2005－CSEC－29（5）
2005／5／19

不正者追跡可能な匿名通信方式の実装と評価

千田 浩司† 小宮 輝之† 塩野入 理† 金井 敦†

†NTT 情報流通プラットフォーム研究所
239-0847 横須賀市光の丘 1-1

{chida.koji,komiya.teruyuki,shionoiri.osamu,kanai.atsushi}@lab.ntt.co.jp

あらまし 本稿では，筆者らが昨年提案した “不正者追跡可能な匿名通信方式” の実装報告を行う．提案方式では匿名プロキシを多段に中継させた通信を行うため，匿名プロキシの段数を増やす程，匿名性は高まるが全体の通信コストも増大するという実用上の課題があった．そこで今回は各種 Web サービスを適用例とし，受信データサイズと匿名プロキシの中継数をパラメータとして提案方式を実装する事で，提案方式の実用性検証を行った．

Implementation and Evaluation of Anonymous Networks with a Robust Anonymity Revocation Scheme

Koji Chida Teruyuki Komiya Osamu Shionoiri Atsushi Kanai

†NTT Information Sharing Platform Laboratories
1-1 Hikarinooka, Yokosuka
Kanagawa, 239-0847 Japan

{chida.koji,komiya.teruyuki,shionoiri.osamu,kanai.atsushi}@lab.ntt.co.jp

Abstract This paper reports on implementation results of an anonymous network with a robust anonymity revocation scheme, which was proposed by our group last year. There was a problem on practical use in the proposed method that the entire communication cost has increased in proportion to the number of relay anonymous proxies. Then, the feasibility of the proposed method was verified by implementing the method for constructing anonymous and reliable Web services.

1 はじめに

1.1 背景

TCP/IP をベースとした WWW では，IP アドレスを元に悪意あるサイト管理者や盗聴者が，ユーザのプライバシ情報を引き出したりマシンに攻撃を仕掛ける恐れがある．この種のセキュリティ侵害を防ぐためには，匿名プロキシと呼ばれる，WWW サーバにユーザ端末の IP アドレスを与えない機能を持つサーバを中継利用する方法が有効である．更に Onion Routing [4] と呼ばれる匿名通信技術を用いて匿名プロキシを多段に中継すれば，匿名プロキシ管理者に対しても「誰がどこにアクセスしたか」を秘匿でき，これによりユーザのセキュリティはより高められるといえる．

しかしながら匿名プロキシはユーザの不正行為を助長する可能性も否定出来ない．特に匿名プロキシが外国で運用されていたり管理が杜撰であれば，不正ユーザの特定が極めて困難と成

りかねない．

筆者らは文献 [1] において，一定数以上の第三者機関が協力すれば不正ユーザを特定出来るが，逆に正規のユーザは一定数以上の第三者機関が不正結託しない限りは Onion Routing 同様の匿名性を有する通信方式を提案した．ユーザの追跡権限を複数の機関に分散させる事で正規ユーザの匿名性と不正ユーザの追跡可能性の両立を狙った提案方式は，匿名掲示板サービスやオークション，そして Peer-to-Peer ファイル交換等，ユーザの匿名性と不正利用防止の双方が望まれるような Web サービスに有効であると考えられる．

1.2 本稿の結果

本稿では筆者らが文献 [1] において提案した“不正者追跡可能な匿名通信方式” の実装報告を行う．特に実装方法は，提案方式 [1] の通信コストを大幅に抑えるプロトコル改良がなされている．詳細は図 2 に示されている．また実装システムの性能結果は表 1 〜 3 に示されている．

3 節で紹介する，提案方式 [1] を実装したシステムを以降 RANS (Revocable Anonymous Network System) と呼ぶ．

以降，2 節で筆者らが文献 [1] において提案した方式の概略を説明する．そして 3 節では，先ず RANS のシステム構成と処理フローを紹介し，次いで 3.1 節で RANS の性能結果を与え，その結果に対する評価を 3.2 節で行う．最後に 4 節で，本稿の結果をまとめるとともに，今後の課題について述べる．

2 基盤方式 [1]

本節では文献 [1] の方式における通信時の処理概要を説明する．初期設定や不正者追跡時の処理については [1] を参照されたい．

U，M_i $(i = 1, \ldots, m)$，R をそれぞれユーザ端末，匿名プロキシ，受信端末とする．$\mathcal{E}_i, \mathcal{P}$ を閾値暗号関数[1]とする．また U, M_i はそれぞれ署名生成関数 $\mathcal{S}_0$, $\mathcal{S}_i$ を保持し，対応する署名検証関数をそれぞれ $\mathcal{V}_0$, $\mathcal{V}_i$ とする．

通信時の処理概要を図 1 に示す．

3 実装方式

図 1 に示した基盤方式を実装した RANS のシステム構成は，ユーザ端末 U，匿名プロキシ M_i $(i = 1, \ldots, m)$，受信端末 R，ユーザの匿名性を失効出来る (複数の) 機関 T_j $(j = 1, \ldots, k)$，そして M_i, R のアドレスや公開鍵を公開するポータルサイト P からなる．U, R は多数存在しているものとする．ここで T_j のうちの一定数 $(= t)$ 以上が協力した場合に限りユーザの匿名性を失効出来る事に注意されたい．

RANS では，ユーザは先ず P にアクセスし，パスワード認証を行う事で匿名 Web サービスを利用出来る．これにより，ユーザはパスワードを入力するだけで様々な端末からサービスを利用出来るようになり，更に P はユーザの利用を動的に制限する事 (例えばユーザは不正が発覚した時点で以降サービスを利用出来なくする等) も容易となる．

RANS の実装方法に関しては以下の特徴がある．

- $\mathcal{E}_i, \mathcal{P}$ を公開鍵暗号ではなく文献 [2] の方式に基づくハイブリッド暗号として実装し，更に鍵交換処理の改良により，$\mathcal{E}_i, \mathcal{P}$ の鍵交換処理を独立に行う場合に比べ通信コストを抑えた．
- メッセージサイズ L_{M}, 匿名プロキシの段数 m をパラメータとしたとき，Γ_i のデータサイズは図 1 の方式では $O(iL_{\mathsf{M}} + im)$ となるが，それを $O(L_{\mathsf{M}} + im)$ となるよう改良した．

RANS の通信時，不正者追跡時の処理フローをそれぞれ図 2, 3 に示す．ここで $\mathcal{G}_q$ は位数 q の可換群，$g(\in \mathcal{G}_q)$ は公開鍵，E_K (D_K) は鍵 K を用いた共通鍵暗号 (復号) 関数，H，$\mathcal{H}$ は一方向性ハッシュ関数とする．以降 $\mathcal{G}_q$ の演算

[1] 閾値暗号とは，秘密鍵を複数の第三者機関に分散させ，その分散鍵を保持する機関のうちの一定数以上が協力した場合に限り復号可能な暗号系を指し，閾値 ElGamal 暗号 [3] 等が知られている．

1. U は以下を行う．
   (a) R のアドレス Addr_{m+1}，中継させる匿名プロキシのアドレス Addr_i $(i = 1, \cdots, m)$ を取得する．
   (b) メッセージ $C_m \stackrel{\text{def}}{=} msg$ に対し，$i = m-1, \ldots, 0$ の順に $C'_i \stackrel{\text{def}}{=} (\mathsf{Addr}_{i+2} \| C_{i+1})$, $C_i \stackrel{\text{def}}{=} \mathcal{E}_{i+1}(C'_i)$ を計算する．
   (c) $\sigma_0 \stackrel{\text{def}}{=} \mathcal{S}_0(C_0)$ を計算し，(C_0, σ_0) を Addr_1 に従って M_1 に送信する．
2. $i = 1, \cdots, m$ の順に M_i は以下を行う．
   (a) $\mathcal{V}_{i-1}(C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1}) = 1$ が成り立つ事を確認する[a](Γ_0 は空の値)．
   (b) C_{i-1} を復号し，$C'_{i-1} = (\mathsf{Addr}_{i+1} \| C_i)$ を得る．
   (c) $\Gamma_i \stackrel{\text{def}}{=} \mathcal{P}(C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1})$，$\sigma_i \stackrel{\text{def}}{=} \mathcal{S}_i(C_i \| \Gamma_i)$ を計算し，$(C_i, \Gamma_i, \sigma_i)$ を Addr_{i+1} に従って M_{i+1} ($i = m$ であれば R) に送信する．
3. R は以下を行う．
   (a) $\mathcal{V}_m(C_m \| \Gamma_m \| \sigma_m) = 1$ が成り立つ事を確認する．
   (b) 入力の組 $(C_m, \Gamma_m, \sigma_m)$ を保管し，メッセージ $C_m (= msg)$ に従った処理を実行する．

[a] ここで $\mathcal{V}_i$ は，$\sigma_i = \mathcal{S}_i(C_i \| \Gamma_i)$ が成り立つ場合に限り 1 を返すような関数とする．

図 1: 基盤方式 [1] の通信時の処理フロー

は乗法的に記述する．P, M_i はそれぞれ署名生成関数 $\mathcal{S}_0, \mathcal{S}_i$ を保持し，対応する署名検証関数をそれぞれ $\mathcal{V}_0, \mathcal{V}_i$ とする．ユーザ ID を UID とする．M_i は乱数 $a_i \in \mathbb{Z}/q\mathbb{Z}$ を選び $h_i \stackrel{\text{def}}{=} g^{a_i}$ を計算，公開しているものとする．同様に P, R はそれぞれ $h_0 \stackrel{\text{def}}{=} g^{a_0}$, $h_{m+1} \stackrel{\text{def}}{=} g^{a_{m+1}}$ を計算，公開しているものとする．不正者追跡処理については説明を簡略化するために，T_j $(j = 1, \ldots, t)$ が処理に協力し，その結果は正しいものとして話を進める．また初期設定は [1]，[2] に基づくものとし，ここではその説明を省略する．

表 2: 通信時に要する処理時間 (2) [単位:秒]

	受信データサイズ ($= L_{\mathsf{M}}$)			
	1KB	10KB	100KB	1MB
送信 (U → R)	0.189	0.189	0.184	0.189
受信 (R → U)	0.030	0.041	0.213	2.151
送受信全体	0.227	0.241	0.430	2.347

表 3: 不正者追跡に要する処理時間 [単位:秒]

匿名プロキシの段数 ($= m$)					
2	3	5	10	20	30
1.122	1.269	1.547	2.250	3.587	4.828

3.1 性能

今回は以下の測定を行った．

[表 1] 受信データサイズを 100KB とし，匿名プロキシの段数 m =3, 4, 5, 10, 15, 20, 30 に対する処理時間

[表 2] 匿名プロキシの段数を 2 とし，受信データサイズ L_{M} =1, 10, 100, 1000 [KB] に対する処理時間

[表 3] 受信データサイズを 100KB, 500KB とし，匿名プロキシの段数 m =2, 3, 5, 10, 20, 30 に対する処理時間

ここで表 1, 2, 3 について，送信リクエスト $(= msg)$ のデータサイズはおよそ 260Bytes と一定である．

試験環境，利用暗号アルゴリズムをそれぞれ表 4, 5 に示す．なおネットワークは Switching HUB 100BASE-TX により構成した．

3.2 評価

図 2 からも分かるように，U から R へ送られる送信データは，その送信過程で各 M_i によって署名生成／検証，鍵生成，および暗号化／復号処理といった各種暗号処理が行われる．一方

1. U は以下を行う．
   (a) パスワード認証により P にアクセスし，R のアドレス Addr_{m+1}，中継させる匿名プロキシのアドレス Addr_i $(i = 1, \cdots, m)$ を取得する．
   (b) $r \in_R \mathbb{Z}/q\mathbb{Z}$ を選び，P への鍵交換用公開鍵 $G_{-1} = g^r$ を計算し，$\Lambda_{-1} = 1$ として $i = 0, 1, \cdots, m, m+1$ の順に $G_i = h_i^{r\Lambda_{i-1}}$, $\Lambda_i = \mathcal{H}(G_i)\Lambda_{i-1}$, および共通鍵 $K_i = H(G_i)$ を計算する．
   (c) メッセージ $C_{m+1} \stackrel{\text{def}}{=} msg$ に対し，$i = m, m-1, \ldots, 0, -1$ の順に $C'_i \stackrel{\text{def}}{=} (\mathsf{Addr}_{i+2} \| C_{i+1})$, および暗号文 $C_i \stackrel{\text{def}}{=} E_{K_{i+1}}(C'_i)$ を計算し，(G_{-1}, C_{-1}) を P に送信する (Addr_{m+2} は空の値)．
2. P は以下を行う．
   (a) $T_0 = G_{-1}^{a_0}$, M_1 への鍵交換用公開鍵 $G_0 = G_{-1}^{\mathcal{H}(T_0)}$, および二つの共通鍵 $K_0 = H(T_0), \mathcal{K}_0 = H(G_0^{a_0})$ を計算し，$D_{K_0}(C_{-1})$ として C_{-1} を復号し，$C'_{-1} = (\mathsf{Addr}_1 \| C_0)$ を得る．
   (b) 不正者追跡用暗号文 $\Gamma_0 \stackrel{\text{def}}{=} E_{\mathcal{K}_0}(G_0 \| \mathsf{UID})$, および署名 $\sigma_0 \stackrel{\text{def}}{=} \mathcal{S}_0(G_0 \| C_0 \| \Gamma_0)$ を計算し，$(G_0, C_0, \Gamma_0, \sigma_0)$ を Addr_1 に従って M_1 に送信する．
3. $i = 1, \cdots, m$ の順に M_i は以下を行う．
   (a) σ_{i-1} の署名検証として，$\mathcal{V}_{i-1}(G_{i-1} \| C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1}) = 1$ が成り立つ事を確認し，$T_i = G_{i-1}^{a_i}$, M_{i+1} ($i = m$ であれば R) への鍵交換用公開鍵 $G_i = G_{i-1}^{\mathcal{H}(T_i)}$, および二つの共通鍵 $K_i = H(T_i)$, $\mathcal{K}_i = H(G_i^{a_i})$ を計算し，$D_{K_i}(C_{i-1})$ として C_{i-1} を復号し，$C'_{i-1} = (\mathsf{Addr}_{i+1} \| C_i)$ を得る．
   (b) 不正者追跡用暗号文 $\Gamma_i \stackrel{\text{def}}{=} E_{\mathcal{K}_i}(G_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1})$, および署名 $\sigma_i \stackrel{\text{def}}{=} \mathcal{S}_i(G_i \| C_i \| \Gamma_i)$ を計算し，(G_i, C_i, Γ_i) を Addr_{i+1} に従って M_{i+1} ($i = m$ であれば R) に送信する．
4. R は以下を行う．
   (a) $\mathcal{V}_m(G_m \| C_m \| \Gamma_m \| \sigma_m) = 1$ が成り立つ事を確認し，$T_{m+1} = G_m^{a_{m+1}}$, および共通鍵 $K_{m+1} = H(T_{m+1})$ を計算し，$D_{K_{m+1}}(C_m)$ として C_m を復号し，$C'_m = C_{m+1} = msg$ を得る．
   (b) 入力の組 $(C_m, \Gamma_m, \sigma_m)$ を保管し，メッセージ $C_m (= msg)$ に従った処理を実行し，U への返信データ $resp$ に対して暗号文 $R_{m+1} \stackrel{\text{def}}{=} E_{K_{m+1}}(resp)$ を計算し，R_{m+1} を M_m に返信する．
5. $i = m, m-1, \cdots, 1, 0$ の順に M_i ($i = 0$ であれば P) は暗号文 $R_i \stackrel{\text{def}}{=} E_{K_i}(R_{i+1})$ を計算し，R_i を M_{i-1} ($i = 1$ であれば P，$i = 0$ であれば U) に返信する．
6. U は $D_{K_{m+1}}(D_{K_m}(\cdots(D_{K_0}(R_0))\cdots)) = resp$ として R_0 を復号し，R からの返信データ $resp$ を得る．

図 2: 実装方式の通信時の処理フロー

表 4: 試験環境

CPU	Pentium4 3.0GHz
RAM	1GB
NIC	1000BASE-T
OS	Windows 2000 Professional SP4
言語	C, Java
プロトコル	SOAP over HTTP

表 5: 利用暗号アルゴリズム

共通鍵暗号	Camellia CBC モード
鍵交換	[2]+独自方式 (OEF 上の楕円演算)
署名	ECDSA on OEF
ハッシュ関数	SHA-1

R から U への返信過程では，返信ルートは送信ルートの逆を辿り，各 M_i は送信時に生成した共通鍵を再利用して暗号化処理のみ行えば良い．なお同一セッションにおける二回目以降の通信は，セッション ID 管理により初回に生成した共通鍵を再利用し，鍵生成不要の仕様とした．即ち同一セッション内であれば，二回目以降の通信は初回よりも高速となる事が期待出来る．

表 1 の結果から，送受信にかかる処理時間は，U, R の通信路に中継させる M_i の段数に対してほぼ比例増加する事が確認出来た．理論上は送受信メッセージの暗号文 C_i, R_i や不正者追跡用暗号文 Γ_i は M_i の段数 m が大きくなるにつれて増大するため，C_{i-1} の復号，および R_{i+1}, Γ_i の暗号化の処理コストは M_i 毎に異なるが，全体として m に対してほぼ比例して増加しているのは，M_i における全体の処理コストに対

1. R は LOG $\stackrel{\mathrm{def}}{=}$ $(G_m, C_m, \Gamma_m, \sigma_m)$ を P に送信し，LOG に対応するユーザの開示を要求する．
2. P は何らかの判断に基づいてユーザ開示の要否を決定し，開示する場合は以下を行う．
3. $\mathcal{V}_m(G_m \| C_m \| \Gamma_m \| \sigma_m) = 1$ が成り立つ事を確認する．成り立たない場合は R を不正として終了する．
4. $i = m, \ldots, 1, 0$ について以下を行う．
   (a) T_j $(j = 1, \ldots, t)$ と協力し，$\mathcal{K}_i = H(G_i^{a_i})$, $D_{\mathcal{K}_i}(\Gamma_i) = (G_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1})$, $K_i = H(G_{i-1}^{a_i})$, $C_{i-1} = E_{K_i}(\mathsf{Addr}_{i+1} \| C_i)$ を計算する $(i = 0$ であれば $D_{\mathcal{K}_i}(\Gamma_i) = (G_{i-1} \| \mathsf{UID}))$.
   (b) $i > 0$ であれば，T_j の処理正当性をゼロ知識証明技術により検証後，$\mathcal{V}_{i-1}(G_{i-1} \| C_{i-1} \| \Gamma_{i-1} \| \sigma_{i-1}) = 1$ が成り立つ事を確認する．成り立たない場合は M_i を不正として終了する．
5. UID から不正ユーザを特定する．

図 3: 実装方式の不正者追跡時の処理フロー

して C_{i-1}, R_{i+1}, Γ_i の処理コストが無視出来る程小さいためである．したがって，送信メッセージ msg あるいは返信データ $resp$ が十分大きい場合は，表 1 の増加傾向とは異なる結果になる事が予測出来る．

表 2 の結果から，受信データサイズが 100KB 程度であっても，受信時の処理コストが鍵交換や不正者追跡用暗号文を生成する送信時の処理コストを上回る事が分かった．また 1MB の受信データであっても 2 段の中継で 2 秒程度と，Web サービスとして見た場合，一般的には十分実用的な性能である事を確認した．ここで受信時における R_i $(i = 0, 1, 2, 3)$ を生成する暗号化処理はおよそ 50 msec と全体から見ればそれ程大きくはなく，むしろデータ送受信にかかる時間の方がボトルネックになるという結果を得た．

不正者追跡時の処理は，通信時と比べ一般に即時性は求められないと考えられるが，表 3 に示すとおり，M_i の段数 m が大きい場合であっても数秒程度で処理出来る事を確認した．

4 まとめと今後の課題

筆者らが昨年提案した “不正者追跡可能な匿名通信方式” の実用性を検証するため，不正者追跡出来る匿名 Web サービスを想定した本方式の実装を行った．結果，例えば HTTP リクエストに対して 100KB 程度の受信データであれば，中継させる匿名プロキシを 10 段としても受信まで 1 秒程度，また 1MB の受信データであっても 2 段の中継で 2 秒程度と，Web サービスとして見た場合，一般的には十分実用的な性能である事を確認した．

最後に今後の課題を以下に挙げる．

- 受信データサイズをより大きくして測定する
- 処理時間の詳細分析を行う
- 独自鍵交換プロトコルの安全性を評価する
- 各サーバのスループットを測定する
- 不正者追跡機能を付加した事で生じるオーバーヘッドを評価する

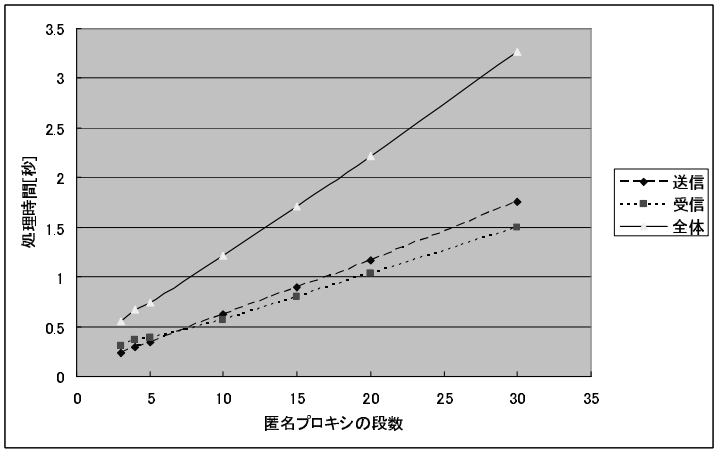
参考文献

[1] 千田浩司, 小宮輝之, 林徹 「匿名性確保と不正者追跡の両立が可能な通信方式」 情報処理学会論文誌, 第 45 巻, 第 8 号, pp. 1873–1880 (2004).

[2] Chida, K. and Abe, M.: Flexible-routing anonymous networks using optimal length of ciphertext, IEICE Trans, Fundamentals, Vol. E88-A, No. 1, pp. 211–221 (2005).

[3] Desmedt, Y. and Frankel, Y.: Threshold cryptosystems, *Advances in Cryptology — CRYPTO '89*, LNCS 435, Brassard, G. (Ed.), pp. 307–315, Springer-Verlag (1990).

表 1: 通信時に要する処理時間 (1) [単位:秒]



	匿名プロキシの段数 ($= m$)						
	3	4	5	10	15	20	30
送信 (U → R)	0.238	0.290	0.343	0.630	0.893	1.169	1.759
受信 (R → U)	0.310	0.362	0.388	0.568	0.807	1.035	1.495
送受信全体	0.549	0.674	0.741	1.213	1.703	2.212	3.261

[4] Syverson, P.F., Goldschlag, D.M. and Reed, M.G.: Anonymous connections and onion routing, *Proc. 1997 IEEE Symposium on Security and Privacy*, pp. 44–54, IEEE Press (1997).