのシリアル番号を保存･管理する必要があり,負荷が大きくなる可能性がある. 一方, 時刻情報を用いる方式は, 通信データに示されている時刻情報と現在時刻との比較を行い, 時刻情報を保存・管理する必要がないため, 負荷は少ないと考えられる. しかし,その時刻情報で示された期限内であればリプレイアタックが行われる危険性がある. モバイル端末のメモリ容量やバッテリーの持続時間を考慮すると,シリアル番号の方式よりも時刻情報の方式が適していると思われるが,その場合は有効期限の適切な設定が必要である[6].

以上をまとめると表 1 のような結果になる. 表 1 において, ○は従来の問題点を解決できたと判断した場合, △は従来通りか従来同様と判断した場合とする.

4.2 従来方式の問題点と本提案方式の比較

本節では, 3.1 で説明した従来方式の問題点と本提案方式の比較を行う.

(1) 従来方式の問題点 1： GCMS と Android 端末間での事前準備の段階で ID や共有鍵が第三者に傍受される恐れがある.

・提案方式 1： Android 端末は TPAS の公開鍵で自身の ID 情報を暗号化して送信する. そのため, TPAS の公開鍵で暗号化された ID 情報は, TPAS が保持している秘密鍵でしか復号できないため, 第三者に傍受されることはない.

・提案方式 2： Android 端末は TPAS の公開鍵で自身の ID 情報を暗号化し送信する. また, TPAS がグループ鍵とパスワードを送信する際は, その ID 情報から生成したセッション鍵を用いて暗号化しているため,第三者に傍受されることはない.

(2) 従来方式の問題点 2： サーバ側での鍵管理が複雑化する可能性がある.

・提案方式 1： TAPS は自身の秘密鍵でメッセージを暗号化して送信するため, Android 端末が何台あろうと自身の秘密鍵さえ保持しておけば良く,鍵管理が複雑化することはない.

・提案方式 2： TPAS は Android 端末グループごとのグループ鍵を持つこととなる. そのため, TPAS はグループ鍵管理テーブルによって鍵管理が複雑化する問題を軽減させてはいるが,このグループ鍵管理テーブルを安全に守る仕組みが必要となる.

(3) 従来方式の問題点 3: 事前準備で送信された情報は, GCMS・Android 端末側双方で保存される. 特に Android 端末においては紛失しやすい等のことからこの情報を安全に保存する仕組みが必要となる.

Table 2 Comparision of proposed methods

従来方式の問題点	提案方式1	提案方式2
事前準備の共有情報が第三者に傍受される	○	○
サーバ側での鍵管理の複雑化	○	△
端末側でのサーバ側の情報管理の安全性	○	△

・提案方式 1： メッセージの復元には TPAS の公開鍵と自身の ID 情報を使用するが, TPAS の公開鍵はいつでも入手可能な情報のため, 自身の ID 情報さえ安全に保存しておけば良く,サーバ側の重要な情報等は Android 端末では保持しない.

・提案方式 2： Android 端末は自身の属しているグループのグループ鍵とパスワードを保持する. もしグループ内の Android 端末 1 台でグループ鍵やパスワードが漏洩した場合,そのグループに属している全ての端末のグループ鍵やパスワードの更新を行わなければならない. そのため, 従来同様 Android 端末ではこの情報を安全に保存する仕組みが必要となる.

以上をまとめた結果を表 2 に示す. ここで, ○は従来の問題点を解決できたと判断した場合,△は従来通りか従来同様と判断した場合とする.

5. まとめ

本研究では,プッシュ型通信での安全性を向上させることを目的として,相互認証が行えない一方向性のプッシュ型通信においても,メッセージを暗号化し安全性を確立する相互信頼プロトコルを提案した. これにより, モバイルクラウドサービスにおける安全性の向上が期待できる. 本提案方式は,事前準備で情報を共有する手段として公開鍵暗号方式を用いていることにより,従来の問題点でもある事前準備の段階で情報が第三者に傍受される危険性を防ぐことができる. また, 提案方式 1 の場合は, TPAS は自身の秘密鍵でメッセージを暗号化するため,自身の秘密鍵さえ保持しておけば良く,サーバ側の鍵管理が複雑化する問題も解決できる. 提案方式 2 の場合は, TPAS にグループ鍵管理テーブルを持たせることにより, 複数の Android 端末の情報を一元管理することができ, 鍵管理の複雑化を軽減させることができる. 本提案方式において, 今後の検討が必要だと思われる課題について以下に示す.

- 提案方式 1・2 共に公開鍵暗号方式を採用している. 公開鍵暗号方式は, 共通鍵暗号方式に比べ, アルゴリズムが非対称であること, 数学的に難しい処理が多いことなどにより, 比較的多くの計算量を要するその結果, Android 端末での処理速度の低下や, 消費電力が多くなる可能性が考えられる.

- 提案方式2の場合，グループ単位での認証としてグループ鍵を用いてデータを暗号化するが，このグループに新しいユーザが参加する場合やグループ内のユーザが離脱する場合，グループ鍵の更新及び再配布を行わなければならない．そのため，ネットワーク上で鍵の配布・更新を行う際には，トラフィックの問題やそれに伴うデータ通信への影響の問題を考慮する必要[7]がある．

参考文献

[1]調査報告書“モバイルクラウドサービス及び端末の市場動向”，H.I. Business Partners，（2010）
[2]市場調査レポート “国内プライベートクラウド市場2012年の実績と2013年～2017年の予測”,IDC Analyze the Future，（2013）
[3]http://developer.android.com/google/gcm/index.html“Google Cloud Messaging for Android | Android Developes”（2013）
[4] 鈴木博文，大岩寛，高木浩光，渡辺創，“フィッシング防止のためのHTTP パスワード相互認証プロトコル”，暗号と情報セキュリティシンポジウム(SCIS2008),3C2-5,（2008）
[5] 松尾真一郎，松尾俊彦，大久保美也子，鈴木幸太郎，“省リソースデバイス向け相互認証プロトコル”，暗号と情報セキュリティシンポジウム（SCIS2008），3E3-4,（2008）
[6] 稲村勝樹，“モバイルクラウド用プッシュ型通信における相互信頼の確立手法”，暗号と情報セキュリティシンポジウム（SCIS 2012），3D1-1，(2012)
[7]朴美娘，岡崎直宣，妹尾商一郎，“ダイナミックグループ暗号通信のための鍵更新システムの提案と評価”電子情報通信学会論文誌 Vol. J87-D-I No.10 pp.907-919,（2004）