内部構造区画化によるプライバシポリシに準拠した Android アプリ開発方式

小林 真也† 鈴木 富明† 可児 潤也† 川端 秀明‡ 西垣 正勝†

†静岡大学大学院情報学研究科 432-8011 静岡県浜松市中区城北 3-5-1

‡株式会社 KDDI 研究所 356-8502 埼玉県ふじみ野市大原 2-1-15

あらまし スマートフォンアプリにおけるプライバシ情報の安易な取扱いが問題になる中，プライバシ保護が意識されたアプリ開発が求められてきている．本稿では，開発者がアプリ向けプライバシポリシに準じた形でプログラムの内部構造を「ドメイン」と呼ばれる単位に区画化し，プライバシ情報の伝搬範囲を限定した形でアプリ開発を行う仕組みを実現する．これにより，Privacy by Designを意識されたアプリ開発が行われるとともに，各ドメイン間のメッセージパッシングを静的検査することで，アプリがプライバシポリシに準じていることが確認できるため，アプリとプライバシポリシの整合性検査も容易となる．

Privacy-by-Design Development of Android Application by Domain Separation based on Privacy Policy

Shinya Kobayashi† Tomiaki Suzuki† Junya Kani†
Hideaki Kawabata‡ Masakatsu Nishigaki†

†Graduate School of Informatics, Shizuoka University
3-5-1 Johoku, Naka, Hamamatsu, Shizuoka, 432-8011, JAPAN

‡KDDI R&D Laboratories, Inc.
2-1-15 Ohara, Fujimino, Saitama, 356-8502, JAPAN

Abstract Recently privacy information leakage by Android applications becomes very serious, and therefore the protection has become more important. In this paper, we introduce a mechanism to limit the propagation range of privacy information in an Android application codes by properly separating the internal structure of the codes into ”domains”. As a result, it can be confirmed that the Android application is in accordance with the privacy policy, just by static analysis of the message passing between each domain. Therefore, this “privacy by design” approach helps to make consistency check between Android application and privacy policy easier.

1 はじめに

近年の不正アプリによるプライバシ情報（端末情報）の漏えいが数多く報告されていることを受け，最近ではモバイル端末アプリに対してもプライバシポリシを表示することが総務省により推奨されている[1]．これとともにアプリマーケットがアプリのプライバシポリシ作成を支援し，

第三者機関としてアプリのプライバシポリシを検査するような動きも始まっている[2].

しかし，アプリ開発者がアプリのプログラムとプライバシポリシを別々に作成する形になっている以上，開発者の誤解やミスによってアプリと不整合なプライバシポリシが記述される可能性が残る. プログラムの静的／動的解析だけではアプリ内におけるプライバシ情報の利用状況を完全に把握することは容易ではなく，端末またはアプリマーケット側で，アプリとプライバシポリシの整合性を検査するには限界がある.

そこで本稿では，プログラムの内部構造を「ドメイン」と呼ばれる区画で分離することによって，プログラム内での端末情報の伝播範囲を限定する仕組みを導入する. 各ドメインにおいては，それぞれプライバシ情報の取得と外部送信における制約が設けられている. 開発者は，プライバシポリシに準じた形でアプリ内の各メソッドを各ドメインに分類した上でアプリ開発することによって，プライバシ情報は適切なドメイン内に囲い込まれる. この結果，各ドメイン間のメッセージパッシングのみを静的検査するだけで，アプリがプライバシポリシに準じていることが確認できるため，アプリとプライバシポリシの整合性（アプリがプライバシポリシ通りの動作をするか）検査も容易となる.

本稿では，現在スマートフォンの世界市場において高いシェアを持つ Android OS を提案方式の適用対象として採用する.

2　プライバシポリシの運用

2.1　課題

モバイル端末アプリにおける端末情報（プライバシ情報）の取り扱いの健全化に対する社会的な要求を受け，総務省によりスマートフォンプライバシーイニシアティブが取りまとめられている[1]. その中で，スマートフォン向けアプリに対してもプライバシポリシ（アプリがどの情報を何の目的で使用するか）を表示することが推奨されている. しかし，プライバシポリシの運用にあたっては，下記の課題がある.

Ⅰ.　アプリ開発者側の課題：

アプリ開発者は，アプリを作成する際に，そのプライバシポリシについても記述しなければならない. すなわち，アプリ開発にあたっての負担がその分増大することになる（課題 1a）. また，アプリ開発の作業とプライバシポリシ作成の作業が分離されているため，アプリ開発の際にプログラムにバグが混入してしまった場合や，プライバシポリシ作成の際に誤りが混入してしまった場合などに，アプリとプライバシポリシの間の整合性（一貫性）が崩れることなる（課題 1b）. 更には，不正なアプリ開発者が，実際のアプリの動作とは異なるプライバシポリシを故意に作成する可能性もある（課題 1c）.

Ⅱ.　ユーザ側の課題：

ユーザがアプリを自身の端末にインストールする際に必ずプライバシポリシが表示されるようになっていたとしても，ユーザがその内容を理解できなかったり，十分確認せずにインストールを許可したりする恐れがある[1]（課題 2）.

Ⅲ.　マーケット側の課題：

プログラムの静的／動的解析によってアプリ内における端末情報（プライバシ情報）の利用状況を完全に把握することは容易でない. このため，アプリマーケット側で，アプリとプライバシポリシとの整合性（アプリが本当にプライバシポリシ通りの動作をするのかどうか）を確認することが難しい（課題 3）. 同様の理由で，アプリとプライバシポリシの整合性をユーザの端末内で検査することも容易ではない.

2.2　既存研究および対策

課題 1a に対しては，アプリ開発者のプライバシポリシ作成を支援する方法が提案されている[4]. 文献[4]では，チェックシート形式の入力方法を採用することによって，開発者がチェックシートにチェックを入れるだけでスマートフォンプライバシーイニシアティブに沿ったプライバシ

ポリシが作成される仕組みを実装している．また，アプリ開発者にプライバシポリシ作成に対するインセンティブを与えるというアプローチも，課題 1a の対策として有効だと考えられる．文献[5]では，実際に，プライバシポリシを正しく記述しているアプリの開発者に対してアプリマーケットが奨励金を支払うという試みを運用した結果，プライバシポリシの充足率の向上が確認されたことが報告されている．

課題 1b，課題 1c の解決には，すべてのプライバシ情報取得 API に対してプライバシポリシに準じた機能制約を与えることにより，アプリ開発とプライバシポリシ作成を一体化させるというアプローチが提案されている[5]．しかし，スマートフォンアプリの利用目的は非常に多岐に渡り，用意すべき機能制約 API の種類が無数に及ぶため現実的な対策となり得ていない．

課題 2 に対しては，文献[3]の中で，ユーザにプライバシポリシを分かりやすく表示する工夫についても示されている．また，関連研究として，アプリに関するワーニング情報やパーミッション情報をユーザに分かりやすく説明するための方法が種々提案されている[6][7]．これら関連研究の知見は，プライバシポリシの表示に対しても活用できるのではないかと考えられる．

課題 3 に対しては，文献[8]が，アプリコードの静的検査によって広告モジュールやプライバシ関連 API の使用を確認し，プライバシポリシに記述するべき内容を提示する機構を提案している．また，アプリ内の端末情報（プライバシ情報）に「色付け」をすることによって，端末情報のトレースを可能にする方法が提案されている[9]．この技術を活用して，アプリにおける端末情報の利用がプライバシポリシに準じているかどうかの検査を端末側で動的に実施することができる可能性がある．

2.3 問題解決に向けてのアプローチ

前述より，現時点では既存対策において課題 1b，1c，2，3 の解決に研究の余地があると考える． 課題 1b，1c については，アプリ開発とプライバシポリシ作成を一体化するアプローチが有用であるが，文献[5]のようにプライバシ情報取得 API のそれぞれに対してすべて個別に対応する方式は現実的ではない．そこで本稿では，アプリの内部をプライバシポリシに準じて区画化し，プライバシ情報を適切なドメイン（区画）に封じ込めることで，各ドメイン内でのプログラム開発の自由度を保ちながら，プライバシポリシと一体化したアプリ開発を可能とする方式を提案する．この結果，各ドメイン間のメッセージパッシングを静的検査するだけで，アプリとプライバシポリシの整合性（アプリがプライバシポリシ通りの動作をするか）検査も容易となり，課題 2，3 の問題についても改善される．提案方式の具体的な方法については次章で説明する．

3 提案方式

3.1 コンセプト

一般的に，モバイル端末用の OS では端末番号や位置情報などのプライバシ情報ごとに情報取得用の API が用意されている．アプリは，必要な情報に対応する API をその都度呼び出すことによって，OS を介して当該プライバシ情報を入手することができる．アプリは，取得したプライバシ情報を利用して，ユーザにサービスを提供する．ここで，「アプリが，これらのプライバシ情報を，プライバシポリシに記された通りの方法によって適正に利用しているか」が重要となる．プライバシ情報取得 API の呼び出しを監視することによって，アプリがいつどのプライバシ情報を入手したかについては確認することは可能である．しかし，アプリに取得されたプライバシ情報は，その後，アプリ内で必要に応じて任意の形に加工されながら利用されるため，アプリ内のプライバシ情報の伝搬を完璧かつ効率的にトレースすることは困難である．

そこで本稿では，アプリの内部構造をプライバシ情報ごとに区画化して分離する開発方法を検討する．アプリ開発者がプライバシポリシに準じた形でアプリ内のメソッドをそれぞれの区

画(ドメイン)内に囲い込むことにより，各ドメインの入出力さえ監視すれば，アプリのプライバシ情報の取扱いがプライバシポリシと整合しているか否かのチェックが可能になる．プライバシ情報がドメインを越えて取り扱われない限り，ドメイン内でいかに加工されようとも不問である．このため，アプリ開発におけるプログラミングの自由度も(ドメインを越えたプライバシ情報の受け渡しはしないという制約内で)維持される．

3.2　区画化ルール

厳密には，プライバシ情報ごとにアプリの内部構造を区画化する必要がある．しかし，ユーザが確認したい内容は「自身のプライバシ情報の内，どの情報がどのサーバに送信されているか」であり，プライバシ情報の外部送信先ごとにアプリの区画化をすれば十分である．故に，

- ドメイン 1:ドメイン内の情報を外部サーバ 1 に送信することが許可されているメソッド群
- ドメイン 2:ドメイン内の情報を外部サーバ 2 に送信することが許可されているメソッド群
- :

という形で，複数のドメインが規定されることになる．また，同時に，

- ドメイン 0:ドメイン内の情報をいかなる外部サーバにも送信することが許可されていないメソッド群
- ドメイン∞:ドメイン内の情報を任意の外部サーバに送信することが許可されているメソッド群

を規定する．

各ドメインの送信制限は，具体的には，外部サーバ m を通信先として指定した形での情報送信用 API の呼び出しを，ドメイン m 内のメソッド群に限定することによって実現される．ただし，ドメイン 0 内のメソッド群は，外部サーバとの通信のための情報送信用 API の呼び出しは一切許可されず，ドメイン∞内のメソッド群は，任意の情報送信用 API の呼び出しが許可される．以下，これを「ルール 1」と呼ぶ．

各ドメインの情報取得は，アプリのプライバシポリシに基づいて制御される．例えば，「位置情報をサーバ n に送信する」ことがプライバシポリシに記述されているアプリにおいて，位置情報取得 API を呼出すことができるのはドメイン n 内のメソッド群に限られる．ただし，ドメイン∞内のメソッド群は，すべてのプライバシ情報取得 API 呼出しが一切許可されず，ドメイン 0 内のメソッド群は，任意のプライバシ情報取得 API 呼出しが許可される．なお，プライバシ情報に該当しない情報は，すべてドメイン∞の情報として扱われる．以下，これを「ルール 2」と呼ぶ．

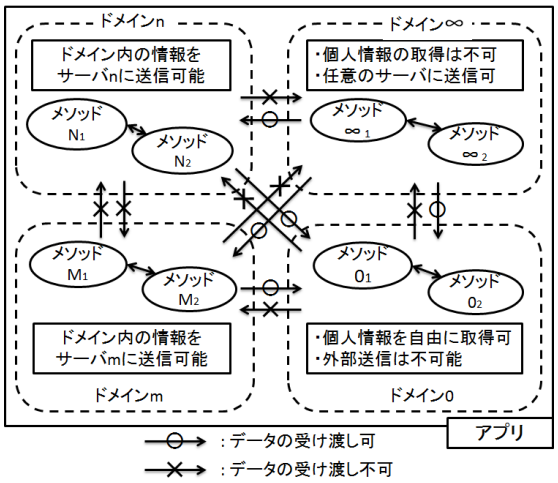

図 1. 区画化ルール

ドメイン間のメッセージパッシングに対しては，ドメイン n とドメイン m(n≠m)間におけるデータの送受信(引数有りのメソッド呼び出しおよび戻り値有りのメソッド呼び出し)を禁止する．ただし，ドメイン n からドメイン 0 へのデータの送信(ドメイン n 内におけるドメイン 0 内の引数有りのメソッド呼び出し，ドメイン 0 内におけるドメイン n 内の戻り値有りのメソッド呼び出し)，ドメイン∞からドメイン n へのデータの送信(ドメイン∞内におけるドメイン n 内の引数有りのメソッド呼び出し，ドメイン n 内におけるドメイン∞内の戻り値有りのメソッド呼び出し)については許可される．以下，これを「ルール 3」と呼ぶ．

アプリの内部構造の区画化に関する上記のルールを図 1 にまとめた．図 1 のルールが守られる限り，アプリ内での各プライバシ情報はそれぞれ適切なドメインの中に閉じ込められるこ

とになるため，アプリとプライバシポリシの整合が保証される．換言すれば，提案方式を利用して開発されたアプリに関しては，図 1 のルールが守られていることが確認できれば，そのアプリはプライバシポリシと整合していることが保証される．図 1 のルールのチェックは，アプリのソースコードがある場合はコードの静的検査によって確認可能である．このため，（アプリの安全性検証を徹底させるなどの目的で）アプリのソースコードの提出を義務付けているアプリマーケットなどにおいては，アプリのコードの静的検査という簡便な方法によって，プライバシポリシに対するアプリの整合性を確認可能となる．

4　Android アプリ開発への適用

提案方式を Android アプリ開発に適用する際の設計・開発方法について以下に示す．

4.1　区画化の方針

Android アプリ開発においては，通常，Activity（アプリの画面を構成するコンポーネント）および Service（アプリをバックグラウンドで動作させるためのコンポーネント）を単位として，プログラムの構造が設計されることになる．ここで，フォアグラウンドプロセスが「画面」を構成するのに対し，バックグラウンドプロセスは「隠し画面」と捉えることができる．この観点に立つと，Android アプリのコンポーネント設計は，アプリの画面遷移を決定することとほぼ等価となると考えられる．そこで，区画化のドメインもコンポーネント単位で設計する方針を採用する．

4.2　区画情報の記述

提案方式のアプリ開発においては，開発者が，アプリのプライバシポリシに応じて，Android マニフェストファイル内にメタデータ(meta-data)タグを用いて各コンポーネント(画面)の区画情報を記述する．区画情報の記述は，アプリのパーミッション定義の記述を踏襲した書式とした．これは，アプリ開発者の利便性に配慮し，従来の Android アプリ開発との親和性の確保を狙ったためである．

4.3　コーディングにおける留意

アプリ開発者は図 1 のルール（主に区画間のメッセージパッシング）に注意してアプリ開発を行わなければならい．同じ区画内のメソッドどうしであれば任意のメッセージパッシングが可能であり，設計時に適切に区画を設定すればある程度のコーディングの自由度が保証される．

提案方式では，区画を越えるメッセージパッシングが禁止されている点が，アプリ開発者に課せられる制約となる．この制約をできるだけ局所化するために，アプリ開発者には，「プライバシ情報については，可能な限りこれを抱えこまない」という点に留意して開発を行ってもらう．例えば，プライバシ情報を不用意にグローバル変数に格納することはしない，プライバシ情報を実際に使用する時点になるまでプライバシ情報取得メソッドを呼び出さない（プライバシ情報に関しては，外部送信メソッドに可能な限り近いポイントで情報取得メソッドを呼ぶ）[1]という点に留意してコーディングするよう開発者に促す．

5　既存アプリの分析と適用評価

提案方式は，プライバシポリシに応じたドメイン管理をアプリ開発者に要求する．また，提案方式の導入により，アプリとプライバシポリシの整合性検査（アプリの静的解析）が容易となる．提案方式が既存のアプリ開発および解析に与える影響について考察するために，プライバシ情報を外部送信するオープンソースアプリを用いて，アプリ内情報の外部送信方法（3.2 節のルール 1），プライバシ情報の取得方法（3.2 節のルール 2），アプリ内の情報処理方法（3.2 節のルール 3）の観点から分析を行った．

1 プライバシ情報取得 API が呼び出されるまではドメイン内にプライバシ情報は存在しないため，ドメイン n 内のメソッドであっても，情報取得 API が呼び出される前に処理を終えてしまうメソッドに対してはドメイン∞（プライバシ情報を含まないため，ドメイン内の情報を任意の外部サーバに送信して問題ない）と同等の扱いが可能である

5.1　分析対象のアプリケーションの収集

分析に利用するアプリは F-Droid[10]というオープンソースソフトウェアを配信するウェブサイトから収集した. 分析時(2015 年 5 月時点)において F-Droid には 1508 のアプリが登録されており, その中からプライバシ情報の取得と外部送信を行うことのできるパーミッションを持つアプリを無作為に抽出して分析を行った.

外部送信を行うために必要な INTERNET パーミッションと端末情報を取得するために必要な READ_PHONE_STATE パーミッションを両方持つアプリは 1508 個中 115 個(群 1)であった. また, INTERNET パーミッションと GPS により位置情報を取得するために必要な ACCESS_FINE_LOCATION パーミッションを持つアプリは 1508 個中 121 個(群 2)であった.

今回は, 群 1, 2 に含まれるアプリの中から, 広告ライブラリを含まないアプリを 10 サンプル無作為に抽出し, コードを分析した.

5.2　外部送信方法の観点からの分析

アプリがプライバシ情報を外部に送信する方法には大別して 3 種類が存在することが確認された. それぞれの分類について, ルール 1(ドメイン n 内の情報は外部サーバ n にのみ送信可能)の観点からの分析を以下に示す.

(1). 標準ライブラリ型

Apache HttpClient や AndroidHttpClient などの Java や Android SDK に備わっている標準ライブラリを用いて, アプリ内情報を外部送信するタイプを「標準ライブラリ型」と分類する.

アプリ開発においては, 開発者は「外部送信メソッドを呼び出す際にはルール 1 を守る」という点にだけ注意すれば良い. また, 標準ライブラリ型の場合, 外部送信メソッド名は決まっているので, 外部送信メソッドの引数(通信先)を解析するだけで, アプリがルール 1 を満たしているか検査できる.

(2). 外部ライブラリ型

google-http-java-client や Volley などのサードパーティ製ライブラリを利用して, アプリ内の情報を外部送信するタイプを「外部ライブラリ」型と分類する. 外部ライブラリは基本的に, 標準ライブラリのラッパーとして構成される.

アプリ開発においては, 開発者は「外部送信メソッドを呼び出す際にはルール 1 を守る」という点にだけ注意すれば良い. 外部ライブラリ型の場合, ライブラリ開発者がメソッド名を自由に設定できるため, メソッド名から外部送信を識別できない. これは, アプリがルール 1 を満たしているか否か検査するにあたっての障害となる.

このため, 外部ライブラリ型の外部送信に関して, 現時点では提案方式の適用対象外としている. これに対しては, 著名なサードパーティ製ライブラリのリストを用意することや, アプリ内で使用する外部送信ライブラリを開発者に明記してもらう方法で対処することを検討している.

(3). コンポーネント型

WebView や MapView など画面コンポーネント自体が通信機能を内包しており, その通信機能によってアプリ内の情報が外部に送信されるタイプを「コンポーネント型」と分類する. WebView は通信先が任意であるのに対し, MapView は基本的に取得情報(位置情報)と通信先(Google Maps サーバ)が固定される.

WebViewアプリの典型がWebブラウザである. ブラウザはユーザが閲覧先の URL を入力することによって様々なサーバにアクセスできるため, Web View コンポーネントはドメイン∞に区画化される. しかし, その一方で, ブラウザアプリは端末の位置情報などのプライバシ情報を利用することも可能である. すなわち, ドメイン∞に区画化されているにも関わらず, プライバシ情報の取得についてはこれを許可する必要があり, 提案方式の区画化ルールに反する. このため, 現時点では提案方式の適用対象外としている. MapView については, 提案方式の区画化ルールに従うと, Google Maps サーバへの送信が許されるドメイン内でのみこのコ

ンポーネントの使用が許可されることになる.

5.3 情報取得方法の観点からの分析

アプリがプライバシ情報を取得する方法は大別して 3 種類存在することが確認された. それぞれの分類について, ルール 2(ドメイン n にて取得が許されるのは, プライバシポリシに「外部サーバ n に送信する」と記載されたプライバシ情報のみ)の観点からの分析を以下に示す.

(1). ゲット型

端末情報や電話番号など, API 呼び出しの返り値として取得できるプライバシ情報を「ゲット型」と分類する.

API はプライバシ情報ごとに用意されているため, アプリ開発者は「プライバシ情報 API を呼び出す際にはルール 2 を守る」という点にだけ注意すれば良い. 同様に, プライバシ情報取得APIの呼び出しの部分を解析するだけで, アプリがルール 2 を満たしているか検査できる.

(2). イベントリスナ型

GPS 情報や通話相手の電話番号など, 動的な(時々刻々と変化する)プライバシ情報については, イベントリスナを用いて取得される. これを「イベントリスナ型」と分類する.

イベントリスナ型ではプライバシ情報は引数の形で授受される. 引数となっているオブジェクトを見ればどのプライバシ情報の取得であるかが明確であるため, アプリ開発者は「イベントリスナを使用する際にはルール 2 を守る」という点にだけ注意すれば良い. 同様に, イベントリスナの引数を解析するだけで, アプリがルール 2 を満たしているか検査できる.

(3). ファイル型

外部またはアプリ内部の記憶領域(変数, ファイル, データベースなど)に格納されているプライバシ情報については, それぞれに応じた読み出し命令によって取得される. コンテンツプロバイダによって提供される電話帳の情報を取得する場合や, アプリの中でユーザ自身が入力したプライバシ情報をプリファレンスに格納しておき, 必要に応じて取り出す場合がこのタイプの典型である. これを「ファイル型」と分類する.

ファイル型の場合は, プライバシ情報を格納する変数やファイルをアプリ開発者が自由に設定できる. すなわち, 読み出し命令のみからは何のプライバシ情報の取得であるか識別できない. よって, ルール 2 を満たすようにアプリ開発を行うとき, 開発者は常に「どこにどのプライバシ情報が格納されているか」を把握し続けておかなければならない. 同様に, 「どこにどのプライバシ情報が格納されているか」を識別できない限り, アプリがルール 2 を満たしているか否かを検査することができない.

このため, ファイル型のプライバシ情報の取得に関しては, 現時点では提案方式の適用対象外としている. これについては, プライバシ情報が格納されるファイル名や, データベースに格納する際のキー文字列を, 開発者に明記してもらう方法で対処することを検討している.

5.4 情報処理方法の観点からの分析

前節までの分析に用いたアプリ 10 サンプルの中から更に半数(アプリ A～E)を無作為に選び, ルール 3(ドメインを越えたデータパッシングは許可されない)の観点からの分析を行った.

ルール 3 の適用がアプリ開発および解析に与える影響を分析するために, アプリ A～E のプログラムの区画化を試行し, ドメイン間のデータパッシングが発生しないような区画化が可能か否かを調査すべきである. しかし, 今回はその事前調査として, アプリ A～E において「プログラム全体の中にプライバシ情報を利用するメソッドがどれだけ含まれるか」を調査した.

各アプリのプログラム全体のメソッド数(総メソッド数)と, アプリが利用するプライバシ情報ごとに「プライバシ情報を内包するメソッド数(プライバシ関連メソッド数)」を計数した結果を表 1 に示す. ここで, メソッド数の計数については Eclipse のメトリクスプラグイン[11]を用いており, プライバシ情報を内包するメソッドであるか否かはソースコードを目視で確認している.

提案方式においては, 同一のプライバシ情報

表 1. アプリ分析結果

アプリ	アプリが利用するプライバシ情報	総メソッド数	プライバシ関連メソッド数
A	位置情報	345	7
B	位置情報	1952	46
	受話番号	1952	1
	連絡先情報	1952	2
C	位置情報	57	3
	IMEI	57	1
D	位置情報	72	55
E	位置情報	770	20

を内包するメソッドは，同一ドメイン内に区画化されなければならない．よって，プライバシ関連メソッド数が少ないアプリほど，プライバシ情報を小さなドメインに閉じ込めること可能となり，区画化が効率良く達成されることを意味する．

表 1 の結果から，アプリ D 以外は，総メソッド数に対するプライバシ関連メソッド数の比は小さいことが分かる．アプリ D においては，プライバシ情報を一度インスタンス変数に格納してから利用していたことが，プライバシ関連メソッド数増加に起因していた．また，アプリ A，B（位置情報に関連するメソッドを除く），C においてプライバシ関連メソッド数はいずれも僅少であることから，アプリの規模の大小（総メソッド数の多少）に依らず，プライバシ情報を小さなドメインに閉じ込められる可能性が示唆される．

6 まとめと今後の課題

本稿では，アプリ開発の自由度をなるべく下げることなく，プライバシポリシに準拠したアプリを開発することができ，同時に，簡単な静的解析によってアプリとプライバシポリシの整合性を検査することができる枠組みを提案した．Android アプリを対象に，提案方式によるアプリ開発ついて検討し，アプリの開発と解析に関する影響について分析した．

今後は，アプリ開発者やアプリ検査者に実際にアプリの開発および解析を実施してもらい，提案方式の適用評価を行っていく必要がある．また，現時点では対応できていないプライバシ情報の取り扱いに対する検討や，開発者の負担を軽減する方法の検討，静的検査ツールの実装・評価が今後の課題として挙げられる．

謝辞

本研究を進めるにあたり KDDI 研究所 竹森敬祐様に貴重なアドバイスを賜りました．ここに感謝の意を表します．

参考文献

[1]. 総務省：「スマートフォン プライバシー イニシアティブ－利用者情報の適正な取扱いとリテラシー向上による新時代イノベーション－」の公表 (http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000087.html, 2015/8/23 参照)

[2]. 総務省：利用者視点を踏まえた ICT サービスに係る諸問題に関する研究会提言「スマートフォン安心安全強化戦略」（案）に対する意見募集, (http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000111.html, 2015/8/23 参照)

[3]. 磯原隆将，川端秀明，竹森敬祐，窪田歩：XML を用いたモバイルアプリのプライバシーポリシーの運用，2013 年暗号とセキュリティシンポジウム，2013.01

[4]. 竹森敬祐，磯原隆将，川端秀明，窪田歩，高野智秋，可児潤也，西垣正勝：アプリ/コンテンツ向けプライバシーポリシーの第三者検証フレームワーク，情報処理学会研究報告，2013-CSEC-62，2013.07

[5]. 鈴木富明，小林真也，可児潤也，川端秀明，磯原隆将，竹森敬祐，西垣正勝：メタ API：プライバシポリシに準じた機能制約を備えた API，コンピュータセキュリティシンポジウム 2013，2013.10

[6]. Adrienne Porter Felt, et.al.: How to Ask For Permission, Proceedings of 2012 USENIX Workshop on Hot Topics in Security, 2012

[7]. Sanae Rosen, et.al.: AppProfiler: A Flexible Method of Exposing Privacy-Related Behavior in Android Applications to End Users, Proceedings of 2013 ACM conference on data and application security and privacy, pp.221-232, 2013

[8]. 磯原隆将，川端秀明，竹森敬祐，窪田歩：Android アプリの静的解析を用いた利用 API と外部モジュール検知によるプライバシーポリシー作成支援機構，情報処理学会研究報告，2013-CSEC-62，2013.07

[9]. William Enck, et.al.: TaintDroid: An Information Flow Tracking System for Realtime Privacy Monitoring on Smartphones, Proceedings of 2010 USENIX Symposium on Operating Systems Design and Implementation, Canada, 2010

[10]. F-Droid | Free and Open Source Android App Repository :(https://f-droid.org/, 2015/8/23 参照)

[11]. Eclipse Metrics Plugin: (http://metrics.sourceforge.net/, 2015/8/23 参照)