情報処理学会論文誌 Vol.58 No.9 1419–1431 (Sep. 2017)

Malvertisingに対するセキュアな広告配信方式

坂本 一仁[1,a)] 稲垣 俊[1] 島岡 政基[1] 松永 昌浩[1]

受付日 2016年11月29日, 採録日 2017年6月6日

概要：インターネット広告の広告配信を悪用するMalvertisingと呼ばれる攻撃が増加している．Malvertisingは短期間に大規模なWebサイト閲覧ユーザへの攻撃が可能であり，その攻撃は年々高度化・巧妙化している．そのため，これまでに提唱されている対策では，十分な効果が得られていない．本稿では，既存のインターネット広告における複雑な広告配信と，高度なMalvertisingに対応可能な，電子署名のプロトコルを導入したセキュアな広告配信方式を提案し，その実装と評価を行った．我々の提案方式は，広告配信に関して統一的なセキュリティが確保でき，ユーザに特別な対策は必要なく，Malvertisingの脅威を軽減するものである．評価結果から，我々の提案方式は脅威を未然に遮断する仕組みを，小さなオーバヘッドで実現できることを明らかにした．さらに，提案方式は拡張性があり，運用のフレームワークと組み合わせれば，より強固なMalvertising対策を実施できる展望を示した．

キーワード：Malvertising，広告配信方式，電子署名

A Secure Advertising Protocol against Malvertising

Takahito Sakamoto[1,a)] Shun Inagaki[1] Masaki Shimaoka[1] Masahiro Matsunaga[1]

Received: November 29, 2016, Accepted: June 6, 2017

Abstract: A mass attacking named malvertising is increasing in Internet advertising. Malvertising is able to attack a lot of users visiting various web sites rapidly. Malvertising has become powerful and clever year by year, thus it is not effective enough to counter the attack with previous proposed plans. In this paper, we show a secure advertising protocol using digital signature. Moreover, we implement and evaluate our protocol. Our protocol is able to block even a powerful and clever malvertising, and apply a complex advertising ecosystem in Internet today. Our results of the evaluation in this paper show that our protocol is able to implement a small overhead. Our protocol mitigates malvertising with a more unified security regard to advertising, and without a special setting by users. Furthermore, this paper shows that our protocol is scalable and able to maintain high security if the protocol combined with an operational framework.

Keywords: Malvertising, advertising protocol, digital signature

1. はじめに

インターネット広告では，多数の広告事業者が相互に連携し，様々な広告配信方式を用いてユーザに広告を配信している．中でも広告配信方式の主流は，RTB（Real Time Bidding）と呼ばれる方式 [1] であり，RTBに関連する広告収益は広告収益全体の約52%に及んでいると報告されている [2]（米国2014年）．この方式では**図 1** のように，ニュースやブログ等の媒体（Publisher）サイトの広告枠を管理する事業者（SSP: Supply Side Platform）と，広告主の広告コンテンツを管理する事業者（DSP: Demand Side Platform）の間で，ユーザが媒体サイトに訪問した時点で自動的なオークションを実施し，配信する広告とその広告料をリアルタイムに決定する．そして，オークションに勝利したDSPは一般にCDNを利用して広告を配信する．RTBのような複雑な広告配信では，ユーザの端末は広告を表示するまでに，複数の広告事業者の広告配信サーバやネットワークを経由することになる．

[1] セコム株式会社 IS 研究所
Intelligent Systems Laboratory, SECOM CO., LTD., Mitaka, Tokyo 181–8528, Japan
[a)] takah-sakamoto@secom.co.jp

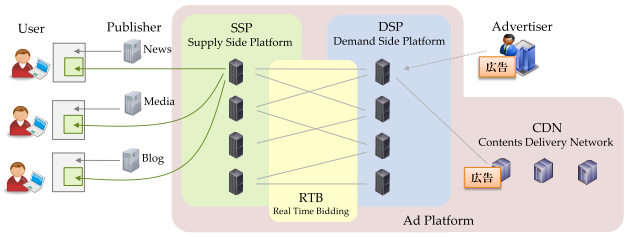

図 1 RTB による広告配信の概要
Fig. 1 An overview of real time bidding.

ここで，上記のような広告配信において，ユーザは広告を表示するまでにセキュリティ面で脆弱な広告事業者のサーバまたは悪意のある広告事業者のサーバを 1 つでも経由すると，マルウェア感染等の脅威にさらされることになる．攻撃者からすれば，たとえば広告事業者の脆弱なサーバへの攻撃が 1 つでも成功すれば，その広告事業者と連携している他の広告事業者のネットワークを通じて，Web サイトを閲覧している広範囲のユーザへ，大規模なマルウェア配信の攻撃が可能となる．このようなインターネット広告の広告配信を悪用した攻撃は Malvertising と呼ばれている．Malvertising は Malware と Advertising を組み合わせた造語である．具体的な攻撃としては，広告配信中のレスポンスである次の広告配信サーバの宛先 URL 等を DbD（Drive-by Download）サイトの宛先へ変更し，不正にユーザを誘導する方法である [3], [4], [5], [6].

これまでに提唱されている Malvertising の対策としては，ユーザ端末の状態を最新に保つこと [7]，ウィルス対策ソフトを導入すること [3] 等，ユーザ側の対策がある．そして，広告事業者が広告配信サーバのセキュリティや入稿される広告コンテンツの検証を強化すること [7]，不正な連携先への転送をフィルタリングすること [8] 等，広告事業者側の対策がある．上記のようにユーザ側と広告事業者側で基本的な対策は打ち出されているが，一方で Malvertising の攻撃は年々高度化・巧妙化している現状がある．Malvertising にゼロデイの脆弱性が利用された攻撃 [9] では，現状のユーザ側の対策が困難である．さらに，正規広告事業者のサブドメインを不正に取得し，巧妙にドメインによるフィルタリングを回避するような Malvertising [10] では，現状の広告事業者側の対策が困難である．以上から，現状の Malvertising 対策では十分な効果を得られているとはいえない．

そこで我々は，こうした広告配信におけるレスポンスの変更と DbD サイトへの誘導を主とした Malvertising の攻撃を防ぐために，広告事業者が電子署名を利用したセキュアな広告配信のプロトコルを導入する提案をしている [11]．我々の提案方式は，広告配信においてセキュリティの信頼点となる認証事業者 TTP（Trusted Tag Provider）を導入し，TTP が広告配信にかかわるすべてのレスポンス（次の宛先 URL や広告コンテンツ）について事前に電子署名を付与して，広告配信時に TTP のタグ（iframe タグと JavaScript）がユーザの端末で署名検証を行うものである．提案方式では，レスポンスの署名をユーザ端末で検証することで，DbD サイトへの誘導を防ぐことが可能となり，さらにゼロデイの脆弱性やフィルタリング回避策等がとられた高度な Malvertising であっても，署名値が異なるため検出可能である．

本稿では，我々の提案方式の Malvertising に対する安全性を示し，プロトタイプの実装と性能評価を行った．そして評価結果の考察および提案方式の実現に向けた議論を行った．本稿の貢献として，提案方式は広告配信中のレスポンスが変更される Malvertising に対して安全性を提供する仕組みとして，小さいオーバヘッド（約 4%）で実施が可能であることを示した．また，提案方式はプロトコルによる対策のため，広告事業者が個別にセキュリティ対策を行うよりも広告配信に関して統一的なセキュリティを確保でき，ユーザは特別な対策の必要なく安全な広告が受信できるといえる．さらに，提案方式は拡張性があり，運用のフレームワーク（トラストフレームワーク）と組み合わせることで，より強固な Malvertising 対策が可能となることを考察した．

以降，本稿の構成は次のとおりである．2 章では関連研究として，Malvertising の事例，既存研究および対策を紹介し，我々のモチベーションを述べる．3 章では，対象とする広告配信モデル，攻撃モデルおよび我々の提案方式について説明する．4 章では，提案方式のオーバヘッド評価を行った結果を示す．5 章では，評価結果の考察および提案方式の利点や拡張性，運用のフレームワークとの組合せについて議論する．6 章でまとめと今後の展望を述べる．

2. 関連研究

本章では Malvertising の調査として，事例や Malvertising を対象とした既存の研究，Malvertising が高度化・巧妙化している点について紹介する．そして，Malvertising の対策を紹介し，既存の対策では困難な点，そして我々の取り組みの位置付けを明確にする．

2.1 Malvertising の調査

まず，Malvertising の事例をいくつか紹介し，攻撃方法の特徴を示す．2016 年 3 月の事例 [3], [4] では，大手広告事業者の AppNexus の広告配信サーバが悪用され，AppNexus の広告配信を通じて大手ニュースサイト等の閲覧ユーザ数万人が DbD サイトへ転送され，マルウェアの脅威にさらされたと報じている．また 2015 年 1 月の事例 [5] では，Huffington Post における AOL Ad-Network での広告配信において DbD サイトへ転送されたことや，2014 年 1 月の事例 [6] では，yahoo.com の広告ネットワークから DbD サイトへ転送されたこと等が報告されている．Malvertising の攻撃では，広告配信中のユーザへのレスポンス（次の宛

先 URL や広告コンテンツ）を DbD サイトへ転送できるような iframe タグや JavaScript に変更し，DbD サイトで Exploit Kit を実行する方法がとられている．

次に，Malvertising の攻撃が年々高度化・巧妙化している事例を紹介する．AdGholas と呼ばれる Malvertising キャンペーンの事例 [9] では，ゼロデイの脆弱性が利用されている．また，広告事業者のドメインによるフィルタリングを巧妙に回避する Malvertising がいくつも報告されている．たとえば，期限切れになった正規の広告事業者のドメインが取得され，Malvertising に利用された事例 [4] や，正規の広告事業者ドメインのサブドメインが不正に取得され，Malvertising に利用された事例 [10] がある．加えて，Malvertising において HTTPS 通信が利用されているといった報告もある [5]．

Cyphort の Malvertising に関する調査報告 [7] では，RTB のような多数の広告事業者が広告配信にかかわる環境において Malvertising が発生していることを示している．また文献 [7] では，i) 広告を見た数日後に悪意のあるペイロードが発生する，ii) 広告アクセスの 10 回や 20 回に 1 度だけエクスプロイットが実施される，iii) Malvertising に SSL が使われる，iv) 攻撃時にユーザエージェントや IP アドレスを検証する等，多岐にわたる巧妙な手法がとられていると報告している．

最後に既存研究における Malvertising の調査を紹介する．Li ら [12] は，Alexa Top 90,000 サイトを 3 カ月にわたり観測し，ドメインの役割（媒体または広告），ドメインの登録期限，URL パターン，ドメインの出現頻度，ドメインペアの出現頻度，主要な広告ネットワークからの距離等で悪性の広告について調査を行っている．Zarras ら [8] は，アンチウィルス会社のデータや Alexa のクロールデータから 60 万以上の広告を調査し，VirusTotal 等のサービスを利用して悪性かそうでないかの判定を行っている．結果として約 1%が悪性な広告であったとしている．また，Sakib ら [13] は，広告だけでなく広告のランディングページまで自動収集するアプリケーションを開発し，ランディングページから送られるバイナリのマルウェアを調査している．

2.2 Malvertising の対策

Malvertising の対策としては，広告を利用するユーザまたは媒体側の対策と，広告を配信する広告事業者側の対策に分けられる．それぞれについて，現状の対策を事例や既存研究から紹介し，課題を言及する．

2.2.1 ユーザまたは媒体側の対策

ユーザ側の対策として，Malvertising の事例 [3] でウィルス対策ソフトを導入すること，また，Cyphort の報告 [7] では，ユーザが端末の状態を最新に保つことが強調されている．

既存研究では，下記のようなユーザ側の対策が提案されている．Li ら [12] は *MadTracer* というツールを開発し，ドメインの出現頻度，役割（媒体または広告等），ドメイン期限，URL（.co や .cc 等）でアノテーションを行い，ドメインをノードとして，3 ノードのリダイレクトパスを特徴量とした学習から Malvertising の判別を行っている．結果として誤検知率は 5%程度と報告している．Zarras ら [8] は，ユーザ側の対策としては広告ブロッキングツール（Adblock Plus 等）が有力であると提言している．Arshad ら [14] は，オリジナルのコンテンツだけでなく，ブラウザ拡張機能や ISP による広告挿入も動的に検知し，ブロックするツールを実装している．プロトタイプのオーバヘッドは 12%程度としている．Dong ら [15] は，信頼できない JavaScript をサンドボックスで実行し，オリジナルの実行環境とはポリシで定めた内容のみをやり取りできるようにしている．プロトタイプのオーバヘッドは 4%程度としている．最後に媒体側の対策として，Ter ら [16] は，媒体サイトが広告に対して厳しいポリシを規定して広告を制御する方法を提案している．

ユーザ側の対策において，ウィルス対策ソフトや広告ブロッキングツールを導入するもの [3], [8] は，ゼロデイの脆弱性への対応が難しい．また Malvertising の挙動でブロックするもの [12], [14] は，正当な広告配信が誤って Malvertising と判定される可能性が存在する．ポリシを定めてサードパーティのスクリプト実行を制御するもの [15], [16] は，ポリシの設定コストが大きい．さらに，ユーザ自身でソフトウェアの導入や設定が必要な場合，ユーザビリティに関する課題も生じる可能性がある．

2.2.2 広告事業者側の対策

広告事業者側の対策として，Cyphort の報告 [7] では，広告事業者が広告配信サーバのセキュリティや入稿される広告コンテンツの検証を強化することをあげている．また，既存研究においては，Zarras ら [8] が不正な連携先への転送のフィルタリングをあげている．

さらに，いくつかの広告事業者はアライアンスを立ち上げて，Malvertising 対策の啓蒙を行っている．Online Trust Alliance（OTA）では，Malvertising 対策の具体的なガイドラインを示している [17]．また，TrustInAds.org では，脅威情報の共有や，ユーザからの通報を促している [18]．

フィルタリングに関しては，2.1 節で示したようにドメインによるフィルタリングを巧妙に回避する事例が多く報告されている．また，広告事業者個別のセキュリティ強化やガイドライン等による啓蒙では，広告事業者によって対応にばらつきが生じる可能性があり，広告配信全体としてセキュリティを高めることが難しい．Malvertising では広告配信にかかわる事業者のサーバが 1 つでも脆弱であれば，大規模な被害が発生している．

2.3 我々のモチベーション

事例から Malvertising の攻撃は，ユーザ端末への広告配

信中のレスポンス（次の宛先 URL や広告コンテンツ）を，DbD サイトへ誘導可能な情報に変更することに起因するといえる．また，現状のユーザ側や広告事業者側の対策は，ゼロデイの脆弱性や巧妙にフィルタリングを回避する高度な Malvertising への対応が難しくなってきている背景がある．そこで我々は，ユーザ端末へのレスポンスの真正性を担保することが高度化・巧妙化する Malvertising に有効な対策になると考え，広告配信に電子署名のプロトコルを導入する対策を提案している [11]．広告配信中のすべてのレスポンスに電子署名が付与されるようなプロトコルとすることで，正当な広告配信のレスポンスかどうかユーザ端末での検証が可能となる．検証に失敗した場合には，広告配信の処理を中断するようにすれば，DbD サイトへの誘導の前に処理を停止できるため，マルウェアにゼロデイの脆弱性が利用された Malvertising であっても有効である．また，レスポンスにおいて宛先 URL 全体への署名とすることで，ドメインによるフィルタリングの回避策がとられたとしても対応が可能である．

提案方式は広告事業者側の対策に位置し，提案方式への準拠が必要となる．しかし，既存の広告事業者側の対策のように，広告事業者が個別にセキュリティ対策を行うよりも，提案方式へ対応すれば広告配信に関して統一的なセキュリティを確保できる．そのため，RTB のように複雑で多数の事業者が広告配信に関与する環境では効果的な対策であると考える．さらに，ユーザは自身の端末にソフトウェア等を導入することなく，従来の Web 閲覧と同様のユーザビリティで広告からの脅威が軽減される．次章では我々の提案方式の詳細を示す．

3. 手法

本章では，まず対象とする広告配信モデルと Malvertising の攻撃モデルを明確にする．次に我々の提案方式を説明し，攻撃モデルへの耐性を示す．

3.1 対象とするモデル

3.1.1 RTB による広告配信モデル

インターネットの広告配信は年々複雑になっており，様々な広告配信方式によって広告が配信されている．中でも近年主流となっている広告配信方式は RTB（Real Time Bidding）と呼ばれる方式である [1], [2]．本稿では RTB を広告配信モデルとして対象とし，RTB のような広告配信で提案方式を適用することを目指す．また，本稿では RTB にかかわる広告事業者の最小構成として，SSP（Supply Side Platform），DSP（Demand Side Platform），CDN（Contents Delivery Network）を想定する．

ニュースやブログサイト等を運営している媒体（Publisher）は，自身のサイトに広告タグを設置する際に，SSP の広告事業者に申請を行う．SSP は各媒体サイトの広告枠を取得・管理しており，広告枠の需要と供給の調整を行っている．また，インターネット広告を出したい広告主（Advertiser）は DSP の広告事業者に広告の入稿を行う．DSP は広告コンテンツとともに，広告の単価（表示単価やクリック単価），ターゲット層を管理しており，広告の需要と供給を調整している．DSP は一般に広告コンテンツの送信に CDN（Contents Delivery Network）を利用する．

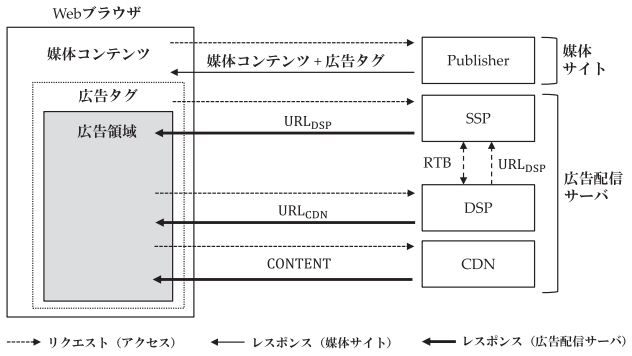

図 2 広告配信モデル
Fig. 2 The model of advertising.

図 2 は，RTB によって広告が配信される際のユーザ端末（Web ブラウザ）との通信を表している．図中の Publisher は媒体が保持している媒体サイトであり，SSP，DSP，CDN は各広告事業者が保持している広告配信サーバである．まず Web ブラウザが Publisher へアクセスすると，Publisher からは媒体のコンテンツと広告タグがレスポンスとして返される．広告タグには SSP への宛先が含まれ，iframe 等によって広告領域が作成されるとする．次に Web ブラウザが SSP にアクセスすると，SSP と DSP 間で RTB による自動的なオークションが開催され，配信する広告とその広告料がリアルタイムに決定される．たとえば，SSP は複数の DSP へ RTB のリクエストを送り，最高値を提示した DSP が RTB の勝者となる．そして，RTB に勝利した DSP は，DSP の宛先（URL_{DSP}）を SSP に返し，SSP からのレスポンスとして URL_{DSP} が Web ブラウザの広告領域に返される．さらに，広告領域において Web ブラウザが DSP にアクセスすると，広告コンテンツの宛先として CDN の宛先（URL_{CDN}）が返され，最後に CDN にアクセスすると広告画像等の広告コンテンツ（CONTENT）が返される．図 2 のように Web ブラウザは広告配信のレスポンスとして，URL_{DSP}，URL_{CDN}，CONTENT を各広告配信サーバから広告領域に受信している．本稿では，上記のような広告領域にレスポンスとして返される情報を対象とし，安全性を確保する提案を行う．

3.1.2 攻撃モデル

広告配信のレスポンスに対する Malvertising の攻撃モデルを定義する．2.1 節において，Malvertising の攻撃は広告配信中にレスポンスが DbD サイトへ誘導可能な情報に変更されることに起因すると述べた．この事象の詳細な原

因は事例や既存研究で明示的に述べられていないが，2つの場合があると想定される．1つは広告配信サーバが外部から不正侵入され，正規のレスポンスを改ざんされる事象，もう1つは広告事業者や広告主がそもそも悪意を持っており，故意にレスポンスをDbDサイトへ誘導可能な情報へ変更する事象である．本稿では攻撃モデルとして前者を扱うとし，後者は対象外とするが，後者の対策の展望は5.4節で言及する．

定義

攻撃者は，3.1.1項で示した$\mathrm{URL_{DSP}}$，$\mathrm{URL_{CDN}}$，CONTENTのうち，少なくとも1つを書き換えるとする．

攻撃者

ここで攻撃者は，広告事業者（SSP，DSP，CDN）に属する人物ではなく，広告配信サーバ等に不正に侵入できるとする．

攻撃者は図2で示したような広告配信サーバのレスポンス（$\mathrm{URL_{DSP}}$，$\mathrm{URL_{CDN}}$，CONTENT）を，攻撃者が掌握しているDbDサイト等へ転送可能な情報に改ざんすることにより，広告配信を通じて大規模なマルウェア配信攻撃が可能となる．ここで媒体や広告主は正当な事業者であり攻撃者と結託しない，さらに媒体サイトや入稿された時点での広告は安全なものであるとする．

3.2 提案方式

前節の攻撃モデルによる攻撃を防止するための具体的な広告配信方式（以降，本方式）を説明する．本方式では，3.1.1項で示した広告配信モデルにおいて，広告配信サーバからユーザ端末へ返すすべてのレスポンスを電子署名付のレスポンスとすることで，改ざんを検知できる仕組みを導入する．電子署名は公開鍵暗号方式による署名とし，信頼できる認証事業者が署名の発行を行うとする．署名フォーマットとしてはJWS（JSON Web Signature: RFC7515 [19]）形式を利用する．また，以降で説明する事前準備時と広告配信時において，本方式の安全性が保障できる前提条件を**表1**にまとめる．

3.2.1 事前準備：信頼点と暗号鍵

本方式では，公開鍵暗号による電子署名のプロトコルを導入する．そのため，3.1.1項で示した広告事業者に，新たに信頼点として認証事業者を加え，認証事業者では暗号鍵の生成を行っておく．

3.2.1.1 認証事業者TTPの設置

認証事業者としてTTP（Trusted Tag Provider）を設置する．TTPは正当な事業者であり，外部からの攻撃を成功させない．さらに自身が生成した暗号鍵等の秘密情報を適切に管理し，漏洩させないものとする．TTPは広告事業者の広告配信情報に署名を実施し，さらに署名検証ができる広告タグを提供して広告配信時にユーザの端末上で署名検証を行う．

表1 提案方式の前提
Table 1 The precondition of our protocol.

	事前準備時	広告配信時
Publisher	安全[1]	安全[1]
TTP	安全	安全
SSP	安全	安全でない[2]
DSP	安全	安全でない[2]
CDN	安全	安全でない[2]
広告主	安全[3]	安全[4]
ユーザ端末	-	安全[5]
Webブラウザ	-	安全[5]
各通信路	安全[6]	安全[6]

[1] 媒体サイトのコンテンツは安全とする．
[2] 広告配信サーバは攻撃者に侵入されているとする．
[3] 入稿される広告は安全とする．
[4] 広告主サイト（ランディングページ）は安全とする．
[5] 悪意のあるソフトウェアおよび拡張機能が導入されていないとする．
[6] 通信路はSSL/TLSで暗号化されているとする．

3.2.1.2 暗号鍵の生成

TTPは署名と署名検証のための公開鍵暗号の鍵ペアを生成する．

$$\texttt{GenerateKey}() \rightarrow (\mathrm{PK}, \mathrm{SK})$$

ここで，GenerateKey()は任意の暗号方式による公開鍵暗号の鍵ペア生成機能であり，PKは署名検証用公開鍵，SKは署名用秘密鍵である．

3.2.2 事前準備：TTPによる署名と広告タグの発行

次に，TTPは広告配信におけるWebブラウザへのレスポンスに対して，事前に署名を発行する．さらに，Webブラウザ上でTTPの権限の下，署名検証が可能となる広告タグを発行しておく．その手順を下記に示す．

3.2.2.1 署名の発行

図3(a)にTTPによる署名の発行を示す．DSPは，図2で示したWebブラウザへのレスポンス（$\mathrm{URL_{DSP}}$，$\mathrm{URL_{CDN}}$，CONTENT）について，TTPへ署名を要求する．TTPは署名要求に対してSKを用いて署名を実施し，署名情報を返す．

$$\texttt{Sign}_{\mathrm{JWS}}(\mathrm{SK}, M) \rightarrow \alpha$$

本方式では，電子署名にJWS [19]形式を利用する．ここで，$\texttt{Sign}_{\mathrm{JWS}}()$はJWS形式の署名情報$\alpha$を出力する機能である．入力SKはGenerateKey()で生成した署名用秘密鍵である．入力Mは，$\{\mathrm{URL_{DSP}}, \mathrm{URL_{CDN}}, \mathrm{CONTENT}\}$のいずれかが想定される．**図4**では，JWSによる$\mathrm{URL_{DSP}}$に対する署名情報$\alpha_{\mathrm{URL_{DSP}}}$の発行例を示す．まずJWS Headerは TTPによって定義される．“alg”はGenerateKey()で利用した暗号方式に該当し，かつJWA（JSON Web Algorithms: RFC7518 [20]）で規定されている公開鍵暗号の署

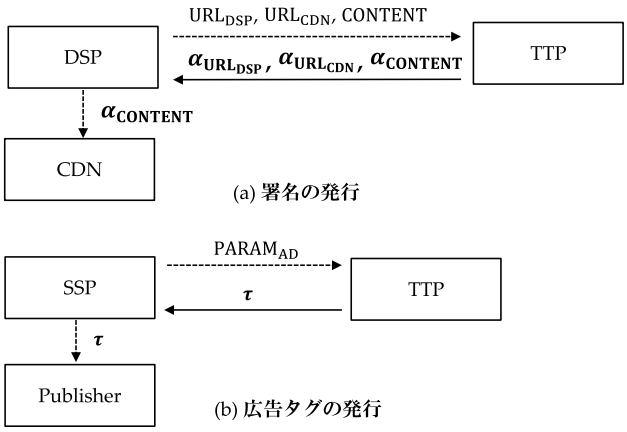

図 3 TTP による署名と広告タグの発行
Fig. 3 Providing signature and tag by TTP.

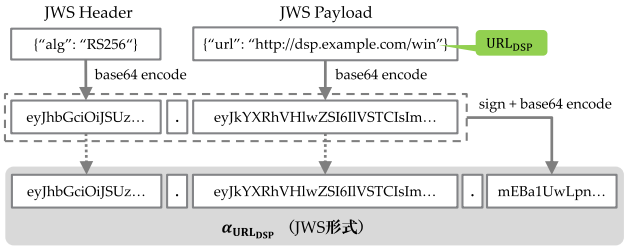

図 4 JWS 形式の署名情報 α
Fig. 4 Signing with JWS.

名方式を設定する．JWS Payload には，ここでは例として $\mathrm{URL_{DSP}}$ が JSON 形式で設定されているとする．JWS では，Base64 エンコードされた JWS Header と JWS Payload を "." ピリオドで連結し，連結した文字列に JWS Header で指定された署名方式と SK を用いて署名し，それを Base64 エンコードしたものをさらに "." ピリオドで連結することで，JWS の文字列（ここでは署名情報 $\alpha_{\mathrm{URL_{DSP}}}$）を作成する．

また，図 3 (a) について，CONTENT に対する署名情報 $\alpha_{\mathrm{CONTENT}}$ に関しては，事前に DSP から CDN に渡され，広告配信が行われるとする．さらに，本方式では画像であっても JSON 形式を利用するため，図 4 の JWS Payload が画像の場合は，JSON の name に "img"，value に "Base64 化された画像データ" 等を設定する等の対応が必要となる．JWS Payload の構成や拡張性については 5.3 節で議論する．

3.2.2.2 広告タグの発行

広告タグは一般に SSP から Publisher へ直接発行され，広告タグによる広告領域（広告が表示される部分）の生成方法は特に定義されていない．本方式では広告領域において Same-Origin Policy [21] を利用し，TTP オリジンを Web ブラウザでの信頼できる領域として署名情報 α の検証を行うことで安全性を確保する．そのため，本方式では TTP が広告タグを間接的に発行し，広告領域としては TTP オリジンの iframe を生成する広告タグを想定する．

図 5 広告タグ τ の概観と機能
Fig. 5 The overview and function of τ.

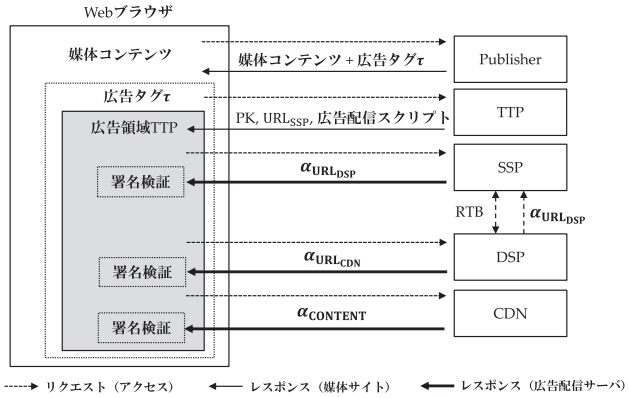

図 6 提案方式による広告配信
Fig. 6 The advertising with proposal method.

図 3 (b) に TTP による広告タグの発行を示す．$\mathrm{PARAM_{AD}}$ は下記のように，広告領域の幅 $\mathrm{W_{AD}}$，高さ $\mathrm{H_{AD}}$，SSP の宛先 $\mathrm{URL_{SSP}}$ を含んでいるとする．

$$\mathrm{PARAM_{AD}} \leftarrow \{\mathrm{W_{AD}}, \mathrm{H_{AD}}, \mathrm{URL_{SSP}}\}$$

TTP が $\mathrm{PARAM_{AD}}$ を受け取ると，$\mathrm{PARAM_{AD}}$ に対応する識別子 $\mathrm{ID_{AD}}$ を生成する．

$$\texttt{GenerateId}(\mathrm{PARAM_{AD}}) \rightarrow \mathrm{ID_{AD}}$$

TTP は広告タグ τ として TTP の iframe 生成機能である `GenerateTtpIframe()` を SSP へ返す．

$$\tau \leftarrow \texttt{GenerateTtpIframe}(\mathrm{W_{AD}}, \mathrm{H_{AD}}, \mathrm{ID_{AD}})$$

SSP は発行された τ を Publisher に提供する．

図 5 に Web ブラウザで広告タグ τ（`GenerateTtpIframe()`）が広告領域を生成する様子を示す．JavaScript が `GenerateTtpIframe()` を実行すると，図 5 (a) のような iframe タグで広告領域 TTP を作成する．広告領域 TTP が TTP にアクセスすると，図 5 (b) のようなレスポンスが返される．具体的な処理に関しては次項で説明する．

3.2.3 広告配信と署名検証

本方式における広告配信全体と署名検証の流れを**図 6** に示す．Web ブラウザは媒体サイト（Publisher）にアクセスすると，媒体サイトのコンテンツと広告タグ τ がレスポン

Algorithm 1 Advertising script

```
 1: pk ← PK
 2: url ← URL_SSP
 3: procedure ADVERTISING(pk, url)
 4:     α ← XMLHttpRequest(url)
 5:     if Verify_JWS(pk, α) = true then
 6:         for all name, value ∈ α[payload] do
 7:             if name = "url" then
 8:                 url ← value
 9:                 return ADVERTISING(pk, url) ▷ Access the url
    of this value.
10:             end if
11:             if name = "img" then
12:                 return value ▷ Show the image of this value.
13:             end if
14:         end for
15:     else
16:         return Exception
17:     end if
18: end procedure
```

スとして返される．広告タグ τ を実行し，iframe を用いて TTP オリジンの広告領域 TTP を生成する．TTP へアクセスすると，広告領域 TTP に署名検証用公開鍵 PK，SSP の宛先 $\mathrm{URL_{SSP}}$，広告配信スクリプトがレスポンスとして返される（図 5 (b) の部分）．広告領域 TTP において，広告配信スクリプトが XML HTTP Request（XHR）で SSP へアクセスすると，SSP は所定の複数の DSP 間で RTB を実施する．RTB に勝利した DSP は署名付の自身の宛先である署名情報 $\alpha_{\mathrm{URL_{DSP}}}$ を SSP に返し，SSP は $\alpha_{\mathrm{URL_{DSP}}}$ を広告配信スクリプトへ返す．広告配信スクリプトは $\alpha_{\mathrm{URL_{DSP}}}$ を検証し，改ざんがないことを確認すると，そこに含まれる DSP の宛先 URL にアクセスする．DSP は広告コンテンツが設置された CDN の宛先 URL に対する署名情報 $\alpha_{\mathrm{URL_{CDN}}}$ を返し，広告配信スクリプトは $\alpha_{\mathrm{URL_{CDN}}}$ を検証して改ざんがなければ内部の URL にアクセスする．最終的に CDN から送られる署名情報 $\alpha_{\mathrm{CONTENT}}$ が改ざんされていなければ，広告コンテンツを表示する．具体的な広告配信スクリプトの XHR 通信と署名検証を Algorithm 1 に示す．

Algorithm 1 において，$\mathtt{Verify}_{\mathrm{JWS}}()$ は，広告配信スクリプトにおける署名検証用公開鍵 PK を用いた署名情報 α（JWS 形式）の署名検証機能であり，成功または失敗を出力する．

$$\mathtt{Verify}_{\mathrm{JWS}}(\mathrm{PK}, \alpha) \rightarrow \mathrm{true/false}$$

広告配信スクリプトは α の検証に成功すると，α の JWS Payload から広告配信情報を取り出す．本項では説明上，name が “url” の場合は次の宛先 URL，“img” の場合は Base64 エンコードされた広告コンテンツとする．name が “url” であれば，次の宛先 URL を取得し，処理を繰り返す．name が “img” であれば，広告コンテンツを表示し，終了する．ここで α の検証に失敗したならば，例外を返し処理を中断する．

なお，署名情報 α のレスポンスを返す際に，SSP や DSP は TTP に対して Cross-Origin Resource Sharing（CORS）を許可しなければならないが，本方式において SSP や DSP は事前に TTP から広告タグや署名を発行されているため，その時点で TTP を信頼しているとする．

3.3 安全性に関して

本方式では，広告領域 TTP へのすべてのレスポンスについて署名検証を行い，その過程で 1 回でも署名検証に失敗した場合，処理を中断する．3.1.2 項で示した攻撃モデルでは，$\mathrm{URL_{DSP}}$，$\mathrm{URL_{CDN}}$，CONTENT の少なくとも 1 つが書き換えられるため，署名検証が失敗する．本方式では，$\mathrm{URL_{DSP}}$，$\mathrm{URL_{CDN}}$，CONTENT の少なくとも 1 回でも署名検証に失敗した場合，処理が中断される．よって，提案方式は攻撃モデルに対して安全であり，ユーザ端末が DbD サイトへ誘導される前に，未然にマルウェア感染等の脅威から防ぐことができるといえる．

3.3.1 署名検証用公開鍵 PK の安全性

本方式における広告配信時の署名検証用公開鍵 PK の安全性について言及する．本方式では具体的に下記の 2 点で PK の安全性を確保している．

(1) Same-Origin Policy の利用

(2) TTP の広告配信スクリプトによる署名情報 α の処理

ここで，広告配信時に PK が媒体コンテンツや署名情報 α によって編集可能な場合，PK が TTP の鍵であるということを保証できない．しかし，上記 (1) によって，媒体コンテンツ側からは広告領域 TTP（iframe 内）にアクセスできず，また (2) によって，TTP の広告配信スクリプトが署名情報 α を取得すると，まず α の署名検証を行う（Algorithm 1）．この時点で広告配信サーバから返された何らかの文字列 α が JWS 形式でない，または署名検証できなければ処理を中断する．そのため，α が PK を編集するような任意の攻撃コードであっても実行されない．よって，広告配信時において TTP は安全であるという前提かつ，媒体コンテンツや署名情報 α から PK の編集が不可能なため，PK は TTP が発行した鍵であると保証できる．

なお，公開鍵の発行者を保証する仕組みとして，Web ブラウザへあらかじめ安全に公開鍵を配布する仕組みである Web の公開鍵基盤（PKI）が存在する．しかしながら，広告配信におけるレスポンスの安全性を確保するために Web の PKI を利用するには，下記の課題がある．

- 広告タグで OS・ブラウザの証明書リストを利用する場合，JavaScript によって証明書リストにアクセスする必要があるが，現状その手段がないため実装が不可能である．
- 証明書リストを利用するためには，ネイティブなソフ

トウェアを開発する必要があり，ユーザはそのソフトウェアを導入しなければならない．

そこで本方式では Web の PKI は利用せず，(1) と (2) によって PK の安全性を保障している．本方式では広告配信時に TTP との通信を必要とするが，TTP を信頼点とした署名の検証を実現することによって，上記課題 2 点を回避している．

3.3.2 安全性を保障する範囲

本方式では，媒体サイトが安全な場合であっても，RTB のような広告配信が悪用されることによって，媒体サイトの広告領域から攻撃者の DbD サイトへ自動的に誘導される点に注目し，その脅威に対応する手法を提案している．本方式で安全性を保障する範囲および対象外は下記のとおりである．

- 安全性を保障する範囲
  - 広告配信サーバ（SSP，DSP，CDN）からのレスポンス
- 対象外
  - 媒体サイトのコンテンツ
  - 広告をクリックした際のランディングページ
  - ユーザ端末および Web ブラウザ

媒体サイトが改ざんされるような攻撃は，Malvertising の有無にかかわらずユーザは脅威にさらされるため，本方式では対象としていない．また，ランディングページ（広告主の Web サイト）も媒体サイト同様，RTB を悪用した Malvertising に直接かかわらないため，対象としていない．このような Web サイトが改ざんされる攻撃に対しては，Web 改ざん検知サービス（たとえば文献 [22], [23]）がすでにいくつか提供されている．媒体サイトやランディングページに関しては，前述のような製品が導入されることを期待する．

4. 実装と評価

本章では提案方式のプロトタイプを実装し，オーバヘッド評価を行った結果を示す．

4.1 実装

提案方式のプロトタイプをプライベートなネットワーク環境に位置する仮想マシン環境を用いて実装した．**表 2** に全体の構成を示し，**表 3** に各サーバの構成を示す．

提案方式の構成は図 6 のように 5 台のサーバとした．また，次節で説明する評価のため，比較方式として RTB における広告配信モデル（図 2）も TTP を除く 4 台のサーバで構成した．媒体サイト（Publisher）はニュースサイトを模したデモサイトに 1 つの広告枠が設置される構成とし，Web ブラウザで表示可能となっている．TTP，SSP，DSP，CDN の各機能は，HTTP GET リクエストによってレスポンスを返す REST API として PHP により実装した．SSP，DSP からのレスポンスは次の宛先 URL，CDN からのレスポンスは幅 300 px，高さ 250 px の広告画像とした．JWS 形式の署名の発行（3.2.2.1）とユーザ端末での署名検証（3.2.2.2，3.2.3 項）の実装は，文献 [24] で公開されているライブラリを利用した．署名の発行には pyjwt を，署名検証には jsrsasign を利用した．

表 2 システム構成と設定
Table 2 The system configuration.

サーバ名	台数（提案）[*1]	台数（比較）[*2]	delay	loss
Publisher	1	1	10 ms	0.1%
TTP	1	-	10 ms	0.1%
SSP	1	1	100 ms [*3]	0.1%
DSP	1	1	10 ms	0.1%
CDN	1	1	10 ms	0.1%

[*1] 提案方式（図 6）における構成
[*2] RTB の広告配信モデル（図 2）における構成
[*3] RTB の実施を考慮した遅延時間

表 3 各サーバの構成
Table 3 A setting of each server.

項目	設定
CPU	Intel Xeon 2.40 GHz（2 コア割当）
メモリ	1,024 MB
SSD	10 GB
OS	Ubuntu 16.04 LTS
WebIO	Apache 2.4.18, PHP 7.0.8

表 4 ユーザ端末（測定用）
Table 4 A client machine.

項目	設定
CPU	Intel Core i7 2.40 GHz
メモリ	8 GB
SSD	248 GB
OS	Windows 8.1 64 ビット
Web ブラウザ	Google Chrome 54（バージョン）
通信	無線 LAN（802.11ac）

また，実際のインターネット環境の挙動に近づける目的から，文献 [25], [26] を参考に遅延（表 2 の delay）とパケットロス（表 2 の loss）を各サーバに設定している．SSP の遅延に関しては，SSP-DSP 間の RTB の処理時間が 100 ms 以下で想定されていること [27], [28] から，100 ms とした．なお，SSP-DSP 間で行われる RTB の実装に関しては，提案方式の評価に直接関係しないためプロトタイプでは実装していない．

4.2 評価方法

前節の実装をもとに，提案方式（図 6）と比較方式（図 2）のオーバヘッドを測定し評価した．測定に使用したユーザ端末は**表 4** のとおりである．提案方式と比較方式の違いは，署名検証の有無であり，評価実験の実装に関して下記の違いが存在する．

表 5 データ量の比較
Table 5 Data size.

	提案方式（byte）[*1]	比較方式（byte）[*2]	差分（byte）
SSP	307	64 [*3]	243
DSP	297	57 [*3]	240
CDN	5,783	4,874 [*4]	909

[*1] JWS 形式
[*2] JWS Payload 部分
[*3] URL の文字列
[*4] Base64 化された画像（300 px，250 px，元サイズ 3,633 バイト）

(1) 提案方式（署名検証あり）
  (a) TTP と通信し，TTP は図 5 (b) のレスポンスを返す．
  (b) SSP，DSP，CDN はレスポンスとして JWS 形式の署名情報 α を返す．
  (c) ユーザ端末は α を検証し，検証が成功すれば，JWS Payload の処理を実行する．
(2) 比較方式（署名検証なし）
  (a) TTP との通信はない．
  (b) SSP，DSP，CDN はレスポンスとして JSON 形式の情報（署名情報 α の JWS Payload に相当する部分）を返す[*1]．
  (c) JSON 形式の値を解析し，そのまま実行する[*1]．

提案方式のオーバヘッド評価として，各サーバとの通信の比較，および Web ブラウザにおける署名検証の有無にかかわる処理の比較に分割できる．通信の比較を「ロード時間」，署名検証処理の比較を「処理時間」として，具体的に下記の評価を行った．

（**比較 1**）ロード時間
- 前記 (1)-(a)，(b) と (2)-(a)，(b) における通信時間を比較する．
- TTP へは，リクエスト開始からレスポンス受信終了までの時間とする．
- SSP，DSP，CDN へは，XHR によるリクエスト開始からレスポンス受信終了までの時間とする．

（**比較 2**）処理時間
- 前記 (1)-(c) と (2)-(c) における処理時間を比較する．
- SSP，DSP，CDN のレスポンス受信終了からレスポンス内容の処理が完了するまで時間とする．

測定はそれぞれ 100 回行い，平均値による比較を行った．比較 1 のロード時間の結果は 4.3.1 項で示し，比較 2 の処理時間の結果は 4.3.2 項で示す．また，比較 1 と比較 2 の総合結果を「総時間」として 4.3.3 項で示す．

4.3 評価結果

本節では前節の評価結果を示す．**表 5** および**図 7**，**図 8**，

[*1] 広告配信において一般にレスポンスの形式は規定されていないが，本稿では提案方式との比較のため JSON 形式とする．

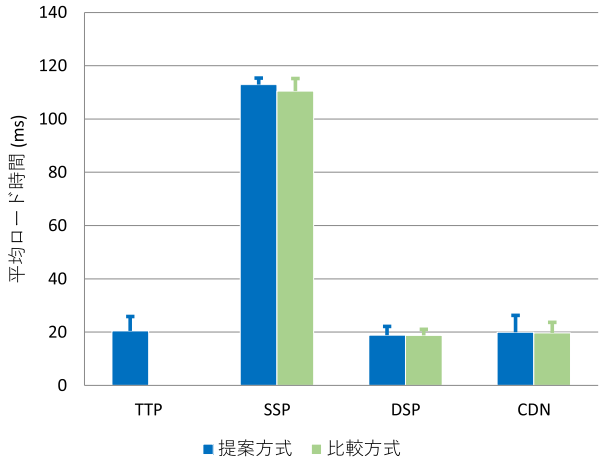

図 7 平均ロード時間
Fig. 7 Means of loading time.

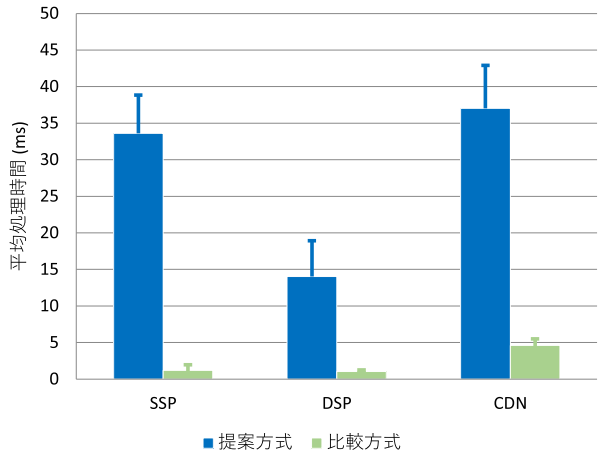

図 8 平均処理時間
Fig. 8 Means of processing time.

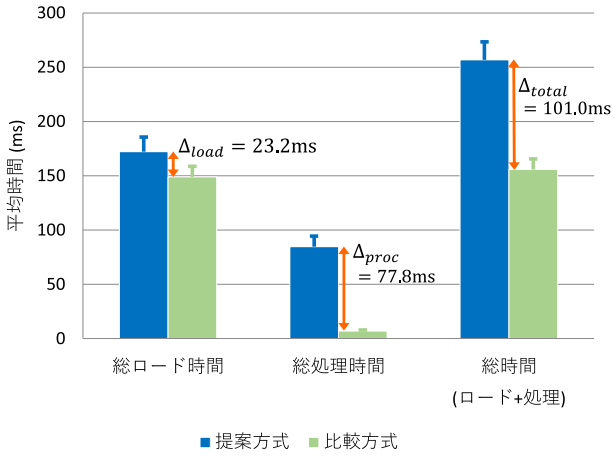

図 9 平均総時間
Fig. 9 Means of total time.

図 9 に関して，“提案方式” は 4.2 節 (1) の提案方式（署名検証あり）であり，“比較方式” は 4.2 節 (2) の比較方式（署名検証なし）である．図中，グラフ上のバーは標準偏差 SD である．また図 7，図 8 に関して，“提案方式” の場合，SSP，DSP，CDN はそれぞれ $\alpha_{\mathrm{URL_{DSP}}}$，$\alpha_{\mathrm{URL_{CDN}}}$，$\alpha_{\mathrm{CONTENT}}$ のロードまたは処理であり，“比較方式” の場合，SSP，DSP，

CDN はそれぞれ $\alpha_{\mathrm{URL_{DSP}}}$，$\alpha_{\mathrm{URL_{CDN}}}$，$\alpha_{\mathrm{CONTENT}}$ における JWS Payload 部分のみのロードまたは処理である．

4.3.1 ロード時間

4.2 節の比較 1 として，SSP，DSP，CDN のそれぞれにおいて，提案方式と比較方式の平均ロード時間を比較した結果を図 7 に示す．比較方式は TTP との通信が存在しないため，TTP のロード時間は提案方式のみである．また，SSP，DSP，CDN のレスポンスに関する提案方式と比較方式のデータ量を表 5 に示す．

表 5 について，提案方式は JWS 形式であり，比較方式の JWS Payload のみと比べデータ量は大きくなっている．JWS 形式は，JWS Payload が Base64 化され，JWS Payload のデータ量に依存しないほぼ固定長の JWS Header と署名値が加わっている（図 4）．

しかし，図 7 について，SSP，DSP，CDN の提案方式と比較方式のロード時間に大きな差は見られなかった．CDN からのレスポンスは画像であるが，ロード時間が大きくなっていない．なお，TTP からのレスポンス（図 5 (b)）のデータ量は 287,409 byte である．

4.3.2 処理時間

4.2 節の比較 2 として，SSP，DSP，CDN のそれぞれにおいて，提案方式と比較方式の平均処理時間を比較した結果を図 8 に示す．処理時間に関しては提案方式と比較方式において，差が観測された．比較方式の JSON 形式の処理ではほぼ実行時間がかからないが，提案方式の署名検証を行い，処理をする内容であると，処理時間の増加が見られる．CDN のレスポンス（広告画像のレスポンス）では，処理時間が大きくなっている．これはレスポンスのサイズが大きいためと考えられる．また，提案方式において SSP と DSP のレスポンス（URL）はほぼ同じ内容だが，処理時間は SSP の方が大きくなっている．これは SSP の最初の処理において，JWS ライブラリを準備する時間がかかっているためと考えられる．

4.3.3 総時間

図 7 の TTP から CDN までの平均ロード時間の総計（総ロード時間），図 8 の SSP から CDN までの平均処理時間の総計（総処理時間），さらにロード時間と処理時間の総計を図 9 に示す．ロード時間の総計として，提案方式は平均 172.1 ms（$SD = 13.53$ ms）であり，比較方式では平均 148.9 ms（$SD = 7.58$ ms）であった．処理時間の総計として，提案方式は平均 84.6 ms（$SD = 9.84$ ms）であり，比較方式では平均 6.8 ms（$SD = 1.19$ ms）であった．ロード時間と処理時間の全体として，提案方式は平均 256.7 ms（$SD = 16.33$ ms）であり，比較方式では平均 155.7 ms（$SD = 7.58$ ms）であった．

結果から，提案方式の通信に関するオーバヘッド Δ_{load} は 23.2 ms であった．これはほぼ TTP との通信に関するオーバヘッドであり，データ量や JWS 形式における増加はあまり影響がないといえる．また，署名検証の処理に関するオーバヘッド Δ_{proc} は，77.8 ms であった．署名検証がない比較方式ではほとんど処理時間はかからないが，署名検証がある提案方式では処理時間が大きくなる．最後に，通信と処理の両方のオーバヘッド Δ_{total} は 101.0 ms であった．

5. 考察および議論

本章では，提案方式と評価結果についての考察を行う．また，提案方式の利点，拡張性，運用について議論を行い，最後に本稿の制限事項と課題を述べる．

5.1 評価結果の考察

4.3 節では，提案方式と比較方式において，通信の比較，処理の比較を行い，提案方式のオーバヘッドを示した．広域ネットワークにおける実験との差異やユーザ端末の性能面も考えられるが，提案方式では主に，TTP との通信のオーバヘッドおよび署名検証の処理に関するオーバヘッド Δ_{proc} が強く関連するといえる．そのため，提案方式を実施する際は，TTP のレスポンス速度，Web ブラウザでの署名検証の速度に注意する必要があると考えられる．

また，本節では提案方式のオーバヘッドが実際の Web ブラウジングにおいて，どの程度の影響があるかを考察する．Web ブラウジングの対象とする Web サイトは，Alexa の日本のカテゴリ別ランキングにおけるニュースカテゴリランキング Top 10 サイト [29] とした（2016 年 10 月時点）．10 サイトの調査はすべてトップページ 1 画面とし，10 サイトすべてのトップページに広告が含まれていることを確認している．各サイトにおいてブラウザ拡張機能で JavaScript の挿入を行い，window.performance.timing 関数においてサイトリクエストから表示完了までの時間を計測した．計測は各サイトとも 100 回行った．**図 10** に各サイトの計測結果として平均表示時間を示す．結果として

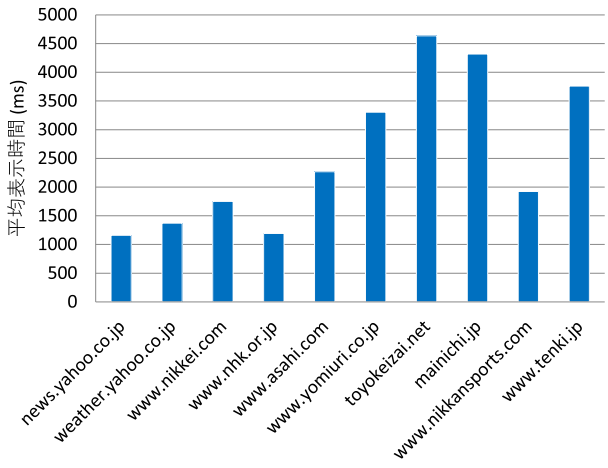

図 10 ニュースサイトの平均表示時間
Fig. 10 Means of loading time on news sites.

ニュースサイトでは最低でも 1,160.1 ms の平均表示時間がかかっており，最大では 4,635.6 ms の平均表示時間であった．10 サイトの平均では 2,568.8 ms の表示時間であり，$\Delta_{total} = 101.0$ ms と比較すると，提案方式のオーバヘッドは約 4%と考えられる．

なお，ブラウザ拡張機能による対策 [14], [15] では Web ブラウザでの処理のオーバヘッドが示されている（2.2.1 項）．提案方式では $\Delta_{proc} = 77.8$ ms であり，上記平均表示時間 2,568.8 ms と比較すると，Web ブラウザでの処理部分では約 3%といえる．この値は参考値ではあるが，文献 [14], [15] よりも小さく抑えられている．

5.2 提案方式の利点

2.2.2 項で述べたように，既存の広告事業者側の対策はそれぞれの広告事業者が個別にセキュリティ対策を行うものである．一方で，我々の提案方式は広告配信のプロトコル自体を Malvertising に耐性があるものにする．提案方式において，広告事業者は広告配信のために新たなプロトコルへの対応が必要になるが，プロトコルに準拠すれば Malvertising 対策が施される．現状の対策では，広告事業者によってセキュリティ対策にばらつきが生じる可能性があるが，提案方式では広告配信に関して統一的なセキュリティを確保できるといえる．

また，2.2.1 項で述べたように，既存のユーザ側の対策はユーザ自身でソフトウェアの導入や設定が必要なものであった．提案方式は広告事業者側の対策であるため，ユーザは通常の Web サイト閲覧と同様のユーザビリティで，広告配信におけるセキュリティが確保される．また，媒体側も通常の広告掲載のように広告タグを自身のサイトに設置すれば，広告配信に関してセキュリティを確保できる．

5.3 提案方式の拡張性

提案方式において，JWS Payload は JSON 形式で記述し，広告配信スクリプトでは JWS Payload を解析して処理しなければならない．そのため，JWS Payload の構成については認証事業者（TTP）および広告事業者（SSP，DSP，CDN）で規定する必要がある．JWS Payload は RTB の様々な処理に対応するように標準化等によって規定できると考える．たとえば，RTB は OpenRTB [30] と呼ばれる標準仕様が規定されている．また，JWT（JSON Web Token: RFC7519 [31]）では JWS Payload のスキーマが規定されている．このように，提案方式についても JWS Payload の内容を規定すれば，様々な広告配信の要求に対応が可能となると考えられる．

また，提案方式では，広告配信のレスポンスである次の宛先 URL や広告コンテンツは事前に固定されているものとしたが，URL や広告コンテンツが動的に変更する場合には，リアルタイムな署名の発行が求められると考えられる．攻撃モデルでは，広告配信時に広告配信サーバが安全でないという前提のため，TTP は署名の要求（$\mathrm{URL_{DSP}}$，$\mathrm{URL_{CDN}}$，CONTENT）に対して，悪意のあるコードが含まれていないかをリアルタイムに判定する必要がある．以上の点を解決できれば，TTP から署名情報 α をリアルタイムに取得しつつ，広告配信を行うことが可能となると考えられる．

5.4 提案方式の運用に向けて

提案方式は，3.1.2 項で示した攻撃モデルに耐性がある．この攻撃モデルでは，攻撃者が広告事業者や広告主でないことを想定した．しかしたとえば，悪意のある広告事業者が正当な広告事業者を偽ってマルウェア配信を行うことや，悪意のある広告主がマルウェアを正当な広告と偽って入稿することも想定される．攻撃者が広告事業者や広告主である場合は本稿では対象外としたが，提案方式が厳格な運用のフレームワークとともに実施されれば，対応が可能であると考えられる．

運用のフレームワークとして代表的なものにトラストフレームワークがあり，ID 連携等で実施されている．ID 連携では OpenID connect 等の認証・認可のプロトコルを用いた Open Identity Trust Framework（OITF）[32] と呼ばれるモデルが知られている．OITF モデルでは，TFP（Trust Framework Provider）と呼ばれる機関を設置し，IdP（Identity Provider）および SP（Service Provider）が遵守すべき技術・運用要件等を規定する．TFP が従う上位の制度や規格を策定する人または組織を Policy Maker と呼ぶ．また，TFP は IdP および SP が規定する各種規準を継続的に遵守していることを確認するため，査定人/監査人（Assessor/Auditor）を認定し，査定や監査を行う．

ここで，本稿の提案方式を技術要素としたトラストフレームワークを想定し，攻撃者が広告事業者または広告主の場合の対策を考察する．攻撃者が広告事業者の場合は，監査人を利用して広告事業者の身元確認，運用実績を厳しく監査することや，内部不正が発生しにくい技術・運用規準を規定することが必要とされる．また攻撃者が広告主の場合は，DSP の運用規定として，i) 広告主の身元確認レベル（保証レベル）を規定，ii) 保証レベルを遵守しているかどうか継続的に査定，iii) 広告主からの広告入稿に対する確認規準を規定（入稿時のマルウェア検知等）することが必要とされる．

上記のように，悪意のある広告事業者または広告主が，新たにトラストフレームワークに加入すること，トラストフレームワーク内で広告事業者または広告主の悪意ある担当者が活動することを抑制して，提案方式を運用していくことで，既存の広告事業者側の運用における対策よりも，Malvertising に高い耐性を維持することが可能となると期待される．

5.5 制限事項と課題

提案方式の実装はプライベートな環境で行われ，PC の Web ブラウザによる評価のみである．また，署名の発行や署名検証においてライブラリ [24] を利用しているため，本稿の実装ではライブラリの安全性や処理性能に依存している．実環境への実装，ライブラリの評価，スマートフォン等の様々な端末での評価は今後の課題である．

また広告配信モデルである RTB では，Cookie 等によるユーザのトラッキングと広告事業者間での Cookie の共有を広告配信に含めている場合がある [33]．トラッキングに関しては本稿の対象外としているが，5.3 節で述べたように，JWS Payload で規定することで対応が可能と考えられる．

提案方式は広告配信に電子署名のプロトコルを導入するものであるが，既存の広告配信から提案方式への対応コストは本稿では議論していない．また広告配信については安全性が確保できることを示したが，媒体サイトやランディングページ，さらにブラウザ拡張機能からの脅威等は本稿では対象としていない．なお広告配信サーバからのレスポンスにおいて，ブラウザの JavaScript エンジンの脆弱性を突く攻撃がされる場合は，提案方式の脆弱性とはいえないため対象としていない．

最後に，本稿において想定した Malvertising に対しては，提案方式を適用することにより，ユーザに特別な対策が必要ないことを強調したが，ユーザのセキュリティ対策全般としてはウィルス対策ソフト導入や端末の状態を最新に保つことは重要であるという点を付言しておく．

6. まとめ

本稿では，複雑化・巧妙化する Malvertising に対して，広告事業者が広告配信に電子署名のプロトコルを導入する対策を示し，実装と評価を行った．我々の提案方式は，広告配信中のレスポンスが変更される Malvertising に対して安全性を提供するものであり，評価の結果から小さいオーバヘッドで実施が可能であることを示した．提案方式は，既存の広告事業者側の対策よりも，広告配信に関して統一的なセキュリティを確保でき，ユーザは新たに特別なソフトウェアを導入する必要なく，通常のブラウジングと同様のユーザビリティで Malvertising の脅威を軽減できる．また，提案方式は広告配信の要求に対して拡張性があり，運用のフレームワーク（トラストフレームワーク）と組み合わせることで，より強固な広告事業者側の対策が実施できる可能性を示した．今後の展望としては，提案方式の一部の標準化や実際の広告ネットワーク上での稼働，提案方式を運用していくためのトラストフレームワークの詳細な設計があげられる．

参考文献

[1] Yuan, S., Wang, J. and Zhao, X.: Real-time bidding for online advertising: Measurement and analysis, *Proc. 7th International Workshop on Data Mining for Online Advertising*, p.3, ACM (2013).

[2] Interactive Advertising Bureau: IAB Programmatic Revenue Report 2014 Results, available from ⟨http://www.iab.com/wp-content/uploads/2015/07/PwC_IAB_Programmatic_Study.pdf⟩ (accessed 2017-03-31).

[3] TrendLabs: Massive Malvertising Campaign in US Leads to Angler Exploit Kit/BEDEP (2016), available from ⟨http://blog.trendmicro.com/trendlabs-security-intelligence/malvertising-campaign-in-us-leads-to-angler-exploit-kitbedep/⟩ (accessed 2017-03-31).

[4] Trustwave: Angler Takes Malvertising to New Heights (2016), available from ⟨https://www.trustwave.com/Resources/SpiderLabs-Blog/Angler-Takes-Malvertising-to-New-Heights/⟩ (accessed 2017-03-31).

[5] Cyphort: HuffingtonPost Serving Malware via AOL Ad-Network (2015), available from ⟨https://www.cyphort.com/huffingtonpost-serving-malware/⟩ (accessed 2017-03-31).

[6] FOX IT: Malicious advertisements served via Yahoo, available from ⟨https://blog.fox-it.com/2014/01/03/malicious-advertisements-served-via-yahoo/⟩ (accessed 2017-03-31).

[7] Cyphort: The Rise of Malvertising (2015), available from ⟨http://go.cyphort.com/Malvertising-Report-15-Page.html⟩ (accessed 2017-03-31).

[8] Zarras, A., Kapravelos, A., Stringhini, G., Holz, T., Kruegel, C. and Vigna, G.: The dark alleys of Madison avenue: Understanding malicious advertisements, *Proc. 2014 Conference on Internet Measurement Conference*, pp.373–380, ACM (2014).

[9] proofpoint: Microsoft Patches CVE-2016-3351 Zero-Day, Exploited By AdGholas and GooNky Malvertising Groups (2016), available from ⟨https://www.proofpoint.com/us/threat-insight/post/Microsoft-Patches-Zero-Day-Exploited-By-AdGholas-GooNky-Malvertising⟩ (accessed 2017-03-31).

[10] TrendLabs: Let's Encrypt Now Being Abused By Malvertisers (2016), available from ⟨http://blog.trendmicro.com/trendlabs-security-intelligence/lets-encrypt-now-being-abused-by-malvertisers/⟩ (accessed 2017-03-31).

[11] 坂本一仁，稲垣 俊，島岡政基，松永昌浩：Trusted Tag Provider（TTP）を導入したセキュア広告プラットフォームの提案，コンピュータセキュリティシンポジウム 2016 論文集，Vol.2016, No.2, pp.1145–1152 (2016).

[12] Li, Z., Zhang, K., Xie, Y., Yu, F. and Wang, X.: Knowing your enemy: Understanding and detecting malicious web advertising, *Proc. 2012 ACM Conference on Computer and Communications Security*, pp.674–686, ACM (2012).

[13] Sakib, M.N. and Huang, C.-T.: Automated Collection and Analysis of Malware Disseminated via Online Advertising, *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol.1, pp.1411–1416, IEEE (2015).

[14] Arshad, S., Kharraz, A. and Robertson, W.: Include Me Out: In-Browser Detection of Malicious Third-Party Content Inclusions (2015).

[15] Dong, X., Tran, M., Liang, Z. and Jiang, X.: AdSentry: Comprehensive and flexible confinement of JavaScript-based advertisements, *Proc. 27th Annual Computer*

Security Applications Conference, pp.297–306, ACM (2011).
[16] Louw, M.T., Ganesh, K.T. and Venkatakrishnan, V.: AdJail: Practical Enforcement of Confidentiality and Integrity Policies on Web Advertisements, *USENIX Security Symposium*, pp.371–388 (2010).
[17] Online Trust Alliance: Malvertising | Online Trust Alliance, available from ⟨https://otalliance.org/initiatives/malvertising⟩ (accessed 2017-03-31).
[18] TrustInAds.org: TrustInAds.org - Keeping people safe from bad online ads, available from ⟨https://trustinads.org/⟩ (accessed 2017-03-31).
[19] IETF: JSON Web Signature (JWS), available from ⟨https://tools.ietf.org/html/rfc7515⟩ (accessed 2017-03-31).
[20] IETF: JSON Web Algorithms (JWA), available from ⟨https://tools.ietf.org/html/rfc7518⟩ (accessed 2017-03-31).
[21] IETF: The Web Origin Concept, available from ⟨https://tools.ietf.org/html/rfc6454⟩ (accessed 2017-03-31).
[22] SecureBrain: GRED Web 改ざんチェック Cloud，入手先 ⟨http://www.securebrain.co.jp/products/gred/⟩（参照 2017-03-31）.
[23] セコムトラストシステムズ：セコムあんしん Web サービス，入手先 ⟨https://www.secomtrust.net/service/web/⟩（参照 2017-03-31）.
[24] JWT.io: JSON Web Tokens, available from ⟨https://jwt.io/⟩ (accessed 2017-03-31).
[25] 吉田　薫，藤井資子，菊池　豊，山本正晃，永見健一，中川郁夫，江崎　浩：ユーザ視点に基づいたブロードバンドインターネット環境における遅延・パケットロスの傾向分析，電子情報通信学会論文誌 B，Vol.91, No.10, pp.1182–1192 (2008).
[26] NTT Communications: Service Level Agreement: Intra-Japan, available from ⟨http://www.us.ntt.net/slas/access/intra_japan⟩ (accessed 2017-03-31).
[27] Zhang, W., Yuan, S. and Wang, J.: Optimal real-time bidding for display advertising, *Proc. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.1077–1086, ACM (2014).
[28] Google: Get Start | Real-Time Bidding Protocol, available from ⟨https://developers.google.com/ad-exchange/rtb/start⟩ (accessed 2017-03-31).
[29] Alexa: Top Sites by Category: World/Japanese/ニュース，入手先 ⟨http://www.alexa.com/topsites/category/Top/World/Japanese/%E3%83%8B%E3%83%A5%E3%83%BC%E3%82%B9⟩（参照 2017-03-31）.
[30] Interactive Advertising Bureau: OpenRTB API Specification Version 2.4, available from ⟨http://www.iab.com/wp-content/uploads/2016/03/OpenRTB-API-Specification-Version-2-4-FINAL.pdf⟩ (accessed 2017-03-31).
[31] IETF: JSON Web Token (JWT), available from ⟨https://tools.ietf.org/html/rfc7519⟩ (accessed 2017-03-31).
[32] Rundle, M., Maler, E., Nadalin, A., Reed, D. and Thibeau, D.: The Open Identity Trust Framework (OITF) Model (2010), available from ⟨http://iop.projectliberty.org/confluence/download/attachments/45059055/OITF+Trust+Model.pdf⟩ (accessed 2017-03-31).
[33] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A. and Diaz, C.: The Web never forgets: Persistent tracking mechanisms in the wild, *Proc. 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp.674–689, ACM (2014).

坂本 一仁　（正会員）

2007 年奈良工業高等専門学校専攻科電子情報工学専攻卒業．2009 年奈良先端科学技術大学院大学修士課程修了．同年セコム株式会社入社．ネットワークセキュリティやプライバシ保護の研究に従事．

稲垣 俊

2012 年千葉大学工学部情報画像学科卒業．2014 年同大学大学院修士課程修了．同年セコム株式会社入社．サイバーセキュリティの研究に従事．

島岡 政基　（正会員）

1998 年慶應義塾大学大学院理工学研究科修士課程修了．同年セコム株式会社入社，認証局システムの開発に従事．2004 年より同社 IS 研究所．2005 年から 2010 年まで国立情報学研究所特任准教授（後に客員准教授）を兼務．現在，IS 研究所主任研究員．認証基盤とトラストの研究開発に従事．博士（情報学）．

松永 昌浩

1997 年慶應義塾大学理工学部電気工学科卒業．1999 年同大学大学院理工学研究科計算機科学専攻修士課程修了．同年セコム株式会社入社．広域分散システムの研究開発およびセキュリティとプライバシ保護の研究に従事．