コンポジットアプリケーションモデルにおける セキュアなフロントエンド連携手法の提案

浦邊 信太郎† 児玉 英一郎‡ 王 家宏‡ 高田 豊雄‡

†株式会社 日立ソリューションズ東日本
980-0014 宮城県仙台市青葉区本町二丁目 16 番 10 号 NBF 仙台本町ビル
shintaro.urabe.dc@hitachi-solutions.com

‡岩手県立大学　ソフトウェア情報学部
020-0693　岩手県滝沢市巣子 152-52
{ kodama, wjh, takata }@iwate-pu.ac.jp

あらまし　Web アプリケーションの開発では，既存資産を組み合わせて一つのアプリケーションを構成するコンポジットアプリケーションモデルという手法が使われている．当該モデルでは個々のアプリケーションをフロントエンドで連携させることができるが，悪意のあるアプリケーションによる情報漏洩の危険性がある．これに対してホストアプリケーション側で連携する経路を予め指定し，通信範囲を限定する方法が知られているが，アプリケーションを柔軟に組み込むことができないという問題があった．本稿では，バックエンドの認証情報から構築する共通鍵を利用することで，柔軟性を損なわずにセキュアな連携を実現するフロントエンド連携手法を提案する．

A Secure Frontend Messaging Method for Web Composite Application Models

Shintaro Urabe† Eiichiro Kodama‡ Wang Jiahong‡ Toyoo Takata‡

†Hitachi Solutions East Japan, Ltd.
2-16-10, Honcho, Aoba-ku, Sendai, Miyagi 980-0014, Japan
shintaro.urabe.dc@hitachi-solutions.com
‡Iwate Prefectural University
152-52, Sugo, Takizawa, Iwate 020-0693, Japan
{ kodama, wjh, takata }@iwate-pu.ac.jp

Abstract　For a composite application, its composed functions generally use the publish/subscribe messaging system to interact with each other. This kind of composite applications potentially have a risk of incorporating malicious functions which eavesdrops on messages exchanged among others. Previous studies suggested the secure communication-based method such that, the host application limits function's interactions to the predetermined communication paths, and composed functions can interact with each other only with the permission of the host applications. This requires that, however, a host must know in advance the qualified correspondents, which may decrease the flexibility of incorporating new composed functions. In this paper, we propose an encrypted communication-based method using the symmetric-key cryptosystem. Users can incorporate new composed functions without loss of flexibility while keeping function's data and communication secure.

1 はじめに

企業での IT システム構築の低コスト化，短納期化を実現するために, 既存の業務アプリケーションを組み合わせたコンポジットアプリケーションが多用されている. コンポジットアプリケーションモデルでは, 必ずしも同一の開発者が開発したとは限らない単純機能を持つアプリケーションをホストアプリケーション上で組み合わせることにより, ユーザの要求に合わせた複雑な機能を実現する. 特に, 近年では Adobe Flash[3] や Microsoft Silverlight[4]に代表されるような, アプリケーションの Web サービス化が進展し，ユーザがブラウザ上で機能や UI を柔軟に組み合わせることが可能になった.

Web 上に構築するコンポジットアプリケーションモデルでの機能連携は, HTTP 等を用いたバックエンドでのマッシュアップの他に, ブラウザ上で動作するフロントエンドでの連携手段を利用できる. フロントエンドの連携として標準的に用いられている手法は，出版-購読型（Publisher-Subscriber）モデルによるメッセージ交換方式である．出版-購読型モデルでは出版者（送信側）がメッセージをメッセージブローカに送信する．購読者（受信側）はメッセージブローカに購読を登録し, 個々のメッセージに対して自身に向けられたメッセージかどうかを判別して受信する. クライアントサーバモデルでは相手の場所や状態を意識する密結合であるのに対し, 出版-購読型モデルは疎結合であり, 互いに相手の位置や状態を知る必要が無いという特長がある.

しかし, 出版-購読型モデルは出版, 購読の機密性が担保できないというセキュリティの問題が知られている[1]. これに対して, この課題に取り組んだ関連研究[5][6]では様々な方法が提案されているが, それらの研究ではアプリケーション内に第三者がアクセスできない秘密情報を持たせることを前提としている．しかし，特定のコンポジットアプリケーションモデルではこの前提が適用できない場合がある．例えば Web 上でのコンポジットアプリケーションのフレームワークとして普及している OpenSocial[2]では，アプリケーションは XML と JavaScript で記述する．アプリケーションが手元にあればそのソースコードを閲覧することができるため, アプリケーション自体に一切の秘密情報を持たせることができない．また，Adobe Flash や Microsoft Silverlight などパッケージ化された Web アプリケーションでも，リバースエンジニアリングにより内部の情報を知られる危険性が存在する.

2 想定しているシステムのモデル

本研究で想定するアプリケーションのモデルは Web 上のコンポジットアプリケーション実現形式の一つである．ホストアプリケーションとして HTML で記述された Web アプリケーションを想定し，これを「コンテナ」と呼ぶ．また，アプリケーションとして同じく HTML で記述された Web アプリケーションを想定し，これを「ガジェット」と呼ぶ. ガジェットはコンテナへのインストール処理によってコンテナ内に配置される. コンテナはユーザ認証機能を持ち, ユーザはログイン画面で正しいユーザ ID とパスワードを入力しなければコンテナ内の情報にアクセスできない. ユーザは権限によって管理者と一般ユーザに分けられる. 管理者はガジェットへのアクセスの他, ガジェットのインストール，ユーザの追加や削除，権限の変更を行うことができ, 一般ユーザはガジェットへのアクセスのみを行うことができる.

コンテナの管理者は, 業務に必要なガジェットを適宜取得してインストールする. ガジェットの取得は，自組織での開発，ベンダからの購入，無料ガジェットの配布サイトからのダウンロードなど様々な形態が考えられる. 前述したようにガジェットには他のガジェットとの出版-購読モデルによる連携インタフェースを持たせることができる．しかし，通常の出版-購読モデルでは，購読を登録している不特定多数のガジェットにメッセージが配信されるため, 特定のガジェットを宛先とする機密情報の送信等には使用できない. これに対応するためには, 宛先のガジェット以外に情報が漏洩しないようにするための仕組みが必要である．本研究では，上述のようなシステム上でのガジェット間のメッセー

ジ交換において，メッセージの機密性を確保する手法を提案する．

3　関連研究

出版-購読型モデルにおいて通信のセキュリティを確保する手法として，通信を行う主体が持つ認証情報を利用する方法が知られている[5][6]．しかし 2 章で述べたように，本研究で前提とする環境では通信の主体はガジェットで，ガジェットを取得したユーザはそのソースコードを見ることができる．そのため当該環境では，ガジェットの公開時にガジェット自体にその信頼性を担保する情報を持たせることができない．

このような環境においてガジェット間通信のセキュリティを確保する技術として，OpenAjax Alliance による OpenAjax Hub 2.0[7]がある．OpenAjax Hub はメッセージの交換を行う通信路（ハブ）としてセキュリティの機能を持たない Unmanaged Hub と，不正なガジェットによる攻撃を防ぐ機能を持つ Managed Hub という二種類のハブを提供する．Managed Hub はアクセスできるガジェットを限定することにより，悪意のあるガジェットからのアクセスを遮断する．Managed Hub の仕組みを図 1 に示す．コンテナでは予めハブ（Managed Hub）を構築し，コンテナがガジェットをロードする際に，個々のガジェットが使用するハブを設定する．例えば，図ではガジェット 1 をロードした際に当該ガジェットをトピック A とトピック B のハブインスタンスに結び付ける．同様にガジェット 2 をロードした際にトピック A のハブインスタンスに，ガジェット 3 をロードした際にトピック B のハブインスタンスに結び付ける．これにより，ガジェット 3 はトピック A にアクセスできないため，ガジェット 1 とガジェット 2 との通信はガジェット 3 からの傍受を防ぐことができる．同様にガジェット 1 とガジェット 3 の通信はガジェット 2 の傍受から守られる．また，個々のガジェットはハブが Managed Hub であることを意識しないで，通常のハブと同じ方法でメッセージのやりとりができる．

このように，OpenAjax Hub 2.0 の Managed Hub は予めコンテナで個々のガジェットの通信経路を定義することにより，ガジェット間通信におけるメッセージの機密性を確保することが可能である．しかし，コンポジットアプリケーションモデルの特長として，コンテナは一定の規格に沿ったガジェットであれば，ガジェットの種類を問わず自由にインストールできることが重要である．Managed Hub ではコンテナが自身に組み込まれるガジェットを意識する必要があり，ガジェット導入の柔軟性を失わせることになる．例えば，企業での業務アプリケーションでコンテナ内のガジェットが頻繁に入れ替わるような場合，通信経路の再定義によるコストの増加や，通信経路の設定ミスに起因するセキュリティの問題などが発生すると考えられる．

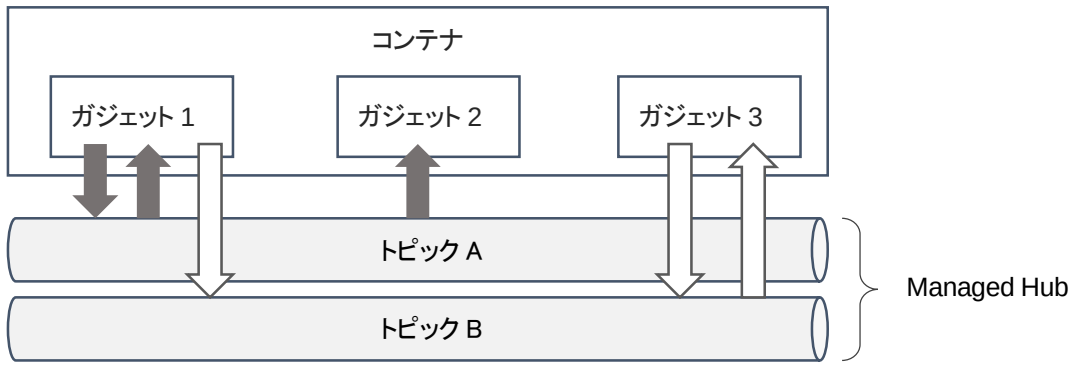

図 1 OpenAjax Hub 2.0 の Managed Hub

4　暗号化によるセキュアなフロントエンド連携手法の提案

4.1　提案手法の概要

セキュアな通信を行う二つのガジェットをそれぞれg,fとし，g,fがインストールされているコンテナをcとする．提案手法では，通信を予定しているガジェット間で予め通信鍵（共通鍵）$k_{g,f,c}$を共有しておき，g,f間のセキュリティを要求する通信は全てこの通信鍵$k_{g,f,c}$を用いた暗号化を行う．$k_{g,f,c}$はコンテナc，ガジェットg，ガジェットfの組み合わせに関連付けられる．これは，コンテナcにおけるガジェットgとf間の通信に対してのみ用いられることを意味する．そのため，コンテナcでの異なるガジェットh,i間では異なる通信鍵$k_{h,i,c}$が用いられ，また，別のコンテナdでのガジェットg,f間の通信でもまた異なる通信鍵$k_{g,f,d}$が使用される．

通信鍵を構成する要素として「シークレット」という文字列を含める．シークレットはガジェット一つ一つに対応付けられる秘密文字列で，「ガジェットサーバ」という主体が保持する．ガジェットサーバはガジェットの信頼性を担保する機能として，「ガジェットサービス」という Web サービスを提供する．ガジェットサービスは通常ガジェットのベンダが運営する Web サービスとして公開される．ガジェット同士のセキュアな連携は「ガジェットベンダ同士の契約」という形で担保する．例えば，あるガジェットベンダsが，自組織のガジェットgを別のベンダtのガジェットhと連携させたい場合，ベンダsとtとで契約を締結する．そして，ベンダ同士で安全な方法（郵送，メールでの送信等）を用いてガジェットのシークレットを交換する．この例では，ガジェットベンダsがgのシークレットsec_gをベンダtに送付し，ベンダtはhのシークレットsec_hをsに送付する．これらのシークレットを共通鍵の要素とすることで，シークレットを持つ正当なガジェットベンダのみが鍵を生成できることを保証する．

ガジェットサーバはコンテナの要求によりガジェットに対して鍵を生成し配布する役割を担う．コンテナcはガジェットgがインストールされるとgに記述されているガジェットベンダsのサーバ（以下，ガジェットサーバs）の URL を参照し，ガジェットサーバsに鍵を要求する．このとき，コンテナcは通信鍵返送用（以下，鍵返送用と呼ぶ）の URL を送付する．また，当該 URL はコンテナ自身を他のコンテナと識別するための ID としても用いられる．コンテナの ID が通信鍵の要素となることにより，コンテナに関連付けられた通信鍵を生成できる．そのため，不正なコンテナが通信鍵を要求した場合でも，その不正なコンテナの ID を要素として生成された通信鍵は正当なコンテナでの通信には使用できない．また，不正なコンテナが自身の ID を正当な ID に偽装して通信鍵を要求した場合，当該通信鍵は正当な ID（返送用 URL）へ返送されるため，

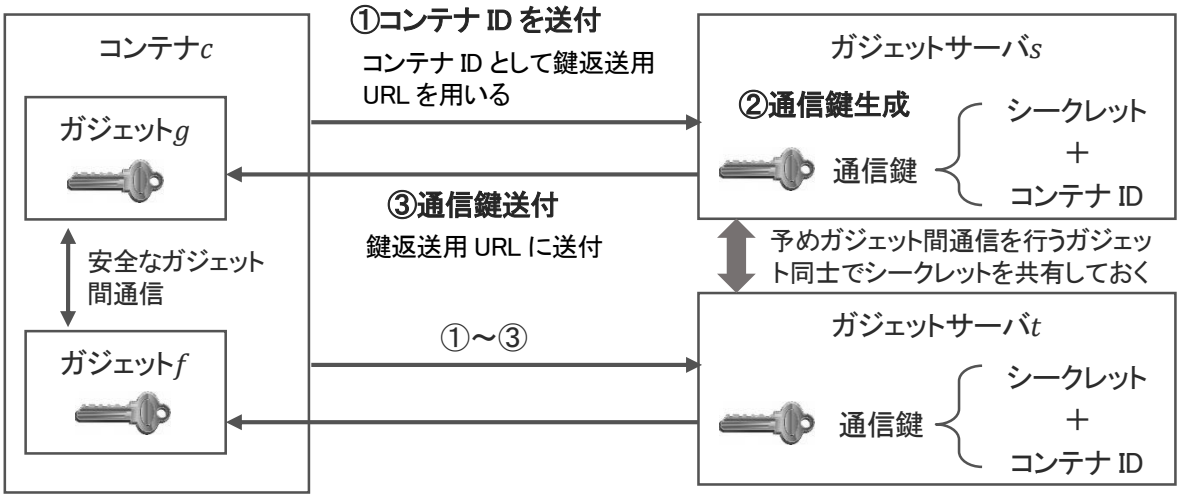

図 2 提案手法の概要

不正なコンテナは通信鍵を受け取ることができない.

提案手法の手順を図 2 に示す. ガジェットのコンテナへのインストールを契機として，①コンテナ ID（鍵返送用 URL）のガジェットサーバへの送付が開始される. ②ガジェットサーバでは，予め交換されたシークレットとコンテナ ID を連結し，ハッシュ関数を用いて通信鍵を生成する. ③生成した通信鍵はコンテナ ID（鍵返送用 URL）に送付される. 次節からは本提案手法の詳細なプロトコルについて説明する.

4.2 定義

以下に記号及び操作の定義を示す.

記号	説明
g, f, h	ガジェット
s, t	ガジェットサーバ
c	コンテナ
id_c	コンテナcの ID （鍵返送用 URL）
G	システム全体のガジェットの集合
S	システム全体のガジェットサーバの集合
C	システム全体のコンテナの集合
G_s	ガジェットサーバsが所有するガジェットの集合
I_c	コンテナcにインストールされているガジェットの集合
SEC_s	ガジェットサーバsが所有するシークレットの集合
KEY_g	ガジェットgが所有する通信鍵の集合
$k_{g,f,c}$	コンテナcにおいて，ガジェットg, f間の通信に使う通信鍵
$digest(g)$	ガジェットgのダイジェストを計算する操作
$digest_g$	ガジェットgのダイジェスト（$digest_g = digest(g)$）
$hash(x)$	xのハッシュ値を計算する操作

4.3 前提条件

本提案手法が前提とする条件を以下に示す.

- 攻撃者が正当なコンテナにログインすることはできないが，攻撃者のガジェットが正当なコンテナにインストールされる可能性はある.
- 攻撃者は不正なガジェットg^*，コンテナc^*，ガジェットサーバs^*をそれぞれ構築することができる.
- 攻撃者の持つ情報は$\langle G, S, C \rangle$であるが，他社のガジェットgの通信鍵KEY_g，及びガジェットサーバsのシークレットSEC_sは分からない.
- 正当なコンテナc，ガジェットサーバsは正常なプロトコルに手順通り従うものとする.
- 正当なコンテナcにインストールされたガジェットI_cの通信鍵はcが意図しない形で外部に漏洩することはない.
- 正当なガジェットサーバsが取り扱うシークレットSEC_sはsが意図しない形で外部に漏洩することはない.
- 任意のガジェットサーバとコンテナ間の通信は https 等を用いた安全な通信であり，第三者から傍受されることはない.
- ガジェットは同じコンテナにある他のガジェットの通信鍵を見ることはできない.

4.4 ガジェットへの鍵の配信

2 章で挙げた通り，想定しているシステムのモデルではガジェットのソースコードが公開されるため, ガジェットの中に通信鍵を埋め込んで公開すると第三者に通信鍵が渡ってしまう危険性がある. そのため, 本提案手法ではガジェットの公開時点では通信鍵を保持せず, ガジェットがインストールされたタイミングで当該ガジェットが属するガジェットサーバからコンテナを通して通信鍵を取得する.

コンテナcはガジェットgがインストールされるとgに記述されているガジェットサーバsの URL を参照し，sに通信鍵を要求する．以下に c からの通信鍵要求と通信鍵の生成，取得の手順を示す.

Step1: cはgのダイジェスト$digest_g$を計算する.
Step2: cからsに$id_c, digest_g$を送付する.
Step3: $sec_g \in SEC_s$ かつ $sec_f \in SEC_s$，かつ $digest_g \in \{digest(i)|i \in G_s\}$の場合，以下の式により通信鍵を生成する．このとき，gのガジェットサーバであるsは当然gと通信するガジェットがfであることを知っており，かつfのベンダとの契約が締結されていればsec_fを持っている.

$$k_{g,f,c} \leftarrow hash(sec_g + sec_f + id_c) \qquad (1)$$

Step4: sからid_cに$k_{g,f,c}$を送付する.
Step5: cはgに$k_{g,f,c}$を組み込む.

上記の Step3 では，通信鍵の要求を受信した際に，要求元のガジェットについて自身が所有するガジェットであるか，当該ガジェットのシークレットを持っているか，当該ガジェットが通信するガジェットのシークレットを持っているかを確認する．これにより，ガジェットの信頼性はガジェットサーバにより担保されることになる.

4.5 通信鍵を利用したガジェット間通信

以下にgからfへのメッセージ送信の手順を示す．また，ここでは予めfはgのトピックを購読（Subscribe）しているものとする.

Step1: gは$k_{g,f,c}$を用いてメッセージmを暗号化し，暗号化メッセージeを生成する.
Step2: gはeを出版（Publish）する.
Step3: fはgの購読からeを受信する.
Step4: fは$k_{g,f,c}$を用いてeを復号し，メッセージmを得る.

上記で示したように，提案手法でのガジェット間通信はメッセージを暗号化する以外で特殊な手順を用いることは無い．そのため，既存のガジェットを提案手法によるセキュアなガジェットに改良する場合でも，その工数は最小限で済む.

4.6 鍵の更新

ガジェットの鍵の更新はコンテナ単位で行われる．コンテナが更新のポリシー（毎月 1 回など）を設定し，更新のタイミングで当該コンテナのすべてのガジェットの通信鍵を一斉に更新する．個々のガジェットでの通信鍵更新の手順は前述の通信鍵取得と同様である．ただし，通信鍵の更新の前にガジェットサーバが保持するシークレットも更新しておく必要がある.

5 評価

本章では提示した課題に対して提案手法が有効であることを示す.

5.1 安全性についての評価

不正なガジェットg^*（$g^* \in I_c, k_{g,f,c} \notin KEY_{g^*}$）が$g$と$f$の通信傍受を試みる場合，$g^*$は$k_{g,f,c}$を持たないため，$k_{g,f,c}$によって暗号化されたメッセージ$e$を復号することはできない．そのため，$g^*$が通信鍵を取得しない限り，本研究で提示した課題は解決できる.

以下では，攻撃者が不正なコンテナc^*，ガジェットサーバs^*（$sec_g \notin SEC_{s^*}$ または $sec_f \notin SEC_{s^*}$），ガジェットg^*（$g^* \notin G_s$）を構築して通信鍵$k_{g,f,c}$の取得を試みる場合において，通信鍵が取得できないことを示す.

- c^*がsに対し$digest_g$とid_{c^*}を送付して通信鍵取得を試みる場合，sではk_{g,f,c^*}を生成しid_{c^*}に送信するが，$k_{g,f,c^*} \neq k_{g,f,c}$であり，$c^*$は$k_{g,f,c}$を取得することはできない.
- c^*が自身の ID をid_cに偽装し，sに対し$digest_g$とid_cを送付して通信鍵取得を試みる場合，通信鍵$k_{g,f,c}$はid_cに返送されるため，c^*は$k_{g,f,c}$を取得することはできない.
- 攻撃者がgのガジェットサーバの URL をs^*に書き換え，cへのインストールを通して通信鍵取得を試みる場合，通信鍵要求（$digest_g, id_c$）がs^*へ送付されるが，s^*はsec_g及びsec_fを知らないため，$k_{g,f,c}$を作成することはできない.

- g^*がcへのインストールを通して通信鍵取得を試みる場合，cは通信鍵要求（$digest_{g^*}$，id_c）をsへ送付する．しかし，$digest_{g^*} \notin \{digest(i)|i \in G_s\}$であるため，$s$における$k_{g,f,c}$の作成は拒否される．

5.2 鍵漏洩の影響についての考察

4.3 節で挙げた前提条件が満たされなかった場合，通信鍵やシークレットが漏洩する危険性がある．その場合は目的としたセキュリティを確保することができないが，提案手法では通信鍵をガジェットやコンテナに関連付けることにより危険性が及ぶ範囲を限定する．以下では，攻撃者が$k_{g,f,c}$を入手した場合について考察する．

- 攻撃者のガジェットg^*がコンテナcにおけるgとfの通信を傍受する場合，ガジェットg^*は当該通信鍵により，コンテナcにおけるガジェットgとfの通信の傍受が可能である．ただし，sec_gとsec_fを取得後にシークレットの更新及びコンテナの鍵更新が実行された場合，通信鍵$k_{g,f,c}$は無効になる．
- 攻撃者のガジェットg^*がコンテナcにおけるガジェットgとhの通信を傍受する場合，通信鍵$k_{g,f,c}$はコンテナcにおけるガジェットgとf間でのみ有効なため，ガジェットgとhの通信の傍受はできない．
- 攻撃者のガジェットg^*がコンテナc'におけるgとfの通信を傍受する場合，当該通信鍵はコンテナcにおけるガジェットgとf間でのみ有効なため，コンテナc'におけるガジェットgとfの通信の傍受はできない．
- 攻撃者が他の通信鍵の取得を試みる場合，通信鍵$k_{g,f,c}$及び攻撃者の保持している情報（G, S, C）から他の通信鍵を作成することはできない．

5.3 シークレット漏洩の影響についての考察

攻撃者がシークレットsec_g，シークレットsec_fのどちらか，または両方を入手した場合について考察する．

- 攻撃者がシークレットsec_gを取得した場合でも，cとsec_gのみでは通信鍵$k_{g,f,c}$を作成することはできない．
- 攻撃者がシークレットsec_gとsec_fの両方を取得した場合，id_cの取得は容易なため通信鍵$k_{g,f,c}$を作成することができる．ただし，sec_gとsec_fを取得後にシークレットの更新及びコンテナの鍵更新が実行された場合，元のsec_gとsec_fから作成した通信鍵は無効になる．

6 おわりに

6.1 結論

コンポジットアプリケーションでは出版-購読モデルによるフロントエンド連携が用いられているが，交換されるデータの漏洩が問題となっていた．また，既存の手法ではガジェット導入の度にコンテナの設定変更が必要になり，管理コストの増加や設定ミスによるセキュリティリスク増加の懸念があった．

この課題に対し，交換されるメッセージを暗号化することによりデータの漏洩を防ぐ手法を提案した．本提案手法の特長を以下に示す．

- OpenSocial のようにガジェットのソースコードが公開されるモデルを考慮し，公開するガジェットには通信鍵を組み込まず，コンテナへのインストールの後に配布する．
- コンテナのユーザは通常のコンテナと同じように柔軟にガジェットを組み込むことができる．コンテナはガジェットの要求に応じて通信鍵を要求するが，ガジェットやガジェットサーバ，通信鍵の信頼性は意識する必要が無い．

- 通信鍵はガジェットベンダで生成し配布される. ベンダ同士の契約という out-of-band の情報を利用することで通信鍵の信頼性を担保する.
- 通信鍵は通信する二つのガジェットと，それらがインストールされているコンテナの組み合わせに対して固有となる. そのため, ある通信鍵が漏洩したとしても，データの傍受が可能な範囲は，当該鍵に関連付けられるコンテナにおける当該鍵を利用する二つのガジェット間通信に限定される.

6.2 今後の課題

提案手法では通信を行う 2 つのガジェットの組み合わせに対して固有の鍵を生成するため, 通信する相手の数に応じて保持する鍵が増加する．また, あるメッセージをマルチキャストする際には送信相手の数だけ暗号化と送信の処理を行わなければならない．一般的な Web コンポジットアプリケーションでは, 画面の制約もありガジェットの数が膨大になることは考えにくいが, メッセージのサイズが大きい場合にはブラウザ上の処理コストが大きくなる可能性がある．この問題への解決策として, マルチキャストのためにガジェットのグループで共有できる鍵を用意する手法が考えられる.

また, 提案手法ではガジェットの信頼性の担保をガジェットサーバに委任する．しかし，すべてのガジェット開発者がガジェットサーバを用意することは現実的ではない. この問題を解消するための手段として, 個人が開発したガジェットの信頼性を保証するための，パブリックな Web サービスなどが考えられる. 開発したガジェットの登録と，開発者間でのガジェット間通信の契約機能を持たせることにより, 本提案手法におけるガジェットサーバとして機能させることができる.

参考文献

[1] C. Wang, A. Carzaniga, D. Evans, A. Wolf: Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems, Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 (2002)

[2] OpenSocial, http://opensocial.org/, Accessed 2014 年 4 月 25 日

[3] Adobe Flash, http://www.adobe.com/jp/products/flashplayer.html, Accessed 2015 年 8 月 6 日

[4] Microsoft Silverlight, https://www.microsoft.com/ja-jp/silverlight, Accessed 2015 年 8 月 6 日

[5] Opyrchal, L., Prakash, A., Agrawal, A.: Designing a Publish-Subscribe Substrate for Privacy-Security in Pervasive Environment, In First Workshop on Pervasive Security, Privacy and Trust (PSPT) (2004).

[6] Dmitrij Lagutin, Kari Visala, Sasu Tarkoma: Publish-Subscribe for Internet PSIRP Perspective, Valencia FIA book, vol. 4 (2010).

[7] OpenAjax Hub 2.0, http://www.openajax.org/member/wiki/OpenAjax_Hub_2.0_Specification, Accessed 2015 年 8 月 5 日