情報処理学会論文誌 Vol.57 No.9 2045–2057 (Sep. 2016)

内部構造区画化によるプライバシに準拠した Androidアプリ開発方式

小林 真也[1,a)] 鈴木 富明[1] 可児 潤也[1] 川端 秀明[2] 西垣 正勝[3]

受付日 2015年12月3日, 採録日 2016年6月2日

概要：プライバシ違反アプリが存在する背景に，アプリ開発の自由度の高さとアプリの内部構造の複雑さにともなう，アプリとアプリケーションプライバシポリシの整合性（アプリがプライバシポリシどおりの動作をするか）検証の難しさがある．そこで本論文では，プログラムの内部構造をプライバシ情報の送信先に応じて「ドメイン」と呼ばれる単位に区画化し，プログラム内でのプライバシ情報の伝搬範囲を限定する形でアプリ開発を行う仕組みを導入する．開発者が，適切に記述されたアプリケーションプライバシポリシに準じた形でドメインを分離しながらアプリ開発を行うことにより，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解析するだけで，アプリとプライバシポリシの整合性検査も容易となる．

キーワード：プライバシ，スマートフォンアプリ，アプリケーションプライバシポリシ，プライバシ・バイ・デザイン

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

Shinya Kobayashi[1,a)] Tomiaki Suzuki[1] Junya Kani[1]
Hideaki Kawabata[2] Masakatsu Nishigaki[3]

Received: December 3, 2015, Accepted: June 2, 2016

Abstract: Recently, privacy-breaching apps have increased in Android market. In the background, due to the high flexibility of application development and the complexity of internal structure of application, consistency check between Android application and privacy policy become more difficult. 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.

Keywords: privacy, Smartphone applications, application privacy policy, privacy-by-design

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（アプリケーションプライバシポリシ）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ケーションプライバシポリシの作成を支援し，第三者

[1] 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
[2] 株式会社 KDDI 研究所
KDDI R&D Laboratories Inc., Fujimino, Saitama 356–8502, Japan
[3] 静岡大学創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
[a)] gs14020@s.inf.shizuoka.ac.jp

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，開発者の誤解や記述ミスによって不正確なアプリケーションプライバシポリシが記述されてしまう可能性が残る．また，昨今のアプリ開発の自由度の高さおよびアプリの内部構造の複雑さにともない，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いため，端末側またはアプリマーケット側でアプリとアプリケーションプライバシポリシの整合性を検査するには限界がある．

そこで本論文では，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，プログラム内でのプライバシ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る．開発者が，アプリケーションプライバシポリシに準じた形でアプリ内の各コンポーネントを各ドメインに分類したうえで，そのドメインの制約に従ったアプリ開発を行うことによって，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，アプリがアプリケーション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どおりの動作をするか）検査も容易となる．

本論文では，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議論を進める．

2. アプリケーションプライバシポリシ運用における課題および既存対策

2.1 課題

モバイル端末アプリにおけるプライバシ情報（端末情報）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た [1]．そのなかで，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，アプリケーションプライバシポリシの運用およびその整合性（アプリとアプリケーションプライバシポリシが一致しているかどうかを確認する）検証にあたっては下記の課題がある．

(1) 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，アプリケーション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アプリケーション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に，アプリとアプリケーションプライバシポリシの間の整合性（一貫性）が崩れることとなる（課題 1b）．さらには，不正なアプリ開発者が，実際のアプリの動作とは異なるアプリケーション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）．

(2) 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アプリケーション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い可能性がある [1]（課題 2）．

(3) 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い．このため，アプリマーケット側で，アプリとアプリケーションプライバシポリシの整合性（アプリが本当にプライバシポリシどおりの動作をするか）を確認することが難しい（課題 3）．同様の理由で，アプリとアプリケーション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 既存対策

課題 1a に対しては，アプリ開発者によるアプリケーションプライバシポリシの作成を支援する方法が提案されている [3]．文献 [3] 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アプリケーション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，アプリ開発者にアプリケーション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 [4] では，実際に，アプリケーション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，アプリケーション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 に対してアプリケーション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，アプリケーションプライバシポリシ作成とアプリ開発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わたるため，用意すべき機能制約 API の種類が無数に及んでしまい現実的な対策とはなり得ていない．

課題 2 に対しては，文献 [3] の中で，ユーザにアプリケーション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，アプリのイン

ストール時に当該アプリに関する警告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おり，これら関連研究の知見をアプリケーションプライバシポリシの表示に対して活用すれば，課題 2 の解決策として有効であろう [6], [7].

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 の取り扱いを確認し，アプリケーション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，アプリ内のプライバシ情報に「色付け（テイント）」をすることによって，プライバシ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，アプリにおけるプライバシ情報の利用がアプリケーションプライバシポリシに準じているかどうかの検査を実施できるが，検出漏れや検出見落としがあるなど精度の点で課題が残されている．

2.3 課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，3 における問題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，アプリケーションプライバシポリシ作成とアプリ開発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．そこで本論文では，アプリの内部構造をアプリケーションプライバシポリシに準じて「ドメイン」と呼ばれる領域に区画化し，プライバシ情報を適切なドメインに封じ込めることによって，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が可能となる開発方式（区画化開発）を提案する．また，この開発方式が実際のアプリ開発に適用可能であるかどうか，その開発負荷や開発限界について，実際のアプリ開発への適用や開発者へのヒアリングを通して考察する．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解析するだけで，アプリがアプリケーションプライバシポリシに準じていることが確認できるため，アプリとアプリケーションプライバシポリシの整合性検査も容易となり，課題 3 の問題についても改善される．

アプリ開発とアプリケーションプライバシポリシ作成の一体化は，プライバシ・バイ・デザインの概念 [14] に基づくアプリ開発の採用を意味する．プライバシ・バイ・デザインは現在の潮流となっている設計指針であり，プライバシポリシイニシアティブの基本原則の 1 つとして謳われている．また，アプリケーションプライバシポリシに準じた区画化をプログラミングにおけるルールとしてアプリ開発者に課すことは，アプリの安全性検査の実効化のためにアプリ開発者にも協力を仰ぐ ADMS（Application Development Management System）型の開発手法 [10] の指向を意図している．ADMS 型のアプリ開発を採用することによって，アプリがアプリケーションプライバシポリシに即した製品であることをアプリマーケットに保証してもらうことが容易となるため，アプリ開発者にとっても，自社製品に対するユーザの信用や評価を高められるというメリットがある．

提案方式の具体的な方法については次章で説明する．

3. アプリの区画化設計

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 をそのつど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，アプリケーションプライバシポリシに記されたとお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が可能である．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．

そこで本論文では，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．アプリ開発者がアプリケーションプライバシポリシに準じた形でアプリ内の処理（メソッド）を区画化し，個々のプライバシ情報をそれぞれの区画内に囲い込むことによって，各区画の入出力の有無さえ監視してやれば（すなわち，どのような情報が伝達されているかは無視して，区画をまたぐメッセージパッシングが発生し得るか否かのみを監視してやれば），アプリの挙動がアプリケーション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．プライバシ情報は，区画を越えての伝搬につながらない限り，区画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（プライバシ情報が区画を越えないという制限の制約内で）維持される．本論文では，この区画のことを「ドメイン」と呼称する．ドメインの正確な定義については次節に示す．

3.2 ドメインの定義

「ドメイン」とは，アプリケーションプログラムを構成するメソッド群を「ある特定の送信先に送信されるプライバシ情報を扱うメソッド」ごとに分離した論理的な区画である．ドメインは，アプリケーションが取得するプライバシ情報とその送信先に応じて規定される．アプリがスマートフォン内のプライバシ情報をどのように取り扱うのか（ど

<FQDN> www.kddilabs.jp </FQDN>
</収集者>
<収集情報一覧> <!-- ②収集情報 -->
<収集情報>
IMEI（端末識別ID）もしくはその特徴値
</収集情報>
<収集情報>
IMSI（加入者識別ID）もしくはその特徴値
</収集情報>
</収集情報一覧>

図 1 アプリケーションプライバシポリシ作成支援ツール [3] によって生成されたプライバシポリシの例（文献 [3] の図 12 を元に作成）

Fig. 1 Application privacy policy generated by a policy description support tool [3].

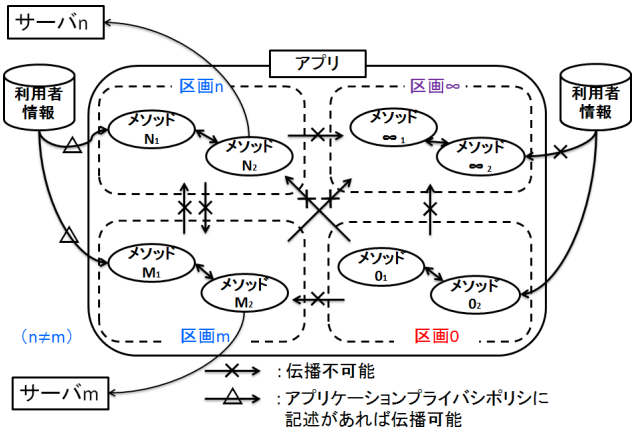

図 2 各ドメインのルール概要

Fig. 2 Overview of data passing rule between domains.

の情報をどこに送信するか）については，アプリケーションプライバシポリシとして明記することが推奨されている [1]．すなわち，ドメインの情報は，アプリケーションプライバシポリシから導出される．

2.2 節で紹介したように，アプリ開発者がチェックシートにチェックを入れるだけで，**図 1** のようなアプリケーションプライバシポリシを自動生成するシステムも提案されている [3]．本論文は，文献 [3] のシステムを利用してプライバシポリシを生成し，そのプライバシポリシからドメイン情報を自動抽出することを想定しており，これにより，アプリケーションごとにプライバシポリシに整合したドメイン情報を設定することができるようになっている．

アプリのドメイン情報については，上述のとおり，アプリケーションプライバシポリシから定まることになるが，プログラムの内部構造（メソッド群）を具体的にどのように分離設計してドメイン情報に沿った区画化を達成するかについては，アプリ開発者の自由裁量に任されることに注意されたい（詳細は 4.3 節で説明する）．

3.3 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うち，どの情報がどのサーバに送られているか」であるため，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．各ドメインのルール詳細について以下に示す．

(1) ルール 1：情報取得の制限

各ドメインの情報取得は，アプリケーションプライバシポリシに基づいて制御される．たとえば，「電話番号をサーバ n に送信する」ことがアプリケーションプライバシポリシに記述されているアプリにおいては，電話番号取得 API を呼び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 の呼び出しがいっさい許可されず，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 の呼び出しが許可される．

(2) ルール 2：情報送信の制限

各ドメインの情報送信制限は，外部サーバ m を通信先として指定した形での外部通信 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外部通信 API の呼び出しはいっさい許可されず，ドメイン ∞ 内のメソッド群は，任意の外部通信 API の呼び出しが許可される．

(3) ルール 3：ドメイン間のメッセージパッシングの制限

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（$n \neq m$）間におけるデータの送受信（静的変数などを利用した値参照），引数ありのメソッド呼び出し，戻り値ありのメソッド呼び出しのすべてを禁止する．ただし，ドメイン n からドメイン 0 へのデータ送信，ドメイン ∞ からドメイン n へのデータ送信，ドメイン n からのドメイン 0 内の任意の引数ありメソッドの呼び出し，ドメイン n からのドメイン ∞ 内の任意の戻り値ありのメソッド呼び出しについては許可される．

アプリの内部構造の区画化における上記のルールを**図 2** にまとめた．図 2 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，アプリとアプリケーションプライバシポリシの整合性が保証される．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 2 のルールが守られていることが確認できれば，そのアプリはアプリケーションプライバシポリシと整合していることが保証される．図 2 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解析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，アプリケーションプライバシポリシに対するアプリの整合性を確認することができる．

3.4 制約の妥当性

2.3 節で述べたように，提案方式は ADMS 型の開発手法 [10] をとっており，アプリケーションプライバシポリシを事前に作成すること，および，アプリケーションプライバシポリシに準じた区画化をプログラミングにおけるルールとして適用することをアプリ開発者に課している．ADMS 型のアプリ開発は，プライバシ・バイ・デザインの考え方に即しており，アプリの安全性検査の実効化にも大きく寄与するが，その代償としてアプリ開発者に課せられるプログラミング上の制約が過大になってしまっては机上の空論となってしまう．

アプリケーションプライバシポリシの作成に対しては，2.2 節で説明したチェックシート形式のアプリケーションプライバシポリシ作成支援の仕組み [3] を利用することによって，アプリ開発者の負担を抑えた形で開発者にアプリケーションプライバシポリシを作成してもらうことが十分可能であると期待される．さらに，3.2 節で説明したように，ドメイン情報はアプリケーションプライバシポリシから抽出可能である．プログラミングの区画化に対しては，6 章および 8.1 節で提案方式の実適用とアプリ開発者へのアンケートを実施し，区画化を遵守した形でプログラミングを行うことによるアプリ開発への影響を考察する．

4. Android アプリ開発への適用

提案方式を Android アプリ開発に適用した際のアプリ開発手順を以下に示す．

4.1 整合性検証までの流れ

Android アプリは一般的にアプリマーケットを介してユーザに配布/販売される．提案方式でアプリとプライバシポリシの整合性検証を行うのはこのマーケット運営者であり，開発者が提案方式に沿って開発を行った後で，アプリマーケットにアプリのソースコードとアプリケーションプライバシポリシを提出する（図 2 におけるアプリ審査）．マーケットは整合性検証を行った後で，アプリを承認または拒否して，その結果をアプリ開発者に告知する．

4.2 開発フロー

アプリ開発者がアプリをマーケットに登録するまでの従来の一般的な開発フローは，「企画・要件定義→コンポーネント設計→コーディング→テスト→マーケット審査」という流れとなる．アプリケーションプライバシポリシの記述は，アプリ開発と独立して行われている．提案方式においては，アプリ開発者はアプリの企画や要件定義を終えた後の設計工程で，アプリケーションプライバシポリシの記述を行う．アプリの区画化の方針についても同時に設計を実施する（詳細は 4.3 節で説明する）．実装の途中で仕様変更が生じた場合，その変更が上流工程にフィードバックされる．**図 3** にアプリの開発フローを示す．

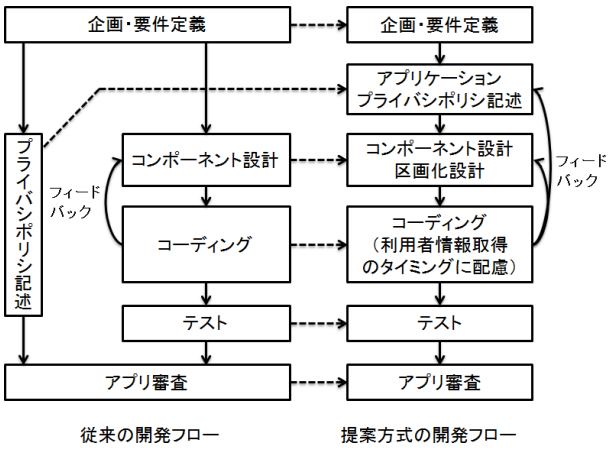

図 3 提案方式におけるアプリ開発フロー
Fig. 3 App development flow in proposed scheme.

4.3 ドメインの設計

Android アプリの開発においては，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（アプリの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，プログラムの内部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」ととらえることが可能である．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，ドメイン情報に基づいてプログラムの内部構造をどのように分離するかを設計する際においても，コンポーネント（画面）単位で区画化する方針を採用することとする．

例として，「近隣レストランの情報を取得するために，端末の位置情報を自社のデータベースサーバに送信する」というアプリケーションプライバシポリシを持つグルメアプリを考える．アプリ開発者がコンポーネント設計を行った結果が，**図 4** の画面遷移であったとする．「メニュー画面」内の「近隣レストラン検索」ボタンをタップすると「近隣レストラン情報画面」に遷移し，「①端末の位置情報をデータベースサーバへ送信→②データベースサーバからの近隣レストラン情報を受信→③受信結果を画面へ表示」のタスクが実行される．

ここで，プライバシ情報の送信に関係するタスクは，「近隣レストラン情報画面」内で実行されるタスク①（端末の位置情報をデータベースサーバへ送信）である．そこで，

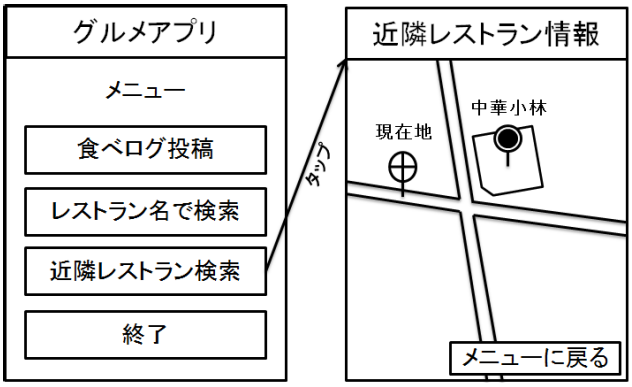

図 4 グルメアプリの画面遷移
Fig. 4 Screen transition of gourmet application.

提案方式においては，「近隣レストラン情報画面」に対応する Activity コンポーネントを「データベースサーバに送信する情報を扱うドメイン」として定義する．この結果，端末の位置情報（データベースサーバに送信する情報）を取得するための API 呼び出しは，この Activity（近隣レストラン情報画面）内でのみ許可されることになる．

Android アプリにおいては，多くの場合で，プライバシ情報取得 API 呼び出しの際にコンポーネントクラスのインスタンスを引数として指定する必要がある．コンポーネント単位で区画化を行うという設計方針は，この観点からも都合がよい．

4.4 ドメインの記述

提案方式においては，アプリ内の区画化に関する情報を何らかの形でアプリ自身に所持させる必要がある．Android アプリにおいては，Activity および Service コンポーネントを使用する場合には，必ずその情報をアプリのマニフェストファイル（AndroidManifest.xml）内に記述しなければならないという仕様になっている．コンポーネント単位で区画化を行うという設計方針であれば，マニフェストファイルにおける各コンポーネントの記述の中に，それぞれの区画の定義を追加するという方法を採用することができる．

マニフェストファイル内へのドメイン情報の記述には，“domain” という XML タグを利用する．domain タグの値にドメイン名を記載する（ドメイン 0，∞ はドメイン名として 0，−1 に指定する）．ドメイン内で扱うことのできるプライバシ情報を記載するために “src” タグを，ドメイン内の情報の送信先を記載するために “dest” タグを，それぞれ用意した．src タグにはプライバシ情報の識別子が，dest タグには送信先となる URL（FQDN），IP アドレス，ファイルパスが記載される．ドメイン情報の記述例を**図 5** に示す．図 5 の例では，MenuActivity という Activity コンポーネントがドメイン 0 に属しており，NeighborSearchActivity という Activity コンポーネントが

```
<activity android:name=".MenuActivity">
  <domain android:value="0"/>
</activity>
<activity android:name=".NeighborSearchActivity">
  <domain android:value="gourmet.com"/>
    <src android:value="location"/>
    <dest android:value="database.gourmet.com“/>
  <domain/>
</activity>
...
```

図 5 マニフェストファイルにおけるドメインの定義
Fig. 5 Domain definition in Android manifest file.

ドメイン「gourmet.com」に属していることが記述されている．また，ドメイン「gourmet.com」から外部に送信可能なプライバシ情報は位置情報（location）のみに限られており，送信先は「database.gourmet.com」のみに限られていることが記述されている．

4.5 整合性検証

3.2 節で述べたように，提案方式においては，図 2 のルールが守られていることが確認できれば，そのアプリがアプリケーションプライバシポリシと整合していることが保証される．提案方式を適用して開発された Android アプリのソースコードを受け取ったマーケットが，アプリケーションプライバシポリシとの整合性検証を行う際の具体的な手順を以下に示す．

Step.1 マニフェストファイルからドメイン情報を抽出
Step.2 ソースコード（全 Java ファイル）を走査して以下を確認する．
A) ドメイン 0 に属するコードについて
- 外部送信を検出→不正（4.7 節参照）
- 他ドメインへの値渡しを検出→不正（4.8 節参照）

B) ドメイン ∞ に属するコードについて
- プライバシ情報の取得を検出→不正（4.6 節参照）
- 他ドメインからの値渡しを検出→不正（4.8 節参照）

C) ドメイン n に属するコードについて
- ドメイン情報に指定されている送信先ドメイン以外への外部送信を検出→不正（4.7 節参照）
- ドメイン情報に指定されているプライバシ情報以外の取得を検出→不正（4.6 節参照）
- ドメイン m（$n \neq m$）間での値渡しを検出→不正（4.8 節参照）

Step.3 上記で不正が 1 つも確認されなければアプリを承認

4.6 プライバシ情報取得 API の検査

4.5 節の整合性検証の Step.2 において，プライバシ情報取得 API の有無の確認については，プライバシ情報と API

表 1 プライバシ情報と API の紐付けリストの一例
Table 1 Example of relation list for privacy information and APIs.

プライバシ情報	識別子	API
IMEI (端末識別 ID)	Imei	TelephonyManager::getDeviceId
IMSI (加入者識別 ID)	Imsi	TelephonyManager::getSubscriberId
位置情報	Location	LocationManager::getLastKnownLocation LocationListener::onLocationChanged
電話番号	phone_number	TelephonyManager::getLine1Number

との紐付けがリスト化されていることが前提となる．プライバシ情報と API の紐付けリストの一例を**表 1** に示す．

本研究では，プライバシポリシの記述に文献 [3] のツールを利用することを想定している．文献 [3] では，プライバシ情報を「ID 情報」と「プライバシ情報」に分類しており，表 1 の IMEI，IMSI，電話番号が ID 情報，位置情報がプライバシ情報に該当する．これらの情報と対応する API については一般的に知られているものであり，その紐付けリストを用意することは容易である．

しかし，最近では悪質な Android アプリが検知ツールによる検出を回避する目的で，一般的に知られていない API を利用してプライバシ情報の取得（や外部送信）を行う場合がある [16]．この問題を解決するためにプライバシ情報取得 API（ソースメソッド）と外部送信 API（シンクメソッド）の包括的なリストを機械学習により構築する既存研究（SuSi プロジェクト）が存在する [16]．SuSi プロジェクトによって提供されているソースメソッドのリストを表 1 の紐付けリストとして用いると有効だと考えられる[*1]．

vCard などのように SD カード上のファイルからプライバシ情報が読み込まれる場合もある．これに対応するためには，ファイル入力 API もプライバシ情報取得 API の 1 つとして考える必要があるだろう．また，Web ページのテキストフィールド内にユーザが入力する住所やクレジットカード番号なども保護すべきプライバシ情報である．この対応については，今後の検討課題としたい．

4.7 外部送信 API の検査

4.5 節の整合性検証の Step.2 において，外部送信 API の有無の確認は，通信 API，ファイル出力 API，インテント API，その他の API に対して行われる必要がある．

通信 API は，一般的に知られている通信用 API 群である．たとえば，DefaultHttpClient クラスの execute メソッドや HttpURLConnection クラスの connect メソッドを用いた HTTP 通信が含まれる．しかし，サードパーティ製の通信ライブラリを用いる開発者も多い．サードパーティ製ライブラリへの対応については 5.4 節で後述する．

あるドメインのプライバシ情報を SD カードなどに保存した後で，他のドメインからそれを読み出して利用することが可能であるため，ファイル出力 API も外部送信 API の 1 つとして取り扱う必要がある．SharedPreference や Content Provider，SQLite などへの書き込みについても同様である．

インテント（Intent）とは，Android OS 上のアプリ間およびコンポーネント間のメッセージパッシングの仕組みである．Intent クラスには値を格納するメソッドが複数定義されている．代表的なものが putExtras メソッドであり，様々なデータ型の値をインテントに格納して値渡しに利用することが可能となっている．他にもいくつかの put メソッドや set メソッドが存在する．提案方式では，インテントに値を格納するこれらのメソッド群を外部送信 API として取り扱うことで，インテントを利用した外部送信に対応する．また，Intent クラスのインスタンスを生成する際，第二引数に URI を指定することが可能である．さらに，parseUri クラスメソッドを使用して同様に URI を指定したインテントを生成することが可能である．この URI に外部送信先 URL を指定すると，インテント送信時に自動的にブラウザが起動して当該ページに接続する．HTTP 通信では，クエリ文字列を URL に含ませることで，そのクエリ文字列を接続先に送信することが可能である．提案方式では，引数を 2 つ以上含む Intent コンストラクタと parseUri メソッドについても外部送信 API として取り扱うことで，インテントを利用した外部送信に対応する．

4.6 節で述べたように，悪質な Android アプリは一般的に知られていない API を利用して（プライバシ情報の取得や）外部送信を行う場合がある．この問題に対しては，SuSi プロジェクト [16] が提供するシンクメソッドのリストに記載されている API を外部送信 API として取り扱うことが有効である．

4.8 ドメイン間の値渡しの検査

4.5 節の整合性検証の Step.2 において，値渡し有無の確認は，パブリック宣言されているメソッドに対する「ドメインをまたぐ形でのメソッド呼び出し」，および，パブリック宣言されている変数に対する「ドメインをまたぐ形での変数アクセス」のすべてを検査することで達成される．

メソッド呼び出しについて，あるドメイン n 内のメソッド α が他のドメイン m 内で呼び出されている例で説明する．まず，メソッド α が (i) 参照型の引数を 1 つ以上有するか，(ii) 参照型の返り値を 1 つ以上有するかを検査する．(i) あるいは (ii) のいずれかが真の場合，そのメソッド呼び出しを「ドメイン m とドメイン n の間の相互の値渡し」と判定する．(i) も (ii) も偽の場合は，続いて，メソッド α が (iii) 基本データ型の引数を 1 つ以上有するか，(iv) 基本

[*1] ただし，SuSi で提供されているソースメソッドは 12 のカテゴリに分類されているため，これを提案方式の紐付けリストとして利用した場合は，プライバシ情報の粒度は 12 種類となる．

データ型の返り値を有するかを検査する．(iii) のみが真の場合，そのメソッド呼び出しを「ドメイン n からドメイン m への値渡し」と判定する．(iv) のみが真の場合，そのメソッド呼び出しを「ドメイン m からドメイン n への値渡し」と判定する．(iii) かつ (iv) が真の場合，そのメソッド呼び出しを「ドメイン m とドメイン n の間の相互の値渡し」と判定する．

変数アクセスについて，あるドメイン n 内の変数（代入演算子の右辺）の値が代入演算子によって他のドメイン m 内の変数（代入演算子の左辺）に受け渡される例で説明する．右辺の変数が基本データ型の場合，この代入演算を「ドメイン n からドメイン m への値渡し」と判定する．右辺の変数が参照型の場合，その代入演算を「ドメイン m とドメイン n の間の相互の値渡し」と判定する．

5. 整合性検証を迂回する手段への対応

提案方式においては，4.5 節に示した手順により「各ドメインの入出力（メッセージパッシング）が図 2 の制約を満たす」ことを静的検査することで，アプリの挙動とアプリケーションプライバシポリシの整合性が確認できる．ただし，Android アプリの場合は，Android 特有の機能や Java 言語特有の機能を用いることで，悪意あるアプリ開発者がこの静的検査を迂回する方法がいくつか存在する．本章では，この問題に対する対処として，アプリ開発に支障をきたさない程度のプログラミング上の制約を加えることを検討する．区画化を遵守した形でプログラミングを行うことがアプリ開発者にとって過負荷となっていないかを 6 章および 8.1 節で確認するが，これに加え 7 章および 8.2 節で，既存アプリの分析とアプリ開発者へのアンケートを実施し，本章で追加された制約がアプリ開発者にとって過負荷となっていないか確認する．

5.1 リフレクション機能の利用制限

Java にはリフレクションという機能が用意されており，これを利用することによって，任意のフィールドやメソッドをプログラム実行時に動的に呼び出すことや，通常であれば外部からアクセスできない private や protected のフィールドやメソッドにアクセスすることが可能となる [15]．すなわち，リフレクション機能を利用することによって，静的解析では発見できない形で，ドメイン n においては許可されていないプライバシ情報取得メソッドを呼び出すことや，ドメイン n 内に封じ込まれているプライバシ情報をドメイン m（$m \neq n$）から参照することが可能である．上記のようなリフレクション機能を利用した整合性検査の迂回は，静的解析だけでは完全に検出することができないため，提案方式における深刻なセキュリティホールとなる．

リフレクションは，Android においては複数のライブラリを動的にリンクさせる目的や，Android SDK で公開されていない非公開の API（Hide API）を利用する目的などで正規アプリにおいても利用される．しかし，リフレクションの使用は，セキュリティ分析をいたずらに複雑にし，容易にぜい弱性を作りこんでしまうという理由から，どうしても使わざるを得ない場合を除いては利用が非推奨とされている [13]．そこで，提案方式においては，現時点ではリフレクション機能の利用を禁止するという制約を設けることとする．

5.2 WebView の HTML5/JavaScript の利用制限

WebView 上で実行される HTML5/JavaScript のプログラムは，実行時に Web サイトから動的に取得される．このため，リフレクション機能と同様，悪意あるアプリ開発者は WebView の HTML5/JavaScript を利用して，静的解析では発見できない形で整合性検査を迂回することが可能である．

WebView における HTML5/JavaScript 実行はデフォルトでは無効に設定されている．そこで，提案方式においては，現時点では HTML5/JavaScript 実行を禁止するという制約を設けることとする[*2]．

5.3 外部送信先 URL（FQDN）のハードコード

外部送信メソッドの呼び出しにあたっては送信先 URL をメソッドの引数として与えることになるが，Java では，この URL を文字列の加工などによって動的に指定することができる．このような場合は，静的解析によって外部送信先を検査することが困難となる．そこで，提案方式においては，現時点では外部送信先 URL はハードコードするという制約を設けることとする．なお，提案方式における整合性検査は，送信先のサーバを確認できれば良いので，ハードコードが要求されるのは，ホスト名とドメイン名（FQDN）までとなる．

5.4 サードパーティ製ライブラリの利用制限

Java では，プログラムの一部がライブラリの形で提供される場合がある．ライブラリは実行形式プログラムとなっているため，ソースコードに基づく静的解析が実行できない．このため，悪意あるアプリ開発者が，プライバシ情報の取得や外部送信の機能を有するライブラリを利用することによって，整合性検査を迂回することが考えられる．

そこで，提案方式においては，現時点ではサードパーティ製ライブラリの利用は禁止するという制約を設けることとする．ただし，広告ライブラリや通信ライブラリなどにおいては広く普及しているものも少なくない．このため，ア

[*2] HTML5/JavaScript を有効にするためには WebView に付属している WebSettings クラスの setJavaScriptEnabled メソッドの引数に True を与える必要がある．したがって，具体的には，setJavaScriptEnabled メソッドの呼び出しを禁止するという制約を設ける．

プリマーケット側で動作確認が取れたライブラリについては，これを内部構造（どのプライバシ情報を取得するか，外部に送信するか）の情報とともに，アプリマーケットがホワイトリストに登録することとする．ホワイトリストに含まれるライブラリについては，ソースコードの代わりに内部構造の情報を用いて整合性検査が可能となる．

6. 評価 1：既存アプリの開発

区画化を遵守した形でプログラミングを行うことがアプリ開発者にどれほどの負担を強いることになるかを評価するために，既存アプリの開発に対して提案方式の適用を試みた．今回は，Google 社がオープンソースソフトウェアとして公開している「My Tracks」というアプリケーション [11] を対象とした．My Tracks はアウトドアで活動した際のコースや速度，距離などのデータを記録し，Google Drive で記録の同期や共有を行うことができるサービスをユーザに提供する Android アプリである．このアプリは，複数のプライバシ情報を取り扱い，そのうちの位置情報を「Google Drive」に送信している．また，ソース量も十分に大きい（208 の Java ファイル，約 38 KLOC）ことから，適用評価の対象として選択している．

6.1 My Tracks が取り扱うプライバシ情報

My Tracks が取り扱うプライバシ情報は，「位置情報」，「連絡先情報」，「携帯端末の状態」，「ID 情報」の 4 種類である．位置情報は，ユーザの移動軌跡を記録するために取得される．連絡先情報は，ユーザのトラックデータ（記録した移動軌跡の情報）を友人と共有するにあたり，「Google Drive へのアクセスが許可された URL」を記したメールを友人のメールアドレスに送信するために用いられている．なお，友人へのメールの送信にあたっては，My Tracks はメールクライアントを起動してメールの送信を依頼する．端末の状態と ID 情報は，My Tracks が利用する音声通知機能がユーザの通話中に作動しないようにするために用いられている．

これらの各情報の利用目的の詳細は My Tracks のサポートページに示されており，端末の状態と ID 情報については，保存や外部送信は行っていないことが明記されている [18]．したがって，端末の状態と ID 情報を利用する機能が含まれるコンポーネントのメソッド群はすべてドメイン 0 に割り当てればよい．ドメイン 0 のコンポーネントは外部送信メソッドを持たないので，プライバシ情報の取り扱いに注意を払う必要はなく，自由度の高いコーディングが許される．本節では，それ以外の，位置情報と連絡先情報を利用するコンポーネントについて，区画化設計/開発が適用可能であるかどうか，実際のコードを元に確認する．

図 6 にトラックデータを共有する操作の実行時の画面を示す．左が Google Drive へのエクスポートダイアログであり，右が Google Drive 上のトラックデータを共有する友人のメールアドレスを入力するダイアログである．提案方式では，アプリケーションプライバシポリシに従い，位置情報を扱うエクスポートダイアログ（図 6 左）のコンポーネント（画面）がドメイン「google.com」として定義され，連絡先情報を扱うアドレス入力ダイアログ（図 6 右）のコンポーネント（画面）がドメイン「mail_client」として定義されることになる．

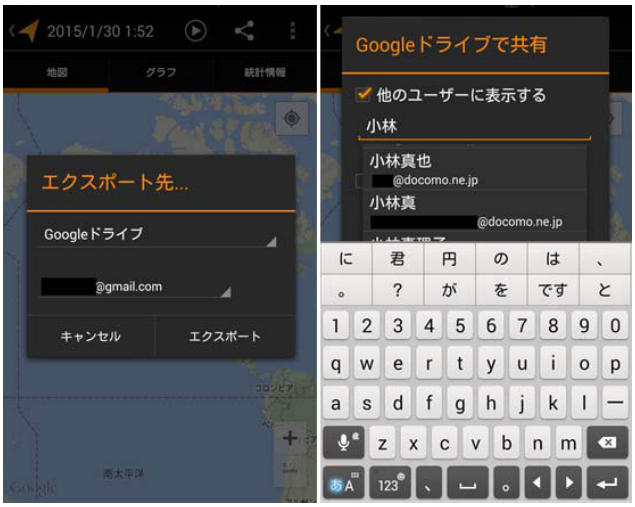

図 6 My Tracks のデータ共有実行画面
Fig. 6 Data sharing views of My Tracks.

6.2 エクスポートダイアログのコーディング

My Tracks のエクスポートダイアログは，SendDriveActivity という名前が付けられたアクティビティコンポーネントである．SendDriveActivity の中で，位置情報（トラックデータ）を記録する機能は，TrackRecordingService と命名されたサービスコンポーネントにおいて実装されている．トラック情報は，MyTrackProvider と呼ばれるコンテンツプロバイダコンポーネントにおいてデータベースに保存され，SendDriveAsyncTask という名前の非同期タスクの中で Google Drive に送信される．したがって，これらのアクティビティ，サービス，およびコンテンツプロバイダのコンポーネントについては，すべてドメイン「google.com」に属するものとして定義し，その旨がマニフェストファイルにも記述される．

アプリ開発者は，「google.com」以外のドメインからドメイン「google.com」への値の受け渡しが禁止されていることに注意してコンポーネントの「中身」をコーディングしていく．たとえば，Google Drive へのトラック情報の送信を担っている SendDriveAsyncTask がドメイン「google.com」の外に出ないように，SendDriveAsyncTask の実装は SendDriveActvitiy 内で行う必要がある．

6.3 アドレス入力ダイアログのコーディング

My Tracks のアドレス入力ダイアログは，AbstractSend-

ToGoogleActivity という名前が付けられたアクティビティの抽象クラスであり，SearchListActivity，TrackDetailActivity，TrackListActivity という 3 つのアクティビティがこの抽象クラスを継承している．AbstractSendToGoogleActivity において，メールアドレスのオートコンプリート機能は，ShareTrackDialogFragment というクラスにおいて実装されている．メールの送信は，インテントを利用した外部メールクライアントの起動により行われている．Google Drive へのトラック情報の送信が行われた後でメール用インテントが送信され，メール本文に Google Drive にアクセスするための URL が挿入される．

したがって，これらのアクティビティについては，すべてドメイン「mail_client」として定義し，その旨がマニフェストファイルにも記述される．外部送信 API に該当するのはこの場合，インテントに関するメソッドである．区画化ルールに従うのであれば，Google Drive の URL については，ドメイン「google.com」内で生成されたデータであるため，これをドメイン「mail_client」に直接受け渡して利用することはできない．したがって，Google Drive の URL 情報をユーザにマニュアルで入力させるようにするか，あるいは Google Drive 情報をプライバシ情報として扱うようプライバシポリシに追記するなど，設計を変更する必要が生じる．今回の開発では，前者の設計変更を採用することとした．

6.4 提案手法による My Tracks の開発負荷

図 7 は，前節までの説明に基づき，My Tracks の区画化を概要図として示したものである．今回は既存のアプリに対して提案方式を適用したが，実際には，まず，コンポーネント設計のフェーズにおいてアプリケーションプライバシポリシに従って図 7 のような区画化を行い，そのうえで，ドメイン間のメッセージパッシング（3.2 節）に注意してコーディングを進めていくという開発フローとなる．

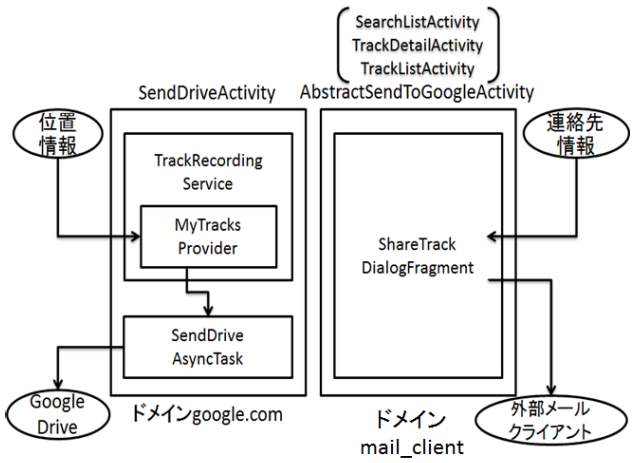

図 7 My Tracks の区画化の概要図
Fig. 7 Overview of domain definition in My Tracks.

My Tracks は比較的規模の大きいアプリであるが，区画化の適用により影響を受ける範囲は小さく提案方式の適用自体は一部設計の変更を要する程度であった．このように，アプリの規模が大きい場合であっても，提案方式の影響を受けるのはプライバシ情報を取り扱うコード群のみであり，適切に区画化に関する設計を行いさえすれば，提案方式による開発負荷は過大にはならないのではないかと考えられる．

7. 評価 2：既存アプリの分析

5 章で課した Android アプリでの実装における 4 つの制約がアプリ開発者にどれほどの負担を強いることになるかを評価するために，これら 4 つの制約に関する機能が実際の既存アプリにおいてどの程度用いられているか分析した．今回は，F-Droid [12] という Android アプリのオープンソースソフトウェアを配信するウェブサイトを利用して分析サンプルを取得した．分析時（2015 年 10 月時点）において F-Droid には 1,661 のアプリが登録されていた．この全アプリのソースコードを分析し，5 章で課した 4 つの制約に関する機能が利用されているアプリの数を調査した．分析結果を以下に示す．

7.1 リフレクション機能の利用

リフレクション機能を用いているアプリは 1,661 サンプル中 368 サンプルであった．3 割弱の利用率（リフレクションを禁止すると 3 割弱のアプリ開発に影響が生じる）は看過できない．リフレクションの利用は一般的には非推奨とされていることを勘案すると，リフレクションの使用を禁止するという制約は妥当であると考えることもできなくはないが，8.2 節のヒアリングの結果からもリフレクションの使用は認める方向で対応する必要があるだろう．この対策については，今後の課題としたい．

7.2 WebView における HTML5/JavaScript の利用

WebView を利用しているアプリは 1,661 サンプル中 371 サンプルであり，そのうち JavaScript を有効にしているアプリは 190 サンプルであった．ただし，これらのアプリは基本的にウェブアプリケーションとして動作するように設計されたものであった．Android アプリへの提案方式の適用を前提とすれば，HTML5/JavaScript を利用しているアプリは僅少であることから，HTML5/JavaScript の使用を禁止することによってアプリ開発者が負う制約は限定的であると考えられる．ウェブアプリケーションに対する提案方式の適用については，今後の課題としたい．

7.3 外部送信先 URL（FQDN）のハードコード

通信を行うことのできる INTERNET パーミッションを持つアプリは 1,661 サンプル中 842 サンプルであり，その

うち 795 サンプルが送信先のハードコードを行っていた．9 割以上のアプリで送信先がハードコードされていたことから，外部送信先 URL のハードコードを強制することによってアプリ開発者が負う制約は限定的であると考えられる．

7.4 サードパーティ製ライブラリの利用

INTERNET パーミッションを持つアプリ 842 サンプルのうち，著名な外部通信ライブラリ（Google-HTTP-Java-Client や Volley など）を利用しているアプリは 89 サンプル，広告モジュールを利用しているアプリは 34 サンプルであった．著名でないライブラリを利用しているアプリについては計数していないが，著名なライブラリの利用率に達することはないであろう．著名なライブラリであっても利用率が 1〜2 割程度であったことから，ホワイトリストに登録されていないライブラリの使用を禁止することによってアプリ開発者が負う制約は限定的であると考えられる．ただし，これは，今回の分析対象がオープンソースソフトウェアであった（ソースコードがオープンになっていないライブラリの利用が許されていない）ことが影響している可能性も考慮すべきである．オープンソースソフトウェア以外の Android アプリの分析については，今後の課題としたい．

8. 評価 3：アプリ開発者へのヒアリング

区画化を遵守した形でプログラミングを行うこと，および，5 章で課した 4 つの制約がアプリ開発者にどれほどの負担を強いることになるかを評価するために，Android アプリ開発者 2 名（開発歴 5 年のソフトウェア開発会社プログラマ 1 名，開発歴 4 年の情報科学系大学生 1 名）にヒアリング調査を行った．得られた意見を以下に示す．

8.1 区画化を遵守したアプリ開発について

区画化を遵守したプログラミングを行うことについては，“設計の時点で適確な区画化がなされていれば，アプリ開発（コーディング）自体は問題なさそうである．ただし，複数のコンポーネントを同じ区画に含めることができると設計しやすいであろう”，“コンポーネント単位での区画化設計は区画の領域が狭すぎて設計がしにくいのではないか，パッケージ単位ぐらいであれば設計しやすいだろう”という意見であった．1 つ目の意見にある，「複数のコンポーネントを同じ区画に含める」設計は現状の方式でも適用可能であるため問題がない．2 つ目の意見について，提案方式では，4.2 節で述べたように Android アプリ開発における画面設計との親和性からコンポーネント単位の区画設計を提案しているが，パッケージ単位で区画化したほうが開発しやすい場合も存在するであろう．最適な区画化の単位の分析については，今後の課題としたい．

また，“アジャイル開発（機能を少しずつ付け加えていく反復的な開発手法）では，追加する機能に応じて上流工程（アプリケーションプライバシポリシや設計）へのフィードバック（図 3 を参照）が頻繁に発生するため開発への負荷が大きいのではないか”というコメントがあった．この問題は提案方式に固有の問題ではなく，プライバシ・バイ・デザイン指向の設計方法全般に当てはまるものではあるが，今後の課題として，提案方式の開発フローがアジャイル開発などの各種開発手法と親和しているかどうかについて，ソフトウェア工学的に検証していくことは重要であると考えている．

8.2 Android アプリ実装における制約について

リフレクションの利用禁止については，“リフレクションは非公開 API の利用やプラグイン機能の実装において用いるため影響が大きい”という意見をいただいた．リフレクションの利用およびそれを用いた非公開 API の利用は非推奨とされているものの，実際のアプリ開発の現場ではリフレクション機能が活用されているという乖離が垣間見える．このような現実に鑑みると，リフレクションは禁止するのではなく，リフレクションの利用を監視する仕組みを用意するアプローチが堅実である可能性がある．具体的には，リフレクション API の呼び出しを検出して，その引数を検査するという方法が候補となる．引数がハードコードされていれば，リフレクションによってどのフィールドやオブジェクトにアクセスされたかを特定できると考えられる．

WebView における HTML5/JavaScript の禁止については，“Web ベースのアプリを作る場合は HTML5/JavaScript の使用は必須”，“Cordova のフレームワークを用いて HTML5 で Android アプリを開発することが不可能になるため影響が大きい”という意見であった．7.2 節で述べたとおり，提案方式は現時点では Java ベースの Android アプリへの適用を対象としているが，HTML5/JavaScript ベースのアプリケーションについても早急に適用の仕方を検討する必要がある．なお，WebView においても，WebChromeClient は位置情報を取得する際にユーザに許諾を得るようになっているなど，提案方式と同様，プライバシ・バイ・デザインを指向する動きがある．

外部送信先のハードコーディングの制約については，“今までもそのような形でアプリ開発をしているため，特に問題はない”という意見であった．

サードパーティ製のライブラリ制限については，“ライブラリのような機能のまとまりは開発時間の短縮のためには必要不可欠であり，制限を加えることは開発の長期化に繋がるため影響が大きい”という意見であった．5.4 節で述べたとおり，利用できるライブラリはマーケットのホワイトリストに依存するため，アプリマーケット側が重要な

ライブラリを適切にホワイトリストに登録するようになっていれば，問題は緩和されると考えられるが，非公開ライブラリや商用ライブラリを開発に利用する場合に問題が残るため，更なる検討が必要である．

9. まとめと今後の課題

本論文では，プログラムの内部構造をプライバシ情報の送信先に応じて「ドメイン」と呼ばれる単位に区画化し，プログラム内でのプライバシ情報の伝搬範囲を限定する形でアプリ開発を行う仕組みを提案した．開発者が，適切に記述されたアプリケーションプライバシポリシに準じた形でドメインを分離しながらアプリ開発を行うことにより，プライバシ情報は適切なドメイン内に囲い込まれるため，これによりマーケット運営者は，簡単な静的解析のみでアプリとアプリケーションプライバシポリシの整合性を検証することが可能となる．提案方式を Android アプリ開発に適用した開発手法を考案し，開発制約によって開発者に課される制約が従来のアプリ開発と比較して過負荷にならないことを評価した．

今後の課題としては，実際に静的解析検査を行うツールを構築するためにプライバシポリシ記述と対応する包括的なソースメソッドのリストおよびその外部送信に利用されるシンクメソッドのリストの構築，および，提案方式を適用したアプリ開発時の負荷をさらに軽減するための開発手法や，従来のアプリ開発と同レベルのソフトウェア品質（保守性や拡張性など）を維持できるような開発手法を考案することがあげられる．これには現在提案している開発方式の見直しや開発補助ツールの考案が含まれている．

謝辞 本研究を進めるにあたり KDDI 研究所の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, 入手先 ⟨http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000087.html⟩.
[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，入手先 ⟨http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000111.html⟩.
[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).
[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).
[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).
[6] Felt, A.P. et al.: How to Ask For Permission, *Proc. USENIX Workshop on Hot Topics in Security* (2012).
[7] Rosen, S. et al.: AppProfiler: A Flexible Method of Exposing Privacy-Related Behavior in Android Applications to End Users, *Proc. 3rd ACM Conference on Data and Application Security and Privacy*, pp.221–232, ACM (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).
[9] Enck, W. et al.: TaintDroid: An Information—Flow Tracking System for Realtime Privacy Monitoring on Smartphones, *Proc. 9th USENIX Symposium on Operating Systems Design and Implementation*, Canada (2010).
[10] 上松晴信，可児潤也，名坂康平，川端秀明，磯原隆将，竹森敬佑，西垣正勝：安全な Android アプリの提供を実現するアプリ開発・管理方式 ADMS の提案，コンピュータセキュリティシンポジウム 2011 論文集，pp.774–779 (Oct. 2011).
[11] My Tracks for Android, available from ⟨https://code.google.com/p/mytracks/⟩.
[12] F-Droid | Free and Open Source Android App Repository, available from ⟨https://f-droid.org/⟩.
[13] Java セキュアコーディングスタンダード SEC05-J，入手先 ⟨https://www.jpcert.or.jp/java-rules/sec05-j.html⟩.
[14] 7 Foundational Principles：Privacy By Design, available from ⟨https://www.privacybydesign.ca/index.php/about-pbd/7-foundational-principles/⟩.
[15] McCluskey, G.: Using Java Reflection, available from ⟨http://www.oracle.com/technetwork/articles/java/javareflection-1536171.html⟩.
[16] Rasthofer, S., Arzt, S. and Bodden, E.: A machine-learning approach for classifying and categorizing Android sources and sinks, *Network and Distributed System Security Symposium* (*NDSS 2014*) (2014).
[17] Arzt, S. et al.: FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps, *Proc. 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp.259–269 (2014).
[18] My Tracks – モバイル Google マップ ヘルプ，入手先 ⟨https://support.google.com/gmm/answer/2391538?hl=ja⟩.

小林 真也

2014 年 3 月静岡大学情報学部情報科学科卒業．2016 年 3 月同大学大学院修士課程修了．同年株式会社 jig.jp 入社．在学中，情報セキュリティに関する研究に従事．

鈴木 富明

2013 年 3 月静岡大学情報学部情報科学科卒業．2015 年 3 月同大学大学院修士課程修了．同年大日本印刷株式会社入社．在学中，情報セキュリティに関する研究に従事．

可児 潤也

2012 年 3 月静岡大学情報学部情報科学科卒業．2014 年 3 月同大学大学院修士課程修了．同年株式会社富士通研究所入社．在学中，情報セキュリティに関する研究に従事．

川端 秀明 （正会員）

2010 年東海大学大学院工学研究科情報通信制御システム工学専攻前期博士課程修了．同年 KDDI 株式会社入社．現在，株式会社 KDDI 研究所で，モバイル OS セキュリティ，ネットワークセキュリティに関する研究開発に従事．2011 年 CSS 優秀論文賞，2012 年 DICOMO 優秀論文賞，2013 年山下記念研究賞，2014 年喜安記念業績賞を各受賞．

西垣 正勝 （正会員）

1990 年静岡大学工学部光電機械工学科卒業．1992 年同大学大学院修士課程修了．1995 年同大学院博士課程修了．日本学術振興会特別研究員（PD）を経て，1996 年静岡大学情報学部助手．同講師，助教授の後，2006 年より同創造科学技術大学院助教授．2007 年同准教授，2010 年同教授．博士（工学）．情報セキュリティ全般，特にヒューマニクスセキュリティ，メディアセキュリティ，ネットワークセキュリティ等に関する研究に従事．2013～2014 年情報処理学会コンピュータセキュリティ研究会主査．2015 年より電子情報通信学会バイオメトリクス研究専門委員会委員長．