暗号化匿名通信プロトコルの提案とそのプロトタイプ実装

江村 恵太[1] 金岡 晃[2,1] 太田 悟史[3,1] 面 和成[3] 高橋 健志[1]

概要：ユーザの匿名性を担保したまま権限を確認することや暗号化を行うためには, 暗号方式と匿名通信プロトコルとの融合が必要不可欠である. 例えば暗号方式として匿名でユーザの権限を確認する機能を保証していたとしても, IP アドレス等の通信時の付加情報によりその匿名性が損なわれてしまう. また現在の PKI を利用すると, ユーザの公開鍵 (の証明書) からユーザを特定可能であるため匿名性を担保できない. そこで本論文では, 暗号方式 (グループ署名及び ID ベース暗号) 及び匿名通信プロトコル (Simpleproxy または Tor) を利用した暗号化匿名通信プロトコルの提案及びプロトタイプ実装を行い, そのフィージビリティを検証する. 特に Tor を用いた場合, 提案プロトコルを用いた場合と SSL 通信の場合 (また暗号化通信を行わない場合) とを比較して, 大きな差異が見られない. これらの結果から, 提案プロトコルが現実的なコストで暗号化匿名通信を実現できることが示される. なお本論文は著者らが ACM SAC 2014 で発表した成果に対し, Tor を用いた場合の評価を加えたものである.

キーワード：暗号化匿名通信, ID ベース暗号, グループ署名, Tor

A Secure and Anonymous Communication Protocol and its Prototype Implementation

Keita Emura[1] Akira Kanaoka[2,1] Satoshi Ohta[3,1] Kazumasa Omote[3] Takeshi Takahashi[1]

1. はじめに

近年のプライバシに対する意識の高まりを受け, ユーザにサービスを提供する際に匿名性, すなわちユーザを特定することなくサービスを提供するための技術が望まれており, これまで様々な研究が行われている [2]. しかしながら, 匿名であるがゆえに “どのように匿名ユーザがサービスを得る権限を有しているのかを確認する” ことは非自明であり, 単純な通信路の匿名性 (例えば IP アドレスを秘匿する) のみでは到底達成できない. そこで匿名システムの多くが構成要素として暗号技術を用いている.

1 情報通信研究機構
National Institute of Information and Communications Technology, Japan
{emura, takeshi_takahashi}@nict.go.jp
2 東邦大学
Toho University, Japan
kanaoka@toho.ac.jp
3 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology, Japan
{s-ohta, omote}@jaist.ac.jp

1.1 背景

匿名性を担保する様々な暗号技術の 1 つとしてグループ署名 [8] が知られている. グループ署名方式では, グループ管理者 (Group Manager, GM) はグループ公開鍵とマスタ秘密鍵を持ち, マスタ秘密鍵を用いてユーザに署名鍵を配布, ユーザはその署名鍵を用いてグループ署名を計算する. 検証者はグループ公開鍵を用いてグループ署名を検証するが, その際にユーザに紐づく情報を必要としないため, 署名者の匿名性 (グループに所属していることのみを示す) を担保できる. また GM は自身の秘密鍵を用いることで, 署名者を特定できる権限 (オープン機能) をも有す. なおグループ署名を実際の通信環境にて活用する際には, パケットに含まれる送信元 IP アドレスを秘匿する必要があることに注意されたい. すなわち, いくら暗号方式として匿名性を保証していても, 通信時の付加情報によりその匿名性が損なわれてしまう.

ユーザから受信した IP パケットには必ずユーザの IP アドレスが入っており, 正常な通信を実現するためにはこのアドレスを単純に消去・改変することは許されない. そこで, 中継機器 (プロキシ) を用いてユーザとプロバイダ

(Service Provider, SP) の間の通信を仲介する方式が考えられており, Simpleproxy [3] や Tor [21] をはじめとしたいくつかの方式が既に提案されている [2]. これら技術は匿名性を実現する一方で, ユーザの正当性を如何にして担保するかは保証されておらず, 例えばチャネルへの無許可アクセスを防止する際などに問題となる. Tor ルータなどの中継機器で認証する手法も考えられるが, SP が直接エンドユーザを認証する手法については, 依然課題のままである.

単純な解決策として, グループ署名をプロキシを介して送信することが考えられる. つまりまずユーザは匿名で検証可能なトークン (グループ署名) を作成し, 匿名チャネル (Simpleproxy や Tor) を経由して SP にトークンを送信する. これにより, SP は匿名性を損ねることなく直接ユーザを検証することができる. しかしながらこれだけでは通信は安全ではなく, 如何にして匿名性を維持したまま通信を暗号化するかを考える必要がある. しかし現在の公開鍵のインフラ (PKI) を利用すると, ユーザの公開鍵 (の証明書) からユーザを特定可能であるため匿名性を担保できない. 共通鍵暗号を利用して暗号化を行う場合だとしても, SP が鍵交換プロトコルの実施にユーザの公開鍵を必要とするため, 同様の問題が発生する.

1.2 本論文の貢献

本論文では, 暗号方式 (グループ署名及び ID ベース暗号 (Identity-based Encryption, IBE)) 及び匿名通信プロトコル (Simpleproxy または Tor) を利用した暗号化匿名通信プロトコルを提案する. グループ署名を用いることでユーザは匿名性を担保したまま自身の権限を SP に証明でき, また ID ベース暗号を用いることで (匿名性に反することなく) SP がユーザに対する暗号化を実現することができる. より具体的には, IBE とは任意の値を公開鍵として設定可能な公開鍵暗号方式であり, ユーザはセッション毎に一時的な ID を選択することで, サーバはその値を公開鍵として利用することができる. また中継機器（プロキシ）を利用することで, SP はユーザの IP アドレスを知ることなしに通信が可能となる.

本論文ではまず提案プロトコルのフレームワーク及び安全性の定義を行う. さらに本方式ではグループ署名のオープン機能が不要である (プロキシが管理する IP アドレステーブルを利用すればよい) ことを鑑み, 古川-今井グループ署名方式 [11] からオープン機能を削除したオープンフリー型グループ署名方式 (Open-Free Group Signature, OFGS) を提案した. 本改良により, オリジナルの方式に比べて署名サイズを 50%削減した.

また提案プロトコルの安全性の定式化及びプロトタイプ実装を行い, そのフィージビリティを検証する. 特に SSL 通信と性能比較してその実用性を示す. なお実装時には前述の古川-今井方式をベースとした OFGS 方式と, Boneh-Franklin IBE 方式 [7] を用いている. プロキシとして Simpleproxy を用いた場合, 提案プロトコルの実行時間が SSL 通信のほぼ 50 倍という結果 (ただしミリ秒オーダの時間に収まる) に対し, プロキシとして Tor を用いた場合, 複数の Tor サーバを経由するための時間の影響が大きく, 提案プロトコルを用いた場合と SSL 通信の場合 (また暗号化通信を行わない場合) とを比較して, 大きな差異が見られない. これらの結果から, 提案プロトコルがユーザと SP 間の通信について, 現実的なコストで暗号化匿名通信を実現できることが示される.

なお本論文では SP からユーザへの暗号通信に焦点を絞っているが, 双方向での暗号化通信を実現する際には, SP がユーザに対して匿名性を維持する必要がないことから, ユーザが SP の公開鍵を用いた暗号化を実施するなどで容易に実現できることに注意されたい. また今回の方式は匿名環境下で署名を SP に送付することのみに着目しているが, 古典的なチャレンジ-レスポンス認証を容易に適用することができる. 例えば, SP がランダムな Nonce を (プロキシを介し) ユーザに送り, ユーザはその Nonce をグループ署名の署名文章に含めればよい. これらの改良は容易であるため, 本論文では議論しない. 以下, 検証 (Verification) と認証 (Authentication) を区別して考える.

国際会議版との差分: ACM SAC 2014 [9] にて, 我々は暗号化匿名通信プロトコルのフレームワーク及び安全性の定義, 古川-今井 OFGS 方式の提案, 及び Simpleproxy を用いたプロトタイプ実装の評価を行った. 本論文では, ACM SAC 2014 で発表した成果に対し, プロキシとして Tor を用いた場合の評価を加え, さらに提案方式のインターネットへの適用可能性を評価する.

1.3 関連研究

Sudarsono ら [19] は IEEE802.1X ベースの匿名認証システムの検討を行っている. クライアントの証明書としてグループ署名を利用することは提案方式と同様であるが, IP ネットワーク上での証明書の送付機構については検討していないことに注意されたい. Lee ら [14] が提案した匿名登録サービス Anon-pass では, (グループ署名と同様に) 非対話ゼロ知識 (Non-Interactive Zero Knowledge, NIZK) を用いた証明方法を採用している一方で, 通信の暗号化については言及していない.

公開鍵を匿名通信路を通して配布する手法が Gilad と Herzberg [12] により提案されている. ユーザは公開鍵をセッション毎にランダムに選択し, 匿名通信路を通して送付することで, 暗号化通信が可能となる. しかしながら公開鍵がランダムに選択されているため公開鍵証明書が付与されず, 結果公開鍵の正当性を確認できないという問題が

発生する.[*1] 例えば, ある攻撃者が公開鍵を取り替えたとしても応答者は検知する事ができない. それに加え, 匿名下での検証について, この方式では言及していない. なお我々の提案方式では, IBE は任意の文字列を公開鍵として利用できるため SP は公開鍵 (すなわち, 一時的な ID) が機能することを理解しており, さらにその ID はグループ署名により署名されているため, 鍵のすり替え攻撃を防止しつつ匿名で署名を検証することができる.

暗号化匿名通信を実現する他の手法として, 代理人再暗号化方式 (Proxy Re-Encryption, PRE) [16] を用いた方式も考えられる. ユーザは自身の秘密鍵と SP の公開鍵を用いて再暗号化鍵を作成しプロキシに預け, SP は自身の公開鍵を使ってメッセージを暗号化し, SP から受け取るメッセージをプロキシは再暗号化して, ユーザに送ることで暗号化匿名通信を実現する. しかしながら, プロキシは全ての再暗号化鍵を常に管理する必要がり, プロキシからこれらの鍵が漏洩する懸念があることから, 他ユーザが予期せず暗号文を復号してしまう可能性は払しょくできない. すなわち, 予期せずに利用されてしまう可能性のある再暗号化鍵を生成してしまうのは望ましくないと考えられる. なお我々の方式では一時的な ID はセッション毎に選択されるため, 予期せぬユーザ (プロキシを含む) が暗号文を復号することはできない. ただし IBE を用いている以上, 鍵生成センタ (Key Generation Center, KGC) が全ての暗号文を復号できるという問題が発生することに注意されたい. しかしながらシステム内部の全プロキシを信頼すること比較して, KGC を信頼し得る第三者としてモデル化することはより現実的であると考えられる.

Tor は TCP/IP における接続経路の匿名化を実現するソフトウェアである. 送信者は複数の Tor ルータを中継ノードとして選択し, それらの Tor ルータはオニオンルーティングを用いてパケットを転送する. オニオンルーティングでは, 送信者があらかじめ経由すべき Tor ルータを, サーキット ID という形で指定する. そして Tor ルータは, 受信したパケットを復号し, その中に記されているサーキット ID に基づき次に転送する Tor ルータを決定し, そこへパケットを送信する (バケツリレー方式). Tor により, 通信の匿名性は担保されるが, 一方でその匿名性を利用した犯罪に利用されるため, Tor による通信を遮断する動きもみられている. 本論文の提案プロトコルは, グループ署名により匿名性を確保しつつ “許可されている組織・グループに所属する” ことのみを確認するため, 本プロトコルとの組み合わせにより問題解決の一助となる.

[*1] ユーザはセッション毎に選択した公開鍵に対応した秘密鍵に対する NIZK 証明を作成することで, 鍵の正当性を保証することも考えられるが, 公開鍵暗号方式の代数的構造を考慮しつつ NIZK 証明を構築しなければならず, 実装の複雑化につながりかねないことから, ここでは検討しないこととする.

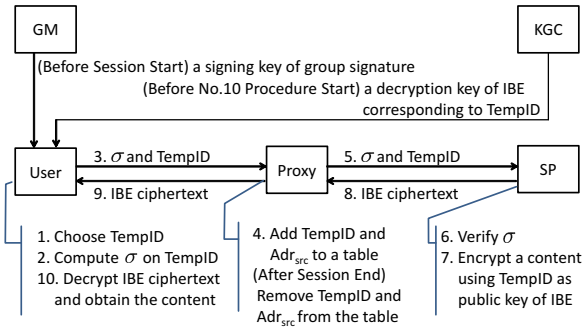


図 1 Framework of the Proposed Protocol

2. フレームワーク

図 1 に, 提案プロトコルのフレームワークを示す. 本フレームワークでは, ユーザ, プロキシ, サービスプロバイダ (SP), GM, 及び KGC の 5 つの役割を定義する. 提案プロトコルの目的は, ユーザが SP にトークンを送付し, SP はそのトークンを (ユーザを特定することなく) 匿名で検証し, 正当な署名である場合のみサービスコンテンツを暗号化し, ユーザに送付することである. その目的を達成するため, プロキシはユーザの IP アドレスを明らかにせずに SP にトークンを中継する役割を担う. ここでプロキシは hondest-but-curious, すなわちプロトコルで定められた手順を逸脱しないものであると仮定する. 尚, プロキシは実際には複数の機器の多段構成により実現されることもあるが, ここではそれらをまとめて 1 つのロールとして定義する. GM はグループ鍵と発行鍵を管理し, トークンを作成するために用いる署名鍵をユーザへ配布する. なお, GM はユーザへの署名鍵の配布前に, ユーザを適切に認証していると仮定する. KGC はユーザの復号鍵を生成する. ただし, KGC は復号鍵の配布前にユーザを適切に認証していると仮定する.

以下, 提案暗号化匿名通信プロトコルの流れを説明する. (1) ユーザ (IP アドレスを $\mathtt{Adr}_{\mathtt{src}}$ とする) はセッション毎の一時的な ID ($\mathtt{TempID}$) を選択, (2) $\mathtt{TempID}$ を署名対象メッセージとしてグループ署名 σ を作成し, (3) プロキシへ $(\sigma, \mathtt{TempID})$ を送信する. (4) プロキシは $\mathtt{Adr}_{\mathtt{src}}$ を $\mathtt{TempID}$ と結びつけて保存しておき, (5) SP へ $(\sigma, \mathtt{TempID})$ を送信する. この手続きを繰り返すことで, 最終的に SP は $(\sigma, \mathtt{TempID})$ を得る. (6) SP はグループ署名を検証することにより, 匿名性を損なうことなくユーザが正当なグループメンバであるかどうかを直接確認することができる (前述の通り, 認証を行う場合は, SP は最初に送付した Nonce に対する正当なグループ署名であることを検証することに注意されたい). (7) 署名が正当である場合, SP は $\mathtt{TempID}$ を IBE の公開鍵としてコンテンツを暗号化する. 署名が正当でない場合, $\perp$ を返す. (8) SP は IBE 暗号文をプロキシへ

送る．なおプロキシは SP とのやり取りを 1 セッションとして管理可能であると仮定する．(9) プロキシは保存してある $\mathtt{Adr}_{\mathtt{src}}$ を参照することで, IBE 暗号文をユーザへ転送する．最終的にユーザは IBE 暗号文を得る．(10) 最後にユーザは KGC から配布された TempID に対応する復号鍵を用いることで, IBE 暗号文を復号しコンテンツを取得する．なおプロキシは通信を中継した後, 直ちに $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ を削除する．もしユーザと SP とが複数回のやり取りを行う場合 (これを 1 セッションと定義), セッション終了後に $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ を削除する．この手続きにより, 通信が完了した後にプロキシが管理するテーブルが漏洩したとしても, 個人が特定される恐れはない．

3. 暗号化匿名通信プロトコル

本節では, 上述のフレームワークの定式化及び安全性定義を行う．

3.1 提案プロトコルの定式化と安全性の定義

$\mathcal{ID}$ と $\mathcal{M}$ はそれぞれ ID 空間およびメッセージ空間とし, $\mathtt{Adr}_{\mathtt{src}}$, $\mathtt{Adr}_{\mathtt{proxy}}$, そして $\mathtt{Adr}_{\mathtt{dst}}$ はそれぞれユーザ, プロキシ, SP の IP アドレスを示す．集合 X と要素 $x \in X$ において, $x \xleftarrow{\$} X$ は x が X から一様ランダムに選ばれる事を意味する．

Definition3.1 (提案プロトコルのシンタックス)

GM.Setup: セキュリティパラメータ λ を入力とし, グループ公開鍵 gpk と発行者鍵 ik を出力する．

KGC.Setup: セキュリティパラメータ λ を入力とし, 公開情報 $params$ とマスタ秘密鍵 msk を出力する．

Join: gpk と ik を入力とし, 署名鍵 sk を出力する．

UserKeyGen: $params$, msk, 一時的な ID $\mathtt{TempID} \in \mathcal{ID}$ を入力とし, 復号鍵 $dk_{\mathtt{TempID}}$ を出力する．

SendRequest: gpk, sk, TempID, 送信元 IP アドレス $\mathtt{Adr}_{\mathtt{src}}$, 宛先 IP アドレス $\mathtt{Adr}_{\mathtt{dst}}$, プロキシの IP アドレス $\mathtt{Adr}_{\mathtt{proxy}}$ を入力とし, トークン σ, TempID, $\mathtt{Adr}_{\mathtt{dst}}$ をプロキシ (IP アドレスを $\mathtt{Adr}_{\mathtt{proxy}}$ とする) へ送信する．

RelayRequest: $\mathtt{Adr}_{\mathtt{src}}$, $\mathtt{Adr}_{\mathtt{dst}}$, ID/IP アドレスのテーブル Tbl, σ, TempID を入力とし, SP (IP アドレスを $\mathtt{Adr}_{\mathtt{dst}}$ とする) へ $(\sigma, \mathtt{TempID})$ と $\mathtt{Adr}_{\mathtt{proxy}}$ を中継する．さらに, $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ を Tbl に追加する．

ValidityCheck: gpk, σ, TempID を入力とし, 0 または 1 を出力する．

SendContent: gpk, σ, TempID, コンテンツ $M \in \mathcal{M}$ を入力とし, σ が正しい場合 (すなわち ValidityCheck が 1 を返す) 暗号文 C をプロキシ (IP アドレスを $\mathtt{Adr}_{\mathtt{proxy}}$ とする) へ送り, σ が不正な場合 (すなわち ValidityCheck が 0 を返す) には $\perp$ を返す．

RelayContent: C と Tbl を入力とし, C をユーザ (Tbl に含まれている IP アドレス $\mathtt{Adr}_{\mathtt{src}}$ を参照) へ中継する．さらに $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ を Tbl から削除する．ここでプロキシは SP から送付された C 及び Tbl から, 中継すべきユーザの IP アドレスを決定できると仮定する．[*2]

GetContent: C と $dk_{\mathtt{TempID}}$ を入力とし, M を返す．

次に Correctness 要件として, アルゴリズムに従ってすべての値が正しく生成された場合に, ユーザは必ずコンテンツを得られる事を要請する．

Definition3.2 (Correctness)

全ての $(gpk, ik) \leftarrow \mathsf{GM.Setup}(1^\lambda)$, $(params, msk) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$, $sk \leftarrow \mathsf{Join}(gpk, ik)$, $\mathtt{TempID} \in \mathcal{ID}$, $M \in \mathcal{M}$, $(\mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ について, 以下が成り立つ．

$$\Pr\left[\begin{array}{l} M \leftarrow \mathsf{GetContent}\Big(\mathsf{RelayContent}\big(C, \mathtt{Tbl}\big), \\ \mathsf{UserKeyGen}(param, msk, \mathtt{TempID})\Big) \end{array}\right] = 1$$

かつ

$$\Pr[1 \leftarrow \mathsf{ValidityCheck}(gpk, \sigma, \mathtt{TempID}) = 1] = 1$$

ただし, $(\sigma, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{dst}}) \leftarrow \mathsf{SendRequest}(gpk, sk, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$, $(\sigma, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{proxy}}) \leftarrow \mathsf{RelayRequest}(\mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Tbl}, \sigma, \mathtt{TempID})$, $C \leftarrow \mathsf{SendContent}(gpk, \sigma, \mathtt{TempID}, M, \mathtt{Adr}_{\mathtt{proxy}})$ である．

次に, Anonymity, Semantic Security, Unforgeability について定義する．ここで 1 つのセッションとは SendRequest から GetContent まで, SendRequest → RelayRequest → SendContent → RelayContent → GetContent の一連のアルゴリズムとして定義する．Anonymity は, プロキシへの接続を許可されている攻撃者 $\mathcal{A}$ (ただし $\mathtt{Adr}_{\mathtt{src}}$ を知ることは許可されていない) に対し, 2 つのセッション実行が同一ユーザによるものか否かを識別できないことを保証する．ここで $\mathcal{A}$ は悪意のある SP としてモデル化されており, $\mathcal{A}$ が署名鍵および KGC のマスタ鍵 msk を入手できた場合にもユーザの識別は不可能である事を証明する．[*3]

Definition3.3 (Anonymity)

(1) チャレンジャー $\mathcal{C}$ は $(gpk, ik) \leftarrow \mathsf{GM.Setup}(1^\lambda)$, $(params, msk) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$, および 2 つの署名鍵 $sk_0, sk_1 \leftarrow \mathsf{Join}(gpk, ik)$ を計算し, 攻撃者 $\mathcal{A}$ に gpk, sk_0, sk_1, $(params, msk)$ を与える．さらに $\mathcal{C}$ は $\mathtt{Tbl} := \emptyset$ とテーブルを初期化する．

(2) $\mathcal{A}$ が送付した SendRequest クエリ $(i, \mathtt{TempID}) \in \{0, 1\} \times \mathcal{ID}$ に対し, $\mathcal{C}$ は $\mathsf{SendRequest}(gpk, sk_b, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ を実行し, (SendRequest アルゴリズムを通じて生成された) σ を $\mathcal{A}$ へ返す．

[*2] 例えばポート番号はセッションの識別に利用できる．我々の実装ではプロキシ内のセッション管理方法をそのまま利用している

[*3] なおユーザが KGC に対し, TempID に対応する復号鍵の生成を依頼をする様子を, TempID を含めて攻撃者が傍受する場合については (復号鍵の発行が $\mathcal{A}$ に傍受されることなく成されると仮定し) 本安全性定義では考慮しない．

(3) $\mathcal{A}$ が送付した RelayRequest クエリ (σ, TempID) に対し, $\mathcal{C}$ は RelayRequest($\mathtt{Adr}_{\mathtt{src}}$, $\mathtt{Adr}_{\mathtt{dst}}$, Tbl, σ, TempID) を実行し, Tbl を更新する.

(4) $\mathcal{A}$ が送付した RelayContent クエリ C に対し, $\mathcal{C}$ は RelayContent(C, Tbl) を実行し, Tbl を更新する.

(5) $\mathcal{A}$ は $\mathtt{TempID}^* \in \mathcal{ID}$ を $\mathcal{C}$ へ送信する. $\mathcal{C}$ は, ランダムに $b \xleftarrow{\$} \{0,1\}$ を選び, $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{dst}}) \leftarrow \mathsf{SendRequest}(gpk,\ sk_b, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ 及び $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{proxy}}) \leftarrow \mathsf{RelayRequest}(\mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Tbl}, \sigma^*, \mathtt{TempID}^*)$ を実行する. $\mathcal{A}$ は $\mathcal{C}$ へ任意の C を返す. $\mathcal{C}$ は $C \leftarrow \mathsf{RelayContent}(C, \mathtt{Tbl})$ を実行する. ここで $\mathcal{A}$ が取得可能な値は, $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{dst}})$, $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{proxy}})$, C, である事に注意されたい. $\mathcal{A}$ は $b' \in \{0,1\}$ を出力する.

λ に対し $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{pro},\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ が無視できる場合, プロトコルは Anonymity をもつと定義する.

次に Semantic Security について定義する. Semantic Security はネットワーク上を流れている情報からはコンテンツ M に関する情報が漏れないことを保証する. ここでは, 攻撃者 $\mathcal{A}$ は悪意のあるプロキシとしてモデル化されている. さらに GM の権限では M に関する情報を得ることができないことを保証するため, $\mathcal{A}$ に GM の秘密鍵 ik を与えている.

Definition3.4 (Semantic Security)

(1) チャレンジャー $\mathcal{C}$ は $(gpk, ik) \leftarrow \mathsf{GM.Setup}(1^\lambda)$ 及び $(params, msk) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$ を実行し, gpk, ik, $params$ を 攻撃者 $\mathcal{A}$ に与える.

(2) $\mathcal{A}$ が送付した UserKeyGen クエリ $\mathtt{TempID} \in \mathcal{ID}$ に対し, $\mathcal{C}$ は UserKeyGen($params$, msk, TempID) を実行し, $dk_{\mathtt{TempID}}$ を返す.

(3) $\mathcal{A}$ は $\mathtt{TempID}^* \in \mathcal{ID}$, $M_0^*, M_1^* \in \mathcal{M}$ と sk^* を $\mathcal{C}$ へ送る. ここで, $\mathtt{TempID}^*$ は UserKeyGen クエリとして送られていない ID とする.
$\mathcal{C}$ はランダムに $b \xleftarrow{\$} \{0,1\}$ を選び, $\mathsf{SendRequest}(gpk, sk^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ と $C^* \leftarrow \mathsf{SendContent}(gpk, \sigma, \mathtt{TempID}, M_b^*, \mathtt{Adr}_{\mathtt{proxy}})$ を実行し, $(\sigma, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{dst}})$ と C^* を $\mathcal{A}$ へ送る.

(4) $\mathcal{A}$ が送付した UserKeyGen クエリ $\mathtt{TempID} \in \mathcal{ID}$ かつ $\mathtt{TempID} \neq \mathtt{TempID}^*$ に対し, $\mathcal{C}$ は UserKeyGen($params$, msk, TempID) を実行し, $dk_{\mathtt{TempID}}$ を返す.

(5) 最後に $\mathcal{A}$ は $b' \in \{0,1\}$ を出力する.

λ に対し $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pro},\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ が無視できる場合, プロトコルは Semantic Security をもつと定義する.

最後に Unforgeability を定義する. Unforgeability は, 署名鍵を持たない攻撃者 $\mathcal{A}$ は ValidityCheck の出力が 1 となる署名を作成できないことを保証する. ここでは, $\mathcal{A}$ は悪意のあるユーザとしてモデル化されている. なお KGC の権限でもそのような署名を作成できないことを保証するために, $\mathcal{A}$ には KGC のマスタ鍵 msk を与えている.

Definition3.5 (Unforgeability)

(1) チャレンジャー $\mathcal{C}$ は $(gpk, ik) \leftarrow \mathsf{GM.Setup}(1^\lambda)$ 及び $(params, msk) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$ を実行し, gpk, msk, $params$ を攻撃者 $\mathcal{A}$ に与える. さらに $\mathcal{C}$ は $\mathcal{S} = \emptyset$ と $\mathcal{S}$ を初期化する.

(2) $\mathcal{A}$ が送付した SendRequest クエリ (i, TempID) に対し, sk_i が生成されていなければ, $\mathcal{C}$ は $sk_i \leftarrow \mathsf{Join}(gpk, ik)$ を実行する. さらに $\mathcal{C}$ は $\mathsf{SendRequest}(gpk, sk_i, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ を実行し, $\mathcal{A}$ へ σ を送り. $\mathcal{S}$ に (σ, TempID) を追加する.

(3) 最後に $\mathcal{A}$ は $(\sigma^*, \mathtt{TempID}^*)$ を出力する. もし, $(\sigma^*, \mathtt{TempID}^*) \notin \mathcal{S}$ かつ $\mathsf{ValidityCheck}(gpk, \sigma^*, \mathtt{TempID}^*) = 1$ であれば, $\mathcal{A}$ の勝利と定義する.

もし $\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{pro},\mathcal{A}}(\lambda) := \Pr[\mathcal{A} \text{ wins}]$ が λ に対し無視できる場合, プロトコルは Unforgeability をもつと定義する.

暗号化匿名通信プロトコルが Correctness, Anonymity, Semantic Security, Unforgeability を満たすならば, 安全な暗号化匿名通信プロトコルであると定義する. なお追加の要件として, SP が送付した暗号文の改ざんや置き換えを行う攻撃も考えられる. この攻撃を防ぐためには, SP が IBE 暗号文に対し署名を作成すればよく, またこの署名は SP の公開検証鍵で検証するため, Anonymity には反しない. このような改良は容易であることから, 本論文では議論しないこととする.

3.2 プロトコル構成

始めに, 提案プロトコルの構成要素として, IBE と OFGS 署名の定義を与える. IBE 方式 $\mathcal{IBE}$ は, (IBE.Setup, Extract, IBE.Enc, IBE.Dec) の 4 つのアルゴリズムで構成されている. $\mathcal{ID}$ と $\mathcal{M}$ はそれぞれ, ID 空間とメッセージ空間とする.

Definition3.6 (IBE のシンタックス [7])

IBE.Setup: セキュリティパラメータ λ を入力とし, 公開鍵 $params$ とマスター秘密鍵 msk を出力する.

Extract: $params$ と msk, $ID \in \mathcal{ID}$ を入力とし, 復号鍵 dk_{ID} を生成する.

IBE.Enc: $params$ と ID, メッセージ $M \in \mathcal{M}$ を入力とし, 暗号文 C_{IBE} を出力する.

IBE.Dec: $params$, C_{IBE}, dk_{ID} を入力とし, M を出力する.

Correctness 要件として, 全ての $(params, msk) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, $ID \in \mathcal{ID}$, $M \in \mathcal{M}$ に対し, $\Pr[\mathsf{IBE.Dec}(params, \mathsf{IBE.Enc}(params, ID, M), \mathsf{Extract}(params, msk, ID)) = M] = 1$ をみたすことを要請する.

OFGS 方式 $\mathcal{GS}$ は, (GS.Setup, Join, Sign, Verify) の 4 つ

のアルゴリズムで構成されている．$\mathcal{M}_{sig}$ を署名対象メッセージ空間とする．

Definition3.7 (OFGS のシンタックス)

GS.Setup: セキュリティパラメータ λ を入力とし，グループ公開鍵 gpk と発行鍵 ik を出力する．

GS.Join: gpk と ik を入力とし，署名鍵 sk を出力する．

Sign: gpk, sk, 署名対象メッセージ M を入力とし，グループ署名 σ を出力する．

Verify: gpk, σ, M を入力とし，σ が M に対して正しい署名，すなわち gpk に対応する ik で作成された sk による M の署名である場合は 1 を，そうでなければ 0 を出力する．

Correctness 要件として，全ての $(gpk, ik) \leftarrow \mathsf{GS.Setup}(1^\lambda)$, $sk \leftarrow \mathsf{GS.Join}(gpk, ik)$, $M \in \mathcal{M}_{sig}$ に対し，$\Pr[\mathsf{Verify}(gpk, \mathsf{Sign}(gpk, sk, M), M) = 1] = 1$ が成り立つことを要請する．

次に，提案暗号化匿名通信プロトコルを記述する．提案プロトコルでは，IBE の公開鍵 TempID をグループ署名の署名対象メッセージ (つまり $\mathcal{ID} = \mathcal{M}_{sig}$) とする．

Construction3.1 (提案暗号化匿名通信プロトコル)

GM.Setup: $(gpk, ik) \leftarrow \mathsf{GS.Setup}(1^\lambda)$ を実行し，(gpk, ik) を出力する．

KGC.Setup: $(params, msk) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$ を実行し，$(params, msk)$ を出力する．

Join: $sk \leftarrow \mathsf{GS.Join}(gpk, ik)$ を実行し，sk を出力する．

UserKeyGen: $dk_{\mathtt{TempID}} \leftarrow \mathsf{Extract}(params, msk, \mathtt{TempID})$ を実行し，$dk_{\mathtt{TempID}}$ を出力する．

SendRequest: $\mathtt{TempID} \xleftarrow{\$} \mathcal{ID}$ を選び，$\sigma \leftarrow \mathsf{Sign}(gpk, sk, \mathtt{TempID})$ を実行し，$(\sigma, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{dst}})$ をプロキシ (IP アドレスを $\mathtt{Adr}_{\mathtt{proxy}}$ とする) へ送付する．

RelayRequest: $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ を Tbl に追加し，$(\sigma, \mathtt{TempID})$ と $\mathtt{Adr}_{\mathtt{proxy}}$ を IP アドレス $\mathtt{Adr}_{\mathtt{dst}}$ の宛先である SP へ送る．

ValidityCheck: もし $\mathsf{Verify}(gpk, \sigma, \mathtt{TempID}) = 1$ であれば 1 を，それ以外は 0 を出力する．

SendContent: もし $\mathsf{ValidityCheck}(gpk, \sigma, \mathtt{TempID}) = 0$ であれば $\perp$ を出力する．それ以外の場合，$C_{IBE} \leftarrow \mathsf{IBE.Enc}(params, \mathtt{TempID}, M)$ を実行し，C_{IBE} を IP アドレス $\mathtt{Adr}_{\mathtt{proxy}}$ のプロキシへ送る．

RelayContent: Tbl 中にある IP アドレスが $\mathtt{Adr}_{\mathtt{src}}$ のユーザへ C_{IBE} を中継する．さらに，Tbl から $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ を削除する．

GetContent: $\mathsf{IBE.Dec}(params, C_{IBE}, dk_{\mathtt{TempID}})$ の結果を出力する．

上記構成はプロキシ 1 台でのものであり，プロキシはユーザ及び SP と直接通信を行っている．そのため，プロキシ視点からは Anonymity は保証されない (なお我々の Anonymity における定義 3.3 には矛盾しないことに注意されたい)．プロキシを複数個定義し，それぞれのプロキシが前後のプロキシに $(\sigma, \mathtt{TempID})$ や C_{IBE} を中継するだけで容易に複数個プロキシ構成に拡張できる．この場合，全てのプロキシが結託しない限り Anonymity が保証されるという利点がある．

4. グループ署名

提案暗号化匿名通信プロトコルは，その構成要素としてグループ署名を利用している．提案プロトコルでは任意のグループ署名を使用することが可能であるが，前述の通りグループ署名のオープン機能を利用しない．本機能を削除することで性能の向上が見込まれるため，提案提案プロトコルではオープン機能のないグループ署名 OFGS を利用している．[*4]．本節では，OFGS の安全性定義を与える．

4.1 OFGS の安全性定義

ここでは，古川-今井グループ署名の定義 (Anonymity, Traceability, Non-Frameability) をオープンフリーに適合するように再定義する．Anonymity では，攻撃者 $\mathcal{A}$ が署名鍵を持っていたとしても，2 つのグループ署名の作成者が同じ署名者か否かを識別できないことを要請していた．ここで Anonymity には，CPA-Anonymity と CCA-Anonymity の 2 種類があることに注意されたい．CCA-Anonymity では，$\mathcal{A}$ はオープンオラクル (グループ署名とメッセージを入力とし，誰の署名鍵で作成されたものであるかを返す) にアクセスできる．今回オープンフリーであるため，CCA-Anonymity に関する検討は不要であることに注意されたい．

Definition4.1 (Anonymity)

(1) 攻撃者 $\mathcal{A}$ は，セキュリティパラメータ λ に対し，gpk, sk_0, sk_1, M をチャレンジャー $\mathcal{C}$ へ送付する．

(2) $\mathcal{C}$ は $b \xleftarrow{\$} \{0, 1\}$ を選択し，$\sigma^* \leftarrow \mathsf{Sign}(gpk, sk_b, M)$ を計算し，σ^* を $\mathcal{A}$ へ送る．

(3) $\mathcal{A}$ は $b' \in \{0, 1\}$ を出力する．

OFGS $\mathcal{GS}$ は，λ に対し $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{GS},\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ が無視できる場合，Anonymity を持つと定義する．

次に我々は Traceability について再定義する．元々の定義では，Traceability は Verify アルゴリズムが 1 を出力するにも関わらず，Open アルゴリズムで署名者を特定できないようなグループ署名を作成できないことを保証していた．今回オープン機能を削除するため，Traceability における攻撃者の勝利条件を変更する必要がある．そこで Traceability の代わりとして，署名鍵を知らずに Verify アルゴリズムが

[*4] OFGS とリング署名 [18] の違いとして，リング署名では 署名者はユーザ公開鍵の集合からメンバを選択し，これらユーザに自身を加えたグループを構成，グループの誰かが署名者であることを証明する．グループ署名と異なり，署名者の Anonymity を無効化する機構は有していない．なお署名者が署名する際には他メンバを周知する必要があり，これは暗号化匿名通信プロトコル用途には不適当であることに注意されたい．

1 を出力するようなグループ署名を作成できないことを保証する安全性として, Unforgeability を以下で定義する.

Definition4.2 (Unforgeability)

(1) チャレンジャー $\mathcal{C}$ は $(gpk, ik) \leftarrow \mathsf{GS.Setup}(1^\lambda)$ を実行し, 攻撃者 $\mathcal{A}$ へ gpk を与える. また $\mathcal{C}$ はリスト $\mathcal{S} = \emptyset$ と初期化する.

(2) $\mathcal{A}$ が送付した署名クエリ (M, i) に対し, もしユーザ U_i に対する署名鍵が作成されていないならば $\mathcal{C}$ は $\mathsf{GS.Join}$ アルゴリズムを実行して sk_i を計算, $\sigma \leftarrow \mathsf{Sign}(gpk, sk_i, M)$ を実行し, σ を $\mathcal{A}$ へ返す. U_i に対する署名鍵が既に作成されていた場合, $\mathcal{C}$ は $\sigma \leftarrow \mathsf{Sign}(gpk, sk_i, M)$ を実行し, σ を $\mathcal{A}$ へ返す. さらに $\mathcal{C}$ はリスト $\mathcal{S}$ に (σ, M) を付加する.

(3) 最後に $\mathcal{A}$ は (σ^*, M^*) を出力する. もし $\mathsf{Verify}(gpk, \sigma^*, M^*) = 1$ かつ $(\sigma^*, M^*) \notin \mathcal{S}$ であれば, $\mathcal{A}$ の勝利であると定義する.

λ に対し, $\mathsf{Adv}^{\mathsf{un}}_{\mathcal{GS}, \mathcal{A}}(\lambda) := \Pr[\mathcal{A} \text{ wins}]$ が無視できる場合, OFGS $\mathcal{GS}$ は Unforgeability をもつと定義する.

最後に, Non-Frameability について再考察する. 元々の定義では, Non-Frameability は攻撃者 $\mathcal{A}$ が 1 人のユーザを除く全て (GM 含む) と結託した状況でグループ署名を生成し, このグループ署名のオープン結果が結託していないユーザとなるようなグループ署名を作成できないことを保証していた. 今回はオープン機能を削除したため, Non-Frameability の考慮は不要である点に注意されたい.

なお古川-今井グループ署名では, Non-Frameability をみたすための方策として, ユーザのみが知る秘密鍵 usk を定義, この usk と署名鍵 sk の両方が揃わないと署名が作成できないように構成している. ここで GM ですら usk を知ることはない (ただしユーザは usk の知識をゼロ知識証明を用いて GM に納得させる. そのため $\mathsf{GS.Join}$ アルゴリズムは通常対話型プロトコルとなる) ことに注意されたい. これらの考察より, Non-Frameability を必要としない今回のような場合, 秘密鍵 usk をグループ署名のシンタックスから省略できることがわかる. これが, 我々の OFGS 定義において $\mathsf{GS.Join}$ アルゴリズムの入力に利用者の秘密鍵を必要とせず, さらに $\mathsf{GS.Join}$ アルゴリズムを非対話型アルゴリズムとして定義可能な理由である.

4.2 提案 OFGS 方式

提案 OFGS 方式は, 古川-今井グループ署名 [11] をベースに構成されている. そこでまず古川-今井グループ署名の構成手法について考察する. 古川-今井グループ署名においては, Boneh-Boyen Short 署名 [6] をユーザ証明書とし, この証明書を保持していることを証明するゼロ知識証明 (3-move Σ プロトコル) を, Fiat-Shamir ヒューリスティック [10] を用いて NIZK 証明に変換したものを使用する. さらにオープン機能を実現するため, ElGamal タイプの CCA 暗号にて証明書を GM の公開鍵で暗号化する. そのために, Decision Diffie-Hellman(DDH) 問題が困難な群を双線型群とは別に用意している. 提案 OFGS 方式ではオープン機能を削除するため, DDH 困難な群を用いる必要はなく, 他の部分については古川-今井グループ署名方式と同様の構成を用いる.

なお自明な OFGS 方式の構成方法として, 署名検証鍵/署名鍵のペア $(\mathsf{VK}, \mathsf{SK})$ に対しグループメンバが SK を共有することが考えられる. しかしながらこの単純な構成では, メンバ削除機能を実現できない [15]. 削除機能は現実のシステムで重要であり, 例えば秘密鍵の紛失などに対応するためには必要不可欠である. 今後削除機能付き OFGS を構成することを見こし, 本論文では古川-今井グループ署名をベースに新たに OFGS 方式を構築する.

Construction4.1 (提案 OFGS 方式)

GS.Setup: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ を素数位数 p の双線形群, g_1, g_2 をそれぞれ $\mathbb{G}_1, \mathbb{G}_2$ の生成元 ($\langle g_1 \rangle = \mathbb{G}_1$ かつ $\langle g_2 \rangle = \mathbb{G}_2$) とし, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ を双線形写像とする.[*5] $\gamma \xleftarrow{\$} \mathbb{Z}_p$ 及び $h \xleftarrow{\$} \mathbb{G}_1$ を選択し, $W = g_2^\gamma$ を計算する. $gpk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h, W, e(g_1, g_2), e(g_1, W), H_3)$ 及び $ik = \gamma$ を出力する. ここで $H_3 : \{0,1\}^* \to \mathbb{Z}_p$ はハッシュ関数であり, 安全性証明時にはランダムオラクルとしてモデル化される.

GS.Join: ユーザ U_i に対して $x_i, y_i \xleftarrow{\$} \mathbb{Z}_p$ を選び, $A_i = (g_1 h^{-y_i})^{\frac{1}{\gamma + x_i}}$ を計算し, $sk_i = (x_i, y_i, A_i)$ を出力する.

Sign: $sk = (x, y, A)$ とする. $\beta \xleftarrow{\$} \mathbb{Z}_p$ を選び, $\delta := \beta x - y$ と定義, $T = Ah^\beta$ を計算する. $r_x, r_\delta, r_\beta \xleftarrow{\$} \mathbb{Z}_p$ を選び, $R = e(h, g_2)^{r_\delta} e(h, W)^{r_\beta} / e(T, g_2)^{r_x}$, $c = H_3(gpk, T, R, M)$, $s_x = r_x + cx$, $s_\delta = r_\delta + c\delta$, $s_\beta = r_\beta + c\beta$ を計算し, $\sigma = (T, c, s_x, s_\delta, s_\beta)$ を出力する.

Verify: $R' = \frac{e(h, g_2)^{s_\delta} e(h, W)^{s_\beta}}{e(T, g_2)^{s_x}} \left(\frac{e(T, W)}{e(g_1, g_2)}\right)^{-c}$ を計算し, もし $c = H_3(gpk, T, R', M)$ であれば 1 を出力し, そうでなければ 0 を出力する.

ここでもし (x, y, A) が正当な証明書であれば, $e(A, g_2^x W) = e(g_1, g_2) e(h, g_2)^{-y}$ が成り立つ. この関係式より, $T = Ah^\beta$ に対し

$$\frac{e(T, W)}{e(g_1, g_2)} = \frac{e(h, g_2)^{\beta x - y} e(h, W)^\beta}{e(T, g_2)^x}$$

が成り立つ. このことから

$$\begin{aligned} &\frac{e(h, g_2)^{s_\delta} e(h, W)^{s_\beta}}{e(T, g_2)^{s_x}} \\ =&\frac{e(h, g_2)^{r_\delta} e(h, W)^{r_\beta}}{e(T, g_2)^{r_x}} \left(\frac{e(h, g_2)^{\beta x - y} e(h, W)^\beta}{e(T, g_2)^x}\right)^c \\ =&R \left(\frac{e(T, W)}{e(g_1, g_2)}\right)^c \end{aligned}$$

が成り立つ.

[*5] 双線型性: すべての $a, b \in \mathbb{Z}_p$ に対し, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$, 非退化性: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ (ここで $1_{\mathbb{G}_T}$ は $\mathbb{G}_T$ の単位元.

ベースとした古川-今井グループ署名方式と比較して, 提案 OFGS 方式では DDH-困難な群の要素 3 つと $\mathbb{Z}_p$ の要素 3 つを削減し, 結果署名サイズが元方式の半分にまで抑えることができた.

5. 実装・評価

5.1 提案プロトコルの安全性評価

利用している IBE 方式が IND-ID-CPA 安全であり, かつ, 利用しているグループ署名方式が Anonymity, Unforgeability をみたしていれば, 提案プロトコルも安全である事を証明可能である. 具体的な安全性証明は ACM SAC 論文 [9] を参照されたい.

Theorem5.1 グループ署名方式が Anonymity をみたしているならば, 提案プロトコルは Anonymity をみたす.

Theorem5.2 IBE 方式が IND-ID-CPA 安全であれば, 提案プロトコルは semantic secure である.

Theorem5.3 グループ署名が Unforgeability をみたしているならば, 提案プロトコルは Unforgeability をみたす.

5.2 提案 OFGS 方式の安全性評価

本節では, 提案 OFGS 方式が Anonymity と Unforgeability をみたしていることを示す. まず, ゼロ知識証明プロトコルについて説明する. 証明者は (T, R) を計算し, 検証者に送る. 検証者はチャンレンジ値 c を証明者へ送る. 証明者は (s_x, s_δ, s_β) を計算し, 検証者へ送る. 検証者は検証式を満たすか否かを検証する. 次に, この 3 パスのプロトコルがゼロ知識であることを示す (ここから提案 OFGS 方式が Anonymity をみたすことが導かれる).

シミュレータは $A \xleftarrow{\$} \mathbb{G}$ 及び $\beta \xleftarrow{\$} \mathbb{Z}_p$ を選択し, $T = Ag_1^\beta$ を算出する. ここで β は一様な乱数を選択している. よって, シミュレータによって生成された T の分布は, 証明者による出力分布と同一である. すなわち, $T \in \mathbb{G}$ に対し, シミュレータは $c, s_x, s_\delta, s_\beta \xleftarrow{\$} \mathbb{Z}_p$ を選択し, $R = \frac{e(h,g_2)^{s_\delta} e(h,W)^{s_\beta}}{e(T,g_2)^{s_x}} \left(\frac{e(T,W)}{e(g_1,g_2)}\right)^{-c}$ を計算する. このとき Transcript $(T, R, c, s_x, s_\delta, s_\beta)$ は実際のプロトコルの Transcript と同じ確率分布となる. 証明する値を知らずにこのような Transcript が生成できることはすなわち, Transcript からは証明する値に関する情報が洩れていないことを示している.

次にプロトコルが知識証明であることを示す. すなわち, $(T, R, c, s_x, s_\delta, s_\beta)$ 及び $(T, R, c', s'_x, s'_\delta, s'_\beta)$ から, SDH ペアを抽出できる extractor が存在することを示す. ここで, $c \neq c'$ であり, 両方の Transcript は検証式を満たすものとする. まず, $\tilde{x} := \frac{s_x - s'_x}{c - c'}$, $\tilde{y} := \frac{(s_x - s'_x)(s_\beta - s'_\beta) - (s_\delta - s'_\delta)(c - c')}{(c - c')^2}$, $\tilde{\beta} := \frac{s_\beta - s'_\beta}{c - c'}$ と定義する. ここで, $\frac{e(T,W)}{e(g_1,g_2)} = \frac{e(h,g_2)^{\tilde{\beta}\tilde{x} - \tilde{y}} e(h,W)^{\tilde{\beta}}}{e(T,g_2)^{\tilde{x}}}$ が成立する. よって, $\tilde{A} = T/h^{\tilde{\beta}}$ に対し, $e(\tilde{A}, g^{\tilde{x}}W) = e(g_1, g_2)e(h, g_2)^{-\tilde{y}}$ が成立することから, $(\tilde{x}, \tilde{y}, \tilde{A})$ を抽出することに成功している. ここから提案 OFGS 方式が Unforgeability をみたすことが導かれる. 詳細な証明については古川-今井方式と同様である事から, ここでは省略する.

5.3 プロトタイプ

本節では, 提案方式の実現可能性と有用性を検証する. まず, 提案方式のプロトタイプを構築し, 本プロトタイプの動作を検証することにより実現可能性を確認する. 次に, HTTP プロキシを経由した通信を実施するケース, その通信を SSL により暗号化したケース, 提案方式を用いて通信したケースのそれぞれの場合の処理時間を比較する. また, 提案方式を用いた通信については, SimpleProxy を用いた通信と Tor を用いた通信のそれぞれの場合に分けて処理時間を比較する. これらの比較により, 提案方式の有用性を考察する.

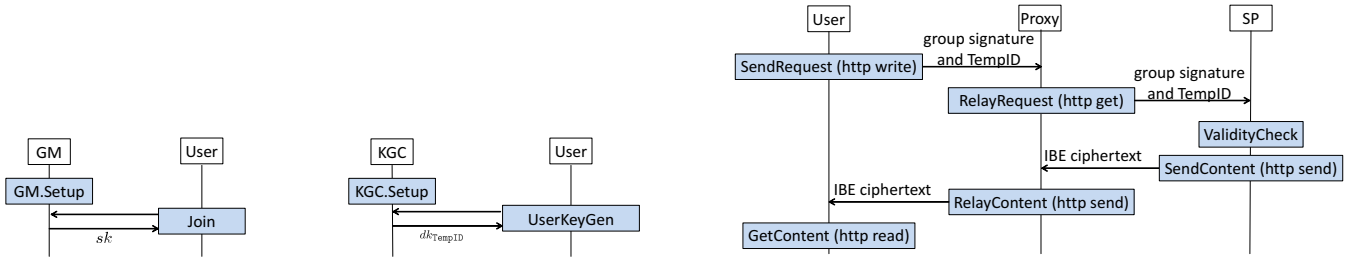
5.3.1 実装

User と SP 用のモジュールを GCC version 4.2.1 を用いて実装した. ここで, Boneh-Franklin IBE 方式, および提案 OFGS 方式の実装には TEPLA ライブラリ [4] を用いた. 本ライブラリは, 254-bit 素数位数を持つ Barreto-Naehrig (BN) 楕円曲線 [5] 上で動作する Optimal Ate ペアリングをサポートし, その埋め込み次数は 12 である (128 ビットセキュリティを保証). プロキシには Tor (version 0.2.4.20) を利用したが, 比較のためプロキシに HTTP プロキシサーバである Simpleproxy [3] を利用した際の実装評価 [9] についても併記することとする.

本実装は, 3 節のフレームワークに従い, 図 2 に示す 3 種類の通信手続きを実現する. ただし署名鍵を付与するため, User-GM 間の手続きは User-プロキシ-SP 間の手続きの前に実行されている必要がある. なおユーザが IBE 復号鍵を取得するために, User-プロキシ-SP 間の手続き中の GetContent モジュールが実行される前に User-KGC 間の手続きが完了する必要はあるものの, User-KGC 間と User-プロキシ-SP 間の手続きは並列に実行しても問題ないことに注意されたい.

5.3.2 性能評価

今回の実装では, Apple MacBookPro (CPU: 2.8GHz Intel Core i7, Memory: 8GB, 1067 MHz DDR3, Darwin Kernel Version 12.4.0) 上にて VM (VMware Fusion 5.0.3) を用いた. MacBookPro 上で User と Tor 接続ツール, VM 上で SP と Tor ツールをそれぞれ動作させる. Proxy の VM は FreeBSD amd64 9.1-RELEASE (1CPU, メモリ 256MB) とし, SP の VM は, CentOS 5.9 x86-64 (1CPU, メモリ 512MB) を使用した. また Tor ネットワークを利用する際には SOCKS proxy を利用し, 長期間リレーしている TOR ルータ 3 台を自動的に選択してルーティンを実施する設定



(a)User-GM (b)User-KGC (c) User-Proxy-SP

図 2 Communication Sequences

とした.

以下, それぞれ (1) HTTP 通信 (暗号関係の演算を含まない), (2)SSL 通信, (3) 提案プロトコル (Simpleproxy), (4) 提案プロトコル (Tor) に対し, 1 セッション (User→Proxy→SP→Proxy→User) の計測時間を表 1 に示す. SSL 通信の実行時間の計測については, OpenSSL ライブラリ (ver. 1.0.1e) [1] の s_server/s_time コマンドを利用した. なお提案方式が 128-bit 安全性を持つことを鑑み, 3072 ビット公開鍵を利用した DHE-RSA-AES128-SHA256 暗号スイートを用いた.

表 1 Running Time (one session)

Scheme	Time(msec)	Cryptographic Operations
None	4.714	-
SSL	12.897	Enc/Auth
Simpleproxy	624.743	Enc/Anon. Auth
Tor	3139.4	Enc/Anon. Auth

提案プロトコル (Simpleproxy) の場合の実行時間は, SSL 通信の場合の約 50 倍となっている. その原因として, 提案プロトコルでは (SSL で使われている通常の公開鍵暗号やデジタル署名とは別に) ペアリング計算を必要としていることが挙げられる. しかしながら提案プロトコルの実行時間はミリ秒オーダ内で収まっており, SSL では実現できていない匿名性をサポートしている点に着目されたい.

提案プロトコル (Tor) の場合の実行時間は, 提案プロトコル (Simpleproxy) の場合の実行時間の約 5 倍となっている[*6]. その原因として, 匿名通信を実現するために複数の Tor サーバを経由し, さらに経由する Tor サーバの接続条件によって実行時間が大きく影響を受けることが挙げられる.

次に各アルゴリズムの実行時間を表 2 に示す. なお GM.Setup, KGC.Setup, Join アルゴリズムはオフラインで実行でき, さらに UserKeyGen アルゴリズムはセッションとは独立に実行可能である. また RelayRequest と RelayContent アルゴリズムは通信の中継を行うのみであり, 暗号処理を行わないため, 表 2 には記述しない.

表 2 Running Time (algorithms)

Algorithm	Time(msec)	Entity
GM.Setup	105.712	GM
KGC.Setup	102.883	KGC
Join	109.036	User-GM
UserKeyGen	102.958	User-KGC
SendRequest	125.069	User
ValidityCheck	199.247	SP
SendContent	198.636	SP
GetContent	101.158	User

計測結果より, ユーザが実行するアルゴリズムの中で, グループ署名を計算する SendRequest アルゴリズムが最も非効率であることがわかる. しかし通信を行う前にオフラインで TempID の選択及びグループ署名の計算を行うことができ, その場合 1 セッションの合計実行時間を 500msec 程度に抑えることができる.

5.4 考察

本節では, 提案暗号化匿名通信において Tor をプロキシとして使用した場合におけるインターネットへの適用可能性について考察する. Tor はインターネット上の通信を匿名化するサービスの 1 つであり, 利用者の送信元情報を秘匿するとともに, サービス提供者側の宛先情報も隠蔽が可能である. これらの特徴を活かしたサービスは知られておらず, むしろ犯罪の温床と成り得るなど暗いイメージを社会に抱かせているのが現状である.

提案暗号化匿名通信では, 利用者の送信元情報を秘匿しつつもその身元 (=ある権限を有するグループに属していること) を保証し, かつ通信を暗号化することを目的としている. 利用事例として, 論文 [20] で挙げられているリスク評価システムへの適用が考えられる. このリスク評価システムでは, ユーザがスマートフォン等にインストールされ

[*6] 本測定では Simpleproxy は PC 内部の仮想ネットワークにて実験ネットワークに接続されているため, Tor ルータを利用したケースと比較する際には, インターネット上での伝搬遅延を考慮する必要がある. しかしながら, 最大 500 ms の伝搬遅延を考慮しても Tor ルータを利用したケースより Simpleproxy を利用したケースのほうが格段に速いことに留意されたい.

ているアプリを分析サーバに送付し, サーバがアプリのリスク評価を行いユーザに返すという流れで実施される. ここで “どのようなアプリを使用しているのか” という情報はユーザの趣味嗜好に関わるため, 個人と紐づいて公にされるべき情報ではない. しかしながら完全に匿名とすると, サービスを利用する権限を有するユーザかどうか分析サーバ側で判断できないという問題が発生する. さらにリスク評価結果は対象ユーザのみが知るべき情報であり, 第三者が知るべきものではないことから, 暗号化されることが望ましい. 提案暗号化匿名通信プロトコルを利用することで, これらの問題を解決することができる. なお本用途では分析サーバは匿名である必要がないため, ユーザはサーバの公開鍵を用いてアプリ情報を暗号化し, 提案暗号化匿名通信プロトコルにおいてグループ署名の署名対象メッセージに暗号化されたアプリ情報を含めればよい.

6. 結論

本論文では, 暗号方式 (グループ署名及び ID ベース暗号) 及び匿名通信プロトコル (Simpleproxy または Tor) を利用した暗号化匿名通信プロトコルを提案し, そのプロトタイプを実装・評価することによりそのフィージビリティを検証した. また, 提案方式をインターネットに展開する際の問題点を検証すべく, 提案方式の Tor サービスへの適用可能性を評価した. その結果, 提案方式は従来の SSL 通信などに比べて処理時間が多くかかり, 提案方式の高速化の余地は未だ大きいものの, Tor ネットワークの利用により生じる処理時間と比較すると微小であることから, その適用可能性は十分高いと判断した.

我々は, 本研究が暗号化匿名通信の利便性を大きく向上することができると考えている. 例えば, 匿名性通信システム Tor [13], [17], [21] は現在最も広範囲に利用されている匿名通信路であるが, ユーザの権限を確認する機能がないために悪用されるリスクが存在し, 犯罪の温床となることが危惧されている. 匿名性を担保しつつも不当ユーザの利用を拒絶することができる提案方式を適用することにより, Tor の安全な利活用が可能となり, その利便性が格段に向上することが期待される. 本研究を通じ, インターネットでの暗号化匿名通信の実現に貢献していきたいと考えている.

謝辞

執筆において貴重なコメントを頂いた花岡悟一郎氏, 大久保美也子氏に感謝する.

参考文献

[1] OpenSSL: Cryptography and SSL/TLS Toolkit. `http://www.openssl.org/`.

[2] Selected papers in anonymity. `http://freehaven.net/anonbib/date.html`.

[3] Simpleproxy: Crocodile group software. `http://www.crocodile.org/software.html`.

[4] TEPLA: University of Tsukuba Elliptic Curve and Pairing Library. `http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html`.

[5] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[6] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[7] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[8] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[9] K. Emura, A. Kanaoka, S. Ohta, and T. Takahashi. Building secure and anonymous communication channel: Formal model and its prototype implementation. In *ACM Symposium On Applied Computing*, 2014.

[10] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[11] J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions*, 89-A(5):1328–1338, 2006.

[12] Y. Gilad and A. Herzberg. Plug-and-play IP security - anonymity infrastructure instead of PKI. In *ESORICS*, pages 255–272, 2013.

[13] A. Houmansadr, C. Brubaker, and V. Shmatikov. The parrot is dead: Observing unobservable network communications. In *IEEE S&P*, pages 65–79, 2013.

[14] M. Z. Lee, A. M. Dunn, B. Waters, E. Witchel, and J. Katz. Anon-pass: Practical anonymous subscriptions. In *IEEE S&P*, pages 319–333, 2013.

[15] B. Libert, T. Peters, and M. Yung. Group signatures with almost-for-free revocation. In *CRYPTO*, pages 571–589, 2012.

[16] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Trans. on Information Theory*, 57(3):1786–1802, 2011.

[17] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. SkypeMorph: protocol obfuscation for Tor bridges. In *ACM CCS*, pages 97–108, 2012.

[18] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565, 2001.

[19] A. Sudarsono, T. Nakanishi, Y. Nogami, and N. Funabiki. Anonymous IEEE802.1X authentication system using group signatures. *JIP*, 18:63–76, 2010.

[20] T. Takahashi, K. Emura, A. Kanaoka, S. Matsuo, and T. Minowa. Risk visualization and alerting system: Architecture and proof-of-concept implementation. In *SESP '13*, pages 3–10. ACM, 2013.

[21] Tor Project. Tor project: Anonymity online. `https://www.torproject.org/`.