# 処理能力の低いスクリプト言語と高速通信路を組み合わせるWebサービスのためのElGamal暗号の依頼計算

村田　純一†　　白石　善明†　　毛利　公美‡

名古屋工業大学†　　岐阜大学‡

## 1. はじめに

クラウドコンピューティングとは，インターネットで高速に接続された外部のITシステムを活用する仕組みのことで，ITシステムの処理能力やデータ容量に対する柔軟性が高いことで注目されている．また，サービス利用者はどこに情報が置かれていても，どこで処理されていても気にすることなくサービスを利用できるという利点がある．

本稿では，サービス利用端末同士が短時間でファイルを共有するクラウドサービスの一つとして，軽量な暗号通信サービスを想定し，その構築のためのElGamal暗号を用いた依頼計算を提案する．そして，提案方式のサービス利用端末側の処理をJavaScriptで，サービス提供サーバ側の処理をJavaServletで実装し，計算時間を計測し，提案方式は短時間でファイルを暗号化・復号できることを示す．

## 2. Webブラウザ上の暗号演算と依頼計算の必要性

JavaScriptは標準的なWebブラウザ上で手軽に利用できるスクリプト言語である．PCや携帯電話，スマートフォン，ゲーム機などのインターネットに接続するほぼすべてのサービス利用端末のWebブラウザで実装されている．

JavaScriptを用いてWebブラウザ上で暗号演算を汎用的に実現する研究[1]がある．JavaScriptの実行速度はネイティブな実行コードに比べて10～1000倍以上低速であるため暗号処理は時間かかる．そこで，本研究では高速ネットワークを介して一部の計算を外部の高速・高性能なサーバに委託する依頼計算を検討する．

## 3. WebサービスのためのElGamal暗号の依頼計算の提案

依頼計算とは，計算能力の小さい端末が計算能力の大きいサーバに計算を依頼する方法で，端末の秘密情報をサーバに隠しながら依頼できる．本稿では，ElGamal公開鍵暗号[2]をWebサービスで利用することを想定し，サービス利用端末にとって暗号化・復号処理に時間のかかる冪乗演算の一部の計算を，秘密情報を隠しながらサービス提供サーバに依頼する方式を提案する．送信者と受信者をサービス利用端末としたとき，提案方式は次のようになる．

【事前準備】

受信者は秘密鍵・公開鍵ペア $(sk, pk)=(x,(p,g,y))$ を生成し，公開鍵 $pk$ を公開する．

【暗号化処理（図1）】

①. 送信者は平文 $m\in Z_p$，乱数 $r\in Z_{p-1}$ を用意する．そして小さな乱数 $r_1, r_3$ と $r=r_1r_2 \pmod{p-1}$ かつ $p-1=r_2r_3$ となる $r_2$ を生成し，$r_2$ をサービス提供サーバに送信する．

②. サービス提供サーバは $r_2$ を受信した後，$r_2$ と受信者の公開鍵 $pk$ から中間暗号文 $Z_1=g^{r_2} \pmod{p}, Z_2=y^{r_2} \pmod{p}$ を計算し，$Z_1, Z_2$ を送信者に返信する．

③. 送信者は $Z_1, Z_2$ を受信した後，サービス提供サーバが $Z_1$ を $g^{r_2} \pmod{p}$ と計算していることを確認するために，$Z_1$ と受信者の公開鍵 $pk$ を用いて $Z_1^{r_3}\equiv 1 \pmod{p}$ を検算する．同様に $Z_2$ と受信者の公開鍵 $pk$ を用いて，サービス提供サーバが $Z_2$ を $y^{r_2} \pmod{p}$ と計算していることを確認するために，$Z_2^{r_3}\equiv 1 \pmod{p}$ を検算する．

④. 送信者は $Z_1, Z_2$ が正しく計算されていることを確認した後，$c_1=Z_1^{r_1} \pmod{p}$ と $c_2=mZ_2^{r_2} \pmod{p}$ を計算し，暗号文 $(c_1,c_2)=(g^r, my^r)$ をサービス提供サーバに送信する．

⑤. サービス提供サーバは $(c_1,c_2)$ を受信した後，それらを保管する．そして受信者から要求されたとき，$(c_1,c_2)$ を送信する．

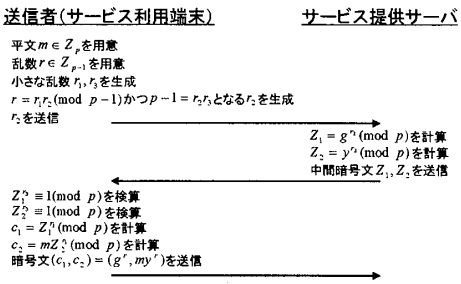


図1：送信者（サービス利用端末）とサービス提供サーバが行う提案方式の暗号化処理

【復号処理（図2）】

①. 受信者は $(c_1,c_2)$ を受信した後，$x'=p-1-x$ を計算する．そして小さな乱数 $r_1, r_3$ と $x'=r_1r_2 \pmod{p-1}$ かつ $p-1=r_2r_3$ となる $r_2$ を生成し，$r_2$ をサービス提供サーバに送信する．

②. サーバは $r_2$ を受信した後，$r_2$ と $c_1$，受信者の公開鍵 $pk$ から中間暗号文 $Z=c_1^{r_2} \pmod{p}$ を計算し，$Z$ を受信者に送信する．

③. 受信者は $Z$ を受信した後，サービス提供サーバが $Z$ を $c_1^{r_2} \pmod{p}$ と計算していることを確認するために，$Z^{r_3}\equiv 1 \pmod{p}$ を検算する．

④. 受信者は $Z$ が正しく計算されていることを確認した後，$m=c_2Z^{r_1} \pmod{p}$ を計算し，平文 $m$ を復号する．

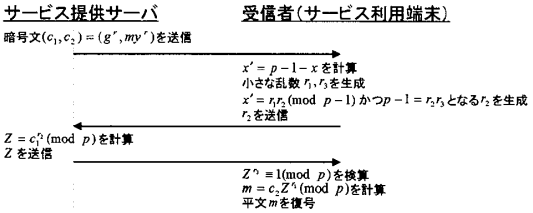


図2：サービス提供サーバと受信者（サービス利用端末）が行なう提案方式の復号処理

## 4. 安全性解析

### 4.1. 受動攻撃に対する提案方式の安全性

受動攻撃とは，提案方式が仕様通りに実行されるときに，サービス利用端末とサービス提供サーバが送信する情報を入手してサービス利用端末の秘密情報を求める攻撃方法である．以下に受動攻撃に対する提案方式の安全性を，暗号化処理と復号処理に分けて記述する．

---

A Server-Aided ElGamal Computation Protocol for Web Service combined with Scripting Language and High-Speed Network

†Jun-ichi MURATA and Yoshiaki SHIRAISHI・Nagoya Institute of Technology

‡Masami MOHRI・Gifu University

【受動攻撃に対する提案方式の暗号化処理の安全性】
提案方式の暗号化処理に対して受動攻撃を行う攻撃者が，送信者の秘密情報である乱数 $r$ と平文 $m$ を求める場合を考える．このとき攻撃者は $pk$ と $Z_1 = g^{r_2} \pmod p$，$c_1 = g^r \pmod p$ から $r_1$ を求められれば $r = r_1 r_2 \pmod{p-1}$ を入手でき，さらに $r$ と $c_2 = my^r \pmod p$ から $m$ を入手できる．しかし受動攻撃を行う攻撃者は $g^r = (Z_1^{r_2})^{r_1} \pmod p$ から $r_1$ を求める離散対数問題を解く必要があり，実際には $r$ を入手できない．したがって，提案方式の暗号化処理は受動攻撃に対して安全である．
【受動攻撃に対する提案方式の復号処理の安全性】
提案方式の復号処理に対して受動攻撃を行う攻撃者が，受信者の秘密情報である秘密鍵 $x$ を求める場合を考える．このとき攻撃者は何らかの方法で入手した平文 $m$ と $pk$，$Z = c_1^{r_2} \pmod p$，$(c_1, c_2) = (g^r, my^r)$ から $x' = p - 1 - x = r_1 r_2 \pmod{p-1}$ を求められれば，$x$ を入手できる．しかし受動攻撃を行う攻撃者は $c_2 Z^{r_1} = c_2 (c_1^{r_2})^{r_1} \pmod p$ から $r_1$ を求める離散対数問題を解く必要があり，実際には $x$ を入手できない．したがって，提案方式の復号処理は受動攻撃に対して安全である．
以上のことから，受動攻撃に対して提案方式は安全である．

4. 2. 能動攻撃に対する提案方式の安全性
能動攻撃とは，提案方式を仕様通りに実行しないようなサービス提供サーバが攻撃者であるときに，サービス利用端末の秘密情報を求めるのに都合の良い値を返信し，サービス利用端末の秘密情報を求める攻撃方法である．以下に能動攻撃に対する提案方式の安全性を暗号化処理と復号処理に分けて記述する．
【能動攻撃に対する提案方式の暗号化処理の安全性】
提案方式の暗号化処理に対して能動攻撃を行う攻撃者は $r_2$ を受信した後，送信者の秘密情報を求めるのに都合の良い値である $Z_1', Z_2'$ を生成してサービス利用端末に返信する．しかし，送信者は次の Fermat の定理を利用して $(Z_1')^{r_3} \equiv 1 \pmod p$ と $(Z_2')^{r_3} \equiv 1 \pmod p$ を検算することで，$Z_1', Z_2'$ が正しく計算された値かを効率良く検証できる．

**[Fermat の定理]** $p$ が素数のとき，任意の $a \in Z_p^*$ に対し，

$$a^{p-1} \equiv 1 \pmod p$$

が成り立つ． □

したがって，提案方式の暗号化処理は，能動攻撃を行うサービス提供サーバに対して安全である．
【能動攻撃に対する提案方式の復号処理の安全性】
提案方式の復号処理に対して能動攻撃を行う攻撃者は $r_2$ を受信した後，受信者の秘密情報を求めるのに都合の良い値である $Z'$ を生成してサービス利用端末に返信する．しかし受信者は Fermat の定理を利用して $(Z')^{r_2} \equiv 1 \pmod p$ を検算することで，$Z'$ が正しく計算された値かを効率良く検証できる．
したがって，提案方式の復号処理は，能動攻撃を行うサービス提供サーバに対して安全である．
以上のことから，能動攻撃に対して提案方式は安全である．

## 5. 効率
サービス利用端末のみで ElGamal 暗号を計算する方式と提案方式の効率を比較する．効率の比較には，サービス利用端末とサービス提供サーバの冪乗剰余演算回数と，サービス利用端末とサービス提供サーバ間の通信量を用いる．
サービス利用端末のみで ElGamal 暗号を計算する方式の効率は表 1 になる．冪乗剰余演算は，暗号化処理のサービス利用端末が 2 回で，復号処理のサービス利用端末が 1 回である．通信量は暗号化・復号処理の両方でゼロである．

表 1：サービス利用端末のみで ElGamal 暗号を計算する方式の効率

| | 冪乗剰余演算 | | 通信量 |
|---|---|---|---|
| | サービス利用端末 | サービス提供サーバ | |
| 暗号化処理 | 2 | 0 | 0 |
| 復号処理 | 1 | 0 | 0 |

提案方式の効率は表 2 になる．冪乗剰余演算は，暗号化処理のサービス利用端末が 3 回とサービス提供サーバが 2 回で，復号処理のサービス利用端末が 2 回とサービス提供サーバが 1 回である．通信量は受信者の公開鍵 p のサイズ|p|に依存し，暗号化処理で 3|p|で，復号処理で 2|p|となる．

表 2：提案方式の効率

| | 冪乗剰余演算 | | 通信量 |
|---|---|---|---|
| | サービス利用端末 | サービス提供サーバ | |
| 暗号化処理 | 3 | 2 | 3\|p\| |
| 復号処理 | 2 | 1 | 2\|p\| |

提案方式は ElGamal 暗号の一部の冪乗剰余演算の依頼に加えて，サービス提供サーバの能動攻撃を防ぐための検算をしている．そのためサービス利用端末のみで ElGamal 暗号を計算する方法と比べて，（1）サービス提供サーバが冪乗剰余演算をしている，（2）サービス利用端末とサービス提供サーバが通信をしている，（3）サービス利用端末の冪乗剰余演算回数が 1 回増えている．
しかし，（i）サービス提供サーバは，サービス利用端末よりもスペックが高く，JavaScript よりも高速な JavaServlet で処理を行う，（ii）高速ネットワークを介して送受信する，（iii）サービス利用端末が行う冪乗剰余演算の指数のサイズが小さいので，提案方式の方が短時間で ElGamal 暗号を計算できる，以上の 3 点で表 2 のように見かけ上は重たくなった処理はより短い時間で完了する．
サービス利用端末のみで ElGamal 暗号を計算する方式と提案方式の実測値を求めた（表 3，4）．計測に用いたサービス利用端末のスペックは，OS：AndroidOS1.5，CPU：Marvell Manahans PXA310，CPU 周波数：624MHz，RAM：128MB で，サービス提供サーバのスペックは OS：CentOS5.4，CPU：Pentium(R) 4，CPU 周波数：3,2GHz，RAM：2GB である．提案方式のサービス利用端末とサービス提供サーバが行う冪乗剰余演算の指数のサイズの組を $(|r_1|, |r_2|)$ = （128bit, 896bit），（256bit, 768bit），（512bit, 512bit）の場合と，サービス利用端末のみで ElGamal 暗号を計算する方式のサービス利用端末が行う冪乗剰余演算の指数サイズ$|r|$=1024bit の場合を比較すると，暗号化処理の場合はそれぞれ 5.9 倍，2.8 倍，1.4 倍，復号処理の場合はそれぞれ 4.5 倍，2.1 倍，1.1 倍の短い時間で計算できる．以上のように，提案方式はサービス利用端末のみで ElGamal 暗号を計算する方式よりも短い時間で暗号化・復号できることが確認された．

表 3：サービス利用端末のみで ElGamal 暗号を実行する方式の実測値

| | 指数のサイズ 端末 $\|r\|$ | 計算時間 | | |
|---|---|---|---|---|
| | | 利用端末 | サーバ | 合計 |
| 暗号化 | 1024bit | 237,568msec | 0msec | 237,568msec |
| 復号 | 1024bit | 118,784msec | 0msec | 118,784msec |

表 4：提案方式の実測値

| | 指数のサイズ $(\|r_1\|, \|r_2\|)$ | 計算時間 | | |
|---|---|---|---|---|
| | | 利用端末 | サーバ | 合計 |
| 暗号化 | (128bit,896bit) | 39,933msec | 56msec | 39,989msec |
| | (256bit,768bit) | 83,550msec | 50msec | 83,600msec |
| | (512bit,512bit) | 166,809msec | 34msec | 166,843msec |
| 復号 | (128bit,896bit) | 26,622msec | 28msec | 26,650msec |
| | (256bit,768bit) | 55,700msec | 25msec | 55,725msec |
| | (512bit,512bit) | 111,206msec | 17msec | 111,224msec |

## 6. まとめ
本研究では，処理能力の低いスクリプト言語と高速通信路を組み合わせる Web サービスのための ElGamal 暗号の依頼計算を提案した．安全性解析を行い，受動攻撃と能動攻撃の提案方式に対する安全性を評価した．効率の評価として実測値を求め，提案方式はサービス利用端末のみで ElGamal 暗号を計算する方式よりも短時間で暗号化・復号できることを示した．

## 参考文献
[1] 小田 哲，諸橋 玄武，“Web 暗号，”信学技報(IEICE Technical Report)，Vol.109，No.271.
[2] T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, ” IEEE Trans. on Inform. Theory, IT31, 4, pp.469-472, 1985.
[3] L. Baird, BigIntegers in JavaScript, http://www.leemon.com/crypto/BigInt.html