情報処理学会論文誌 Vol.55 No.9 2061–2071 (Sep. 2014)

推薦論文

Web Workersを用いた多変数公開鍵暗号Rainbowの並列実装

鷲見 拓哉[1] 石黒 司[2] 清本 晋作[2] 三宅 優[2] 小林 透[3] 高木 剛[4,a)]

受付日 2013年11月30日, 採録日 2014年6月17日

概要：W3C は，HTML5 および JavaScript 上で並列計算を行うための規格である Web Workers の勧告候補を 2012 年に公開した．JavaScript および Web Workers はプラットフォーム非依存である．JavaScript で書かれたウェブアプリケーションは広く普及しており，インターネット選挙運動やブロードキャスティングサービス等を行うウェブアプリケーションの中には安全な通信を必要とするものが存在する．そのようなウェブアプリケーションにディジタル署名を組み込むことにより，安全な通信を実現することができる．Rainbow 署名は，Ding と Schmidt により 2005 年に提案された多変数公開鍵暗号方式のディジタル署名である．Rainbow 署名は有限体上の多変数 2 次多項式の連立方程式に対する求解問題が NP 困難であることを安全性の根拠とし，ポスト量子暗号の 1 つとして期待されている．本稿では，Web Workers を用いた Rainbow 署名の並列実装手法を提案し，その応用例を述べる．また，マルチコア CPU を搭載する汎用 PC および Android タブレット端末を用いて提案方式の実行時間を計測した．その結果，汎用 PC 上においては 0.62 ミリ秒，Android タブレット端末上においては 4.54 ミリ秒で署名検証を行うことができた．

キーワード：Web Workers，JavaScript，並列実装，多変数公開鍵暗号，ディジタル署名

Parallel Implementation of Multivariate Public Key Cryptosystem Rainbow Using Web Workers

Takuya Sumi[1] Tsukasa Ishiguro[2] Shinsaku Kiyomoto[2] Yutaka Miyake[2] Toru Kobayashi[3] Tsuyoshi Takagi[4,a)]

Received: November 30, 2013, Accepted: June 17, 2014

Abstract: Web Workers is a specification that defines an API which allows Web application developers to use background workers running scripts in parallel. The W3C published its candidate recommendation in 2012. Web Workers is used with JavaScript and is platform-independent. Hence, Web applications written in JavaScript can be used for a wide variety of purposes. There are many Web applications and some of them, for instance Internet election campaign and real-time broadcasting, need secure communications. We can include digital signatures with such Web applications to guarantee their security of communications. The Rainbow signature scheme is a multivariate public key cryptosystem proposed by Ding and Schmidt in 2005. The security of the Rainbow signature scheme is based on the NP-hardness of solving the multivariate quadratic equations over a finite field, and it is expected to be one of the candidates of post-quantum cryptography. In this paper, we propose a parallel implementation of the Rainbow signature scheme using Web Workers and assess the performance of it on several Web browsers. We also propose cryptography applications for Web browsers. With our implementation, it is possible to verify a message in 0.62 milliseconds on a Windows PC and 4.54 milliseconds on an Android tablet.

Keywords: Web Workers, JavaScript, parallel implementation, MPKC (multivariate public key cryptosystem), digital signature

1. はじめに

HTML5 [1] は，HTML の次世代標準であり，WWW に関する国際的な標準化団体である W3C が策定作業を進めている．W3C は，HTML5 に関する最初の作業草案を 2008 年に公開した．その後，W3C は作業草案の改定を重ね，2012 年に勧告候補を公開した．HTML5 の登場により，高機能なウェブアプリケーションを開発することが可能となった．ウェブアプリケーションは，JavaScript 言語で記述される．JavaScript は，オブジェクト指向のスクリプト言語であり，国際的な標準化団体である Ecma International により標準化され，ECMAScript（ECMA-262 [2]）としてその仕様が公開されている．ウェブブラウザは，ECMA-262 に基づき，JavaScript を実装している．しかし，JavaScript の実行はシングルスレッドであり，高機能なウェブアプリケーションを開発するうえでウェブブラウザの応答性の低下が問題となることがあった．W3C は，この問題を解決するために，JavaScript で記述されたウェブアプリケーション上で並列処理を行うための規格を設けた．それが，Web Workers [3] である．Web Workers は，ウェブアプリケーションの実行環境にバックグラウンドワーカ[*1]を提供する．ウェブアプリケーションの開発者は，バックグラウンドワーカを用いて処理を並列化することができる．W3C は，Web Workers に関する最初の作業草案を 2009 年に，勧告候補を 2012 年に公開した．JavaScript および Web Workers は，特定の実行環境に依存しないプラットフォーム非依存な技術である．

現在実用化されている公開鍵暗号には，RSA 暗号や楕円曲線暗号等がある．これらの暗号は，素因数分解問題や楕円曲線上の離散対数問題が困難であることを安全性の根拠とする．しかし，Shor のアルゴリズム [4] によれば，これらの問題は量子コンピュータを用いれば多項式時間で解くことができる．したがって，量子コンピュータを用いた場合においても解くことが困難な公開鍵暗号を構成する必要がある．有限体上の連立多変数非線形方程式において各係数をランダムに選んだ場合，その求解問題は NP 困難であることが知られている [5]．また，有限体上の連立多変数非線形方程式の求解は，量子コンピュータを用いた場合においても困難であると考えられている [6]．ゆえに，多変数多項式を用いた公開鍵暗号が活発に研究されている．多変数公開鍵暗号には，Mixed-Field 型と呼ばれる松本今井暗号 [7] や HFE 署名 [8]，Single-Field 型と呼ばれる UOV 署名 [9] 等が提案されている．

Rainbow 署名は，Ding と Schmidt により 2005 年に提案された多変数公開鍵暗号方式のディジタル署名である [10]．Rainbow 署名は多変数 2 次多項式の連立方程式に対する求解問題が困難であることを安全性の根拠とし，ポスト量子暗号の 1 つとして期待されている．Rainbow 署名の署名生成および署名検証は，位数の小さな有限体上の多項式の演算により実装するため，広く普及している RSA 署名と比較して処理速度が高速である [11]．さらに，Rainbow 署名の署名検証は，扱う多項式のデータ並列性が高いため，並列計算に向いているという特徴を持つ．したがって，Web Workers と Rainbow 署名を組み合わせることにより，より安全なウェブサービスをハイパフォーマンスで実現することができる．また，Rainbow 署名を，JavaScript を用いてウェブブラウザ上に実装することで，ウェブアプリケーション内において署名生成および署名検証を行うことが可能となる．したがって，特別なソフトウェアやハードウェアを用いることなく，一般的なウェブブラウザのみでメッセージの送信者の認証と，メッセージの改ざん検出を行うことができる．この方法を用いることにより，たとえば，インターネット選挙運動における選挙運動用文書図画の頒布にあたり，その正当性を示すことができる．インターネット上のディジタルコンテンツを保護する仕組みとしては，ディジタル著作権管理（DRM）や SSL/TLS 技術が存在する．DRM では，コピーを制限することでディジタルコンテンツの再頒布を禁止することができるが，これを実現するために特定のソフトウェアやハードウェアに依存している．また，SSL/TLS 通信においては，通信相手の認証および通信内容の改ざん検出機能の他に，通信内容の暗号化を行うことで通信の盗聴を防ぐことができる．しかし，インターネット選挙運動における選挙運動用文書図画の頒布においては，文書図画の再頒布や通信における盗聴対策は重要ではなく，送信者の認証および内容の改ざん検出ができれば十分であると考えられる．したがって，そのような用途においては，本稿で提案する方法は，既存の DRM や SSL/TLS 通信と比較して，より軽量な方法として利用することが可能である．

本稿では，Rainbow 署名を，JavaScript を用いてウェブブラウザ上に実装した．Rainbow 署名の実装にあたり，署名検証については Web Workers を用いて並列化を施した．実装した Rainbow 署名を，マルチコア CPU を搭載する汎

1 九州大学大学院数理学府
Graduate School of Mathematics, Kyushu University, Fukuoka 819–0395, Japan
2 株式会社 KDDI 研究所
KDDI R&D Laboratories, Inc., Saitama 356–8502, Japan
3 長崎大学大学院工学研究科
Graduate School of Engineering, Nagasaki University, Nagasaki 852–8521, Japan
4 九州大学マス・フォア・インダストリ研究所
Institute of Mathematics for Industry, Kyushu University, Fukuoka 819–0395, Japan
a) takagi@imi.kyushu-u.ac.jp

[*1] 文献 [3] における “background workers” の訳語である．
本稿の内容は 2013 年 10 月のコンピュータセキュリティシンポジウム 2013 にて報告し，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

用 PC および Android タブレット端末上で動作させた場合の署名生成および署名検証に必要な実行時間を計測した．

2. Rainbow 署名

本章では，Rainbow 署名 [10] について説明する．

2.1 Rainbow 署名の構築

$\mathbb{K}$ を有限体とする．$n \in \mathbb{N}$ に対し，$V = \{1, 2, \ldots, n\}$ とおく．$t \in \mathbb{N}$ に対し，$v_1, v_2, \ldots, v_{t+1} \in \mathbb{N}$ を

$$0 < v_1 < v_2 < \cdots < v_t < v_{t+1} = n$$

なるものとし，$m = n - v_1$ とおく．$i = 1, 2, \ldots, t+1$ に対し，$V_i = \{1, 2, \ldots, v_i\}$ とおくと，$\#V_i = v_i$ である．$i = 1, 2, \ldots, t$ に対し

$$O_i = V_{i+1} \setminus V_i = \{v_i + 1, v_i + 2, \ldots, v_{i+1}\},$$
$$o_i = v_{i+1} - v_i$$

とおく．$\#O_i = o_i$ である．

Rainbow 署名は，t 個のレイヤから構成される．$h = 1, 2, \ldots, t$ に対し，第 h レイヤでは，次に示す o_h 個の n 変数多項式を使う．$f_k(x_1, x_2, \ldots, x_n) = f_k(\boldsymbol{x})$ を次に示す 2 次多項式とする．

$$f_k(\boldsymbol{x}) = \sum_{i \in O_h, j \in V_h} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i,j \in V_h, i \le j} \beta_{i,j}^{(k)} x_i x_j + \sum_{i \in V_{h+1}} \gamma_i^{(k)} x_i + \eta^{(k)} \qquad (k \in O_h)$$

ここで，$\alpha_{i,j}^{(k)}, \beta_{i,j}^{(k)}, \gamma_i^{(k)}, \eta^{(k)} \in \mathbb{K}$ である．x_i $(i \in O_h)$ を Oil 変数，x_j $(j \in V_h)$ を Vinegar 変数と呼ぶ．Rainbow 署名の中心写像 $\mathcal{F}\colon \mathbb{K}^n \to \mathbb{K}^m$ を次のとおりとする．

$$\mathcal{F}(\boldsymbol{x}) = \overbrace{(f_{v_1+1}(\boldsymbol{x}),\ f_{v_1+2}(\boldsymbol{x}),\ \ldots,\ f_n(\boldsymbol{x}))}^{m \text{ 個の } n \text{ 変数多項式}}$$

$f_k(\boldsymbol{x})$ $(k \in O_h)$ は Oil 変数が 1 次線形の形で現れるという特徴を持ち，$f_k(\boldsymbol{x})$ に現れるすべての Vinegar 変数を固定すると，Oil 変数に関する 1 次式になる．

いま，$f_k(\boldsymbol{x})$ と固定された $(b_1, b_2, \ldots, b_{v_h}) \in \mathbb{K}^{v_h}$ に対して，$x_1 = b_1$, $x_2 = b_2$, …, $x_{v_h} = b_{v_h}$ なる代入を行えば，Oil 変数 x_i $(i \in O_h)$ に関する 1 次多項式 $\tilde{f}_k(x_{v_h+1}, x_{v_h+2}, \ldots, x_{v_{h+1}}) = \tilde{f}_k(\tilde{\boldsymbol{x}})$ が作れる．$\tilde{f}_k(\tilde{\boldsymbol{x}})$ と適当な $(a_{v_h+1}, a_{v_h+2}, \ldots, a_{v_{h+1}}) \in \mathbb{K}^{o_h}$ を用いて連立 1 次方程式を解くことにより，中心写像 $\mathcal{F}$ の逆像（の 1 つ）を簡単に求めることができる．以下に，署名生成において $\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$ を計算する方法を示す．

$\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$ の計算方法

(1) $(b_1, b_2, \ldots, b_{v_1}) \in \mathbb{K}^{v_1}$ をランダムに決める．

(2) $\boldsymbol{a} = (a_{v_1+1}, a_{v_1+2}, \ldots, a_n) \in \mathbb{K}^m$ とする．$h = 1, 2, \ldots, t$ に対して，Oil 変数 x_i $(i \in O_h)$ に関する連立 1 次方程式

$$\tilde{f}_k(x_{v_h+1}, x_{v_h+2}, \ldots, x_{v_{h+1}}) = a_k \quad (k \in O_h)$$

の解 $(b_{v_h+1}, b_{v_h+2}, \ldots, b_{v_{h+1}}) \in \mathbb{K}^{o_h}$ を計算する．解がなければ (1) へ戻る．

(3) $(b_1, b_2, \ldots, b_n) \in \mathbb{K}^n$ が中心写像 $\mathcal{F}$ の逆像（の 1 つ）である．

鍵生成

秘密鍵　中心写像 $\mathcal{F}$ と 2 つのアフィン同型写像

$$\mathcal{S}(\boldsymbol{y}) = S\boldsymbol{y} + \boldsymbol{c}_S : \mathbb{K}^m \to \mathbb{K}^m,$$
$$\mathcal{T}(\boldsymbol{x}) = T\boldsymbol{x} + \boldsymbol{c}_T : \mathbb{K}^n \to \mathbb{K}^n.$$

ここで，$S \in \mathbb{K}^{m \times m}$, $T \in \mathbb{K}^{n \times n}$ は正則行列であり，$\boldsymbol{y}, \boldsymbol{c}_S \in \mathbb{K}^m$, $\boldsymbol{x}, \boldsymbol{c}_T \in \mathbb{K}^n$ である．

公開鍵　合成写像 $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{K}^n \to \mathbb{K}^m$．ここで

$$\mathcal{P}(\boldsymbol{x}) = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}(\boldsymbol{x}) = (p_{v_1+1}(\boldsymbol{x}), p_{v_1+2}(\boldsymbol{x}), \ldots, p_n(\boldsymbol{x}))$$

であり，$p_k(\boldsymbol{x})$ $(v_1 + 1 \le k \le n)$ は $\boldsymbol{x}$ に関する n 変数 2 次多項式である．

署名生成　署名対象のメッセージを $\boldsymbol{M} \in \mathbb{K}^m$ とする．$\boldsymbol{a} = \mathcal{S}^{-1}(\boldsymbol{M})$, $\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$, $\boldsymbol{s} = \mathcal{T}^{-1}(\boldsymbol{b})$ の順に $\boldsymbol{a} \in \mathbb{K}^m$, $\boldsymbol{b} \in \mathbb{K}^n$, $\boldsymbol{s} \in \mathbb{K}^n$ を計算する．$\boldsymbol{s}$ が $\boldsymbol{M}$ に対する署名である．

署名検証　$\mathcal{P}(\boldsymbol{s}) = \boldsymbol{M}$ ならば署名は有効である．

この方式を Rainbow$(\mathbb{K}; v_1, o_1, o_2, \ldots, o_t)$ と表し，$(\mathbb{K}; v_1, o_1, o_2, \ldots, o_t)$ を Rainbow 署名のパラメータと呼ぶ．

2.2 パラメータと鍵長

本稿では，安全性が 128 ビットとなる $\mathbb{F}_{31}$ 上の Rainbow 署名を実装する．Rainbow 署名で用いるパラメータは，$(\mathbb{F}_{31}; 27, 26, 26)$ である [12]．文献 [12] において，Rainbow$(\mathbb{F}_{31}; 27, 26, 26)$ は 128 ビットの安全性を持つとされるが，文献 [12] の発表以降に提案された新しい攻撃方法により，安全性が低下する可能性がある．新しい攻撃方法として，たとえば文献 [13] がある．

秘密鍵および公開鍵の構成に必要な $\mathbb{K}$ の元の個数を鍵長と呼ぶ．鍵長は次のとおりである．

秘密鍵長　$m(m+1) + n(n+1) + \sum_{h=1}^{t} o_h \left(v_h o_h + v_h(v_h+1)/2 + v_{h+1} + 1\right)$

公開鍵長　$m(n+1)(n+2)/2$

表 1 に，秘密鍵および公開鍵の大きさを示す．$\mathbb{F}_{31}$ の 1 つの元は，5 ビットで表現できる．表 1 に示す大きさは，元の表現に必要なビット数を基に算出した理論値である．

表 1 秘密鍵および公開鍵の大きさ
Table 1 The size of the private key and the public key.

方式	署名長（ビット）	秘密鍵		公開鍵	
		鍵長[1]	大きさ（kB）	鍵長[1]	大きさ（kB）
Rainbow($\mathbb{F}_{31}$; 27, 26, 26)	395	113,674	69.4	168,480	102.9

[1] 鍵の構成に必要な $\mathbb{F}_{31}$ の元の個数である．

表 2 実装・計測に使用した計算機の性能
Table 2 Specifications of the computers.

端末	OS	CPU	コア数	RAM
PC	Windows 7 64 ビット	AMD Phenom II X6 1090T 3.2 GHz	6	4.0 GB
Nexus 7	Android 4.4	NVIDIA Tegra 3 T30L (Cortex-A9) Quad-core 1.3 GHz	4	1.0 GB

3. 実装手法

本章では，Web Workers を用いた Rainbow 署名の並列実装手法について説明する．

3.1 有限体の演算と実装環境

Rainbow 署名で用いる有限体 $\mathbb{F}_{31}$ は

$$\mathbb{F}_{31} = \{0, 1, \ldots, 30\}$$

とする．$\mathbb{F}_{31}$ 上の加算，減算および乗算はつど計算，逆元算は $a \in \mathbb{F}_{31}$ に対し，$a^{-1} = a^{29}$ として計算する．

実装・計測には，汎用 PC および Android タブレット端末の 2 つの計算機を使用する．Android タブレット端末は，Google 社の Nexus 7 を使用する．PC は 6 つのコアで構成される CPU を搭載し，Nexus 7 は 4 つのコアで構成される CPU を搭載する．**表 2** に，PC および Nexus 7 の性能を示す．また，JavaScript プログラムの実行環境として **表 3** に示すウェブブラウザを使用する．表 3 に示すウェブブラウザのうち，Google Chrome および Opera は同一の JavaScript エンジンである V8 を搭載する．JavaScript エンジンの Chakra および SpiderMonkey は，JavaScript コードをバイトコードへ変換し，インタプリタ方式で実行する．その後，実行中の状況に応じて，バイトコードをコンパイルしネイティブコードを生成する．V8 は，JavaScript コードを最初の実行時にコンパイルし，バイトコードへ変換することなくネイティブコードを生成する．

3.2 並列計算の性能指標

並列計算に関する性能指標について説明する [14]．以降の説明において，ある処理のうち，並列実行できる部分の実行時間の割合を r，使用する CPU のコア数を c とする．

高速化率 S を次の式で定める．

$$S = \frac{\text{逐次実行の所要時間}}{\text{並列実行の所要時間}}$$

高速化率は，逐次実行と比較して，並列実行がどの程度速くなったかを表す．

例 1 ある処理の逐次実行に必要となる時間が 1,000 ミリ秒であり，同じ処理を並列実行した際の所要時間が 200 ミリ秒であるとする．このときの高速化率は $S = 5$ である．

実行効率 E を次の式で定める．

$$E = \frac{S}{c}$$

実行効率は，計算機のリソースがどの程度消費されているかを表す．並列数 W の並列計算を行うとき，$W < c$ ならば $E = \frac{S}{c} < \frac{S}{W}$ である．

例 2 64 コアの CPU を用いて，53 倍の高速化率が得られたならば，実行効率は $E = 0.828$ である．これは，処理全体を平均して，実行時間の約 17%は CPU の全コアがアイドル状態になっているという意味である．

また，アムダールの法則 [15] を用いることにより，達成可能な高速化率の上限を知ることができる．アムダールの法則によれば

$$S \leq \frac{1}{(1-r) + \frac{r}{c}}$$

である．

表 3 実装・計測に使用したウェブブラウザとそのバージョン
Table 3 Specifications of the Web browsers.

ウェブブラウザ	JavaScript エンジン	バージョン	
		PC	Nexus 7
IE	Chakra	11.0.9600.16428	N/A[1]
Chrome	V8	31.0.1650.57	31.0.1650.59
Opera	V8	18.0.1284.49	18.0.1290.66961
Firefox	SpiderMonkey	25.0.1	25.0.1

[1] Android 版 Internet Explorer は提供されていない．

3.3 Web Workers

Web Workers の策定目的は，JavaScript の実行がシングルスレッドであることに起因するウェブブラウザのユーザインタフェーススレッド（UI スレッド）の応答性の低下を解消することである．Web Workers 登場以前は，JavaScript の実行環境はつねにシングルスレッドであり，JavaScript 上で並列計算を行うことはできなかった．しかし，Web

表 4 署名生成における演算の回数
Table 4 The number of operations in signature generation.

ステップ	加算	減算	乗算	逆元算
$\boldsymbol{a} = \mathcal{S}^{-1}(\boldsymbol{M})$	2,704	52	2,704	0
$\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$	105,898	65,416	219,648	52
$\boldsymbol{s} = \mathcal{T}^{-1}(\boldsymbol{b})$	6,241	79	6,241	0

Workers の登場により，JavaScript の実行環境に専用のバックグラウンドワーカを導入できるようになった．ウェブブラウザの UI スレッドとバックグラウンドワーカは，メッセージパッシング方式による通信を行うことができる．本稿における実装では，UI スレッドとバックグラウンドワーカ間の通信には postMessage 関数を使用する．この通信を用いて，UI スレッドからバックグラウンドワーカへデータを転送し，計算量の多い問題をバックグラウンドワーカ上で処理することができる．これにより，UI スレッドの応答性低下を解消することができる．また，Web Workers を用いれば，ウェブアプリケーションの開発者はバックグラウンドワーカをいくつでも生成できる．生成された各バックグラウンドワーカは，それぞれが独立したアドレス空間を持つスレッドとして独立に動作する．したがって，計算対象のデータを分割し，分割したデータを各バックグラウンドワーカへ転送することにより，ウェブブラウザ上で並列計算を行うことができる．

3.4 署名生成

Rainbow 署名の署名生成は，署名者の秘密鍵 $\mathcal{S}$，$\mathcal{F}$，$\mathcal{T}$ を用いて行う．HTML5 登場以前は，セキュリティ上の理由により，JavaScript プログラムからローカルストレージへアクセスすることはできなかった．しかし，HTML5 で新たに組み込まれた File API [16] を用いることにより，JavaScript プログラムからローカルストレージへアクセスすることができるようになった．本稿における実装では，JavaScript プログラムが署名者の秘密鍵を読み込むために，File API で定める FileReaderSync インタフェースをバックグラウンドワーカ上で使用する．

表 4 に，署名生成における演算回数を示す．表 4 より，Rainbow 署名の署名生成では，$\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$ の計算における乗算回数が，署名生成全体における乗算回数の約 96%を占めることが分かる．したがって，署名生成をより高速に行うには，$\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$ の計算を高速化する必要がある．しかし，$\boldsymbol{b} = \mathcal{F}^{-1}(\boldsymbol{a})$ の計算で現れる連立 1 次方程式は，Rainbow 署名を構成する各レイヤごとに存在し，各連立 1 次方程式は前のレイヤの連立 1 次方程式に依存している．したがって，これらの連立 1 次方程式を並列に解くことはできない．また，本実装では，連立 1 次方程式の求解アルゴリズムとしてガウスの消去法を採用した．本稿で用いるパラメータの場合には，連立 1 次方程式の係数行列の大きさは 26×26 であり，ガウスの消去法を逐次実行することで十分高速に解ける．ガウスの消去法を並列化する場合においても，係数行列の大きさが小さいため，並列計算することにより削減できる実行時間の割合も小さく，また並列計算を行う各バックグラウンドワーカと UI スレッド間の通信により，オーバヘッドが発生する．以上の 2 つの理由により，ガウスの消去法を並列化することによる実行時間の削減は期待できないため，本稿では署名生成の並列化については実装していない．

3.5 署名検証

Rainbow 署名の署名検証は，署名者の公開鍵 $\mathcal{P}$ を用いて行う．$\boldsymbol{M} \in \mathbb{K}^m$ を署名対象のメッセージとし，$\boldsymbol{s} \in \mathbb{K}^n$ を $\boldsymbol{M}$ に対する署名とする．署名の受信者は，署名者の公開鍵 $\mathcal{P}$ に $\boldsymbol{s}$ を代入し，$\mathcal{P}(\boldsymbol{s})$ が $\boldsymbol{M}$ と一致するか否かを調べることにより署名検証を行う．$\mathcal{P}(\boldsymbol{s})$ は，m 個の n 変数多項式 $p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ へそれぞれ $\boldsymbol{s}$ を代入することにより求められる．$p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ は互いに干渉しないため，$p_k(\boldsymbol{s})$ $(v_1 + 1 \leq k \leq n)$ を独立に計算することができる．本稿では，JavaScript プログラム上において Web Workers で定めるバックグラウンドワーカを用いて，$\mathcal{P}(\boldsymbol{s})$ の並列計算を実装する．

並列計算は，複数のバックグラウンドワーカが同時に $p_k(\boldsymbol{s})$ $(v_1 + 1 \leq k \leq n)$ を計算することにより実現する．$p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ は

$$p_k(\boldsymbol{x}) = \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(k)} x_i x_j + \sum_{i=1}^{n} p_i^{(k)} x_i + p_0^{(k)}$$

である．ここで，$p_{ij}^{(k)}, p_i^{(k)}, p_0^{(k)} \in \mathbb{K}$ である．$p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ の 2 次部分における $x_i x_j$ の乗算は，一度計算した値を使い回すことができる．したがって，各バックグラウンドワーカへの $p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ の分割を工夫することにより，乗算の回数を削減することができる．本稿で用いるパラメータの場合には，各 $p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ について単項式 $p_{ij}^{(k)} x_i x_j$ が 3,160 個現れる．並列計算に W 個のバックグラウンドワーカを用いる場合，各バックグラウンドワーカに番号 $w_p = p$ $(p = 1, 2, \ldots, W)$ を割り当てる．各バックグラウンドワーカが計算する i の範囲を得るために，関数 $i(p)$ を次のように定める．

$$i(p) = \left\lfloor \frac{2n + 1 - \sqrt{(2n+1)^2 - 8 \cdot 3160 \cdot \frac{p-1}{W}}}{2} \right\rfloor$$

ここで，$\lfloor\ \rfloor$ は床関数である．各バックグラウンドワーカは，$p_k(\boldsymbol{x})$ $(v_1 + 1 \leq k \leq n)$ に現れる 2 次部分については

$$\sum_{i=i(w_p)+1}^{i(w_p+1)} \sum_{j=i}^{n} p_{ij}^{(k)} x_i x_j$$

表 5 Rainbow 署名の署名生成 1 回の実行時間（ミリ秒）
Table 5 Running time of the signature generation.

端末	IE	Chrome	Opera	Firefox
PC	10.60	3.70	3.59	4.03
Nexus 7	N/A	19.75	21.62	21.57

アルゴリズム 1 Rainbow 署名の署名検証の実行時間計測

```
入力： 公開鍵 P
出力： 署名検証の実行時間
 1: N ← 並列数の最大値
 2: L ← ランダムなメッセージの個数
 3: M ← L 組のランダムなメッセージ
 4: S ← M に対する，L 組の署名
 5: for W = 1 to N do
 6:    バックグラウンドワーカを W 個生成する．
 7:    t0 ← 時刻
 8:    W 個のバックグラウンドワーカへ P, S を転送する．
 9:    for i = 1 to L do
10:       W 個のバックグラウンドワーカを用いて，P(Si) を計算する．
11:    end for
12:    UI スレッドへ P(Si) (i ∈ {1, 2, ..., L}) を転送する．
13:    t1 ← 時刻
14:    if P(Si) = Mi (i ∈ {1, 2, ..., L}) then
15:       t ← t1 − t0
16:       並列数 W，署名の個数 L のときの実行時間として t を出力する．
17:    else
18:       エラーを出力して終了する．
19:    end if
20: end for
```

を計算する．このように $p_k(\boldsymbol{x})$ $(v_1+1 \le k \le n)$ の計算を分割することにより，乗算表を用いることなく，効率的に $\mathcal{P}(\boldsymbol{s})$ を計算することができる．この場合，$\mathcal{P}(\boldsymbol{s})$ の計算全体で必要な乗算の回数は 171,588 回である．$x_i x_j$ の値を使い回さない場合は，$\mathcal{P}(\boldsymbol{s})$ の計算全体で 332,748 回の乗算が必要であるから，この方法を用いることで約 48%の乗算を削減することができる．

また，Rainbow 署名の署名検証における $\boldsymbol{M'} = \mathcal{P}(\boldsymbol{s})$ を計算するステップと，$\boldsymbol{M}$ と $\boldsymbol{M'}$ が等しいかを判定するステップのうち，後者の判定に必要な時間は，$\boldsymbol{M'}$ の計算時間と比較して十分に小さい．そのため，Rainbow 署名の署名検証においては，$r = 1$ と考えることができる．したがって，アムダールの法則より $S \le c$ となり，Rainbow 署名の署名検証における高速化率の上限は，使用する CPU のコア数と等しい．

4. 実装結果

本章では，Web Workers を用いた Rainbow 署名の並列計算の実装結果について説明する．

4.1 実行時間

表 5 に，Rainbow 署名の署名生成の実行時間を示す．**表 6** に，Rainbow 署名の署名検証の実行時間，高速化率および実行効率を示す．表 6 に示す実行時間は，アルゴリズム 1 を用いて計測した．アルゴリズム 1 において，メッセージの個数を $L = 1{,}000$ とし，PC 上では並列数の最大値を $N = 6$，Nexus 7 上では $N = 4$ として Rainbow 署名の署名検証の実行時間を計測した．計測した実行時間を基に，1 回の署名検証に必要な時間を算出した．

表 6 に示す結果より，Rainbow 署名の署名検証は，PC 上においては並列数 6 のとき，Nexus 7 上においては並列数 4 のとき最も高速に動作することが分かる．特に，PC 上においては，Google Chrome，Opera および Firefox を使用した場合は，1 回の署名検証を 1 ミリ秒未満で完了している．Nexus 7 については，最も高速に署名検証を行うことができたウェブブラウザは Google Chrome であり，その実行時間は 4.54 ミリ秒であった．表 6 に示す高速化率および実行効率に着目すると，PC 上においては，並列数 6 のとき，およそ 5 倍程度の高速化率であり，Nexus 7 上においては，並列数 4 のとき，およそ 4 倍程度の高速化率を達成していることが分かる．

全体を通して見ると，PC 上と Nexus 7 上の両方において，Google Chrome と Opera が同様の挙動を示していることが分かる．これは，両ウェブブラウザの使用する JavaScript エンジンが，同一の V8 であるからだと考えられる．また，Rainbow 署名の署名生成および署名検証を行うにあたって，高速に動作した順にウェブブラウザを並べると Google Chrome，Opera，Firefox，Internet Explorer であることが分かる．Firefox および Internet Explorer に搭載されている JavaScript エンジンの SpiderMonkey および Chakra は，JavaScript コードをバイトコードへ変換しインタプリタ方式で実行する．インタプリタ方式で実行したのちに，実行状況のプロファイリングを行い，必要であればバイトコードをコンパイルし，ネイティブコードを生成する．それに比べて，Google Chrome および Opera に搭載されている JavaScript エンジンの V8 は，JavaScript コードを最初の実行時にコンパイルし，バイトコードへ変換することなくネイティブコードを生成する．したがって，Google Chrome および Opera を用いた場合の実行速度が最も高速になったと考えられる．

4.2 RSA 署名との比較

Rainbow 署名の実行時間と比較するために，3,072 ビット RSA 署名を，JavaScript を用いて実装した．3,072 ビット RSA 署名の安全性は，本稿で実装した Rainbow 署名と同一の 128 ビットである [17]．RSA 署名においては多倍長整数を扱う必要があるが，JavaScript は標準の多倍長演算機能を提供していない．本稿における実装では，Leemon Baird による多倍長演算ライブラリを使用する [18]．**表 7** に，RSA 署名の実行時間を示す．表 7 に示す結果より，署名生成については，Rainbow 署名の方が約 50 倍高速であ

表 6 Rainbow 署名の署名検証 1 回の実行時間と高速化率 S および実行効率 E
Table 6 Running time, speedup and efficiency of the signature verification.

端末	並列数	実行時間（ミリ秒）				IE		Chrome		Opera		Firefox	
		IE	Chrome	Opera	Firefox	S	E	S	E	S	E	S	E
PC	1	7.17	3.20	3.18	3.91	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	2	4.05	1.62	1.61	2.03	1.77	0.295	1.98	0.330	1.98	0.330	1.93	0.322
	3	2.85	1.12	1.12	1.45	2.52	0.420	2.86	0.477	2.84	0.473	2.70	0.450
	4	1.81	0.90	0.92	1.15	3.96	0.660	3.56	0.593	3.46	0.577	3.40	0.567
	5	1.63	0.75	0.75	0.94	4.40	0.733	4.27	0.712	4.24	0.707	4.16	0.693
	6	1.49	0.62	0.64	0.80	4.81	0.802	5.16	0.860	4.97	0.828	4.89	0.815
Nexus 7	1	N/A	16.33	16.68	20.95	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	2	N/A	8.53	8.55	10.69	N/A	N/A	1.91	0.478	1.95	0.488	1.96	0.490
	3	N/A	5.92	5.90	7.43	N/A	N/A	2.76	0.690	2.83	0.708	2.82	0.705
	4	N/A	4.54	4.56	6.35	N/A	N/A	3.60	0.900	3.66	0.915	3.30	0.825

表 7 RSA 署名の実行時間（ミリ秒）
Table 7 Running time of RSA.

	端末	IE	Chrome	Opera	Firefox
署名生成[1]	PC	304.87	202.19	234.10	203.12
	Nexus 7	N/A	1,307.39	1,341.08	1,305.31
署名検証[2]	PC	39.03	31.03	31.27	25.87
	Nexus 7	N/A	57.61	58.75	68.59

[1] 署名生成は，中国剰余定理を用いて指数を分割し，2 並列で行う．
[2] 暗号化指数は $e = 2^{16} + 1$ である．

表 8 ネイティブ実装した Rainbow 署名の実行時間（ミリ秒）
Table 8 Running time of the Rainbow with native code.

実装方法	PC		Nexus 7	
	署名生成	署名検証[1]	署名生成	署名検証[1]
ネイティブ	1.91	2.12	19.05	17.10
JavaScript	3.59[2]	3.18[2]	19.75[2]	16.33[2]

[1] 並列化せずに行った場合の実行時間である．
[2] Opera および Google Chrome を用いた場合の実行時間である（表 5 および表 6）．

り，署名検証については，Rainbow 署名の方が PC 上で約 40 倍，Nexus 7 上で約 10 倍高速であった．

4.3 ネイティブ実装との比較

JavaScript プログラムが，どの程度高速なのかを調べるために，Rainbow 署名をネイティブ実装した．**表 8** に，ネイティブ実装した場合の Rainbow 署名の実行時間を示す．表 8 に示す実行時間は，Rainbow 署名の署名生成および署名検証を並列化せずに行った場合の実行時間である．ネイティブ実装は，JavaScript コードを C++コードへ移植することで行い，SIMD 演算は使用しない．JavaScript コードおよび C++コードにおけるアルゴリズムは同一のものを使用する．C++コードのコンパイルには，Microsoft Visual Studio Express 2012 for Windows Desktop および Android NDK, Revision 9b に含まれるコンパイラを使用する．表 8 に示す結果より，Nexus 7 上においては，署名生成および署名検証ともに，ネイティブ実装した Rainbow 署名と JavaScript 実装した Rainbow 署名の実行時間は同程度である．PC 上においては，署名生成については，ネイティブ実装した Rainbow 署名の方が約 1.88 倍高速であり，署名検証については，ネイティブ実装した Rainbow 署名の方が 1.50 倍高速であった．

5. 応用例

JavaScript を用いた Rainbow 署名の応用例について述べる．JavaScript を用いてウェブブラウザ上に実装した Rainbow 署名は，他のディジタル署名と同様に，メッセージの送信者の認証と，メッセージの改ざん検出機能を提供する．

5.1 ブロードキャスティングサービス

ここでは，YouTube 等のブロードキャスティングサービスから配信される映像を，ウェブブラウザ上でリアルタイムに閲覧できるウェブアプリケーションを考える．映像の配信にあたり，配信者の認証および映像の改ざん防止を行いたい場合は，映像に対し，Rainbow 署名を用いて署名を付与すればよい．

具体的な応用例として，インターネット選挙運動を考える．2013 年 4 月 26 日に公職選挙法の一部を改正する法律が公布され，2013 年 5 月 26 日に施行された．公職選挙法の改正は，インターネット等を利用する方法による選挙運動を解禁することを目的として行われた．インターネット選挙運動解禁にともなう改正公職選挙法第 142 条の 3 第 1 項により，ウェブサイト等を利用する方法による選挙運動用文書図画の頒布が解禁された．これにより，何人も，ウェブサイト等を利用する方法により，選挙運動を行うことが可能となった．したがって，立候補者は，自身のウェブサイトやブロードキャスティングサービス等を利用した政見放送を行うことができる．このようなインターネット

を通じた政見放送を行う場合，政見放送の送信者を認証し，政見放送の内容が改ざんされていないことを保証する必要がある．政見放送の送信にあたり，送信者は，Rainbow 署名を用いて署名を生成したうえで，映像と署名を同時に送信する．政見放送の閲覧者は，ウェブアプリケーションにアクセスした際，送信者の公開鍵が記録されたローカルストレージ上のファイルを File API により指定し，閲覧者の使用するウェブブラウザ上で署名検証を行うことで，送信者の詐称や内容の改ざんを検出することが可能となる．なお，送信者の公開鍵については，PKI と同様の仕組みにより，その正当性を確認する必要がある．

インターネット上のディジタルコンテンツを保護する仕組みである DRM は，コピーを制限することでディジタルコンテンツの再頒布を禁止することができる．しかし，DRM は，コピーを制限するために特定のソフトウェアやハードウェアに依存している．また，SSL/TLS 通信を用いることにより，通信相手であるウェブサーバの正当性を保証し，ウェブサーバから送信されたディジタルコンテンツが，通信経路において改ざんされていないことを保証することができるが，ディジタルコンテンツ自体の正当性（閲覧者の意図する送信者により作成され，署名付与時点から改ざんされていないこと）については何ら保証されない．一方，JavaScript を用いた Rainbow 署名では，ディジタルコンテンツの作成者が，ディジタルコンテンツに対しディジタル署名を直接施すことにより，ディジタルコンテンツ自体の正当性を保証することが可能となる．さらに，インターネット選挙運動における政見放送という応用例においては，各立候補者のウェブサイト等を一次配布元として，第三者が任意の方法で二次配布することが可能となる．この場合，政見放送自体にディジタル署名が施されているため，二次配布の方法がどのようなものであっても政見放送の正当性は保証され，より広く配布することができるという利点がある．したがって，インターネット選挙運動等の，ディジタルコンテンツの再頒布が問題とならない場合は，DRM および SSL/TLS 通信と比較してより軽量かつ柔軟な手段として JavaScript 実装の Rainbow 署名を利用することが可能である．

実際にブロードキャスティングサービス等から映像を配信する場合は，映像の各フレーム等の小さなデータに対し署名を付与し，連続的に送信することになる．**表 9** に，各ウェブブラウザにおいて 1 秒間に実行可能な Rainbow 署名の署名検証の回数を示す．一般に，映像は，毎秒 15 フレーム，30 フレーム，60 フレーム等の FPS（Frames Per Second）で撮影される．したがって，たとえば映像の各フレームに対して署名を付与し送信する場合は，閲覧者のウェブブラウザにおいて，配信される映像の FPS と同じ回数だけ 1 秒間に署名検証を行う必要がある．しかし，表 9 に示すとおり，Nexus 7 において，署名検証の並列計算を行わない場合は，Google Chrome および Opera を用いたときで毎秒 60 回程度，Firefox を用いたときは毎秒 47 回しか署名検証を行うことができない．これではウェブアプリケーション上で映像をリアルタイムに閲覧することはできない．Web Workers を用いて署名検証を並列に行う場合は，Google Chrome および Opera を用いたときで毎秒 220 回程度，Firefox を用いたときにおいても毎秒 157 回の署名検証を行うことができる．したがって，ウェブアプリケーション上においても，映像をリアルタイムに閲覧することが可能となる．

表 9 1 秒間に実行可能な Rainbow 署名の署名検証の回数
Table 9 The number of signature verifications.

ウェブブラウザ	PC		Nexus 7	
	非並列	6 並列	非並列	4 並列
Internet Explorer	139	671	N/A	N/A
Google Chrome	312	1,612	61	220
Opera	314	1,562	59	219
Firefox	255	1,250	47	157

5.2 署名付きリクエスト

ウェブアプリケーションとウェブサーバ間の通信に関する応用例をあげる．あるサービスについてのウェブアプリケーションが，サービス提供元のウェブサーバと双方向通信を行い，ウェブサーバからのコマンドを基にその振舞いを決定する場合を考える．この場合，ウェブアプリケーションが受信したコマンドが，正規のウェブサーバから送信されたものであり，内容が改ざんされていないことが重要である．コマンドの送受信にあたり，SSL/TLS 通信を用いる場合は，送信者の認証および通信内容の改ざん検出に加え，通信内容を暗号化することができる．しかし，コマンドのみを送受信する場合においては，通信内容の盗聴対策は重要ではない．したがって，この場合は送信者の認証および通信内容の改ざん検出を行うことができれば十分であり，署名付きリクエストという方法を利用することでこれを実現できる．署名付きリクエストとは，ウェブアプリケーションおよびウェブサーバがコマンドを送信する際に，コマンドの内容から生成される署名を同時に送信することにより，その正当性を示す方法である．たとえば，Facebook は，開発者が Facebook と連動したウェブアプリケーションを構築する場合に，PHP 等のサーバサイドスクリプト言語を利用したメッセージ認証符号による署名付きリクエストを提供している [19]．ここでは，認証の手段として，広く普及している PHP の hash_hmac 関数を利用した方法を想定する．hash_hmac 関数を利用した認証は，サービス提供元による署名生成とウェブアプリケーションによる署名検証において，同一の鍵を用いる共通鍵暗号方式である．したがって，ウェブアプリケーションの開発

者は，鍵を事前に共有する必要があり，またウェブアプリケーションの運用にあたっては，鍵を安全に管理する必要が生じる．実際にウェブサーバ（CPU: Intel Xeon E5620 2.40 GHz，RAM: 2.0 GB，Apache HTTP Server 1.3，PHP 5.2.5）上において，hash_hmac 関数による SHA-256 アルゴリズムを用いたハッシュ生成の実行時間を計測したところ，約 15 マイクロ秒であった．これは，Rainbow 署名より 100 倍程度高速である．しかし，Rainbow 署名による署名検証は 1 ミリ秒程度で行うことができるため，ウェブアプリケーションの運用にあたって，この遅延が許される場合には，hash_hmac 関数による認証を Rainbow 署名を用いた認証に置き換えることができる．その場合は，ウェブアプリケーションの開発者は，ウェブサーバの公開鍵のみを用いてコマンドを認証することができ，また鍵を安全に管理する必要もない．さらに，hash_hmac 関数は，PHP がインストールされたウェブサーバでしか使用できず，また他のサーバサイドスクリプト言語を利用する場合も特定の実行環境に依存したものとなる．しかし，JavaScript 実装の Rainbow 署名にはそのような制限はなく，すべての処理がウェブブラウザ内で完結する．なお，SSL/TLS 通信において，暗号化を行わず認証のみを行う方法として，SSL_RSA_WITH_NULL_SHA 等を指定する方法がある．SSL_RSA_WITH_NULL_SHA 等を指定し認証のみを行う場合においても，ウェブサーバは，クライアントからの接続要求がある度に認証処理を実行する必要がある．一方，署名付きリクエストという応用例においては，認証処理の実行環境を，ウェブサーバからウェブブラウザへ移すことができるという特徴がある．この特徴により，ウェブサーバはディジタル署名が施されたコマンドを送信するだけで，SSL_RSA_WITH_NULL_SHA 等を指定した SSL/TLS 通信と同等の認証機能をクライアントへ提供することができ，SSL/TLS 通信の認証処理を行わないため，ウェブサーバの負荷を軽減することが可能となる．

6. まとめ

本稿では，安全性が 128 ビットとなる Rainbow 署名を，JavaScript 言語を用いてウェブブラウザ上に実装した．Rainbow 署名の署名生成においては，File API を用いることにより，JavaScript プログラムからユーザのローカルストレージ上に保存されている秘密鍵へアクセスすることができるようになった．また，JavaScript 上で並列計算を行うための新しい規格である Web Workers を用いて，Rainbow 署名の署名検証を並列実装した．実装は，汎用 PC および Android タブレット端末上で行った．汎用 PC は 6 つのコアで構成される CPU を搭載し，Android タブレット端末は 4 つのコアで構成される CPU を搭載する．これらの端末上で，Rainbow 署名の実行時間を計測し，並列計算を行った場合の高速化率および実行効率を算出した．その結果，汎用 PC 上においては，Rainbow 署名の署名検証は並列数が 6 のとき最も高速に動作し，そのときの実行時間は 0.62 ミリ秒であった．Android タブレット端末上においては，並列数が 4 のとき最も高速に動作し，そのときの実行時間は 4.54 ミリ秒であった．

参考文献

[1] W3C: HTML5, W3C (online), available from ⟨http://www.w3.org/TR/html5/⟩ (accessed 2013-11-29).

[2] Ecma International: Standard ECMA-262, Ecma International (online), available from ⟨http://www.ecma-international.org/publications/standards/Ecma-262.htm⟩ (accessed 2013-11-29).

[3] W3C: Web Workers, W3C (online), available from ⟨http://www.w3.org/TR/workers/⟩ (accessed 2013-11-29).

[4] Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, *SIAM Journal on Computing*, Vol.26, pp.1484–1509 (1997).

[5] Garey, M.R. and Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company (1979).

[6] Bernstein, D.J., Buchmann, J. and Dahmen, E. (Eds.): Post-Quantum Cryptography, Ding, J. and Yang, B.: *Multivariate Public Key Cryptography*, pp.193–241, Springer Berlin Heidelberg (2009).

[7] Matsumoto, T. and Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption, *EUROCRYPT 1988, LNCS*, Vol.330, pp.419–453 (1988).

[8] Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms, *EUROCRYPT 1996, LNCS*, Vol.1070, pp.33–48 (1996).

[9] Kipnis, A., Patarin, J. and Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes, *EUROCRYPT 1999, LNCS*, Vol.1592, pp.206–222 (1999).

[10] Ding, J. and Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme, *ACNS 2005, LNCS*, Vol.3531, pp.164–175 (2005).

[11] Chen, A.I.-T., Chen, M.-S., Chen, T.-R., Cheng, C.-M., Ding, J., Kuo, E.L.-H., Lee, F.Y.-S. and Yang, B.-Y.: SSE Implementation of Multivariate PKCs on Modern x86 CPUs, *CHES 2009, LNCS*, Vol.5747, pp.33–48 (2009).

[12] Petzoldt, A., Bulygin, S. and Buchmann, J.: Selecting Parameters for the Rainbow Signature Scheme, *PQCrypto 2010, LNCS*, Vol.6061, pp.218–240 (2010).

[13] Thomae, E.: A Generalization of the Rainbow Band Separation Attack and its Applications to Multivariate Schemes, Cryptology ePrint Archive (online), available from ⟨http://eprint.iacr.org/2012/223.pdf⟩ (accessed 2014-03-10).

[14] Breshears, C.（著），千住治郎（訳）：並行コンピューティング技法，オライリー・ジャパン (2009).

[15] Amdahl, G.M.: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, *Proc. AFIPS Conference*, Vol.30, pp.483–485 (1967).

[16] W3C: File API, W3C (online), available from ⟨http://www.w3.org/TR/file-upload/⟩ (accessed 2013-11-29).

[17] Barker, E., Barker, W., Burr, W., Polk, W. and Smid,

M.: *NIST Special Publication 800-57 Recommendation for Key Management — Part 1: General* (*Revision 3*), NIST (2012).
[18] Baird, L.: Big Integers in JavaScript, Leemon Baird's Home Page (online), available from ⟨http://www.leemon.com/crypto/BigInt.html⟩ (accessed 2013-11-29).
[19] Facebook: Login for Games on Facebook, Facebook developers (online), available from ⟨https://developers.facebook.com/docs/facebook-login/using-login-with-games/⟩ (accessed 2013-11-29).

推薦文

本稿は，Rainbow 署名を様々なプラットフォームで実装・評価しており，今後の Rainbow 署名の可能性について示唆した論文である．実際に，著者らは Rainbow 署名方式の並列化実装をウェブブラウザベースのプログラミングを用いて評価を行い，その結果，PC とタブレット端末で異なる傾向の結果を示している．それだけでなく当該署名の応用例も検討しており，今後のシステム応用に貢献できるものと判断できる．以上の結果から，このような先端的な暗号技術について，実システムでの導入を見据えた綿密な実装実験を行っている点を評価できる．よって，今後の同研究分野の発展に有用であるため，推薦論文として推薦したい．

（コンピュータセキュリティシンポジウム 2013
プログラム委員長　越前　功）

鷲見 拓哉

2012 年九州大学理学部数学科卒業．2014 年同大学大学院数理学府修士課程数理学専攻修了．2013 年情報処理学会コンピュータセキュリティシンポジウム CSS2013 学生論文賞受賞．

石黒 司

2008 年公立はこだて未来大学システム情報科学部卒業．2010 年情報セキュリティ大学院大学情報セキュリティ専攻博士前期課程修了．同年 KDDI（株）入社．現在，（株）KDDI 研究所情報セキュリティグループ研究員．暗号プロトコル，暗号の安全性解析等の研究に従事．

清本 晋作　（正会員）

2000 年筑波大学工学研究科物質工学専攻博士前期課程修了．同年 KDD（株）入社．現在，（株）KDDI 研究所情報セキュリティグループ主任研究員．ストリーム暗号，暗号プロトコル，モバイルセキュリティ，プライバシ保護技術等の研究に従事．日本物理学会会員．2008～2009 年，London 大学 Royal Holloway 校客員研究員．2004 年，電子情報通信学会学術奨励賞受賞．博士（工学）．

三宅 優　（正会員）

1988 年慶應義塾大学理工学部電気工学科卒業．1990 年同大学大学院修士課程修了．2009 年電気通信大学大学院博士課程修了．現在，（株）KDDI 研究所情報セキュリティグループリーダー．高速通信プロトコルの実装，インターネットアクセス，インターネットセキュリティの研究に従事．1989 年度電気・電子情報技術振興財団猪瀬学術奨励賞，1995 年度情報処理学会学術奨励賞，2003 年電波産業会電波功績賞，2009 年日本 ITU 協会活動奨励賞受賞．

小林 透　（正会員）

1985 年東北大学工学部精密機械卒業．1987 年同大学大学院工学研究科修士課程修了．同年 NTT 入社．以来，ソフトウェア生産技術，情報セキュリティ，データマイニング，Web 技術等の研究開発に従事．2013 年から長崎大学大学院工学研究科教授．IEEE，電子情報通信学会（シニア）各会員，博士（工学）．

高木 剛 （正会員）

1993 年名古屋大学理学部数学科卒業．1995 年同大学大学院理学研究科修士課程修了．同年日本電信電話株式会社入社．2001 年理学博士（ダルムシュタット工科大学）．その後，ダルムシュタット工科大学准教授，公立はこだて未来大学教授を経て，現在に至る．第 8 回船井情報科学振興賞，2012 年度情報処理学会喜安記念業績賞受賞．暗号および情報セキュリティに関する研究に従事．電子情報通信学会，日本数学会各会員．