安全な Web サービスの実現にむけた複数プロバイダに跨がる関係性の相互宣言機構

大谷 亘[1,a)]　近藤 賢郎[2,b)]　ルーク コリー[3,c)]　甲斐 賢[4,d)]　植原 啓介[5,e)]　手塚 悟[5,f)]

概要：TLS によるセキュリティモデルでは，host-to-host の通信路の識別・秘匿化 を可能にするが，PaaS プロバイダや CDN プロバイダなど複数のサービスプロバイダを跨がる Web サービスにおいて，TLS はサービスプロバイダ同士の関係性を保証できない．ユーザは第三者の攻撃やサービスプロバイダの運用事故などにより，サービスプロバイダの意図しない通信先に reroute される可能性がある．本稿では，サービスプロバイダ同士で相互に署名した TLS 公開鍵を DNSSEC で保護された権威 DNS ゾーンで公 開する，軽量な自己管理型相互宣言機構 M2DMRT を提案する．M2DMRT により，サービスプロバイダは第三者に頼らずサービスプロバイダ同士の関係性を相互に宣言でき，ユーザは署名を検証することで容易に関係性を信頼し脅威を回避することができる．本稿では M2DMRT における相互宣言の登録にかかるプロトコルを設計して，そのサーバサイドにおける Proof of Concept の実装を行い，基本性能を評価した結果，実用に耐えうる性能を持つことがわかった．

キーワード：Web service integrity, Verifiable HTTP redirection, PaaS security, DNSSEC, PKI

Mutual Declaration Mechanism of Multi-provider Relationship for Trusted Web Services

Wataru Ohgai[1,a)]　Takao Kondo[2,b)]　Korry Luke[3,c)]　Satoshi Kai[4,d)]　Keisuke Uehara[5,e)]
Satoru Tezuka[5,f)]

Abstract: The TLS security model enables the identification and secrecy of the host-to-host communication channel; however, TLS cannot guarantee the relationship between service providers in Web services that cross multiple service providers such as PaaS providers and CDN providers. The user may be rerouted to an unintended destination due to an attack or operational accident. In this paper, we propose a lightweight self-managed mutual declaration mechanism, M2DMRT, where service providers mutually sign their TLS public keys and publish them in DNSSEC-protected zones. With M2DMRT, service providers can mutually declare their relationships with each other without relying on a third party, and users can easily trust the relationships and avoid threats by verifying the signatures. We designed a protocol for registering mutual declarations in M2DMRT, implemented a server-side proof of concept, and, after evaluating its basic performance, found it to be sufficiently performant for practical use.

Keywords: Web service integrity, Verifiable HTTP redirection, PaaS security, DNSSEC, PKI

[1] 慶應義塾大学総合政策学部
Faculty of Policy Management, Keio University
[2] 慶應義塾情報セキュリティインシデント対応チーム
Computer Security Incident Response Team, Keio University
[3] 慶應義塾インフォメーションテクノロジーセンター
Information Technology Center, Keio University
[4] 慶應義塾大学 SFC 研究所
Keio Research Institute at SFC, Keio University
[5] 慶應義塾大学環境情報学部
Faculty of Information and Environment Studies, Keio University
[a)] alt@sfc.wide.ad.jp
[b)] latte@itc.keio.ac.jp
[c)] koluke@sfc.wide.ad.jp
[d)] skai@sfc.keio.ac.jp

1. はじめに

今日のインターネットにおけるセキュリティモデルは Transport Layer Security (TLS)[1] による host-to-host の通信路の識別・秘匿化を基本としている．TLS では通信相手のホストが提示する公開鍵証明書に含まれる公開鍵を Web PKI などの公開鍵基盤に基づいて検証して，公開鍵証明書に記載されたドメイン名を認証したり (DV 証明書)，ドメイン名の認証に加えてドメイン名を管理する組織を認証したりする (OV 証明書，EV 証明書)．また TLS では，DHE/ECDHE[2], [3] などの鍵交換の仕組みを用いて前方秘匿性の伴った共通鍵を生成して通信路を暗号化して，通信内容の秘匿化と真正性保証を実現する．TLS は host-to-host の通信路の識別に基づいたセキュリティモデルである一方で，大規模なサービスプロバイダ環境における公開鍵証明書の運用上のコストなどを考慮して，複数のドメイン名に跨がって公開鍵証明書を共有する仕組み (Subject Alternative Name; SAN[4]) も普及している．

インターネット上の Web サービスを中心とするサービスプロバイダは，Web アプリケーションの高機能化・利用者の大規模化・コンテンツの広帯域化などに対応して，バックエンド構成の複雑化が進行する．このため複数の PaaS プロバイダや CDN プロバイダなどのサービスプロバイダに跨がったサービストラフィックの reroute によって，一つの Web サービスが構成される場合も多い．例えば e-commerce を運用する Web サービスは，決済サービスを請け負うサービスプロバイダや 3-D Secure 認証 [5] を請け負うクレジットカード会社に HTTP リダイレクトの仕組みよってサービストラフィックを reroute する．また CDN プロバイダはコンテンツ配信時の効率的な帯域消費や低遅延性を実現することを目的として，DNS ラウンドロビンによる名前解決や IP anycast に基づいてユーザ毎にコンテンツ配信サーバを割当ててサービストラフィックを reroute する [6].

本稿では Web サービスを構成するために実施されるサービストラフィックの reroute の真正性に関する脅威モデルを取り扱う．この脅威モデルは，結合されたサービスプロバイダ間は契約関係に基づく信頼を前提としているものの，その信頼が利用者の視点から可視性がないことに由来する．HTTP リダイレクトに基づく reroute 時の脅威としては，URL 改ざん，DNS spoofing, ARP spoofing などが挙げられる．これらの脅威への対策として HSTS[7] 等による常時 SSL 化のポリシを強制した上での HTTPS の利用が考えられる．HTTPS を利用することで通信相手のホストが提示してきた公開鍵証明書によって通信相手が認証され，通信路の暗号化に基づいたコンテンツ改ざんの検知・防止が可能となる．しかし HSTS によるポリシの強制が適用できない場合には，中間者による HTTP への downgrade[8], [9] が可能となる．また HSTS によるポリシの強制が可能な場合でも，SAN に基づいて公開鍵証明書が複数のホストで共有されている場合 [10] などの一定の条件下でポリシの強制を回避できる可能性が指摘されている [11]．また DNS に基づくサービストラフィックの reroute 時の脅威としては，運用者による設定ミスなどの運用事故に由来する subdomain takeover[12], [13] が挙げられる．Web サービスを構成するサービスプロバイダ間の契約関係が消滅した際に，仮に片方のサービスプロバイダの運用者が設定ミスを生じさせても，両者の間に双方の意思に基づいた関係性が定義されているかをユーザの視点から確認することができれば，このような脅威を防ぐことができる．

以上の脅威モデルを解決するため，本稿では Web サービスを構成する複数のサービスプロバイダに跨がった関係性を相互に宣言できる機構として，*M2DMRT* (*Mechanism of Mutual Decralation of Multi-provider Relationship for Trusted Web Services*) を提案する．M2DMRT は Web PKI や TLS に基づいた host-to-host の通信路の識別が保証されている環境を前提とする．このもとで，サービストラフィックの reroute 元のサービスプロバイダは reroute 先のサービスプロバイダの公開鍵に対して，自身の秘密鍵でデジタル署名を生成する．またサービストラフィックの reroute 先のサービスプロバイダも同様に，reroute 元のサービスプロバイダの公開鍵に対して，自身の秘密鍵でデジタル署名を生成する．これらのデジタル署名と署名対象の公開鍵に対応するサービスプロバイダ名は，Web サービスの利用者の視点から検証可能なレポジトリに登録される．

本稿では，レポジトリの自己管理性やトランザクション数に関するスケーラビリティの要件を考慮して，このレポジトリとして DNSSEC が有効化されていることを前提に DNS を採用する．DNSSEC により DNS のルート zone を信頼の基点とする trust chain 上で，ゾーン毎に権威を持つ DNS サーバの認証と，そこに登録されたリソースレコードの真正性が保証される．Web サービスの利用者は，サービスプロバイダ間のサービストラフィックの reroute 時に，DNSSEC の仕組みに基づいて，reroute 元と reroute 先のサービスプロバイダそれぞれのサービスプロバイダ名及び公開鍵，それに付与されたデジタル署名を取得する．これらのデジタル署名を検証することで，当該サービストラフィックの reroute が，reroute 元と reroute 先のそれぞれのサービスプロバイダが意図したものかどうかを，サービス利用者の視点で検証することが可能となる．

e) kei@wide.ad.jp
f) tezuka@sfc.keio.ac.jp

2. 関連研究

2.1 HTTPS を前提とする機構

Certification Transparency (CT)[14] は Web PKI における商用認証局の運用事故 [15] に対応して，認証局による証明書の誤発行や悪意のある攻撃者などによる不正な証明書発行を検知する仕組みである．誤発行や不正に発行された証明書は，ユーザの視点からは正当な証明書と区別がつかず，攻撃者があたかも正しいサイトであるかのように見える偽のサイトに接続させることで入力したデータなどの通信内容が傍受される危険性が生じる．CT では認証局が証明書を発行する際に第三者が運用する CT ログサーバーに証明書を登録する．登録時には CT ログサーバーから証明書に対して Signed Certificate Timestamp (SCT) と呼ばれるタイムスタンプが付与される．証明書の検証時には SCT をもとに CT ログサーバに着目する証明書に関する発行の記録があるかを問い合わせる．検証に失敗した場合には同一のホスト名を持つ偽の証明書である可能性があると判断できる．

Cross Origin Resource Sharing (CORS)[16] は同一オリジンポリシに基づいた構成されている Web アプリケーションにおいて，その制約を一部解除して別のサーバへの HTTP リクエストを許可するための仕組みである．同一オリジンポリシは Cross-site Scripting (XSS) や Cross-site Request Forgeries (CSRF) といった Web セキュリティ脅威を防ぐために重要な役割を果たす．CORS が適応される場合，HTTP リクエストヘッダには Origin フィールドが追加されて，HTTP サーバに対して元の Origin 名が示される．Origin フィールド付きの HTTP リクエストを受け取った HTTP サーバは Origin フィールドを参照して元の Origin を認証する．Origin が認証された場合，HTTP レスポンスヘッダには Access-Control-Allow-Origin フィールドが追加されて，HTTP クライアントに対して認証された Origin 名を示す．

Subresource Integrity (SRI)[17] は Web サービスの構成時に CDN などのサードパーティ上でスクリプトなどの subresource がホストされる状況を想定して，その subresource が改ざんされる可能性を脅威と捉える．Web サービス内で script タグや link タグでサードパーティ上でホストされた subresource を参照する際に，あらかじめ subresource をもとに計算しておいたハッシュ値を integrity 属性として付加する．Web サービスのユーザが subresource にアクセスする際には，integrity 属性に記載されたハッシュ値と実際に取得されたコンテンツから計算したハッシュ値を比較して真正性を検証する．

2.2 分散レポジトリに基づく仕組み

昨今 Distributed Hash Table (DHT)[18] や Blockchain[19] 技術に基づいた分散レポジトリの構築が盛んである．DNS を分散レポジトリとして見る際，プライバシや Transaction ID spoofing[20] などのセキュリティ上の懸念から，DHT や Blockchain を用いて DNS を再構成しようとする取り組みがある [21], [22]. Handshake[23] はブロックチェーンを用いた非中央集権型命名プロトコルである．Handshake ネットワークに参加するピアは，ブロックチェーンによりトップレベルドメインを管理するトランザクションを生成し検証できる．また，同様にドメイン名の管理者はブロックチェーン上で管理権を証明できるため，Handshake ネットワークは PKI における認証局の役割も果たす．Handshake はドメイン名の管理権証明や取引に経済的なインセンティブを与えることで中央集権化を防いでいる．

2.3 DANE TLSA

DANE TLSA[24] は DNSSEC によって保護された DNS ゾーンで，ゾーン内のホストで使用する TLS の公開鍵証明書や公開鍵証明書を署名する認証局を指定するプロトコルである．ゾーンの管理者は DANE TLSA クライアントに対して証明書の検証方法を指定することができ，PKI に頼らずに DNSSEC の trust chain のみを信頼して検証させることができる．なお，指定したホストにおける証明書や信頼方法の提示はできるが，複数のホストに跨がる提示や，ゾーン外のホストを指定することはできない．

2.4 関連研究間の比較

本稿では関連研究を比較するための評価軸として以下を定義する．これらの評価軸に基づいて関連研究を比較すると **表 1** を得る．

1. Mutual and verifiable declaration of service relationship: Web サービスを構成するサービスプロバイダ間の関係性をサービストラフィックの reroute 前後の双方の視点から定義できる必要がある．またここで定義されたサービスプロバイダの関係性はユーザの視点から検証可能である必要がある．

2. Localization of transaction of declaration modification: サービスプロバイダ間の関係性の定義にかかるトランザクションは，インターネット規模のグローバルな Web 環境を支える意味でスケーラブルである必要がある．このためサービスプロバイダ間の関係性の定義にかかるトランザクションは局所的である必要がある．

3. Self-manageable declaration of service relationship: サービスプロバイダ間の関係性は当事者同士の契約関係に関わる性質を有する．このためここで定義される関係性は当事者の管理するレポジトリ内に集約されていて，当事者

表 1　関連研究との比較.

	Web PKI / CT	CORS / SRI	Handshake / DHT	DANE TLSA	M2DMRT
Mutual and verifiable declaraion	No	No	No	No	Yes
Localization of transaction	Middle	Yes	No	Yes	Yes
Self-manageable declaration	No	No	No	No	Yes
Localization of failure points	No	Yes	Yes	Yes	Yes
Minimum disclosure	No	No	No	Yes	Yes

の意思のみに基づいてレポジトリから改変や削除できる必要がある.

4. Localization/minimization of failure points (independent from central authority): サービスプロバイダ間の関係性を定義するレポジトリは，インターネット規模のグローバルな Web 環境を支える意味で可用性を有する必要がある．このためこのレポジトリは中央集権的な権威などの単一障害点に依存しない構成とする必要がある.

5. Minimum disclosure of each party's components: サービスプロバイダ間の関係性は当事者同士の契約関係に関わる性質を有する．このためここで定義される関係性は Web サービスの利用者の視点で必要最小限な開示範囲とする必要がある.

3. 提案: M2DMRT

本章では，複数のサービスプロバイダ間で HTTP リダイレクトを伴う Web サービスの事例について，ユーザと e-commerce の購買情報をやりとりする EC サイトと，ユーザから入力されたクレジットカード情報などをもとに決済処理を行う Payment サイトの 2 者からなる Web サービスを想定する.

3.1 想定する環境

図 1 は既存の Web PKI によるホストの真正性担保のみを利用した環境を模している．ユーザが EC サイトで商品を購入したいが，決済しようとすると別の Web サイトである Payment サイトへリダイレクトされ，決済を要求されたことを想定する．ユーザのデバイスは各々の Web サーバと TLS を利用して通信しており，ユーザのデバイスは Web サーバから提供される公開鍵証明書をもとに，認証局の公開鍵証明書を trust anchor とする Web PKI の trust chain を利用し，各々の Web サーバの真正性を検証できる．しかし，ユーザは EC サイトと Payment サイト各々のサービスプロバイダ同士の関係性を検証できないため，リダイレクトされた Payment サイトに対し決済情報を提供することで本当に EC サイトの商品を購入できるかわからない．また，HTTPS ハイジャック攻撃や CDN 環境におけるサブドメインテイクオーバ攻撃により，ユーザは EC サイトの意図しない通信先にリダイレクトされうる.

図 2 は Web PKI と本稿で提案する機構による Web サイト同士の関係性担保を併用した環境を模している．本機構では，関係性を持つ複数のサービスプロバイダの Web サーバが利用している TLS 公開鍵について，TLS 秘密鍵で相互に署名する．署名は関係性の相互宣言として，DNSSEC で保護されたサービスプロバイダの権威 DNS サーバで公開する．DNS と DNSSEC を利用することで，相互宣言はサービスプロバイダ自身により管理でき，非中央集権性をもつ．また，サービスプロバイダは第三者に頼ることなく相互宣言を作成，変更，破棄することができる．署名はサービスで使用するすべての TLS 鍵ペアについて相互に行い，各々を独立した DNS リソースレコードとして公開するため，サービスプロバイダはすべての Web サーバをリストとして公開する必要がない．本機構により，ユーザはサービスプロバイダ同士の関係性が容易に検証できる.

3.2 相互宣言の登録時のシーケンス

図 3 は EC サイトと Payment サイトの 2 つのサービスプロバイダそれぞれに属する *MRDA* (*Mutual Relation Declaration Agent*) 同士の認証と鍵交換のシーケンスを示す．**図 4** は MRDA による TLS 公開鍵への署名と，署名により生成された相互宣言 DNS リソースレコードを公開するシーケンスを示す.

両サービスプロバイダでは，署名に使用する Web サーバの TLS 公開鍵をあらかじめ MRDA に登録しておく．また，両 MRDA では認証に非対称鍵暗号による鍵ペアを使用するため，サービスプロバイダ間で互いの公開鍵自己署名証明書を交換しておく.

相互宣言の新規作成が必要となった場合には，まず，両 MRDA 間で認証と鍵交換を行う．図 3-(1) - 図 3-(3) では互いに乱数 ($nonce_1$, $nonce_2$) を送信し，秘密鍵で署名されたものを相手の公開鍵証明書で検証することで認証する．図 3-(4), 図 3-(5) では Diffie-Helman Ephemeral 鍵交換を行い，共通鍵 (SK) を生成することで以降の通信を暗号化できるようにする.

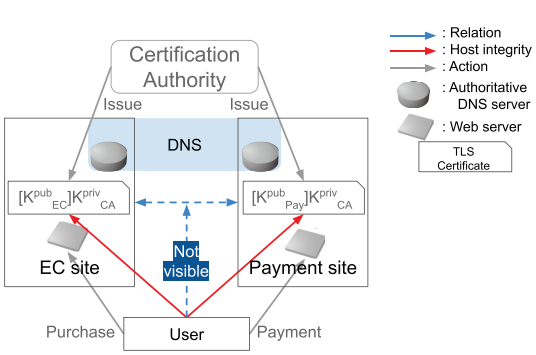

図 1 Web PKI におけるホストの真正性保証.

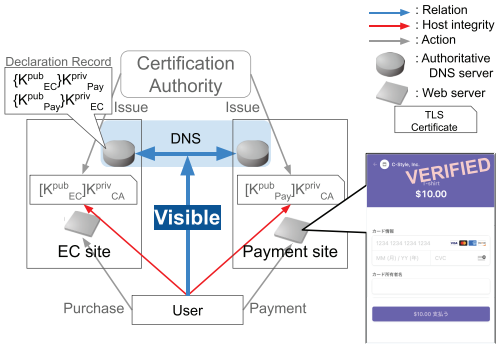

図 2 M2DMRT に基づいたサービストラフィックの reroute の真正性保証.

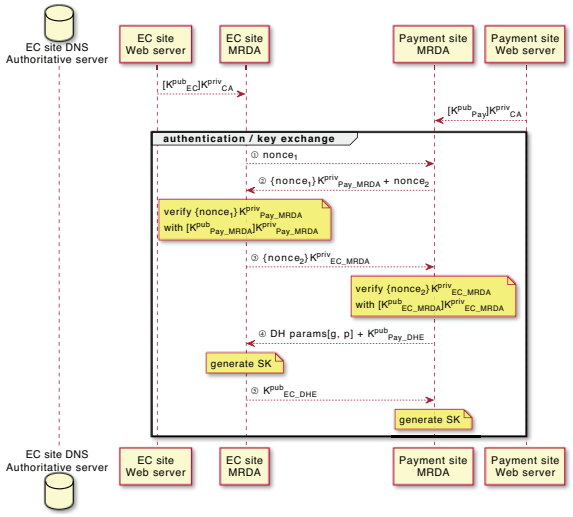

図 3 相互宣言の登録時のシーケンス 1.

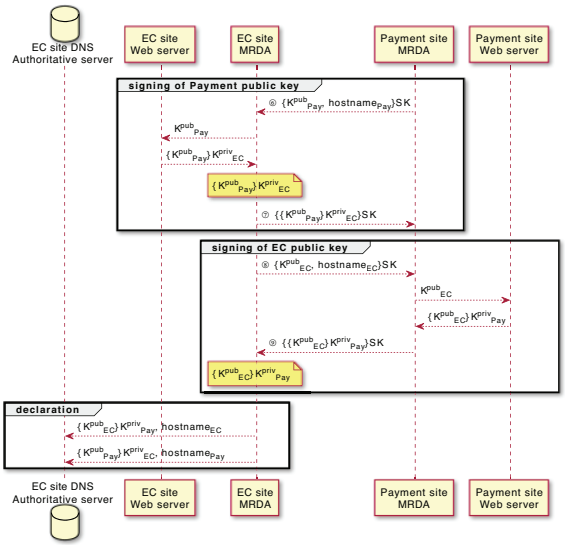

図 4 相互宣言の登録時のシーケンス 2.

次に，一方の TLS 公開鍵を他方の TLS 秘密鍵で相互に署名する．図 4-(6), 図 4-(7) では Payment サイトの TLS 公開鍵への署名を，図 4-(8), 図 4-(9) では EC サイトの TLS 公開鍵への署名を行う．図 4-(6), 図 4-(8) で署名を要求する際にその TLS 公開鍵を使う Web サーバのホスト名を送信し，すべての鍵について相互に署名することで，複数の Web サーバで TLS 通信を終端する場合でもこの機構が利用できる．署名に使う TLS 秘密鍵は MRDA ではなく各々の Web サーバが管理しているため，MRDA は TLS 秘密鍵を使って署名できる別のエージェントに署名を要求する．署名は TLS 公開鍵を所有者する側の MRDA に返送され検証される．検証に失敗した場合，再度署名を要求する．

最後に，相互宣言を DNS リソースレコードとして公開する．リソースレコードは，署名する TLS 秘密鍵と署名される TLS 公開鍵の対の数だけ生成され公開される．

4. 実装

本稿では，図 3, 図 4 に示した EC サイト及び Payment サイトの MRDA を PoC 実装した．すべての機能を Python3.8.2 で実装した．通信には標準の socket ライブラリを利用した．暗号処理については，Diffie-Helman Ephemeral 鍵交換には cryptography ライブラリを，TLS 公開鍵への署名とその検証には PyOpenSSL ライブラリを，通信の AES 暗号化と復号には PyCryptoDome ライブラリをそれぞれ利用した．

MRDA 同士の認証に先立ち，各々のサービスプロバイダは RSA 鍵ペアを生成し，自己の秘密鍵で署名した公開鍵証明書を安全に交換している前提を設ける．認証は互いに nonce を送信し，秘密鍵で署名し返送した後に公開鍵証明書で署名を検証する事で行った．また，認証以後の通信の秘匿性と完全性を確保するため，Diffie-Helman Ephemeral 方式による鍵交換を行い，以降の通信はその共通鍵によって内容を AES で暗号化する．

表 2 評価環境.

OS	Apple MacOS Big Sur 11.4
CPU	Apple M1 (3.2 GHz, 8 cores)
RAM	16GB

表 3 認証・鍵交換の各区間にかかった時間の平均.

(1)	(2)	(3)	(4)	(5)
11.7 ms	0.616 ms	11.2 ms	0.409 ms	49.0 ms
(6)	(7)	(8)	-	-
6.87 ms	0.227 ms	0.282 ms	-	-

表 4 TLS 公開鍵署名の各区間にかかった時間の平均.

(9)	(10)	(11)	(12)	(13)
1.43 ms	1.40 ms	11.0 ms	0.366 ms	0.379 ms
(14)	(15)	(16)	(17)	(18)
0.20 ms	0.183 ms	10.9 ms	0.176 ms	0.201 ms
(19)	-	-	-	-
0.393 ms	-	-	-	-

相互宣言は，一方のサービスプロバイダの Web サイトがサービスに使用する TLS 公開鍵に対し，他方のサービスプロバイダの Web サイトがサービスに使用する TLS 秘密鍵で署名する事で生成される．まず，一方の MRDA が他方の MRDA に対し，TLS 公開鍵とその鍵を使用する Web サーバのホスト名を暗号化して他方の MRDA に送信する．次に，他方の MRDA は，自身の Web サイトで使用する TLS 秘密鍵を使用して受信した TLS 公開鍵に署名し，署名をもう一方の MRDA に送信する．最後に，署名を受信した MRDA は，他方の Web サイトで使用している TLS 公開鍵を使用して検証する．これを双方の TLS 公開鍵に対して行うことですべての相互宣言の生成が完了する．なお，複数の Web サーバを使用してサービスを提供しているなど，一つのサービスプロバイダに対し TLS 鍵ペアが複数存在する場合は，すべての TLS 鍵ペアについて相互宣言を生成する．

5. 評価

前章の PoC 実装について，暗号処理にかかるノード内処理時間を評価した．**図 5** と**図 6** に評価時間の区間を示す．評価環境として，**表 2** に示すコンピュータを用いた．**表 3**，**表 4** に，それぞれの区間を 100 回計測して算出した平均を示す．計測では，2 つのサービスプロバイダがそれぞれ 1 つの Web サーバと TLS 鍵ペアを使用していることを前提とした．

5.1 認証・鍵交換

表 3 において，(1)，(3) は MRDA の秘密鍵による nonce への署名にかかる時間を，(2)，(4) は (1)，(3) で得た署名の公開鍵による検証にかかる時間を，(5)，(6) は Diffie-Helman Ephemeral 鍵交換に使用する Paymen サイト MRDA の公開鍵算出にかかる時間を，(7)，(8) は Diffie-Helman Ephemeral 鍵交換によって得られる共通鍵算出にかかる時間を，それぞれ示している．nonce への署名や Diffie-Helman Ephemeral 鍵交換に使用する公開鍵の算出には時間がかかっているものの，署名検証や共通鍵の算出は高速で実行できていることがわかった．

5.2 TLS 公開鍵への署名

表 4 において，(9)，(14)，(17) は共通鍵によるメッセージの暗号化にかかる時間を，(10)，(12)，(15)，(18) は共通鍵によるメッセージの復号にかかる時間を，(11)，(16) は EC サイト TLS 秘密鍵による Payment サイト TLS 公開鍵への署名にかかる時間を，(13)，(19) は (11)，(16) で得た署名の EC サイト TLS 公開鍵による検証にかかる時間を，それぞれ示している．TLS 公開鍵への署名そのものは 10 ミリ秒程度かかることがわかった．

以上の測定結果を踏まえ，実際の通信では DNS キャッシュサーバによりユーザは相互宣言リソースレコードのキャッシュにアクセスできることから，本機構は実用に耐えうる性能を持つと考えられる．

6. まとめ / 今後の課題

本稿では TLS 公開鍵の相互署名と DNSSEC で保護された DNS 権威サーバでの署名の公開による，サービスプロバイダ同士の関係性についての軽量な自己管理型相互宣言機構である M2DMRT を提案した．提案では，M2DMRT が既存の TLS のみでは解決できないサービスプロバイダ間の reroute における脅威モデルを解決できる可能性について確認できた．Proof of Concept 実装と基本性能評価では，署名処理にかかる速度やユーザが利用できるキャッシュを考慮した結果，M2DMRT が実用に耐えうる性能を持つことがわかった．実用にあたっては，古い DNS キャッシュによる宣言変更時の影響や，ユーザが容易に署名を検証しサービスプロバイダ同士の関係性を信頼できるクライアントの Proof of Concept 実装などが今後の課題として挙げられる．

参考文献

[1] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, *IETF*, 2018.

[2] D. Gillmor. Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS). RFC 7919, *IETF*, 2016.

[3] Y. Nir, S. Josefsson, and M. Pegourie-Gonnard. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier. RFC 7919, *IETF*, 2016.

[4] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, *IETF*, 2008.

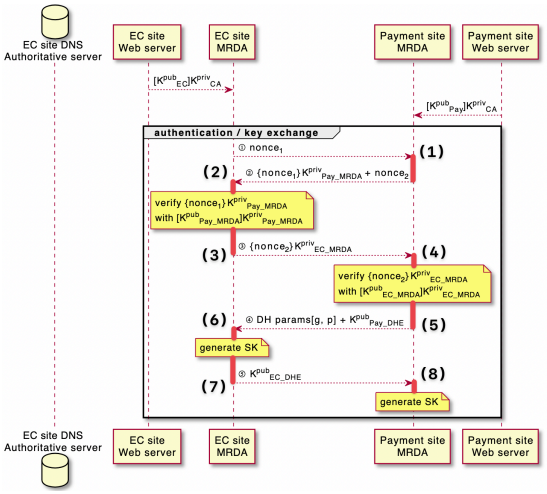

図 5 評価時間の範囲 1.

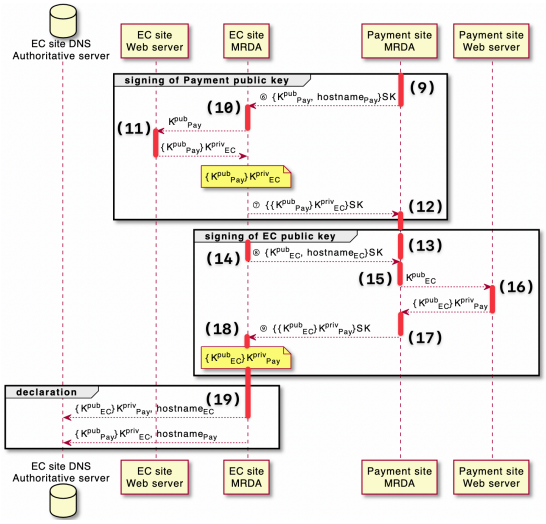

図 6 評価時間の範囲 2.

[5] 3-D Secure - EMVco. `https://www.emvco.com/emv-technologies/3d-secure/`.

[6] J. Zhao, P. Liang, W. Liufu, and Z. Fan. Recent Developments in Content Delivery Network: A Survey. In *Parallel Architectures, Algorithms and Programming*, pp. 98–106. Springer Singapore, 2020.

[7] J. Hodges, C. Jackson, and A. Barth. HTTP Strict Transport Security (HSTS). RFC 6797, *IETF*, 2012.

[8] M. Marlinspike. New Tricks For Defeating SSL In Practice. In *Proc. of BLACKHAT Europe '09*, 2009.

[9] X. Li, C. Wu, S. Ji, and R. Gu, Q.and Beyah. HSTS Measurement and an Enhanced Stripping Attack Against HTTPS. In *Security and Privacy in Communication Networks*, pp. 489–509. Springer Intl. Pub, 2018.

[10] M. Zhang, X. Zheng, K. Shen, Z. Kong, C. Lu, Y. Wang, H. Duan, S. Hao, B. Liu, and M. Yang. Talking with Familiar Strangers: An Empirical Study on HTTPS Context Confusion Attacks. In *Proc. of ACM CCS'20*, pp. 1939–1952, 2020.

[11] J. Selvi. Bypassing HTTP Strict Transport Security. In *Proc. of BLACKHAT Europe '14*, 2014.

[12] S.M.Z.U. Rashid, M.I. Kamrul, and A. Islam. Understanding the Security Threats of Esoteric Subdomain Takeover and Prevention Scheme. In *Proc. of ECCE '19*, pp. 1–4, 2019.

[13] M. Squarcina, M. Tempesta, L. Veronese, S. Calzavara, and M. Maffei. Can I Take Your Subdomain? Exploring Same-Site Attacks in the Modern Web. In *Proc. of USENIX Security '21*, pp. 2917–2934, 2021.

[14] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, *IETF*, 2013.

[15] N. Leavitt. Internet security under attack: The undermining of digital certificates. *Computer*, Vol. 44, No. 12, pp. 17–20, 2011.

[16] WHATWG - Fetch Living Standard. `https://fetch.spec.whatwg.org/#cors-protocol`.

[17] D. Akhawe, F. Braun, F. Marier, and J. Weinberger. Subresource Integrity. REC-SRI-20160623, *W3C*, 2016.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM '01*, pp. 149–160, 2001.

[19] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.

[20] V. Ramasubramanian and E.G. Sirer. Perils of Transitive Trust in the Domain Name System. In *Proc. of USENIX IMC '05*, pp. 379–384, 2005.

[21] Y. Liu, Y. Zhang, S. Zhu, and C. Chi. A Comparative Study of Blockchain-Based DNS Design. In *Proc. of ACM ICBTA '19*, pp. 86–92, 2019.

[22] K. Matsuoka and T. Suzuki. Blockchain and DHT Based Lookup System Aiming for Alternative DNS. In *Proc. of ICCCI '20*, pp. 98–105, 2020.

[23] Handshake. `https://handshake.org`.

[24] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, *IETF*, 2012.