# 安全性が証明可能な
# 追跡不能アクセス制御プロトコルと
# デジタルコンテンツ流通への応用
# に関する研究

申吉浩

平成18年11月6日

# 目次

# 図目次

# 表目次

# 第1章　序論

## 1.1 ユビキタス情報社会とプライバシー

「ユビキタス情報社会」という言葉が人口に膾炙するようになって久しく、その実現に向けて産学官の取り組みも本格化してきている。市場では、より高機能を追求した携帯電話や携帯端末が続々と導入され、商用のモバイル・ネットワーク環境も急速に整備されつつある。大学等の研究機関では、ユビキタス情報社会を実現する基礎技術として、通信技術やセンサー技術の研究が盛んである。また、総務省においても、「ユビキタスネットワークの将来展望に関する調査研究会 (2001 – 2002)」、「ネットワーク・ロボット技術に関する調査研究会 (2002 – 2003)」、「ユビキタスセンサーネットワーク技術に関する調査研究会 (2004 – 2005)」、「安心・安全な社会の実現に向けた情報通信技術のあり方に関する調査委員会 (2006 –)」と来るべきユビキタス情報社会のあり方に関する調査研究活動を加速している。

さて、ユビキタス情報社会を技術的に表現する用語としては、「ユビキタスコンピューティング」が一般的である。ユビキタスコンピューティングの概念は、坂村健東京大学教授が 1980 年代に TRON プロジェクトにおいて提唱した事例をもって嚆矢とする。一方、Ubiquitous Computing という用語そのものは、米 Xerox 社 PARC 研究所の Mark Weiser が Scientific American 誌に発表した論文「The computer for the 21st Century (1991)」[63, 64] に由来する。この記念碑的な論文において、Weiser は、人々の生活にとけ込んだコンピュータがいかに魅力的な効用を提供し得るかを活々とした筆致で描き、その命名の妙もあって、Ubiquitous Computing は瞬く間に IT 研究者の耳目を集めるところとなった。爾来、IT 技術の一つの集大成と目され、長年にわたる熱心な研究の対象となっている。近年になり、インターネット・無線通信・携帯端末等の技術が飛躍的な進歩と普及を遂げるに伴い、前述したように、ユビキタスコンピューティングは研究室を飛び出し、本格的な実用化に向けた道を歩み始めている。

さて、前掲の論文において、Weiser は、21 世紀の社会におけるコンピュータは invisible であるべきことを強調している。技術にとっての最も明確な成功の証は、技術を利用するためのリテラシが社会に浸透している事実である。例えば、万人が文字を自由に理解するようになったことにより、「キャンディの包み紙さえも文字で埋め尽くされ (Candy wrappers are covered in writing [63, 64])」、その効用が遍く社会に行き渡ることとなった。即ち、技術は、意識されなくなる程度にまでリテラシが社会に浸透することにより、最大限の効用を提供し得るのである。これに論拠を得て、Weiser は、コンピュータ利用のためのリテラシの習得を一般の人々に強要するのではなく、逆に、コンピュータが「背景にとけ込む (vanish into the background [63, 64])」、即ち、既存のリテラシをコンピュータが取り込むことで、人々の生活を直接に支援するように進化するべきであると主張する [63, 64, 61]。

実際には、識字の普及は営々たる社会的努力の果実であり、また、氏が「アイデア自体が的外れである (the idea of a "personal" computer itself is misplaced, and that the vision of laptop machines, dynabooks and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology)」と批判した PC が人々と情報との関わりを根本的に変革してきたことは否定のしようのない事実であるので、Weiser の議論には聊か牽強付会の部分があることは否定できないが、コンピュータによる全く新しい効用を追求するバーチャルリアリティや人工知能に対する強烈なアンチテーゼとして、「何も難しいことをしなくても、現在のコンピュータの能力で人々の生活を十分以上に豊かに変革できる」ことを魅力的に示した Weiser の論文の意義は大きい。

では、Ubiquitous の原義である遍在とはどういう意味であろうか。ユーザがコンピュータを携帯し、場所に制約されずに利用する、所謂、モバイル・コンピューティングも「遍在」といっていえないわけではない。実際には、ユビキタスコンピューティングの本質は、ユーザが携帯するコンピュータと環境の

一部となった数多くのコンピュータとが、相互に協調・連携しながらサービスを実現する点にあり、単に、ユーザのコンピュータが場所を選ばずネットワークに接続可能であることを求めるモバイル・コンピューティングとは、格段に積極的な概念である。モバイル・コンピューティングは、ユビキタスコンピューティングの前提であるといってもよい。

このように、移動するコンピュータと環境に埋め込まれたコンピュータとの相互作用としてユビキタスコンピューティングを捉えた場合、最重要のキーワードは「シームレス（seamless）」である。

環境側のコンピュータは、その所属するドメインに依存して、運用ポリシー、機器構成、ネットワーク構成、サービス種別等が異なる。ユーザがドメインからドメインへと移動した場合にも、携行するコンピュータのコンフィギュレーションやデバイスの変更をユーザに強制することなく、コンピュータ間の相互作用を保証し、サービスをシームレスに提供し続けることができること、これがユビキタスコンピューティングの最も基本的な要件である。

総務省の政策懇談会が公表している「u-Japan 政策」では、ユビキタスの実現によって、ユーザは「ネットワーク」、「端末」、「サービス及びコンテンツ」、「ネットワークリスク」の 4 項目において「制約から解放される」としているが、これは、この 4 項目においてシームレスな機能提供が必須であることを述べているに他ならない。文才溢れる Weiser は、この問題を、「Nomadic Issues（遊牧民の課題）」と表現している。

サービスのシームレスな提供は、ユーザに大きな利便を提供する一方、プライバシーに対して深刻な脅威となり得る諸刃の剣でもある。サービスのシームレスな提供は、サービスの提供者の側から見れば、シームレスなアクセス制御を意味する。現行のアクセス制御技術では、個人の身許の認証を前提とすることが通常であり、そのため、ユーザのプライバシーは否応無く露出せざるを得ない。しかも、商業的であれ公共的であれ、適正な提供のためにアクセス制御を必要としないサービスは存在しないといってよいのである。例えば、商業的な観点からは対価を支払った利用者にのみサービスが提供されるべきことは資本主義の大前提であり、個人情報の閲覧や証明書の発行等の公共サービスを本人以外が利用することの脅威は広く認識されている。

このように、ユビキタス情報社会では、サービスや情報資源の「安全」の観点から、アクセス制御そのものもシームレス、透過的、統合的、広範囲に行われることとなるが、これは個人のプライバシーにとっての「安全」とは相反するのは明らかである。収集された個人情報はそれこそ「シームレス」にシステムによって共有される危険があるからである。警句的な表現をとるならば、ユビキタス情報社会は、「市民がいつ何処で何をしたか」を当局が不断に監視し得る、検閲社会である可能性もあるのである。

## 1.2 アクセス制御とプライバシー保護の境界に関する問題

前節で述べたように、プライバシーの保護なしでは、ユビキタス情報社会は検閲社会に陥る危険性が高い。プライバシーの問題は人々の中心的な関心であるという調査報告もある [6, 43]。では、ユビキタス情報社会におけるプライバシー保護は、いかにあるべきであろうか？

一般に、アクセス制御におけるプライバシー保護の概念として、匿名性（Anonymity）と追跡不能性（Unlinkability）の二つが重要である。

匿名性をサポートするアクセス制御は、サービスへのアクセスに際して、身許を明らかにすることをユーザに求めない。一見、ユーザに依存してアクセスの可否を制御するアクセス制御は、匿名性と相矛盾する概念であるようにも思えるが、実際にはこの直感は正しくない。サービスの提供の可否を判断するためにユーザを認識する場合、その目的は必ずしもユーザの身許を知ることではなく、「適切な権限者によってサービスの提供が許諾されている」事実の確認が第一義であるからである。例えば、商用サービスの多くでは、サービスの提供者は、ユーザが対価を支払っているかという事実にのみ興味があり、ユーザの身許を知ることは本意ではなかろう。このようなケースでは、個人の識別情報を知ることは必要ではないだけではなく、認証の目的に照らして十分ですらない（身許を特定できても、サービスへのアクセスを許可するべきか否かは判断できない）。また、個人情報保護法の施行により、必要でもない個人情報を取得することのリスクもある。つまり、注意深く考えれば、多くのケースで、ユーザの身許を完全に隠蔽したとしても、検証者の目的は十分に達せられ、そして、個人情報を取り扱わないことにより、法的、社会的リスクを軽減するという積極的な効果が得られることが分かる。ここに匿名によるアクセス制御の意義がある。

追跡不能性は、匿名性を更に進めた概念である。単に個々のアクセスが匿名で行われることを要求するだけではなく、2つの独立したアクセスが同一のユーザによるという事実さえも隠蔽されることを要求する。例えば、匿名性を実現する手段としては仮名（**Pseudonym**）の利用が考えられるが、一連のアクセスイベントが同一のユーザによって行われた事実を隠蔽することはできない。アクセス制御が、個々のアクセスイベントにおける匿名性のみをサポートし、アクセスイベントの追跡を許すとすると、実際にはユーザの匿名性すらも侵害される危険がなお残る。アクセスのパターンから個人を特定できるかも知れないし、また、特定のアクセスイベントにおいて個人が特定された場合、それを手掛りに、芋蔓的に他のアクセスイベントにおける匿名性が破られる惧れもある。

このように、アクセス制御におけるプライバシー保護に完全を求める場合、追跡に利用可能な情報がユーザの手許から一切漏洩することのない、絶対的な追跡不能性を要件とするのが正しい。

一方、サービスの保護、或いは、社会の安全の観点から、匿名性や追跡不能性を優先することが困難な場合があることは、前節でも述べた。戸籍の閲覧のために身分証の提示を求めることは防犯上合理的であり、また、米国の公安当局による通信の傍受はテロの予防に効果をあげているとされる。

同様に、個人の観点からも、プライバシーの完全な保護が、必ずしも、金科玉条であるわけではない。Palen 等は、論文 [55] において、プライバシーの問題について以下のように考察している。

> *Privacy management is not about setting rules and enforcing them; rather, it is the continual management of boundaries between different spheres of action and degrees of disclosure within those spheres. Boundaries move dynamically as the context changes. (中略) The significance of information technology in this view lies in its ability to disrupt or destabilize the regulation of boundaries.*

確かに、プライバシーとパブリシティー (publicity) との間の境界は、コンテキストにより動的に変動する。人間は、「周囲との協調の意思を示すために、また、場合によっては、自身が周囲とは際立った存在であることを誇示するために、自らの意見や行動に関する情報を発信する (disclose or publicize information about ourselves, our opinions and our activities, as means of declaring allegiance or even of differentiating ourselves from others [55])」からである。

このように、他を犠牲にしてもプライバシーの保護を最優先にするという方針は、「プライバシーとパブリシティーの境界はいかにあるべきか」という問いかけに対する模範解答とはなりえない。プライバシーを考えるコンテキストによって、複数の解答が存在する筈である。

では、ユビキタス情報社会のアクセス制御というコンテキストにおいては、何を解答とするべきであろうか？この問いは、ユビキタス情報社会を考える上で、最重要の問題のひとつである。

## 1.3 本論文における研究課題

本論文の主たる研究テーマは、ユビキタス情報社会におけるプライバシーに対する要件を明確にし、かつ、その要件をサポートするアクセス制御方式を提案することであり、当然、1.2 で述べたプライバシーとパブリシティーの境界に関する問いへの回答も含まれる。

本論文では、ユビキタス情報社会においてはドメインを横断するアクセス制御インフラストラクチャが提供され、プライバシーとパブリシティの境界に関しては、以下の条件を満たすべきであると主張する。

- アクセスのための認証が透過的かつシームレスに実行される場合には、インフラストラクチャはアクセスが絶対的に追跡不能であることを保証する。
- アクセスの追跡を可能とする情報がユーザの端末からインフラストラクチャを経由して開示される場合には、ユーザの明示的な同意を必須とする。また、インフラストラクチャを経由して開示される追跡情報は必要最小限とし、付加的な情報が必要な場合にはアプリケーションが独自に対応する。

本論文では、プライバシーとパブリシティの境界に関する上記の考え方を、**Consensual Disclosure** と名付け、ユビキタス情報社会におけるプライバシーの基本原則として強く主張する。

この観点から、本論文の問題意識の一端を述べると、ベースラインとして絶対的な追跡不能性を保証し、かつ、サービスと情報資源の保護とプライバシーのバランスの観点からは、Consensual Disclosure を機能としてサポートするアクセス制御インフラストラクチャのための技術を提案することを目的とする。

しかしながら、本論文の問題意識は、追跡不能性と Consensual Disclosure に限定されるものではない。ユビキタスコンピューティングにおいてアクセス制御インフラストラクチャに求められる要件をより広い視点から考察し、更に、それらの要件をサポートするための具体的な技術的提案を行う。

最後に、本研究は、経済産業省商務政策局「平成 17 年度新世代情報セキュリティ研究開発事業（平成 17・11・25 財情第 2 号）」及び「平成 18 年度新世代情報セキュリティ研究開発事業」の研究として行われたものであることを附記する。

# 第2章　ユビキタス・アクセス制御関連技術と未解決問題

## 2.1 ユビキタスコンピューティングにおけるトラスト管理

典型的なアクセス制御のシステム構成では、認証 (entity authentication) のレイヤの上に、アクセス制御の機能を構築する。即ち、システムがユーザからアクセスの要求を受けると、公開鍵基盤 (Public Key Infrastructure) によりユーザをまず認証し、次いで、ユーザの識別名を鍵として ACL (Access Control List) を検索して、ユーザに許諾されているアクセス権限を取得する。

認証とアクセス制御を分離するこの構成をユビキタスコンピューティングに適用するには、不都合がある。これは、ユビキタスコンピューティングでは、アクセス権限を付与する主体 (権限発行者) が必ずしもサービスが提供される同一のドメインに帰属するとは限らないという事情による。ユーザによるシームレスで透過的なアクセスを保証するためには、ユーザへのアクセス権限の付与の事実を即座にアクセス定義テーブルに反映することが必要となる。管理者がドメイン内のサービスやリソースを管理するのであれば、ユーザへのアクセス権限の発行と同時に、ドメイン内の ACL を更新すればよいであろうが、権限発行者のドメインとサービスが提供されるドメインが異なる場合、ドメインをまたいで分散するかもしれないアクセス定義テーブルを即座に更新することは容易でない。この問題は、同一のサービスが特定多数或いは不特定多数のドメインを横断して提供される、ユビキタスコンピューティングで一般的に発生すると想定されるケースでは、特に顕著であろう。

このような事情に基づき、ユビキタスコンピューティングでは、以下の要件を設けて、この問題を解決することが提案されている [33, 10, 9]。

- 認証とアクセス制御とを統合されたプロセスとして取り扱う (Authorization)。
- アクセス権限の委譲 (Delegation) を公開鍵基盤に準拠する既存の信頼管理方法論に統合する。

以下では、SPKI [33] と Distributed Trust Management [10, 9] の二つの重要な事例により、上記のアプローチを考察する。

### 2.1.1 SPKI (Simple Public Key Infrastructure)

Simple Public Key Infrastructure (SPKI) [33] は、Authorization Certificate と Authorization Certificate の譲渡 (Delegation) の概念を導入することにより、前掲の要件を満足することを狙う。

SPKI の Authorization Certificate は、公開鍵ペアとアクセス権限とを結びつける内容を有し、Authorization Certificate に基づいて認証を実行することにより、証明書の主体 (subject) に関して、以下の 2 つの事実が同時に確認される。

- 証明書に指定される公開鍵に対応する個人鍵を保持している。
- 証明書に指定されるサービスやリソースに対するアクセス権限が付与されている。

即ち、Authorization Certificate は、ユーザ毎の ACL (Access Control List) を記載した公開鍵証明書であると考えることができる。従って、検証者は Authorization Certificate を認証した時点で、ユーザのアクセス権限を知ることが可能となり、別に設けられたアクセス定義テーブルを参照する必要がなくなる。

このように、Authorization Certificate の利用でアクセス定義テーブルの不整合の問題を解決することが可能となるが、次には、Authorization Certificate の発行そのものが問題になる。

即ち、Authorization Certificate の発行は認可 (authorization) の行為に他ならないが、認可の行為にもアクセス制御が必要だからである。実際、発行や無効化を含むアクセス権限の変更は、ACL というリソースの変更に他ならず、ACL へのアクセスの可否は別の ACL で定義されなければならない。このように、現実には、リソースの管理構造や組織の構造を反映して、ACL はネスト構造に構造化されることが通常である。

SPKI では、Authorization Certificate の主体が新たに Authorization Certificate を発行し、オリジナルの Authorization Certificate に記載されているアクセス権限に基づいて、他のユーザにアクセス権限を付与する機能 (delegation) を実現することにより (証明書の主体がアクセス権限を委譲できるか否かの属性も同じ証明書中に指定される)、ネスト構造をもつ ACL (Access Control List) による集中的なアクセス制御に起因する問題を回避し、アクセスの分散管理を可能にする [33]。

### 2.1.2 Distributed Trust Management

*Distributed trust management* の概念は、Blaze 等 [10, 9] によって最初に導入された。

Distributed Trust Management は、SPKI と同様に「認証とアクセス制御の統合」と「アクセス権限の委譲」の 2 項目を達成すると共に、SPKI の枠組みを包含するより柔軟な定式化を提案し、更には、ドメイン毎のローカルポリシーを整合させる具体的なシステムを提案した点に重要な意義がある。

実際、Distributed Trust Management は、ユビキタスコンピューティングにおけるトラスト管理のフレームワークとして広く受け入れられており、Distributed Trust Management に準拠したシステムの実装例も報告されている [44, 45]。

以下では、まず、Blaze 等による *PolicyMaker* システムの概要を紹介し、その後に、その意義について考察する。

ユーザは、ドメイン内に散在する Service Agent を介して、サービスやリソースへのアクセスを行う。Service Agent は、ユーザからサービスやリソースへのアクセスの要求を受けると、ドメインのポリシーを管理する *PolicyMaker* と協調して以下の手順で処理を行う。

1. Service Agent は、Query を *PolicyMaker* システムに発行する。
2. Query を受け取った *PolicyMaker* は、Proof を生成して返信する。
3. Service Agent は、Proof を評価して、アクセスを要求しているユーザに対して要求しているアクセスを許可するべきか否かを判断する。

*PolicyMaker* に発行される Query は、以下の形式をとる

$key_1, key_2, \ldots, key_n$ REQUESTS *ActionString*

ここで、*key* は、ユーザが対応する個人鍵を保持していることが想定される公開鍵であり、*ActionString* はユーザが要求しているアクションとする。

一方、*PolicyMaker* が生成する Proof は、ひとつ以上の *Assertion* から構成される。それぞれの *Assertion* は、以下の形式をとる。

*Source* ASSERTS *AuthorityStruct* WHERE *Filter*

Assertion は、「*Filter* が指定するコンテキストにおいて、*Source* が *AuthorityStruct* を保証する」という意味をもつ。Assertion の各フィールドは以下のように定義される。

- Assertion には、Policy Assertion と Signed Assertion の区別がある。

  Policy Assertion の *Source* には、適用されるローカルポリシーの識別子が指定される。Policy Assertion は常に真と評価され、Proof における公理の役割を果たす。

  一方、Signed Assertion の *Source* には公開鍵が指定される。Signed Assertion が真と評価される必要かつ十分条件は、*Source* に指定された公開鍵が信頼でき、かつ、Assertion に付された署名がこの公開鍵を使って検証できることとする。

- *AuthorityStruct* は、この Assertion により保証される、ひとつ以上の公開鍵を指定する。*AuthorityStruct* に指定される公開鍵は、公開鍵の単純な集合であってもよいし、何らかの構造が与えられることもある。例えば、指定された公開鍵中の任意の $k$ 個の部分集合という指定も可能である。

- *Filter* は、*ActionString* を入力として真或いは偽を返す命題である。

Query を入力された *PolicyMaker* が出力する Proof は、以下の条件を満足する非循環有向グラフである。

- Assertion をグラフの各頂点 (vertex) とする。

- 有向グラフの始点は常に Policy Assertion である。始点とは、その頂点を終端とする辺 (edge) が存在しない頂点である。

- 有向グラフの各辺において、終端の Signed Assertion の *Source* に指定される公開鍵は始端の Assertion の *AuthorityStruct* により保証される。

- 有向グラフの終点の Assertion は、Query に指定される公開鍵を保証する。終点とは、その頂点を始端とする辺が存在しない頂点である。

*PolicyMaker* が出力する Proof は、上記の構成をとった上で、以下の条件を満足する時に、真と評価される。Proof が真と評価された時に、Service Agent は Query で指定される公開鍵で認証されるユーザに対して、*ActionString* が指定するアクションを許可する。

- 全ての Assertion が真と評価される。

- 全ての頂点の Assertion に指定される *Filter* が、*ActionString* を入力として、真と評価される。

Blaze 等による上記の Assertion は、以下のように、X.509 証明書、Authorization Certificate、及び、証明書の委譲を包含する。

- Signed Assertion が、*AuthorityStruct* が単一の公開鍵を指定し、かつ、*Filter* が恒真命題 (入力によらず常に真を返す命題) であるとき、Signed Assertion は X.509 証明書と意味的に同等である。
- *Filter* は、*ActionString* を入力として、真偽を返す命題であり、Service Agent は命題の値が真である場合に *ActionString* を許可する。即ち、*Filter* は、*AuthorityStruct* が識別するユーザのアクセス権限を指定していると考えてよい。
- *Source* の公開鍵の所有者として、認証局ではなく、ユーザを指定することもできる。そのユーザは *Filter* が指定するアクセス権限を委譲する場合、ユーザは *Source* に自身の公開鍵を指定して、Assertion を生成する。ユーザにアクセス権限を委譲する権限が付与されているか否かは、Proof 中で直前に位置する Assertion の評価により判断される。

Blaze 等による *PolicyMaker* の提案は、認証において従来区別せずに考えられてきた、検証と評価を明示的に分離した点に意義があると考えられる。

検証とは、証明書 (鎖) に付された署名を検証する処理を指す。

一方、評価とは、署名の検証の結果を具体的にどのような行為に結びつけるかを決定するプロセスである。通常の X.509 証明書による公開鍵基盤においても、検証と評価とは必ずしも一致しない。証明書はそれを発行した認証局のポリシーによってその信頼性が決定されるからである。認証局のポリシーは Certificate Policy/Certification Practices Statements [29, 28] として公表されるが、その中には証明書発行に先立つ登録手続きとして、例えば、身許の証明としてどのような証拠を求めるか等の規定が記載される。例えば、身許の証明に写真付き公文書 (運転免許証等) を求める認証局が発行した証明書と、電子メールで身許の確認を行う認証局が発行した証明書とでは、その信頼性には大きな隔たりがあるのは当然である。従って、証明書 (鎖) の検証者は、単に署名の検証を行うだけでは十分ではなく、証明書を発行した認証局のポリシーを評価して、証明書の信頼性を判断する必要がある可能性がある。

Authorization Certificate のように、アクセス制御の機能を併せ持つ証明書では、検証と評価の区別は益々明瞭である。検証は機械的に実行できるのとは対照的に、評価のプロセスはドメインのローカルポリシーに強く依存するからである。

Blaze 等の *PolicyMaker* システムでは、検証と評価を明確に区別する。即ち、検証は Service Agent が実行し、評価は *PolicyMaker* が Proof を生成することで実行することで、ドメインのローカルポリシーの一貫性を図っているのである。

## 2.2 下位通信層における匿名性と追跡不能性

本論文で提案するアクセス制御プロトコルは、OSI (Open Systems Interconnection) 参照モデルでいうところのアプリケーション層で機能提供される。当然、システムによるユーザの追跡を不能とするためには、アプリケーション層でのみ追跡不能性をサポートし得ても不十分であり、セッション層、トランスポート層、ネットワーク (IP) 層、データリンク層の全ての下位通信層で追跡不能性をサポートする必要がある。実際、下位通信層では、IP アドレスや MAC アドレス等のアドレスを手がかりにパケットやデータグラムを配達するが、これらのアドレスをユーザの追跡に利用することが可能である。

この節では、セッション層から下位の階層において、追跡不能性を実現するための手法を見ることとする。これらの手法は、その性質によって、*Deducible Unlinkability* か *Trusted Unlinkability* のいずれかに分類することが可能である。以下では、$V$ と $P$ との間の通信において、$P$ は通信における追跡不能性を要求するものとして、Deducible Unlinkability と Trusted Unlinkability について説明する。

**Deducible Unlinkability** 下位通信層が仮に何らかのアドレスをパケットの配達に利用するとしても、そのアドレスをユーザの追跡に利用できないことが理論的に演繹できるレベルの安全性を指す。仮名アドレス *pseudonyms* を用いる手法と、ブロードキャスト通信 *broadcasting* を用いる手法が知られている。

**Trusted Unlinkability** 下位通信層は、ユーザの追跡に利用できるアドレスを用いてパケットの配達を行うが、機能としてアドレスを上位のアプリケーションから秘匿するレベルの安全性を指す。即ち、実際の安全性は下位通信層の信頼性に依存する。Trusted Unlinkability を提供する通信層を、信頼できる通信層と呼ぶ。

以下では、仮名アドレスとブロードキャストを利用して Deducible Unlinkability を実現する手法と、Trusted Unlinkability のための手法について述べる。

### 2.2.1 仮名アドレスを利用した通信方式

仮名アドレスを用いた追跡不能通信とは、ユーザが動的に選択したアドレスを利用することで、追跡不能性を実現しようとする通信方式であり、赤外線通信やスマートカードにおいて実現性がある。

IrDA (Infrared Data Association) の通信規約である IrLAP (Infrared Link Access Protocol) [39] や、NFC (Near Field Communication) の通信規約である ISO/IEC 18092 [41] では、アドレスの衝突を回避するメカニズム (address collision avoidance mechanisms) を規定している。IrDA は赤外線通信の規格であり、NFC は、Type C と呼ばれる非接触型 IC カードのために開発された通信方式を、10cm の距離で 100〜400kbps の伝送速度を提供する近接通信に汎化した規格である。IrDA や NFC では、少数のノード間の通信を想定しており、通信の都度動的にアドレスを決定しても衝突が発生する確率は小さい。従って、ワールドワイドで非衝突性を保証するようにアドレスを静的にデバイスに割り当てる方法 (MAC アドレスに適用されている方法) は、オーバーヘッドが大きいだけで効果を得る機会は少ない。逆に、接続の都度動的にアドレスを生成するようにして、衝突が発生したときにアドレスを変更して衝突を回避する手段を講

じる方が効果的である。Type A、Type B の非接触型 IC カードの通信規約である ISO/IEC 14443 [40] でも、動的にアドレスを生成して利用する仕様となっている。

このように、赤外線通信、及び、非接触型 IC カード通信では、$P$ は自らのアドレスを動的に決定する自由度を有しているので、仮名アドレスを適当なタイミングで更新する (例えば、セッションの更新の都度) ことで、演繹可能な追跡不能性を実現することが可能となる。

動的に仮名アドレスを生成する手法は、衝突回避メカニズムが存在しない IP 通信層やデータリンク層にも適用可能できる可能性はある。即ち、IP アドレスと MAC アドレスを動的に生成して、仮名アドレスとして利用する。

但し、アドレスの衝突回避メカニズムを伴わないため、LAN のような大規模なネットワークに適用すると、アドレスが衝突する危険が高まる。例えば、仮名に利用できる MAC アドレスの有効長は 3 バイトであるので、Birthday Problem の理論から、$n$ ユーザの間で衝突が発生する確率は約 $1 - e^{-n(n-1)/2^{13}}$ となる。これは、ネットワークに 581 以上のデバイス (NIC 等) が同時に存在すると、$\frac{1}{100}$ より大きい危険率で衝突が発生することを意味し、また、危険率を $\frac{1}{10}$ まで許容しても、許容されるデバイス数は 1,880 程度である。この数値が大きいと考えるか小さいと考えるかは、適用場面に依存するが、衝突が発生することは想定しなければならない。

Ethernet のように、アドレスの衝突が発生しても、パケットが喪失することはなく、配達が重複するのみであることが仮定できれば、アプリケーション側で衝突に対応することができる。但し、アドレスが衝突している他のデバイスに宛てたパケットも混在するため、アプリケーションは、下位の通信スタックから渡されるペイロードを検査して、自分宛のペイロードを取捨できる必要がある。

### 2.2.2 ブロードキャストを利用した通信方式

ブロードキャスト通信も、追跡不能性を実現する目的に利用することができる。

`255.255.255.255` は IP アドレスにおけるブロードキャスト・アドレス (broadcast address) であり、`FF:FF:FF:FF:FF:FF` は MAC アドレスにおけるブロードキャスト・アドレスである。例えば、$P$ は、パケットを送信する際、ブロードキャスト・アドレスをパケットの Source フィールドに指定するものとする。パケットを受信した $V$ は、Source フィールドを含むパケットヘッダを根拠に、送信者を特定することはできないだけではなく、それ以前に受信したパケットとリンクすることもできない。

一方、$V$ が返信を行う際には、ブロードキャストでパケットを送信する。ブロードキャストされたパケットから、$P$ が自分宛のパケットを選択できるためには、パケットのペイロードにそのための情報が指定されなければならない。

加えて、ブロードキャスト通信では $V$ は $P$ と TCP セッションを構成できないので、本来ならば TCP スタックが提供している通信の信頼性 (*e.g.* reliability, in-order delivery) はアプリケーション層でサポートする必要がある。

### 2.2.3 信頼できる通信層を仮定した通信方式

インターネット・プロキシにより IP アドレスを隠蔽したウェブアクセスが可能であることは知られている。また、NAT 或いは NATP をサポートするルータは、プライベート IP アドレスとルータのグローバル IP アドレスの変換を行う。このように、インターネット・プロキシやある種のルータは、通信において仮名アドレスによる通信を提供する。

しかしながら、実際のユーザの追跡不能性は、プロキシとルータの信頼性に依存している。極端な例では、プロキシはウェブサイトと共謀している可能性もあるし、ルータの管理者が追跡情報を追跡している可能性もある。このように、インターネット・プロキシやルータを用いた仮名通信は Deducible Unlinkability をサポートせず、下位通信層の信頼性に依存する Trusted Unlinkability をサポートするに留まる。

## 2.3 認証における匿名性と追跡不能性

### 2.3.1 グループ署名

グループ署名は、以下の特徴を備えた署名方式である。

1. グループのメンバーはグループを代表して、任意に署名を生成することができる。
2. グループの公開検証鍵にアクセスできる誰でもが署名を検証することが可能であるが、検証により署名者の身許が明らかになることはない。
3. グループの管理者 (Trusted Group Authority, TGA) は、署名から署名者を特定することが可能であり、紛争等、必要に応じて署名者の身許を特定する。

グループ署名は Chaum と Heijst [26] によって最初に提案され、電子オークション [27]、匿名注文システム [68] 等に応用がある。

例えば、電子オークションでは、以下のような手続きを行い、オークションの参加者のプライバシーを守りつつ、不正な参加者を排除する。

1. 入札者 (bidder) はオークションの開催者が管理するグループに登録し、グループ署名用の鍵を生成する。
2. 入札者は入札時に入札金額等を記載したデータにグループ署名を施して申し込みを行う。
3. 出品者はグループ署名を検証することにより、入札資格者による申し込みであることを検証できるが、入札者の身許を知ることはできない。
4. オークション成立後は、開催者は入札の署名から落札者の身許を明らかにし、品物の配送や決済を行う。

グループ署名のスキームは、以下に述べる、SETUP、JOIN、SIGN、VERIFY、及び、OPEN の手続きから構成される。

**SETUP** TGA はグループの公開鍵ペア (group public key pair) を生成する。ペアの個人鍵は TGA の鍵として安全に保管する。

**JOIN** ユーザは、TGA と協力してグループ署名鍵 (group signing key) を生成し、グループに参加する。

**SIGN** グループのメンバーは、グループ公開鍵と自身のグループ署名鍵を用いて、任意のメッセージに署名する。

**VERIFY** 検証者は、メンバーが生成した署名を検証するために、グループ公開鍵のみを用いる。

**OPEN** TGA は、グループ公開鍵ペアを用いて、署名を生成したメンバーを特定する。

グループ署名が満たすべき要件は、以下のように整理される。

**Correctness** グループメンバーにより生成された署名は受理される。

**Unforgeability** グループメンバー以外により生成された署名は拒否される。

**Anonymity** TGA を除く何人にとっても、署名からメンバーを特定することは、計算量的に困難である。

**Unlinkability** TGA を除く何人にとっても、異なる二つの署名が同一のメンバーにより生成されか否かを判定することは、計算量的に困難である。

**Exculpability** 結託したグループメンバー、TGA、いずれも、他のメンバーの署名を偽造できない。

**Traceability** TGA は、グループ公開鍵及び TGA の個人鍵を用いて、署名からメンバーを特定することができる。

更に、グループの公開鍵の更新の観点から、グループ署名には static と dynamic の 2 種類の区別がある。

**Static** グループへの新しいメンバーの加入、既存のメンバーの脱退、或いは、グループメンバーによる個人用鍵ペアの更新の都度、グループの公開鍵を更新する必要がある。

**Dynamic** 新規加入、脱退、個人用鍵ペアの更新が行われても、グループの公開鍵を更新する必要がない。

実用を考えると dynamic の要件を満足することが望ましいことは明らかであるが、当初提案された方式はいずれも static のタイプであった [19, 26, 27]。最初に提案された dynamic なグループ署名方式は、Camenisch と Stadler [19] による。Camenisch 等が提案したグループ署名方式は、公開鍵のサイズ、署名のサイズのいずれも、グループのサイズに依存しないという特長も併せ持つ。

更に、Ateniese、Camenisch、Joye と Tsudik [5] による方式は、証明可能な安全性を有するグループ署名方式としては従前の方式に比較して効率的であり、日本においても東芝はこの方式をベースに携帯電話による匿名注文システムを開発している [68]。

以下に、Ateniese 等によるグループ署名方式の概要を示す。

1. $QR_n$ を安全な RSA 合成数 $n$ の平方剰余がなす群とする。合成数 $n = pq$ が安全であるとは、別の素数 $p', q'$ に対して、$p = 2p' + 1, q = 2q' + 1$ が成立することである。

   TGA は、$a, d, g, g_1 \in_{\mathcal{R}} QR_n$ を生成し、更に、$x \in_{\mathcal{R}} \mathbb{Z}^*_{p'q'}$ を選び、$y = g^x \bmod n$ を計算する。$(n, a, d, g, g_1, y)$ をグループ公開鍵とし、$(p, q, x)$ を TGA の秘密鍵とする。

2. ユーザ U は以下の手順でグループに加入する。

   (a) U と TGA は協調して $x_u$ を選ぶ。即ち、U と TGA は、相手がどのような戦略をとろうとも、$x_u$ が定義域中で一様に分布することを確信できる。$x_u$ は U の秘密であり、$y_u = a^{x_u} \bmod n$ は TGA によって保持される。

   (b) TGA は素数 $e_U$ をランダムに選び、下式により $c_u$ を計算する。

   $$c_u = (y_u d)^{\frac{1}{e_u}} \bmod n$$

   強 RSA 仮説により、$p, q$ を知る TGA のみが $(c_u, e_u)$ を計算することができる。

(c) TGA は $(c_u, e_u)$ を U に発行し、U は $(x_u, c_u, e_u)$ をグループ署名鍵とする。

3. メンバーによるメッセージ $m$ への署名は、

$$c_u^{e_u} = da^{x_u} \bmod n$$

を満足する $c_u, e_u, x_u$ の知識に関する非対話証明であり、Fiat-Shamir heuristic に従って、チャレンジを $m$ に依存させることで、$m$ への署名とする。

更に、$c_u$ を $y$ により暗号化し、正しい暗号化であることの非対話証明を付与する。

4. OPEN において、TGA は、U を識別する目的で $c_u$ を復号する

Song [59] は新たな安全の概念として forward security と retroactive public revokability を導入し、Ateniese 等の方式を拡張して forward security と retroactive public revokability を満足するグループ署名方式を提案している。

**Forward-security** メンバーの (グループ) 署名鍵がある時点で危殆化し、グループ公開鍵が無効化されても、その時点以前の署名の有効性は保存される。

**Retroactive public revokability** メンバー (グループ) 署名鍵が危殆化した時点に遡って、メンバーの該 (グループ) 署名鍵を無効化する。他のメンバーによる署名、及び、該メンバーの無効化以前の署名に関しては、匿名性と追跡不能性が保証される。

グループ署名をユビキタスアクセス制御に適用する上での最も重大な課題は、計算量が大きい点にある。実際、Iachello 等は、Camenisch 等が提案した Anonymous Credential システム [17, 16] を評価して、ユビキタスコンピューティングに適用するには不適当であると結論し、安全性を犠牲にしても、暗号処理を含まない単純なシステムを提案している [38]。

この観点から、Ateniese 等による提案 [5] 以降、グループ署名の計算量を軽減し、計算効率を向上させようとする試みがなされてきた [12, 15, 14, 60, 51, 36]。表 2.1 に、佐古等 [69] によって報告された各グループ署名方式の計算量の比較を示す。比較に当たっては、楕円曲線上のスカラー倍演算の実行回数に換算してある。

表 2.1: グループ署名の計算量の比較 [69]

| アルゴリズム | 署名 | 検証 | 合計 |
|---|---|---|---|
| Ateniese-Camenisch-Joyce-Tsudik [5] | 2700 | 2475 | 5175 |
| Boneh-Boyen-Shacham [12] | 20 | 33 | 53 |
| Camenisch-Lysyanskaya [15] | 42 | 69 | 111 |
| Camenisch-Groth [14] | 28 | 38 | 66 |
| Teranishi [60] | 65 | 86 | 151 |
| Nguyen-Safavi-Naini [51] | 38 | 37 | 75 |
| Furukawa-Imai [36] | 19 | 39 | 49 |

(楕円曲線上のスカラー倍演算の実行回数に換算)

Pentium 4 3.2GHz を搭載した PC と、名古屋工業大学で開発している暗号ライブラリ [46] とを用いて、192 ビット長標数の素体上に定義された楕円曲線のスカラー倍演算の実行時間を実測し、その計測値を

もとに古川等 [36] の方式の実行時間を推定したところ、一回の署名の生成と検証に要する合計時間は約 600 ミリ秒と計算された。現実の適用においては、ユーザが携帯する端末或いはデバイスは計算能力が制限されるので、上記の数倍程度は時間がかかることが予想される。即ち、実際の実行時間は少なくとも 2 秒程度にはなるものと想定され、アクセスの頻度が高いことが想定されるユビキタスコンピューティングでは、満足できる数値ではないであろう。

# 第3章　抽出研究課題と提案する解決手法

## 3.1 追跡不能性の実現と計算量の抑制の両立に関する課題と解決手法

序論でも述べたように、本論文の主要な研究テーマのひとつは、追跡不能性と Consensual Disclosure とを実現するユビキタスアクセス制御方式を提案する点にある。この目的のために利用できる暗号技術として、グループ署名があることを 2.3.1 で見た。

グループ署名は、グループメンバーが各々の個人鍵 (グループ署名鍵) を用いて生成した署名を、グループ毎に一意に定まる公開鍵 (グループ公開鍵) で検証する。グループ公開鍵は個別のメンバーに依存しないため、署名の検証において署名者の匿名が守られる。グループ署名をユビキタスアクセス制御に適用する際には、アクセス権限のユーザへの付与をグループ署名鍵の発行によって行う。ユーザがサービスにアクセスする際には、グループ署名鍵を用いて生成した署名を権限の証明として提示し、検証者は、署名を検証してアクセスの可否を決定する。

このように、グループ署名は、追跡不能アクセス制御の基礎となる認証方式として有効であるが、計算量に関わる以下の 2 点の問題を含む。

- 署名の生成・検証に必要な計算量が大きく、アクセスイベントが高頻度で発生するユビキタスコンピューティングにおいては、実行速度上のボトルネックとなる可能性がある。
- グループ署名は認証のための仕組みであり、ユビキタスアクセス制御で求められる要件、例えば、アクセスルールの完全性や検証者の認証等をサポートしない。これらユビキタスアクセス制御で求められる要件を満足するためには、機能追加が必要であり、その結果として計算量の更なる増加は避けられない。

本論文では、グループ署名に代わる認証方式で、追跡不能性をサポートしつつも基本的な計算量が小さい方式を提案することを目的の一つとする。

本論文における解決手法を述べるに先立って、グループ署名において大きな計算量が必要であった背景から解説する。

一般的な公開鍵暗号では、公開鍵と個人鍵とは一対一に対応する。

- RSA 法数 $n = pq$ に対して、RSA 暗号 [56] の公開鍵ペア $(e, d)$ は下式によって定義される。

  $$ed \equiv 1 \bmod lca(p-1, q-1)$$

  公開鍵 $(n, e)$ が与えられた時、上式を満足する $d$ は $lca(p-1, q-1)$ を法として一意である。
- DSA[52] 等、素体 $\mathbb{F}_p$ の離散対数問題 (Discrete Logarithm Problem) の困難性に安全性の根拠を置く暗号系では、公開鍵ペア $(y, x)$ は以下を満足する。

  $$y = g^x \bmod p$$

  $g$ は乗法群 $\mathbb{F}_p^*$ の生成元であるので、$x \in [0, p)$ と $y \in \mathbb{F}_p^*$ は一対一に対応する。

公開鍵基盤 (Public Key Infrastructure) では、公開鍵と個人鍵が一対一に対応する性質を利用して、個人認証の基盤を構築する。即ち、特定の公開鍵で検証可能な電子署名は、一意に定まる特定の個人鍵を用

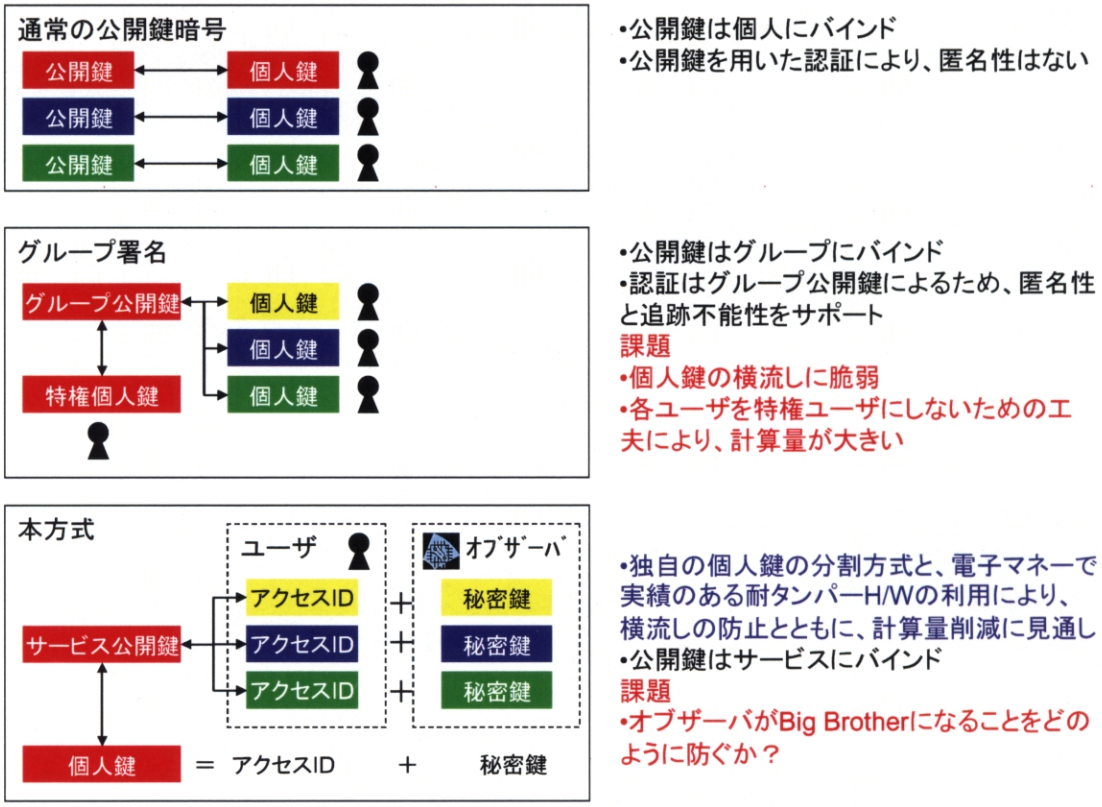


図 3.1: 公開鍵ペアの管理

いてのみ生成される。従って、公開鍵と個人の識別名を証明書に併せて記載すれば、公開鍵による電子署名の検証から署名を行った個人を特定することができるのである。

さて、公開鍵と個人鍵が一対一に対応する通常の署名方式を用いて、最も安直に追跡不能性を実現するには、グループに属するユーザ全てに同じ個人鍵を配布すればよい。しかし、この方法では、必要な場面においてすらも、署名を行った個人を特定することができない。例えば、電子オークションでは、入札時には入札者の資格を匿名で確認することが求められるが、オークションが終了した時点では、オークションの開催者は署名した入札者を特定できる必要がある。全ての入札者が同じ個人鍵を用いて署名を生成する方法では、落札者の特定は原理的に不可能である。

一方、グループ署名では、ひとつの公開鍵 (グループ公開鍵) に対して複数の個人鍵 (グループ署名鍵) を設けることで、上記の方法の欠陥を改善する。即ち、署名の検証は共通のグループ公開鍵で行われるため、署名者の匿名性は守られる上に、署名の生成は署名者毎に異なる署名鍵で行われるので、特別な秘密情報を有するグループ管理者は、グループ署名鍵の違いに由来する署名の差異を識別することができるのである。

グループ署名の問題点は、公開鍵と個人鍵とを一対多に対応させる工夫により、署名の生成・検証の計算量が大幅に増大してしまう点にある。実際、通常の署名とグループ署名の計算量を、楕円曲線上のス

カラー倍演算の実行回数[1]に換算して比較すると、Schnorr 署名、DSA 署名等の通常の署名方式では、署名・検証を合計して 3 回の実行で済むのに対して、現時点で最も効率がよいグループ署名方式である [36] でも 49 回のスカラー倍演算の実行に相当する計算量を必要とする。

先に述べたように、ユビキタス・コンピューティングでは、これまで物理的に提供されてきた多くのサービスが、ユーザ端末と環境側計算機資源との間の協調によって提供されるようになり、それに伴う認証イベントの発生回数も飛躍的に増大することが予想される。加えて、認証処理はユーザに意識されることなく実行されるべきことを考えると、認証処理の実行速度は実用上非常に重要な問題となる。

そこで、本論文では、従来の署名方式を利用することで高速な処理を可能とし、かつ、公開鍵ペアの取り扱いの工夫により、グループ署名の要件、即ち、ベースラインで署名者の匿名性を保証しながら、必要な場合には署名者を識別することを可能とする方式を提案する。

本論文で提案する方式では、耐タンパーハードウェア (Tamper-Resistant Hardware) がユーザの手許に存在することを仮定する。耐タンパーハードウェアとは、マイクロコンピュータとプログラムを内蔵したハードウェアであり、マイクロコンピュータが I/O を管理して内部データへのアクセス制御を行い、また、プローブ等、I/O を迂回した内部データへの不正アクセスは、超細密実装やセンサーによりこれを防止する。Tamper とは、「改竄する」という意味の英語であり、Tamper-Resistant とは、内部に保持されるデータの改竄や窃取に対抗する機能を有することを指す。電子マネーに使われる IC チップや携帯電話の UIM/SIM は、実用化されている耐タンパーハードウェアの典型的な例である。以下では、Chaum 等 [25] の用語に従い、ユーザの手許の耐タンパーハードウェアをオブザーバ (Observer) と呼ぶこととする。

では、本論文で提案する方式の基本的なアイデアと技術上の研究課題について述べる。

本論文で提案する方式は、一対一で対応する公開鍵と個人鍵に対して、個人鍵をランダムに二つの数の算術和に分解して、一方 (Access ID) をユーザに与え、他方 (秘密鍵) をユーザが保持するオブザーバに秘密裏に格納するというアイデアに基づく。ユーザとオブザーバは、互いに協調しながら、公開鍵で検証可能な署名、即ち、追跡不能な署名を生成する。一方、一つの個人鍵からユーザ毎に非常に多数の (実用上は無制限に) 算術和の組み合わせを作ることができるので、通常の署名方式に準拠しながらも、多数のユーザを区別することが可能となり、計算量の抑制と署名者の識別を両立させることができる。

算術和の一方をオブザーバに記録することは、ユーザによる不正を防止するために必要である。即ち、ユーザが自身に知らされる Access ID に加えて、その片割れをも知りえると、両者の算術和をとって個人鍵を計算することが可能となる。個人鍵が得られると、自身で算術和を自由に計算することが可能となり、他のユーザに成りすまして署名を行ったり、また、グループの管理者になりかわって、他のメンバーに有効な Access ID を発行することが可能となる。

10 年前であれば、耐タンパーハードウェアは安全性や価格の点でその実現性に疑義がはさまれることが多かったかもしれないが、現在では、市場において機能的な実証が行われ、製品の低価格化も進んでいる。従って、耐タンパーハードウェアを十分に実際的であるといっても差し支えないが、グループ署名との比較において、本方式のデメリットであると思われるかもしれない。しかしながら、グループ署名もグループ署名鍵の横流しに関する脆弱性を内包しており、これを防止しようとすると、結局、耐タンパーハードウェアか、それに相当する手段が必要であるという実情がある。グループ署名鍵を他の重要

[1] *Rules* 暗号計算において最も計算量の多い演算は、基礎となるアーベル群上でのスカラー倍演算である。アーベル群が素体の乗法群の場合は冪乗剰余演算であり、楕円曲線の場合は点のスカラー倍演算である。このため、暗号計算の計算量は、スカラー倍演算の実行回数で評価される。■

情報とバインドし、横流しにより重要情報も一緒に漏洩するような抑止策も提案されているが、実用における有効性は疑わしい。例えば、住民基本台帳カードでは、耐タンパー特性を有するICチップに署名用の個人鍵を格納し、不正なコピーを防止しようとしている。つまり、耐タンパーハードウェアの仮定は、十分に現実的であるとともに、重要情報の横流し防止の観点からは実用上は必須であるともいえる。

さて、以上のアイデアに基づいて認証方式を設計する際の安全上の課題を述べる。

1. 一つの個人鍵に対して、多数の算術和の組み合わせがユーザとオブザーバに対して与えられる。このような環境において、複数のユーザが結託し、また、攻撃者が複数のオブザーバを自由に利用するケースを想定しても、なお、証明可能な安全性を保証しなければならない。

2. オブザーバは、ユーザの手許には存在しても、ユーザはその内部処理に干渉することは出来ない。特に、オブザーバは、認証の目的に照らして、暗号学的に計算したデータを出力するので、ユーザが検知できないように好ましくない情報をを紛れ込ませることができるかもしれない。つまり、オブザーバがBig Brotherとして振舞う脅威を排除しなければならない。

3. Consensual Disclosureでは、ユーザの同意に基づいて追跡情報が開示されるため、その実現に特に技術的な課題はないと思われるかもしれない。しかしながら、実際には、ユーザが意図している情報のみを開示し、かつ、意図している相手にのみ情報が開示されなければならない。また、不正なユーザが虚偽の情報を開示する可能性を排除するという要件も満足しなければならない。

3.2では、この二つの研究課題に対して、本論文の取り組みを概説する。

## 3.2 安全性に関する課題と解決手法

### 3.2.1 証明可能な安全性

本論文において提案するプロトコルでは、その全ての安全性について証明可能なレベルを求める。アルゴリズムの証明可能な安全性とは、アルゴリズムの安全性がいくつかの原理的な仮説に論理的に帰着されることを意味する。

以下では、認証に求められる最も基本的な安全性である**健全性**を例に取り上げて、証明可能な安全性を得るための本論文の取り組みを述べる。

認証の健全性は、正当な証明者 (ユーザ) のみが、認証をパスできることを保証する安全性を意味するが、本論文では更に以下のように具体化する。

まず、ユーザが、権限者が発行した Access ID と、Access ID に対応する秘密鍵を格納するオブザーバとにアクセスできる時、そのオブザーバをそのユーザにとって **Authentic なオブザーバ**と定義する。その上で、健全性とは、「ユーザが検証可能な署名 (証明) を提出するならば、その署名は Authentic なオブザーバを利用して生成されたものである」と定義する。

5.3 で提案するプロトコルの健全性は、「選択メッセージ攻撃 (Adaptive Message Attack) のもとで偽造不能 (Unforgeable) な」署名アルゴリズムが存在するという仮説に立脚する。署名アルゴリズムが選択メッセージ攻撃の元で偽造不能であるとは、攻撃者が正規の署名者を署名オラクルとして利用可能であっても、新しいメッセージに対する署名を偽造できない安全性を指す。署名オラクルとは、攻撃者が自由にアクセスできる正規の署名者であり、攻撃者が任意に選んだメッセージに対して無条件に署名を施して、結果を出力する。

ランダムオラクルモデル (Random Oracle Model) と離散対数問題の困難性を仮定すると、Schnorr 署名、DSA 署名等が、選択メッセージ攻撃のもとで偽造不能性を満たしていることが知られている。5.3 では、Schnorr 署名をベースとしてプロトコルを構成しているが、DSA 署名等、上記の安全性を満足する他の署名方式の上でも同様のプロトコルを構成することができる。

5.3 のプロトコルが健全であることは、以下のような工夫により、証明可能である。

- 個人鍵の Access ID と秘密鍵の算術和への分解を一様にランダムに行う。この工夫により、不正なユーザによる Authentic でないオブザーバへのアクセスは、不正なユーザ自身によってシミュレート可能となる。即ち、不正なユーザは、Authentic なオブザーバにのみアクセスすると考えてよい。
- Authentic なオブザーバは署名オラクルに置き換えることが可能であることを示す。この性質により、プロトコルの健全性を選択メッセージ攻撃における偽造不能性に帰着させる。即ち、Authentic なオブザーバを利用して認証におけるなりすましが可能であれば、準拠している署名アルゴリズムにおける偽造が署名オラクルを利用して可能であることを示す。

### 3.2.2 Big Brother の回避

本論文で提案する方式では、個人鍵を Access ID と秘密鍵の算術和に分解し、Access ID を保持するユーザと、秘密鍵を保持するオブザーバとが協調して署名を作成する。ユーザが個人鍵を知ることがあってはならないので、オブザーバが秘密鍵をユーザに出力することはない。代わりに、秘密鍵を用いた暗号計算の結果のみを出力し、それによりユーザから秘密鍵を秘匿する。

実は、この仕組みを逆用して、不正なオブザーバが出力に何か情報を紛れ込ませると、ユーザがそれを検知することは困難となる。例えば、本論文で提案する基本的なプロトコルでは、オブザーバの出力は数のペア $(W,r)$ である。オブザーバが正しい処理を行えば、$(W,r)$ は、あらかじめ定められた関係式、

$$F(W,r) = 0$$

を満足する。$F(W,r) = 0$ を方程式と考えると、不定方程式となり、実際その解空間は非常に多くの (実質的に無限個の) 解 $(W,r)$ を含む。解空間に冗長性を与える理由は、オブザーバが内部に保持する秘密鍵を保護するためであり、正しいオブザーバは方程式 $F(W,r) = 0$ の解空間から一様にランダムに $(W,r)$ を選んで出力することで、秘密鍵の漏洩を防ぐ。

一方、不正なオブザーバは、解空間の冗長性を逆手にとって、$(W,r)$ の分布を敢えて偏らせることで、ユーザに検知されることなく何かしら情報を外部に出力することが可能となる。

比較的単純な攻撃例としては、オブザーバに固有の ID を内部に持たせ、かつ、異なる ID のオブザーバは同じ $(W,r)$ を出力しないように実装する。このような不正な実装のもとでは、外部の検証者に $(W,r)$ からオブザーバの ID を逆引きするテーブルを与えれば、検証者はオブザーバの出力からオブザーバの ID を割り出すことが可能となる。もし、オブザーバが十分に広く一様な空間から $(W,r)$ を選ぶとすると、相当数の出力を検査しない限り、ユーザがこの攻撃を検知することは不可能となる。

オブザーバを仮定する本論文の方式では、不正なオブザーバによるこのような攻撃をいかに防止するかが、安全上の研究課題となる。

本論文では、オブザーバの出力をユーザが操作して、出力の確率分布の偏りを取り除く方法により、この問題を解決する。具体的には、オブザーバから受け取った $(W,r)$ を確率的な操作により変換し、変換後の $(W,r)$ は $F(W,r) = 0$ を満足しつつも、確率分布としては解空間 $\{(W,r) \mid F(W,r) = 0\}$ の上を一様に分布するようにする (図 3.2)。

### 3.2.3 Consensual Disclosure における情報開示の安全性

Consensual Disclosure は、検証者から要求があり、かつ、ユーザが同意した場合に限り、サービスの提供と引き換えにユーザが何らかの追跡情報を開示する機能である。Consensual Disclosure では、以下が求められる。

- ユーザが意図している以上の情報が開示されることはない。本論文ではユーザはアクセスの根拠となる Access ID のみを開示するものとする。Access ID の開示によって、権限の発行者は、Access ID の発行イベントとアクセスイベントとを紐付けることが可能となる。

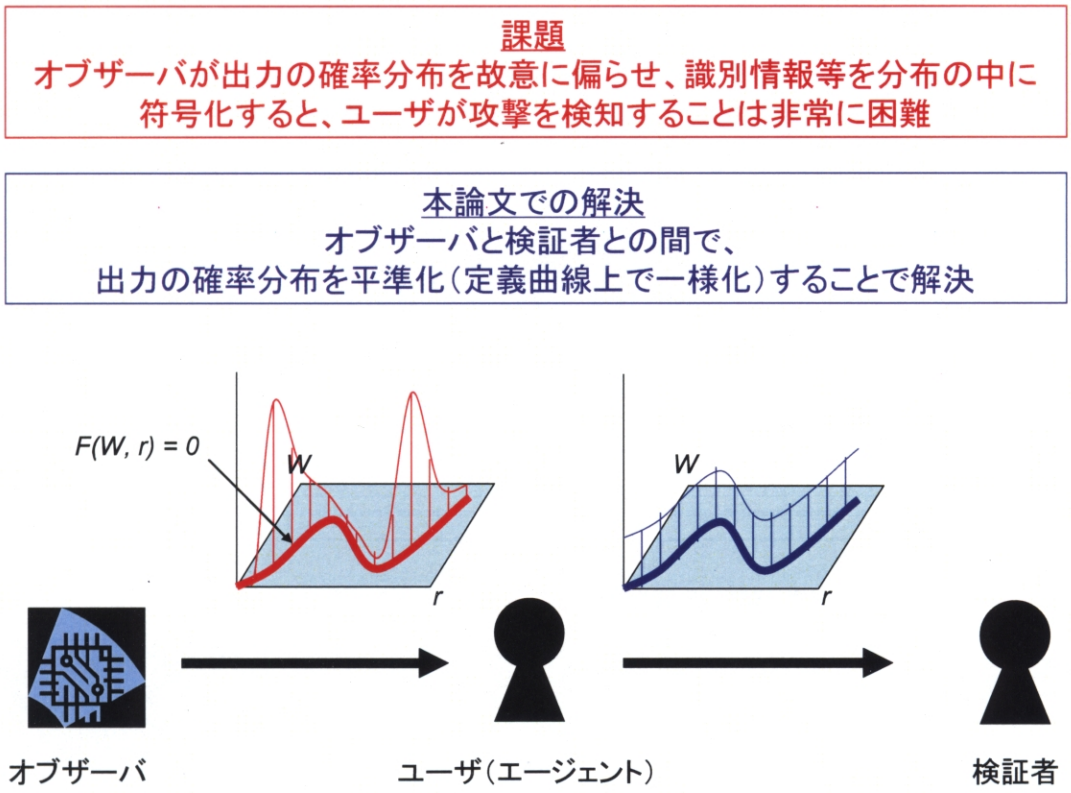


図 3.2: Big Brother の回避

- Access ID の開示は権限の発行者に対してのみ行われる。特に、ユーザと発行者の間で追跡情報を仲介する検証者は、ユーザが正しく Access ID を開示している事実のみを検証することが可能で、Access ID 自体については一切知ることができない。

本論文で提案する方式では、ユーザが Access ID をオブザーバに入力し、オブザーバは公開鍵により Access ID を暗号化して出力することにより、上記の機能を提供する[2]。公開鍵に対応する個人鍵は、権限の発行者のみが保有するため、暗号文を復号できるのはオブザーバに限定される。

安全性の観点から、上記の条件を満足するためには、以下の二つの攻撃を防止できなくてはならない。

1. 不正なユーザが、オブザーバに虚偽の Access ID を入力し、正しい Access ID の開示を妨害する。
2. 不正なオブザーバが、Access ID と偽って別の情報を暗号化し、ユーザが意図していない情報を外部に開示する。

本論文では、署名生成と暗号化の方法を工夫し、上記の攻撃を防止する。具体的には、上記の攻撃が行われた場合、検証者、或いは、ユーザは以下のように攻撃を検知できる。

[2] *Rules*Access ID はユーザの追跡につながる情報であるため、実際には、Access ID をマスクする乱数 $\rho$ をオブザーバに入力する

1. オブザーバに入力されたAccess IDが虚偽であると、検証者はユーザが示す署名の検証に失敗する。
2. 平文のAccess IDを知っているユーザは、オブザーバが生成した暗号文がAccess IDの正しい暗号化であることを検証できる。

## 3.3 ユビキタスアクセス制御の要件のサポートに関する課題と解決手法

### 3.3.1 要件の整理

2.1 で述べたように、Blaze 等が提案する Distributed Trust Management 等では、ユビキタスコンピューティングにおけるサービスへのシームレスなアクセスを実現するために、認証とアクセス制御の統合を求める。本論文においても、ユビキタスコンピューティングで提供される認証方式としては、ここまで述べてきた追跡不能性と Consensual Disclosure に関する要件や、安全性や認証速度の向上に関する要件を満足するだけでは不十分であり、アクセス制御の機能との統合に関する要件をサポートする必要があると考える。

また、グループ署名では、検証者によるユーザの認証のみが考慮されていたが、ユビキタスコンピューティングでは、このような片方向認証のみでは不十分であると考えられる。これは、ユビキタスコンピューティングでは、同じサービスが複数のドメインを横断して提供され、従って、サービスのオーナーはサービスを提供するドメインに必ずしも帰属するわけではないいう事情に由来する。つまり、サービスのオーナーにとって、ドメイン中の検証者 (サービスの提供ポイント) は必ずしも信頼できるエンティティであるとは限らず、サービスへのアクセスに際しては、検証者が信頼できる検証者であるかをまず認証し、認証が成功した場合にのみ権利認証とそれに引き続く処理を許したいという要求は理に適っている。

このように、ユビキタスコンピューティングにおける認証に対しては、単機能の認証に求められる以上の要件が求められるが、それを体系的に整理した報告事例はない。本論文では、ユビキタスアクセス制御に求められる要件の整理を、その成果の一部とする。

### 3.3.2 要件のサポートと計算量の抑制の両立

ユビキタスアクセス制御における要件を満足するためには、安全性の観点から処理の重い暗号計算が必要となることが容易に予想される。従って、これらの要件を満足するための機能を、単に、独立に設計された権利認証プロトコルに追加するだけの方法では、追加的に必要となる暗号計算の実行が認証の実行速度に大きな悪影響を及ぼす惧れがあり、また、機能の界面において脆弱性を内包する危険がある。

本論文では、権利認証プロトコルを、認証本来の要件とユビキタスアクセス制御の固有の事情に由来する要件とを同時に満足するように、最初から統合的に設計することにより、両者で必要とされる暗号計算の共有を図り、機能の界面における脆弱性の排除と、要件のサポートと計算量の抑制とを両立させることを目指す。

# 第4章　ユビキタスアクセス制御の要件の提案

## 4.1 概要

この章では、ユビキタスアクセス制御においてサポートされるべき要件を整理する。

以降、本論文では、サービスやリソースへのアクセスを行う**ユーザ**、アクセス権限をユーザに発行する**(権限) 発行者**、及び、ユーザのアクセス権限を認証し、アクセスへの可否を判断する**(権限) 検証者**の3者間の相互作用 (interaction) により機能提供されるものとして、ユビキタスアクセス制御をモデル化する。Blaze 等のフレームワークにおける Service Agent は、検証者に相当する。また、*PolicyMaker* は、ローカルポリシーに照らしてユーザによるアクセスの可否を判断するための手段を Service Agent に提供するが、本論文のモデルにおいては検証者の一機能に写像する。

ユビキタスアクセス制御の最も基本的な機能は認可 (authorization) と認証 (authentication) であり、従って、ユビキタスアクセス制御に関する要件の最も基本的な部分は、認可と認証に関するものとなる。認可は発行者とユーザの間の相互作用によって実現され、認証は検証者とユーザの間の相互作用によって実現される。相互作用はプロトコルの実行により、発行者とユーザの間での認可のためのプロトコルの実行を**権利発行のイベント**、検証者とユーザの間での認証のためのプロトコルの実行を**権利認証のイベント**と呼ぶ。

前章までに見たように、ユビキタスアクセス制御に対しては、認可と認証以外に以下の項目において要件が存在することが分かっている。

1. Consensual Disclosure に基づく追跡不能性 (4.3)
2. 権限発行者と検証者の分離 (4.4)
3. 認証とアクセス制御の統合 (4.5)
4. 相互運用性 (4.6)

この章では、認可と認証 (4.2) を含めた上記の5項目において求められる要件を整理する。

## 4.2 認可と認証に関連する要件

Completeness と Soundness は、認可と認証に関わる最も基本的な要件であり、それぞれ、判定手法における false positive と false negative と呼ばれる誤判定に関わる。

また、Non-transferability は、

### 4.2.1 Completeness — 完全性

Completeness は、正しい対象を不正なものと判定してしまう誤判定、false positive の発生率が非常に小さいことを求める。

ユビキタスアクセス制御における認可及び認証に関する Completeness を以下のように定義する。

- 認可が Complete であるとは、ユーザが発行者に対して負う義務要件、具体的には、後述する non-transferability, verifier authentication, revocation, access rule enforcement, access rule update の要件を満足することを証明できる限りにおいて、ユーザに対して権利発行が拒否される確率が身視しえる程小さいことと定義する。
- 認証が Complete であるとは、正しい手順によってアクセス権限の発行を受けたユーザが権利認証において拒否される確率が無視しえる程小さいことと定義する。

本論文で提案するプロトコルでは、認可・認証の両方において false positive の発生率が 0 である。

### 4.2.2 Soundness — 健全性

Soundness は、正しくない対象を正しいものと判定してしまう誤判定、false negative の発生率が非常に小さいことを求める要件である。

ユビキタスアクセス制御における認可及び認証に関する Soundness を以下のように定義する。

- 認可が Sound であるとは、発行者に対する義務要件を満足し得ないユーザが認可において受容され、アクセス権限の発行を受ける確率が無視しえるほど小さいことと定義する。
- 認証が Sound であるとは、正しい手順によってアクセス権限の発行を受けていないユーザが認証において受容され、サービスの提供を拒否される確率が無視しえるほど小さいことと定義する。

### 4.2.3 Non-transferability — 横流し不能

Non-transferability は、正当な手続きを経ることなく、ユーザがアクセス権限を他のユーザに譲渡 (横流し) することが不可能であることを要求する。

横流しに対しては、不正が行われると後に不正の事実が検知される事後検知 (after-the-fact) や、横流しが不利益を伴うように設定を行う抑止による対処が知られている。例えば、Consumable Credential では、予め設定された回数以上、Credential を使用すると所有者の身許が開示される機能を実現しており [21, 22, 23, 24, 32, 13, 35]、横流しの検知に有効である。また、Ateniese 等 [5] は、Credential を横流しすると、他の全ての Credential も同時に横流し先に利用される All-or-nothing non-transferability により、横流しを抑止する方法を提案している。

しかしながら、事後検知や抑止の有効性は、不正行為により得られる利益と不正行為により蒙る不利益とのバランスに依存する。不正行為に対して十分な不利益を設定しえるかは、適用領域の特性や事情に依存し、抑止効果を有する制裁を常に設定できるかは保証できない。特に、アクセスが非常に重要なサービスやリソースに対するものである場合、この問題の解決は極めて困難であることが予想される。

この考察に基づき、本論文では、事後検知や抑止による横流しへの対処は効果は不十分であるとし、横流しそのものを困難とする防止策 (prevention) を要件とする。

## 4.3 Consensual Disclosure に基づく追跡不能性の要件

### 4.3.1 Unlinkability — 追跡不能

あるイベントが Unlinkable であるとは、このイベントと別の任意のイベントとが同一のユーザによって引き起こされたものであるか否かの判別が不可能、或いは、非常に困難であることを意味する。より正確には、判別が不可能であるとは、実用的な計算量では、いかなる判別法の成功確率も無視しえるほど小さいことと定義する。

本論文で提案するユビキタスアクセス制御プロトコルにおいては、権利認証と権利発行の両方において、Unlinkability を要求する。

- 権利認証においては、検証者が要求し、かつ、ユーザが同意した場合 (4.3.2) を除いて、Unlinkability が要求される。

  Unlinkable な権利認証のイベントは、別の任意の権利認証のイベント (Unlinkable であるか否かに関わらず)、或いは、任意の権利発行のイベントに対して、同一ユーザによって実行されたものか否かの判別が不可能である。

- 一方、権利発行に対しては、常に Unlinkability を要求する。

  権利発行のイベントは、任意の権利認証のイベント、或いは、別の任意の権利発行のイベントに対して、同一ユーザによって実行されたものか否かの判別が不可能である。

権利発行における匿名性及び追跡不能性については、これまであまり考察してこなかったので、以下で少し詳しく述べる。

サービスやリソースへのアクセス権限の発行と匿名性や追跡不能性とは互いにそぐわないという直感を持つかもしれないが、例えば、以下のケースでは、権利の発行に際してユーザの身許の証明は必要ではない。

- 有料サービスへのアクセス権限の付与において、現金のような匿名性を有する決済手段を用いて対価の支払いを行うケース
- 従業員であれば希望しさえすれば無条件に福利厚生プログラムの利用を許可する等、特定の資格を有していれば無条件にアクセス権限を付与するケース

本論文における権利発行の Unlinkability は、ユーザが発行者に対して負う義務要件、具体的には、non-transferability, verifier authentication, revocation, access rule enforcement, access rule update の要件を満足することをユーザが証明しさえすれば (4.2.2)、ユーザの身許情報、及び、他の権利認証・権利発行のイベントとのリンク情報を含む、いかなる情報も発行者に明かすことなく、アクセス権が発行されるべきことを要求する。

一方、アクセス権限の発行において、身許の証明が必要であるケースが存在することも事実であるが、権利認証では Consensual Disclosure (4.3.2) として、追跡情報を開示する際の要件を定めているが、権利発行においてはイベントは常に Unlinkable であることを求める。

この違いは、ユビキタスコンピューティングの基本的な要請により、権利認証は可能な限り透過的かつシームレスに行われるべきであるのに対し、権利発行は、サービスの保護の観点からも、または、ユーザの保護の観点からも、ユーザによる意識的な手続きを免れないという事情に由来する。

即ち、権利認証では、Consensual Disclosure に基づいてユーザの追跡情報を開示する場合においても、ユーザによる確認作業は最小限であるべきで、できれば YES/NO を答えるのみで済むのが最上である。この要請は、YES と答えた場合に開示される追跡情報が標準化されていることを暗に求めている訳で、そうでなければ、ユーザは開示される情報を都度確認しなければ YES とは答えられない。

対照的に、権利発行では、発行イベントがユーザにとって透過的であることを前提として求める根拠はない。逆に、サービス保護の観点からは、身許の証明や対価の支払い等、一定の義務をユーザを求めることの方が前提であり、また、ユーザの保護の観点からも、ユーザは自分が取得した権利について意識することが重要である。

従って、権利発行における追跡情報の開示は、アクセス制御のインフラストラクチャではなく、適用領域での要請に応じて、それぞれのアプリケーションで制御するべきであり、インフラストラクチャではアプリケーションで制御する情報に何も付加しないこと、即ち、常に Unlinkable な権利発行を保証することが重要である。

### 4.3.2 Consensual disclosure — 同意に基づく開示

序論で述べたように、ユーザのプライバシーの保護は社会やシステムの安全と常に鬩ぎ合う関係にあり、プライベートとパブリシティとの境界はコンテキストに依存して動的である。

本論文では、ユビキタスアクセス制御において有効なプライベートとパブリシティとの境界として、Consensual Disclosure を提案する。Consensual Disclosure では、検証者が明示的に要求し、かつ、ユーザが明示的に同意した場合に限り、サービスへのアクセスと引き換えにユーザが追跡情報を開示する。

更に、追跡情報の開示範囲と開示対象とについて、以下の要件を設ける。

- 追跡情報の開示は権限の発行者に対して行われ、他のエンティティは開示された追跡情報を知りえない。特に、ユーザと発行者との間で開示される追跡情報の仲介をする検証者は、ユーザが追跡情報を正しく開示していることを確認できるのみで、追跡情報の内容については知りえないことを求める。
- ユーザが開示する追跡情報は、権利認証イベントと権利発行イベントとの間のリンクに限定し、それ以外の情報は開示されない。

上記の要件は、以下の理由に基づく。

追跡情報の開示範囲は最小限に限定されるべきであることは、明らかである。追跡情報は、特定のサービスの利用履歴の情報であるが、その情報は第一にユーザの所属に帰すべきであり、次席で所有権を主張できるとすれば、サービスを資産として所有するオーナであろう。サービスのオーナはモデル上ではアクセス権限の発行者に擬されるので、Consensual Disclosure の結果としてユーザの同意を得て追跡情報が開示される場合、その最小限の開示範囲は、認証イベントの根拠となる権限の発行者となる。

一方、開示が要求される追跡情報は、あるケースではユーザの身許であり、別のケースでは性別や年齢等の属性情報のみかも知れない。また、特定のイベントとのリンクが求められるケースもあるであろう。このように、追跡情報として何が要求されるかはアプリケーションに依存する。これを理解した上で、ユビキタスアクセス制御のインフラストラクチャが機能としてサポートする開示範囲をどうするべきかを考えると、過剰な追跡情報の開示は、ユーザにとっても、また、サービスの提供者にとっても好ましくないことから、インフラストラクチャによる開示対象は最小限に限定されるべきであることが分かる。

権限の発行者にとって、権限の認証イベントと発行イベントのリンクは、唯一とはいえなくとも、最小限の追跡情報と考えて妥当である。

発行イベントと関連付けてユーザの情報を収集するようにすれば、発行イベントとのリンクによって、認証イベントにユーザの情報を関連付けることが可能となる。

## 4.4 権限発行者と権限検証者の分離に起因する要件

### 4.4.1 Verifier authentication — 検証者認証

権利認証は、検証者によるユーザの認証であるが、ユビキタスコンピューティングでは、このような片方向認証のみでは不十分であると考えられる。

ユビキタスコンピューティングでは、同じサービスが複数のドメインを横断して提供され、サービスのオーナーはサービスを提供するドメインに必ずしも帰属するわけではないので、サービスのオーナーにとって、ドメイン中の検証者 (サービスの提供ポイント) は必ずしも信頼できるエンティティであるとは限らないからである。

Verifier authentication では、サービスへのアクセスに際して、検証者が信頼できる検証者であるかをまず認証し、認証が成功した場合にのみ権利認証とそれに引き続く処理を検証者に許すことを求める。

## 4.5 認証とアクセス制御の統合に起因する要件

認証とアクセス制御の統合を考えるにあたり、本論文では、アクセス権限には**アクセスルール**が付随するものと考える。

アクセスルールは、SPKI [33] の Authorization Certificate に指定される属性に相当し、Distributed Trust Management の Assertion に指定される *Filter* に相当する。但し、アクセスルールは、Authorization の属性や Assertion の *Filter* に比較して、より複雑なものを想定しており、例えば、XrML (eXtensive rights Markup Language) [31] 等の権利記述言語 (Rights Expression Language) により記述されるレベルを想定する。特に、アクセスルールには、自律的に自身を更新するルールが記述されることを許し、検証者は、サービスの提供に伴って、必要であればルールを変更する (4.5.3)。

### 4.5.1 Revocation — 権限無効化

権限の発行者により信頼され、認証された検証者は、アクセス権限の認証と同時に、認証したアクセス権限を無効化できることが求められる。

アクセス権限の無効化を求める理由として、以下の 2 項目を想定する。両者を区別する際には、前者を **Contextual revocation**、後者を **Named revocation** と呼ぶ。

- アクセスルールに従って、無効化が実行される。例えば、1 回限り有効なアクセス権限、有効期限が定められたアクセス権限、その他特定の条件を満たす時無効化するべきことがルールに指定されているアクセス権限については、無効化の指示がアクセス権限に付随するアクセスルール中に記載される。無効化に際してサービスの提供を行うか否かは、アクセスルールの指定及び解釈に依存する。
- 権限の検証者は、権限の発行者が発行する無効化する権限のリストを参照し、無効化の処理を行う。無効化に際して、アクセスルールの記載には依存せず、また、サービスの提供は行わない。

### 4.5.2 Access rule enforcement — アクセスルールの遵守

権限の発行者により信頼され、認証された検証者は、アクセス権限の認証と同時に、アクセスルールの検査を行い、以下の 2 点を検証する。

- アクセスルールが認証したアクセス権限に固有に付随している。即ち、当該アクセスルールは、権限の発行者が権限の発行時に付随して生成したものであるか、アクセスルールの指定に従って正当に更新されたものであるかのいずれかである。
- 発行後、或いは、正当な更新後、改竄をうけていない。

### 4.5.3 Access rules update — アクセスルールの更新

権限の発行者により信頼され、認証された検証者は、アクセス権限の認証と同時に、アクセス権限に付随するアクセスルールを更新することができる。

アクセスルールの更新は、同アクセスルール中に記載される指示に従う。例えば、利用回数が制限されている権限では、アクセスの都度、アクセスルール中に記載されるアクセス回数の上限値を減算して更新する。

### 4.5.4 Rights delegation — アクセス権限の委譲

アクセス権限の委譲は、認証とアクセス制御の統合と並んで、ユビキタスコンピューティングにおけるトラスト管理の要件として認識されている ( 2.1)。SPKI [33] や Distributed Trust Management[10, 9] では、委譲はユーザが保有している公開鍵ペアへの Authorization Certificate 或いは Assertion の発行という形で行われる。

本論文におけるアクセス権限の委譲の要件は、権限の発行者により信頼され、認証された検証者が、発行済みの正当なアクセス権限とアクセスルールに従って、新たなアクセス権限とアクセスルールを発行する機能が提供されることとする。

## 4.6 相互運用性に関連する要件

### 4.6.1 Message specification — メッセージ規定

ユビキタスコンピューティングにおいて、シームレスにサービスを提供するためには、アクセス制御のためのプロトコルは標準化され、ユーザ端末と環境側計算機の間で紛れなく解釈されなければならない。

また、ユビキタスコンピューティングでは、下位通信層において利用される通信手段も、赤外線通信、非接触 IC カード、無線 LAN、Bluetooth と、環境によって多様であることが想定される。従って、アクセス制御プロトコルは、下位通信層の通信手段に依存しない仕様である必要がある。

以上の要求を満足する有効な方法は、ユビキタスアクセス制御のプロトコルで交換されるメッセージに関して、構文、解釈、及び、符号化方式を厳密に定義することである。本論文では、メッセージを明確に規定することを、相互運用性のための要件のひとつに上げる。

### 4.6.2 Lower-layer connectivity — 下位通信層との接続性

2.2 で述べたように、アプリケーション層のみではなく、それより下位の通信層 (セッション層、トランスポート層、ネットワーク層、データリンク層) においても追跡不能性が提供されなければならない。赤

外線通信やICカード通信では、アドレスの衝突防止のメカニズムを用いて、追跡不能性を提供することが可能である。しかし、例えば、IP/Ethernetの上でBroadcastにより追跡不能性を提供する場合には、アプリケーションにおいて、必要なペイロードを取捨し、また、ステートフルなTCPを利用できないため、通信の信頼性を保証するメカニズムを用意する必要がある。

下位通信層との接続性における要件は、下位通信層においてどのような方法でアドレスの追跡不能性を実現しても、アプリケーション層において接続性を維持できるよう、完結したプロトコルを用意することである。

## 4.7 要件の一覧

以下では、前節までで定義した要件を、一覧として記述する。

**Completeness** 正しいユーザはアクセス権限の認証に成功する。

**Soundness** 不正なユーザがアクセス権限の認証に成功することはない。

**Non-transferability** ユーザは、発行されたアクセス権限を、正当な手続き (Rights Delegation) を経ずに、他のユーザに委譲することはできない。

**Unlinkability** ユーザ以外は、Unlinkable な権限認証イベント、或いは、権限発行イベントを、他の権限認証イベント、或いは、権限発行イベントとリンクすることはできない。

**Consensual disclosure** 検証者が要求し、かつ、ユーザが明示的な同意を行った場合に限り、認証イベントと発行イベントとをリンクする情報が開示される。また、開示されたリンク情報は、発行者のみが知ることができる。

**Verifier authentication** 検証者は発行者により信頼されていることが検証された後に、アクセス権限の認証とそれに引き続く処理を実行することができる。このような検証者を、**認証された検証者**と呼ぶ。

**Revocation** 認証された検証者は、アクセスルールに従って、認証したアクセス権限を無効化できる (Contextual Revocation)。

更に、発行者は、無効化するアクセス権限のリストを発行し、認証された検証者を介して、発行済みのアクセス権限を特定して無効化できる (Named Revocation)。

**Access rule enforcement** 認証された検証者は、アクセス権限の認証と同時に付随するアクセスルールの検査を行い、当該アクセスルールが認証したアクセス権限に固有に付随し、かつ、改ざんされていないことを検証する。

**Access rule update** 認証された検証者は、アクセス権限の認証後に、付随するアクセスルールの指定に従って、当該アクセスルールを変更することができる。

**Rights delegation** 認証された検証者は、発行済みの正当なアクセス権限とアクセスルールに基づき、新たなアクセス権限を発行することができる。

**Message specification** アクセス制御の目的のために、発行者、検証者、ユーザの間で交換されるメッセージに関して、その構文、解釈、符号化方式が厳密に規定される。

**Lower-layer connectivity** ユビキタスアクセス制御における追跡不能性の提供は、下位通信層における通信手段にも、また、下位通信層における追跡不能性の実現手段にも依存しない。

# 第5章　Consensual Disclosureに基づく追跡不能プロトコルの提案

## 5.1 基本モデル

### 5.1.1 プレイヤー

本論文で対案するアクセス制御プロトコルでは、*Service Provider Agent* (*SpA*)、*User Agent* (*UA*)、*Service Provider* (*SP*) 及び *Service Appliance* (*SA*) の 4 つのプレイヤークラスから構成される構成モデルを仮定する。この構成モデルにおいて、各プレイヤークラスは互いに独立であり、プレイヤークラスのインスタンス相互の信頼関係は、インスタンスのペア毎に定まる合意に依存する。

以下に、上記のプレイヤークラスの定義を述べる。

#### 5.1.1.1 *SpA*: Service Provider Agent

*SpA* はモノリシックなモジュールであり、ユーザが、サービスに対するアクセス権限を取得する行為、及び、サービスの利用時にアクセス権限を証明する行為を助けると同時に、サービスの所有者 (*SA*) の代理として、アクセス権限が濫用されることを防止する機能を果たす。

*SpA* の性質として以下を仮定する。

- *SpA* は常にスレイブとして機能し、定常状態では非活性化されている。外部の適正なプログラムからアクセスされた場合に活性化され、そのプログラムからのアクセスが終了するまでの間のみ、活性化の状態を維持する。
- *SpA* は耐タンパー特性を有するハードウェアとする。*SpA* 中に存在するプログラムやデータは外部からのいかなる不正アクセスからも保護される。

*SpA* を耐タンパーハードウェアとする目的は、Non-transferability (4.2.3) の要件を満足する点にある。

Camenisch と Lysyanskaya [17] は、credential の non-transferability を耐タンパーハードウェアを仮定せずに実現する方法として、all-or-nothing non-transferability を提案した。All-or-nothing non-transferability は、不正なユーザが自身の credential をひとつでも開示すると、そのユーザが所有する全ての credential が計算できてしまう仕組みにより、credential の不正な開示を抑止することを狙いとする。卑近な例で述べれば、運転免許証を他人に貸すと、クレジットカードからパスポートまで全ての証明書が利用される危険があるので、運転免許証を貸すことができないという理屈である。

Camenisch 等による all-or-nothing non-transferability のスキームは非常に大きな計算量を要求する点で実用に供することはできないが、それ以前に、その抑止効果にも疑問が呈されている。例えば、不正なユーザ同士が credential を交換すると、互いに相手の credential を自由に使用することができるため、濫用は相互に抑制される。例えば、一方が相手のクレジットカードを許可なく利用すると、報復として自分のクレジットカードも不正に利用される危険があるので、相手の信頼を裏切る行動には出ることができない。従って、当初期待された抑止効果は期待できない。

Camenisch と Lysyanskaya [17, 19] は、事故や不審の際に、別に anonymous credential を無効化するスキームを提案しているが、計算量はなお大きい。

技術的に見れば、証明書の不正な開示や共有を検知することは不可能ではないであろうが、実用として受け入れることは困難であると考えられる。実際、一回の不正アクセスによって取り返しのつかない被害が生じるケースでは、検知に基づく after-the-fact の対策は有効でない。また、不正行為に対しては動機を失わしめるほど十分な懲罰が、社会的にも法律的にも用意されていなければ、検知の効力はない。

このような観点から、本研究では、non-transferability は耐タンパーハードウェアを仮定して実現するものとする。

耐タンパーハードウェアは、内部状態の読み出しや改変が困難なように構成されたハードウェアであり、電子マネーに利用されているスマートチップ等が知られている。耐タンパーハードウェアに関しては、安全性のレベルとコストが長きにわたって議論されてきたが、最近になり IC カード電子マネーとして普及し、有効性が確認されている。

耐タンパーハードウェアは、その所有者、即ち、ユーザを最も強力な攻撃者と想定して保護機能を実現するため、ユーザにとってはコントロールの及ばないブラックボックスとなる。プライバシーの観点から言えば、耐タンパーハードウェアは、ユーザの匿名性を脅かす情報を外部に漏洩する最も危険なチャネルとなる。

このように、耐タンパーハードウェアを所有するユーザの利害と耐タンパーハードウェアが代表する利害とは互いに相克するが、両者を協調させることを目的に提案されたスキームが、Chaum と Pedersen による wallet-with-observer である [25, 11]。

Wallet-with-observer では、耐タンパーハードウェアは、ユーザの管理下にある wallet を介してのみ外部と通信するように構成される。Wallet はユーザに有利なデータ (positive credential) を保持し、耐タンパーハードウェアはユーザに不利なデータ (negative credential) をユーザによる干渉を排除して保持する。更に、耐タンパーハードウェアと外部との通信は、ユーザの目を逃れて不利な情報がやり取りされないように、wallet による監視と干渉を受ける。

本研究でも、アクセス権限の non-transferability を実現することを目的に、wallet-with-observer のスキームを踏襲するものとする。

#### 5.1.1.2 *UA*: User Agent

*UA* のインスタンスは、ユーザの完全な管理下におかれるモジュールであり、ユーザに代わって *SpA* にアクセスするとともに、*SpA* が匿名性及び追跡不能性を損なう情報を漏洩しないよう管理する役割を果たす。

*UA* と *SpA* の実装としては、ユーザが日常生活において携行するデバイス中に共存する構成が自然である。例えば、*SpA* を携帯電話に装着するスマートチップ (*e.g.* UIM) として実装し、*UA* を携帯電話にインストールするスマートチップとのインタフェースをとるデバイスドライバとする実装が考えられる。

#### 5.1.1.3 *SP*: Service Provider

*SP* は、サービスの所有者を表すプレイヤーであり、サービスに対するアクセス権限を発行が許される唯一のエンティティである。

*SP* は、これまで権限の発行者と呼んでいたエンティティと一致する。

*SpA* は、アクセス権限を発行する際、*SP* は *UA* を介して *SpA* と通信を行う。

#### 5.1.1.4 *SA*: Service Appliance

*SA* は、*SP* の代理として、サービスをユーザに直接提供するエンティティであり、ソフトウェア、デバイス、機器、装置等、任意の実装形態を取る。

*SA* は、これまで権限の検証者と呼んでいたエンティティと一致する。

*SA* は、ユーザに発行されたアクセス権限を検証する目的で、*UA* を経由して、*SpA* と通信を実行する。

### 5.1.2 プレイヤー間の信頼

ユビキタス・コンピューティングは、インターネット、無線通信 (Bluetooth, IEEE 802.11 等) や携帯電話の急激な普及に後押しされて、現実味を加えてきている。この事実は、ユビキタス・コンピューティングを実現するに当たって、既存のインフラストラクチャがもつ複雑性と多様性とを前提としなくてはならないことを同時に意味している。

ビジネスレベルにおいて考慮するべき複雑性・多様性としては、第一に、プレイヤー間の信頼性の問題がある。デジタルコンテンツ流通に関するケーススタディを通して、この問題を見てみよう。

このケーススタディでは、以下の仮定を置く。

- *SP* はレコードレーベル (*e.g.* Sony BGM Music Enterprise) であり、デジタル化された楽曲コンテンツをユーザに提供 (配信) する。
- ユーザは携帯プレイヤ (*e.g.* iPod) である *SA* を用いて楽曲を聴取する。
- *SpA* はユーザが携行する携帯電話に組み込まれる。

上記の設定では、*SP*、*SA* の製造業者 (*e.g.* Apple Corp.)、及び、*SpA* の製造業者 (Motorola) は互いに独立した事業者であり、相互の信頼関係は当事者の恣意に任され、非均質的である。例えば、レコードレーベルは、複数の *SA* 製造業者 (*e.g.* Apple Corp., Panasonic) と契約を締結し、これらの製造業者が製造した *SA* での楽曲の再生を許可する一方で、安全上の理由から、第三者のベンチャーが製造した *SA* による再生は忌避するかもしれない。逆に、*SA* の製造業者は、自社の *SA* で、正規のレコードレーベル (*e.g.* Sony

BGM Music Entertainment, Toshiba EMI) が配信した楽曲は再生できても、海賊行為の疑いのある配信業者が配信した楽曲の再生は拒否したいかもしれない。

このように、現実世界でのプレイヤー間の信頼関係は、複雑で非均質的であり、このような信頼関係に対応するために、本モデルでは以下のポリシーをとる。

- *SP* は自身が信頼する *SA* にのみサービスの実行を許可し、逆に、*SA* は自身が信頼する *SP* のサービスのみを実行する。
- *SP* は自身が信頼する *SpA* を所有するユーザに対してのみ、サービスに対するアクセス権を発行する。
- *SpA* は *SP* を選り好みせず、いかなる *SP* に対しても、アクセス権取得のための処理を等しく実行する。但し、*SpA* は、ユーザからの指示なしに同処理を実行することはなく、信頼できる *SP* と信頼できない *SP* とを区別する役割はユーザが果たす。
- *SA* は、自身のポリシーで、特定の *SpA* を拒否することはない。*SA* は、*SpA* がアクセス権の保持に対する正当な証明を提示する限り、所定のサービスを実行する。*SpA* が正当な証明を提示できるということは、*SP* がその *SpA* を信頼していることに他ならないので、*SA* は、*SP* のポリシーに従って、*SpA* を篩い分けていることになる。
- *SpA* は、自身のポリシーで、特定の *SA* を拒否することはない。*SpA* は、通信相手である *SA* が *SP* が信頼するプレイヤであることを認証し得る限り、通信を行う。

### 5.1.3 アクセス制御プロトコル

匿名性と追跡不能性を実現するための基本的なアイデアは、サービスに対して公開鍵ペアを割り当て、サービスに割り当てた公開鍵に依拠してユーザのアクセス権を認証する点にある。

#### 5.1.3.1 サービスの識別

サーブすの識別のためには、*SP* は公開鍵ペアを生成し、それを自身がユーザに提供するサービスに対して割り当てる。*SP* は公開鍵をサービスの公開識別子として発行し、対応する個人鍵は安全に秘匿する。

#### 5.1.3.2 アクセス権の発行

*SP* は、ユーザからの発行要求に応えて、特定のサービスに割り当てられた個人鍵から Access ID 5.1.5 と呼ばれるデータを生成し、該ユーザに発行する。個人鍵から Access ID への変換は落とし戸のある一方向であり、即ち、変換に先立って *SP* と *SpA* が交換する秘密鍵なしでは、Access ID から個人鍵を復元することはできない。

#### 5.1.3.3 アクセス権の認証

*SA* は、**匿名化**された Access ID を検査し、それが要求されているサービスの個人鍵から正しく生成されていることを検証する。匿名化された Access ID の検証はサービスに割り当てられた公開鍵のみを使って実行されるので、*SA* と *SpA*(ユーザ) との間の通信において、ユーザの身許を指示する情報や、今回のアクセスをユーザによる過去或いは将来のアクセスと結びつける情報が漏洩することはない。

#### 5.1.3.4 匿名性及び追跡不能性の要件

サービスを不正なアクセスから保護するために、*SpA* は *SP* を代理して、以下の重要な機能を果たす。

- 発行済みアクセス権の不正な譲渡を防止する。
- *SA* と協力して、正当なアクセス権を伴わない、不正なアクセスを防止する。
- *SA* が、*SP* の財産であるサービスを、不正に実行することを防止する。

このように、*SpA* は、ユーザのデバイス中に存在する「*SP* のエージェント」と考えることが出来る。この特性は、*SpA* はユーザの利益に反して行動し、特に、ユーザの匿名性と追跡不能性とを毀損する脅威となる可能性がある。

*UA* の役割は、まさに、*SpA* がユーザの利益に反して匿名性と追跡不能性とを毀損することを防止する点にある。*UA* は、*SpA* と *SP/SA* の中間に存在し、*SpA* 及び *SP*/SA の両者に対してインタフェースを持つ。各々のインタフェースにおいては、ユーザの匿名性と追跡不能性とを保護するために、以下の機能が求められる。

***UA/SP* 間、及び、*UA/SA* 間のインタフェース** *UA* は、*SpA* が *SP/SA* に対して送信するメッセージを乱数化することで、*SP/SA*、或いは、通信の傍受者に対して、ユーザの匿名性や追跡不能性を毀損する情報が漏洩することを防止する。

***UA/SpA* 間のインタフェース** *SpA* は、通信の成功を犠牲にして、いかなるメッセージを送信することも可能である。従って、攻撃者が *UA*・*SpA* 間の通信を観察出来るならば、ユーザの識別情報を通信から得ることが可能である。*UA* がこの攻撃に対して実施できる唯一の対策は、*SpA* がプロトコル規定を逸脱した場合には、いつでもそれを検知できることである。

### 5.1.4 記法

本稿で用いる記法について述べる。

- $\mathcal{G}_T$ 及び $\mathcal{G}_S$ は共に加法群である。$\mathcal{G}_T$ 及び $\mathcal{G}_S$ は、それぞれ *SpA* 及び *Service* に帰属しており、アクセス権の発行及び検証のための基礎群として利用される。
- 式 $x \stackrel{\mathcal{G}}{=} y$ は、$x$ 及び $y$ が加法群 $\mathcal{G}$ の元として同一であることを意味する。
- $\mathcal{G} = \mathcal{G}_T$ または $\mathcal{G} = \mathcal{G}_S$ とするとき、ランダムに選んだ $A, B \in_{\mathcal{R}} \mathcal{G}$ に対して、「$x$ に関する方程式 $A \stackrel{\mathcal{G}}{=} xB$ の解を求めることは計算量的に困難である」という性質を満足する。この性質は、乗法群においては、離散対数問題 (Discrete Logarithm Problem) の困難性として知られている。
- $G_T$ 及び $G_S$ は、$\mathcal{G}_T$ 及び $\mathcal{G}_S$ 上の基点であるとし、その位数を、それぞれ、$n_T$ 及び $n_S$ と表す。
- $T \stackrel{\mathcal{G}_T}{=} \tau G_T$ が成り立つとし、$T$ を *SpA* の公開鍵、$\tau$ を *SpA* の個人鍵と呼ぶ。

  $\tau$ は、*SpA* 中に安全に保管される。

  $(T, \tau)$ は *SpA* のインスタンス毎に与えられるのではなく、*SpA* のクラス (例えば、同じ製造業者が製造した同じタイプの *SpA*) により共有される。
- $S \stackrel{\mathcal{G}_S}{=} \sigma G_S$ が成り立つものとし、$S$ を *Service* の公開鍵、$\sigma$ を *Service* の個人鍵と呼ぶ。*SP* は、$\sigma$ を安全に保管する。
- $A \stackrel{\mathcal{G}_S}{=} \alpha G_S$ が成り立つものとし、$A$ を *UA* の公開鍵、$\alpha$ を *UA* の個人鍵と呼ぶ。*SP* は、$\sigma$ を安全に保管する。$\alpha$ は、*UA* 中に安全に保管される。
- $\pi(x)$ は、$\mathcal{G}_T$ 或いは $\mathcal{G}_S$ の元 $x$ を、所定の長さのビット列に一対一に変換する関数である。
- $\omega(x)$ は、任意長のビット列を入力とし、固定長のビット列を出力とする擬似乱数関数であるとし、一方向性と非衝突性を備えるものとする。実用的には HMAC [7] を利用して構成することを想定する。
- $\mu(key, x)$ は、安全な MAC 生成 (検証) 関数であるとする。$\mu(key, x)$ は、$\omega(\cdot)$ を用いて、例えば、$\mu(key, x) = \omega(key|x)$ のように定義され、同様に一方向性と非衝突性を備える。実用的には HMAC [7] を利用して構成することを想定する。
- $A, B \in \mathcal{G}$ に対して、$b = \mathrm{mult}_A B$ は $B \stackrel{\mathcal{G}}{=} bA$ を意味する。

### 5.1.5 アクセス ID

Access ID *aid* は *SP* がユーザ (*UA*) に発行するデータであり、*SP* がユーザに許諾するアクセス権を表現する。Access ID (*aid*) は、*SP* と *SpA* との通信を介して生成される。

実際には、*aid* は、*Service* の個人鍵 $\sigma$ を *SP* と *SpA* との間で共有されるランダムな秘密鍵 $k$ によりマスクして得られるデータであり、

$$aid = \sigma - \mu(k, *) \bmod n_S$$

により定義される。

ここで、データ $*$ は *SP* が定める任意のデータであり、アクセス権限に固有の属性を指定するものである。以降、***認証子**と呼ぶこととする。*SA* は、ユーザのアクセス権限を認証する際に、*認証子が認証対象のアクセス権限に正しく対応しており、かつ、不正に改変されていないことを同時に検証する。

*認証子の例としては、*aid* が表現する権利に固有に付随するアクセスルール (*e.g.* 有効期限、有効利用回数等) のハッシュ値を指定する等が考えられる。

Access ID という名称が示すように、*SP* は *aid* を発行したアクセス権限を識別する情報として取り扱う。従って、*UA* が *aid* を提示して権利の証明を行う場合、少なくとも、権利発行のイベントと権利行使のイベントとが紐付けられ、利用の追跡が可能となる。

権利の証明を匿名性と追跡不能性とを保証して実行したい時には、*UA* は、Access ID を以下のように乱数化して *anm* を生成し、*aid* の代わりに *anm* を提示する。匿名化された *anm* は、*UA* が生成する秘密の乱数 $\rho \in [0, n_S)$ により、*aid* を更にマスクした値となる。

$$anm = aid - \rho \bmod n_S$$

$\rho$ は $[0, n_S)$ 中で一様に分布するので、*anm* も $[0, n_S)$ 中で一様に分布する。即ち、*anm* の分布は、*aid* の値に依存しないことから、*SP* は *anm* から *SpA* のインスタンスを区別することはできない。

### 5.1.6 アクセス ID の発行プロトコル

以下は、*SP* がユーザ (*UA* と *SpA*) に Access ID を発行する**権利発行プロトコル**の基本的な要件を述べたものである。

**実装クラスの認証** *SP* は、自らが認定する *SpA* を認証できる場合に限り、*UA* に対して *aid* を発行するが、その際、匿名性と追跡不能性とを確保するために、個別の *SpA* のインスタンスを識別することがあってはならない。即ち、*SpA* のインスタンスを知ることなく、*SpA* の実装クラスのみを認証して、*aid* を発行しなければならない。

実装クラスの認証は、*SpA* の実装が、Non-transferability 4.2.3、Verifier authentication 4.4.1、Revocation 4.5.1、Access rule enforcement 4.5.2、及び、Access rule update 4.5.3 の要件を満足することを確認する。現実的には、*SP* が *SpA* の実装を検査し、合格した場合に、公開鍵 $(\mathcal{G}_T, G_T, T)$ を信頼できる *SpA* の公開鍵として登録する。

**健全性** 不正な *UA* が *SpA* への入力を恣意的に操作したとしても、権利発行プロトコルを開始した *SpA* インスタンスと異なる *SpA* インスタンスに同一の $k$ を生成させることができてはならない。特に、*SP* は *SpA* の実装クラスを認証して $k$ を共有するので、同じ実装クラスに属する異なる *SpA* インスタンスが $k$ を共有しないことを保証することが重要である。

**匿名性及び追跡不能性** *SP* がプロトコルを通じて、ユーザの匿名性や追跡不能性を毀損する情報を知りえてはならない。特に、*SP* と *SpA* とが共謀し、プロトコルの成否を無視して、*SpA* が *SP* に送付するメッセージに識別情報等を混入する攻撃に対抗できなければならない。

権利発行プロトコルでは、*Service* の個人鍵をマスクするための秘密の乱数 $k$ を *SP* と *SpA* の間で共有する。図 5.1 で規定する権利発行プロトコルでは、MQV(Menezes-Qu-Vanstone) 鍵交換プロトコル [50] の片方向バージョンに準拠して *SP* と *SpA* とが $k$ を交換し、*SP* は交換した $k$ に基づいて *aid* を生成・発行する。

また、ユーザがアクセス権限を取得する際には、ユーザが *SP* を認証することが必要であるが、片方向 MQV 鍵交換をベースとする図 5.1 の権利発行プロトコルでは、ユーザによる *SP* の認証はサポートしない。これは以下の理由による。

- 双方向 MQV 鍵交換をベースとすることで同時に *SP* を認証することは可能であるが、*SP* が、*SpA* の実装に依存する $\mathcal{G}_T$ 上に公開鍵を保有するという前提は技術的に不合理である。
- 実用上、ユーザによるアクセス権限の取得に際して、権利発行プロトコルが単独で実行されるとは考えづらく、認証機能をサポートする SSL のようなプロトコルの上で動作すると考える方が自然である。例えば、ユーザが *SP* のサイトからアクセス権限を受ける場合、ホームページにはサービスに関する重要事項 (*e.g.* 使用許諾に関する規約等) が記載されていたり、アクセス権限発行に伴ってユーザが何らかの行為 (*e.g.* 規約への同意、決済等) を行うと考えることは自然であり、その場合、ユーザは SSL 等の手段を用いて、記載内容の完全性や個人情報の保護を図る。

| *SP* | | *UA* | | *SpA* |
|---|---|---|---|---|
| | | | $\epsilon_T \in_{\mathcal{R}} [0, n_T)$ | |
| | | | $\xleftarrow{E_T}$ $E_T \overset{\mathcal{G}_T}{=} \epsilon_T G_T.$ | |
| | | $\epsilon_U \in_{\mathcal{R}} [0, n_T)$ | | |
| | | $E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$ | | |
| | $\xleftarrow{E_U}$ | | | |
| $\epsilon_P \in_{\mathcal{R}} [0, n_T)$ | | | | |
| $E_P \overset{\mathcal{G}_T}{=} \epsilon_P G_T$ | | | | |
| $k' = \pi(\epsilon_P(E_U + \bar{E}_U T))$ | | | | |
| $k = \mu(k', \#)$ | | | | |
| $aid = \sigma - \mu(k, *) \bmod n$ | | | | |
| | $\xrightarrow{aid,*}$ | Store *aid* and $*$. | | |
| | $\xrightarrow{E_P}$ | $\# = \omega(aid)$ | $\xrightarrow{E_P,\#,\epsilon_U}$ | $E_U \overset{\mathcal{G}_T}{=} (\epsilon_U + \epsilon_T) G_T$ |
| | | | | $k' = \pi((\epsilon_U + \epsilon_T + \bar{E}_U \tau) E_P)$ |
| | | | | $k = \mu(k', \#)$ |
| | | | | Store $k$ and #. |

図 5.1: 権利発行プロトコル: Rights-Issue

図 5.1 で規定する権利発行プロトコル (Rights-Issue) では、データ#を *SpA* に対して発行する。#は、*aid* の一方向ハッシュ値とであり、*SP* のスコープ内で一意に *aid* を特定する識別子として利用される。ハッシュ関数の一方向性より、*SpA* は#から *aid* を計算することはできない。以降では、**#識別子**と呼ぶ。*SpA* は、$k$ と#識別子とを含む**鍵レコード**を生成し、内部データベースに検索可能な形式で記録する。

*SP* が特定の *aid* を無効化したい場合、*SP* は**権利失効リスト** (rights revocation list) を作成する。権利失効リストは、無効化対象となる *aid* の#識別子のリストであり、*SA* を経由して *SpA* に送付される。*SpA* は、権利失効リストに対応する#識別子が記載されている場合、#識別子と紐付けられている $k$ を削除する。削除により、対応する *aid* に基づく権利認証は永久に不可能となる。

一方、図 5.1 中の $*$ で表されるデータを、$*$ **認証子**と呼ぶ。$*$ 認証子は、*SP* が発行するアクセス権限に付随して指定するデータに関して、その完全性を権限認証イベントにおいて検証する目的で用いられる (7.1)。アクセス権限に付随して指定するデータの典型的な例として、アクセスルール (4.5) を挙げることができる。以降では、$*$ 認証子は、少なくとも、アクセスルールのハッシュ値を含むものとする。

*SP* と *SpA* とが、$k$ を共有することは、下式により証明される。

$$\epsilon_P(E_U + \bar{E}_U T) \overset{\mathcal{G}_S}{=} \epsilon_P(\epsilon_E + \epsilon_T + \bar{E}_U \tau) G_S \overset{\mathcal{G}_S}{=} (\epsilon_E + \epsilon_T + \bar{E}_U \tau) E_P$$

Rights-Issue は前節で述べた要件を満足することを示す。

**実装クラスの認証** *SP* が *SpA* との鍵交換に当たって使用する公開鍵 $(\mathcal{G}_T, G_T, T)$ は、*SpA* のインスタンスに対してではなく、*SpA* の実装クラスに対して固有に割り当てられた公開鍵である。

**インスタンスの制限** *SpA* が適正に実装されていることを仮定すると、即ち、$\epsilon_T$ を一様にランダムに生成し、かつ、プロトコル規定に逸脱する隠しチャネルを持たないとすると、*SP* 以外では $\epsilon_T$ を生成した *SpA* のみが $k$ を計算することが可能である。

*SpA* が不正に実装されている場合は、典型的な例では、*SpA* が隠しチャネルを利用して $k$ を外部に漏洩するような場合、プロトコルのレベルで $k$ の漏洩を防止する原理的な手立ては存在しない。

*UA* が、*SpA* の隠しチャネルを監視する方法は有効な対抗手段であり、匿名性と追跡不能性の確保のために動機付けも存在するものの、*UA* と *SpA* が共謀する可能性を否定することはできない。

**匿名性と追跡不能性** *SP* が *UA* から受け取る乗法は、$\mathcal{G}_T$ 上で一様に分布する $E_U$ のみである。従って、たとえ *SpA* と *SP* が共謀したとしても、*SP* に対してユーザの匿名性と追跡不能性を毀損する情報が漏洩することはない。

## 5.2 零知識対話型証明を利用したプロトコル

### 5.2.1 プロトコルの定義

図 5.2 に、零知識対話型証明に安全性の基礎を置く追跡不能権利認証プロトコルを示す。

| *SA* | *UA* | | *SpA* |
|---|---|---|---|
| | $\rho \in_{\mathcal{R}} [0, n_S)$ | | |
| | $anm = aid - \rho \bmod n_S$ | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | $\xleftarrow{anm}$ | | |
| $e_0 = \omega(c)$ | $\xrightarrow{e_0}$ $\longrightarrow$ | $\xrightarrow{e_0}$ | $w' \in_{\mathcal{R}} [0, n_S)$ |
| | | $\xleftarrow{W'}$ | $W' \overset{G_S}{=} w'G_S$ |
| | $w'' \in_{\mathcal{R}} [0, n_S)$ | | |
| | $W \overset{G_S}{=} w''G_S + W'$ | | |
| | $\xleftarrow{W}$ | | |
| | $\xrightarrow{c}$ | | |
| | | $\xrightarrow{c}$ | |
| | | | Unless $e_0$ is valid, abort the session. |
| | | $\xleftarrow{r'}$ | $r' = c\mu(k, *) + w' \bmod n_S$ |
| | Unless $r'$ is valid, abort the session. | | |
| | $r = r' + c\rho + w'' \bmod n_S$ | | |
| Unless $r$ is valid, abort the session. | $\xleftarrow{r}$ | | |

図 5.2: 零知識対話型証明を利用した権利認証プロトコル: Unlink-Verify-ZKIP

*SA* は *UA* を介して *SpA* と通信を行い、*SpA* を保持するユーザが正しくアクセス権限を有していることを検証する。

- *UA* は *aid* を匿名化し、*anm* を生成して、*SA* に提示する。
- *SA* は、*anm* を検査し、*SpA* が *aid* に対応する $k$ を保持していることを検証する。
- *SpA* が *SA* への通信中に追跡に繋がる情報を符号化することを防止するために、*SpA* の出力が所定の検証式を満足することを検証するとともに、検証式の解空間中で一様に分布するように出力を乱数化する。

### 5.2.2 安全性の評価

#### 5.2.2.1 実行効率を改善した 1-Round Schnorr 識別アルゴリズムの安全性

図 5.3 で示したプロトコルは、Schnorr の識別アルゴリズム (Schnorr identification algorithm) [57] をベースとしている。Schnorr の識別アルゴリズムは、完全零知識性を有する対話型証明 (perfect zero-knowledge interactive proof) であることが知られており、不正な検証者を含む任意の検証者 $\tilde{\mathcal{V}}$ に対しても、正しい証明者 $\mathcal{P}$ との対話がシュミレートできる、即ち、$\mathcal{P}$ との対話と同一の確率分布の対話を出力する多項式時間 Simulator の構築が可能であるという性質を満足する。

図 5.3 のプロトコル (以降、1-RND-Schnorr と表記する) では、Schnorr の識別アルゴリズムに以下の変更を加えた対話型証明に基づく。

- 検証者 $\mathcal{V}$ は、証明者 $\mathcal{P}$ から Witness $W$ を受信する前に Challenge $c$ を生成し、その暗号学的ハッシュ $\omega(c)$ を、Commitment として検証者 $\mathcal{P}$ に提示する。暗号学的ハッシュ関数とは、一方向性と非衝突性とを満足するハッシュ関数であるとする。
- 検証者 $\mathcal{V}$ は、証明者 $\mathcal{P}$ から Witness $W$ を受信した後、Challenge $c$ を証明者 $\mathcal{P}$ に開示する。
- 証明者 $\mathcal{P}$ は開示された $c$ の一方向ハッシュ値が、Witness $W$ の送信に先立って受信した Commitment と一致することを検証する。
- 検証者 $\mathcal{V}$ と証明者 $\mathcal{P}$ は、Commitment、Witness、Challenge、及び、Response の交換を 1 ラウンドのみ行う。

| Verifier $\mathcal{V}$ | | Prover $\mathcal{P}$ |
|---|---|---|
| $c \in_{\mathcal{R}} [0, n)$ | | |
| $h = \omega(c)$ | $\xrightarrow{e_0}$ | |
| | | $w \in_{\mathcal{R}} [0, n)$ |
| | | $W \stackrel{\mathcal{G}}{=} wG$ |
| | $\xleftarrow{W}$ | |
| | $\xrightarrow{c}$ | |
| | | Unless $h$ is valid, abort the session. |
| | | $r = cx + w \bmod n$ |
| | $\xleftarrow{r}$ | |
| Verify $rG \stackrel{\mathcal{G}}{=} cX + W$ | | |

The public key $X$ and the private key $x$ satisfy the equality $X \stackrel{\mathcal{G}}{=} xG$.

図 5.3: 1-round Schnorr identification algorithm: 1-RND-Schnorr

Schnorr の識別アルゴリズムでは、零知識性を保証するために、即ち、Simulator がセキュリティパラメータに関して多項式時間で終了することを保証するために、Challenge $c$ を多項式サイズの空間から選択する必要があった。この場合、Challenge の選択空間の大きさを仮に $C$ と表すとすると、不正な証明者 $\tilde{\mathcal{P}}$ は、あらかじめ $c$ を予想することにより確率 $\frac{1}{C}$ で成り済ましに成功してしまう。即ち、$c$ を予測した後、任意に $r \in [0, n_S)$ を選択し、

$$W \stackrel{\mathcal{G}_S}{=} rG_S - cX$$

により $W$ を計算する。検証者 $\mathcal{V}$ が、予測通りの $c$ を証明者 $\mathcal{P}$ に開示した場合は、先に選択した $r$ を Response として送信することで、個人鍵 $x$ を知っていなくても認証に成功する。

この攻撃を無効化するために、検証者 $\mathcal{V}$ は、Witness、Challenge、及び、Response の交換を複数ラウンド ($N$) 実行して、成り済ましの確率を $(\frac{1}{C})^N$ に低減する必要があるが、ラウンドの増加は計算処理時間と通信処理時間の増加を招くので、安易に受け入れることはできない。

そこで、本論文では、検証者 $\mathcal{V}$ が前もって Challenge $c$ の暗号学的ハッシュを発行するように Schnorr の識別アルゴリズムを変更することで、零知識対話型証明の安全性を継承しながら、Witness、Challenge、及び、Response の交換を 1 ラウンドに抑制する方法を採用する。

以下では、1-RND-Schnorr が統計的零知識性を満足する対話型証明であることを証明する。

### 1-RND-Schnorr の完全性の証明

**命題 1.** *1-RND-Schnorr* は完全である。

**証明** $rG \stackrel{\mathcal{G}}{=} (cx+w)G \stackrel{\mathcal{G}}{=} cX + W$ が成り立つことにより明らかである。 □

### 1-RND-Schnorr の健全性の証明

**命題 2.** 擬似乱数関数 $\omega : \{0,1\}^* \to \{0,1\}^k$ の安全性を仮定すると、*1-RND-Schnorr* は健全である。

**証明** $\mathcal{O}$ を以下のように定める。公平なコインを投げ、表が出た時は $\mathcal{O} = \mathcal{O}^k_{\mathrm{prf}}$ であるとし、裏が出た時は $\mathcal{O} = \mathcal{O}_{\mathrm{rnd}}$ であるとする。$\mathcal{O}$ と $\tilde{\mathcal{P}}$ をオラクルとして利用し、$\mathcal{O}^k_{\mathrm{prf}}$ と $\mathcal{O}_{\mathrm{rnd}}$ の識別を試みる多項式時間チューリング機械 $\mathcal{D}$ を以下のように構成する。

---

$\mathcal{D}$ のアルゴリズム

---

1. $\mathcal{D}$ はランダムに $c \in [1,n]$ を選択する。
2. $c$ を $\mathcal{O}$ に入力し、出力 $h_0$ を受け取る。
3. $h_0$ を $\tilde{\mathcal{P}}$ に入力し、出力 $X$ を受け取る。
4. $c$ を $\tilde{\mathcal{P}}$ に入力する。
5. ステップ 2 と 3 において、$\tilde{\mathcal{P}}$ による $\mathcal{O}^k_{\mathrm{prf}}$ への問い合わせ $q$ を受け取り、問い合わせ $q$ を $\mathcal{O}$ に発行し、得られる出力 $\mathcal{O}(q)$ を $\tilde{\mathcal{P}}$ に返す。$\tilde{\mathcal{P}}$ が停止するまでこの処理を繰り返す。
6. $\tilde{\mathcal{P}}$ の出力が `abort` ならば 0 を出力して停止し、$r$ であれば次のステップに進む。
7. $rG \stackrel{\mathcal{G}}{=} cZ + X$ を検査し、成り立てば 1 を出力、成り立たなければ 0 を出力して停止する。

---

$\mathcal{D}$ の優位性 $\mathsf{Adv}^{\mathcal{D}}$ は、

$$\mathsf{Adv}^{\mathcal{D}} = \Pr[\mathcal{D} \to 1 | \mathcal{O} = \mathcal{O}^k_{\mathrm{prf}}] - \Pr[\mathcal{D} \to 1 | \mathcal{O} = \mathcal{O}^k_{\mathrm{rnd}}]$$

と定義されるが、擬似乱数関数 $\omega(\cdot)$ は安全であると仮定すると、$\mathsf{Adv}^{\mathcal{D}}$ は無視しえるほど小さい。一方、

$$\Pr[\mathcal{D} \to 1 | \mathcal{O} = \mathcal{O}^k_{\mathrm{rnd}}] \leq \frac{1}{n} \cdot \Pr[\mathtt{abort} \not\leftarrow \tilde{\mathcal{P}}]$$

が成立するので、$\Pr[\mathcal{D} \to 1 | \mathcal{O} = \mathcal{O}^k_{\mathrm{rnd}}]$、即ち、$\Pr[\mathcal{D} \to 1 | \mathcal{O} = \mathcal{O}^k_{\mathrm{prf}}]$ は無視しえるほど小さい。 □

### 1-RND-Schnorr の統計的零知識性

**命題 3.** *1-RND-Schnorr* は統計的零知識性を満足する。

**証明** 今、$\tilde{\mathcal{V}}$ があるラウンドで出力する Commitment $h$ と Challenge $c$ が、$h = \omega(c)$ を満足しないとすると、$\tilde{\mathcal{V}}$ がこのラウンドで得る情報はラウンド毎に独立な乱数 $W$ のみであり、$\mathcal{P}$ に関する知識を付け加えることは一切ない。従って、このようなラウンドは最初から削除してしまって、常に $h = \omega(c)$ が成立するものと仮定して一般性を失わない。以下では、この仮定を置く。

不正な検証者 $\tilde{\mathcal{V}}$ を仮定して、以下のいずれかを出力する Simulator $\mathcal{S}$ を構成する。

- 擬似乱数関数 $\omega(x)$ の衝突 (collision)
- $\tilde{\mathcal{V}}$ と正当な証明者 $\mathcal{P}$ との間の対話と完全に確率分布が一致する対話

---

Simulator $\mathcal{S}$ のアルゴリズム

---

1. $\mathcal{S}$ は、$\tilde{\mathcal{V}}$ から Commitment $h$ を取得するとともに、$\tilde{\mathcal{V}}$ のステートを記録する。
2. $W \in_{\mathcal{R}} \mathcal{G}_S$ をランダムに生成して、$\tilde{\mathcal{V}}$ に入力し、Challenge $c$ を得る。
3. $r \in_{\mathcal{R}} [0, n_S)$ をランダムに生成して、$W \stackrel{\mathcal{G}_S}{=} rG_S - cX$ を計算する。
4. $\tilde{\mathcal{V}}$ のステートを Step 1. まで戻して、再度、$W$ を入力し、Challenge $c'$ を得る。
5. $c \neq c'$ であれば、ハッシュ関数 $\omega(x)$ の衝突 $(c, c')$ を出力して停止する。
6. $c = c'$ であれば、対話 $(h, W, c, r)$ を出力して停止する。

---

Simulator $\mathcal{S}$ が対話 $(h, W, c, r)$ を出力する確率を $\Pr(\mathcal{S} \to (h, W, c, r))$、Simulator $\mathcal{S}$ が $\omega(x)$ の衝突 $(c, c')$ を出力する確率を $\Pr(\mathcal{S} \to (c, c'))$、$\tilde{\mathcal{V}}$ と $\mathcal{P}$ とのプロトコルが対話 $(h, W, c, r)$ を出力する確率を $\Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (h, W, c, r))$ と表す。

1-RND-Schnorr が統計的零知識性を満足することを示すためには、

$$\sum_{(h,W,c,r)} |\Pr(\mathcal{S} \to (h, W, c, r)) - \Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (h, W, c, r))| + \sum_{\omega(c)=\omega(c')} |\Pr(\mathcal{S} \to (c, c')) - \Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (c, c'))|$$

が無視できるほど小さいことを示せばよい。

$\Pr(\tilde{\mathcal{V}} \to h)$、及び、$\Pr(\tilde{\mathcal{V}}_{h,W'} \to c)$ は、それぞれ、$\tilde{\mathcal{V}}$ が Commitment として $h$ を出力する確率、及び、

Commitment $h$ を出力し、Witness $W$ を受信した $\tilde{\mathcal{V}}$ が Challenge $c$ を出力する確率を表すものとする。

$$\begin{aligned}
&\Pr(\mathcal{S} \to (h, W, c, r)) \\
&\quad = \Pr(\tilde{\mathcal{V}} \to h) \cdot \left( \sum_{W' \in \mathcal{G}_S} \frac{1}{n_S} \cdot \Pr(\tilde{\mathcal{V}}_{h,W'} \to c) \right) \cdot \frac{1}{n_S} \cdot \Pr(\tilde{\mathcal{V}}_{h,W} \to c) \\
&\Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (h, W, c, r)) \\
&\quad = \Pr(\tilde{\mathcal{V}} \to h) \cdot \frac{1}{n_S} \cdot \Pr(\tilde{\mathcal{V}}_{h,W} \to c)
\end{aligned}$$

が成り立つので、特に、

$$\Pr(\mathcal{S} \to (h, W, c, r)) < \Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (h, W, c, r))$$

が成り立つ。従って、

$$\begin{aligned}
&\sum_{(h,W,c,r)} |\Pr(\mathcal{S} \to (h, W, c, r)) - \Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (h, W, c, r))| \\
&\quad + \sum_{\omega(c)=\omega(c')} |\Pr(\mathcal{S} \to (c, c')) - \Pr(\langle \tilde{\mathcal{V}}, \mathcal{P} \rangle \to (c, c'))| = 2 \cdot \Pr(\mathcal{S} \to (c, c'))
\end{aligned}$$

を得る。$\omega(x)$ には非衝突性を仮定しているので、右辺はセキュリティパラメータに対して無視しえるほど小さく、従って、1-RND-Schnorr は、統計的零知識性を満足する。 □

#### 5.2.2.2 Unlink-Verify-ZKIP の完全性の証明

**定理 4.** *Unlink-Verify-ZKIP* は完全である。

**証明** *UA* による検証式 (5.1)、及び、*SA* による検証式 (5.2) の両方が成立することを示せば十分である。

$$(r' + c \cdot aid) G_S \overset{\mathcal{G}_S}{=} cS + W' \tag{5.1}$$

$$(r + c \cdot anm) G_S \overset{\mathcal{G}_S}{=} cS + W \tag{5.2}$$

検証式 (5.1) が成立することは、$r' = ck + w' \bmod n_S$ かつ $aid = \sigma - k \bmod n_S$ より、下式で示される。

$$(r' + c \cdot aid) G_S \overset{\mathcal{G}_S}{=} (ck + w' + c(\sigma - k)) G_S \overset{\mathcal{G}_S}{=} cS + W'$$

一方、$r = r' + c\rho + w'' \mod n_S$ であるので、

$$(r + c \cdot anm) G_S \overset{\mathcal{G}_S}{=} (r' + c(anm + \rho) + w'') G_S \overset{\mathcal{G}_S}{=} (r' + c \cdot aid) G_S + w'' G_S \overset{\mathcal{G}_S}{=} cS + W' + W''$$

より、検証式 (5.2) も成立する。 □

#### 5.2.2.3 Unlink-Verify-ZKIP の健全性の証明

**定義 1.** *UA* にとって *Authentic* な *SpA* とは、以下の条件を満足する *SpA* を指す。

- *UA* は *aid* を保持している。
- *SpA* は $k$ を保持している。
- *UA* が *SpA* に入力した $*$ に対して、下記の関係式が成立する。

$$aid = \sigma - \mu(k, *) \bmod n_S$$

**定理 5.** *Unlink-Verify-ZKIP* は健全である。

**証明** Unlink-Verify-ZKIP の健全性を示すためには、*SA* が *UA* の認証に有意な確率で成功するならば、*UA* は必ず Authentic な *SpA* から出力を得ていることを証明すればよい。

1-RND-Schnorr の証明者 $\tilde{\mathcal{P}}$ を以下のように構成する。

---

$\tilde{\mathcal{P}}$ のアルゴリズム

---

1. *UA* から *anm* を受け取る。
2. 1-RND-Schnorr の検証者 $\mathcal{V}$ から Commitment $h$ を受け取り、*UA* に入力する。
3. *UA* から Witness $W$ を受け取り、$\mathcal{V}$ に出力する。
4. $\mathcal{V}$ から Challenge $c$ を受け取り、*UA* に入力する。
5. *UA* から $r$ を受け取り、

$$\tilde{r} = r + c \cdot anm \bmod n_S$$

を $\mathcal{V}$ に出力する。

---

$\mathcal{V}$ は式

$$\tilde{r}G_S \stackrel{?}{=} cS + W$$

の検証に有意な確率で成功するため、1-RND-Schnorr の健全性、即ち、命題 5 により、*UA* は少なくともひとつの *SpA* と通信を行って $r$ を計算しなければならない。

一方、*UA* がアクセスする *SpA* が Authentic でない場合、*UA* は確率的に *SpA* をエミュレートできる。従って、*UA* は、ひとつ以上の Authentic な *SpA* と通信を行い、$r$ を計算したことが分かる。 □

#### 5.2.2.4 Unlink-Verify-ZKIP の追跡不能性の証明

Unlink-Verify-ZKIP の追跡不能性は、定理 6 の証明に示すように、1-RND-Schnorr の零知識性に帰着する。

**定理 6.** *Unlink-Verify-ZKIP* は追跡不能である。

**証明** *UA* が *SpA* の検証式 (5.1) の検証に成功し $r$ を *UA* に送付した場合と、検証に失敗しセッションを中止した場合とに分けて示す。

まず、*UA* が式 (6.11) が成立することを検証した場合を考える。*UA* と *SA* の対話は、公開鍵 $S - anm \cdot G_S$ に関する 1-RND-Schnorr であるので、命題 3 により統計的零知識性を満足する。即ち、*SA* が対話から学習できる唯一の情報は *anm* であるが、*UA* は定義域 $\rho$ を $[0, n_S)$ から一様にランダムに選択するので、*anm* も $[0, n_S)$ 中を一様に分布する。更に、*UA* による $\rho$ の選択は、Unlink-Verify-ZKIP の実行の都度、確率的な独立事象として行われるため、*SA* は追跡に繋がる有意な情報を得ることはない。

一方、*UA* が式 (6.11) を否定的に検証した場合には、*SA* は *anm* と Witness $W$ を得るのみである。*anm* と $W$ は、互いに独立な乱数であり、それぞれ $[0, n_S)$ と $\mathcal{G}_S$ において一様に分布するため、やはり、*SA* は対話を通じてなんら有意な情報を得ることはない。 □

### 5.2.3 計算量の評価

Unlink-Verify-ZKIP の計算量を表 5.1 に示す。

表 5.1: Unlink-Verify-ZKIP の計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Unlink-Verify-ZKIP | 2 | 3 | 1 | 6 |

(楕円曲線上のスカラー倍演算の実行回数)

## 5.3 電子署名を利用したプロトコル

### 5.3.1 プロトコルの定義

この章では、電子署名の Non-malleability に安全性の根拠を置く権利認証プロトコルを示す。Unlink-Verify-ZKIP と同じく、この章で示す権利認証プロトコルは、*SA*、*UA* 及び *SpA* 間のプロトコルであり、*UA* は *SpA* の出力を平準化することにより、追跡情報が *UA* にもれることを防ぐ。

Unlink-Verify-ZKIP は追跡不能性のみをサポートしていたが、電子署名に基づく本章のプロトコルは、追跡不能性をサポートする Unlink-Verify と追跡性をサポートする Link-Verify との 2 つのプロトコルから構成される。

**Unlink-Verify** 追跡を可能とする情報が一切外部に漏れることのない、追跡不能性をサポートする。プロトコルの健全性はベースとなる署名方式の Non-malleability に依拠し、追跡不能性は *UA* による署名の確率的平準化に依拠する。

**Link-Verify** Consensual disclosure の原則に基づき、ユーザが明示的に合意した場合に限り、アクセスの根拠となる *aid* のみが開示される。但し、*aid* は *aid* を発行した *SP* のみが知り得るように暗号化され、*SA* 等からは秘匿される。*aid* の暗号化は *SpA* が行うが、*UA* は暗号文の確率的平準化を行うとともに、*aid* 以外の情報が暗号化されていないことを確率的手法で検査する。

Unlink-Verify 及び Link-Verify の定義は、それぞれ、図 5.4 及び図 5.5 で与えられる。

#### 5.3.1.1 Unlink-Verify の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w' \in_{\mathcal{R}} [0, n_S)$ |
| | $\xleftarrow{W'}$ | $W' \overset{G_S}{=} w' G_S$ |
| | $\rho \in_{\mathcal{R}} [0, n_S)$ | |
| | $anm = aid - \rho \bmod n_S$ | |
| | $w'' \in_{\mathcal{R}} [0, n_S)$ | |
| | $W \overset{G_S}{=} w'' G_S + W'$ | |
| | $\xleftarrow{W, anm}$ | |
| $c \in_{\mathcal{R}} [0, n_S)$ | | |
| | $\xrightarrow{c, *}$ | |
| | $\xrightarrow{c, *, w''}$ | |
| | | $W \overset{G_S}{=} w'' G_S + W'$ |
| | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | $r' = a\mu(k, *) + w' + w'' \bmod n_S$ |
| | $\xleftarrow{r}$ | |
| | Unless $(W, r')$ is valid, abort the session. | |
| | $a = \omega(\pi(W) \mid c \mid *)$ | |
| | $r = r' + a\rho \bmod n_S$ | |
| | $\xleftarrow{r}$ | |
| Unless $(W, r)$ is valid, abort the session. | | |

図 5.4: 追跡不能権利認証プロトコル: Unlink-Verify

所謂、Fiat-Shamir heuristic [34] により、$(W, r')$ 及び $(W, r)$ は電子署名となり、*UA* 及び *SA* は以下の検証式に従い、それぞれ、*SpA* 及び *UA* が出力した署名を検証する。

**UA による検証式**

$$r' G_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - aid \cdot G_S) + W \tag{5.3}$$

**SA による検証式**

$$r G_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - anm \cdot G_S) + W \tag{5.4}$$

*SA* はランダムに Challenge $c$ を生成し、$(r, W)$ は Challenge $c$ に対する署名であるので、*SA* は Challenge-Response 手法に基づいて *UA* の権利を認証したこととなり、*UA* による Replay 攻撃を防止する効果を有する。

#### 5.3.1.2 Link-Verify の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w', q' \in_{\mathcal{R}} [0, n_S)$ |
| | | $W' \overset{G_S}{=} w'G_S$ |
| | | $Q' \overset{G_S}{=} q'G_S$ |
| | $\xleftarrow{W', U'}$ | |
| | $\rho, w'', q'', x, y \in_{\mathcal{R}} [0, n_S)$ | |
| | $anm = aid - \rho \bmod n_S$ | |
| | $W \overset{G_S}{=} w''G_S + W'$ | |
| | $Q \overset{G_S}{=} q''G_S + Q'$ | |
| | $U \overset{G_S}{=} xG_S + yQ$ | |
| | $\xleftarrow{anm, W}$ | |
| $c \in_{\mathcal{R}} [0, n_S)$ | | |
| | $\xrightarrow{c, *}$ | |
| | $\xrightarrow{c, *, w''}$ | |
| | $\xrightarrow{q'', U, \rho}$ | |
| | | $W \overset{G_S}{=} w''G_S + W'$ |
| | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | $r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$ |
| | | $Q \overset{G_S}{=} q''G_S + Q'$ |
| | | $e_P = \pi((\mu(k, *) + \rho)Q) \oplus \rho$ |
| | | $V \overset{G_S}{=} (\mu(k, *) + \rho)U$ |
| | | $b = \omega(r \mid e_P \mid \pi(Q))$ |
| | | $s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$ |
| | $\xleftarrow{r, s, e_P, V}$ | |
| | Unless $W, r, Q, s, e_P, V$ are valid, abort the session. | |
| Unless $(W, r, Q, s, e_P)$ is valid, abort the session. | $\xleftarrow{r, Q, s, e_P}$ | |

図 5.5: 追跡可能権利認証プロトコル: Link-Verify

*UA* 及び *SA* による検証式は、以下のように与えられる。

**UA による検証式**

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)\,(S - anm \cdot G_S) + W \tag{5.5}$$

$$sG_S \overset{G_S}{=} \omega(r \mid e_P \mid \pi(Q))\,(S - anm \cdot G_S) + Q \tag{5.6}$$

$$V \overset{G_S}{=} x(S - anm \cdot G_S) + y \cdot \pi^{-1}(e_P \oplus \rho) \tag{5.7}$$

**SA による検証式** (5.5) 及び (5.6)

**SP による *aid* の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で *aid* を復号する。

1. 検証式 (5.5) 及び (5.6) を検査する。
2. 下式により $\rho$ 及び *aid* を計算する。

$$\rho = \pi((\sigma - aaid)Q) \oplus e_P$$

$$aid = anm + \rho \bmod n_S$$

Link-Verify の定義において、$(anm, W, r)$ の計算に用いられた署名スキームと、$(anm, Q, s)$ の計算に用いられた署名スキームとは互いに異なる署名スキームである。具体的には、前者においてハッシュを計算するときには $\pi(W)$ は **Prefix** として組み込まれるのに対し、後者では $\pi(Q)$ は **Suffix** である。**Prefix** であるか、**Suffix** であるかに本質的な意味はないが、両者が異なる署名スキームであり、かつ、両方とも Non-malleable であることが、後述する定理 12 の証明において本質的な意味を有する。

### 5.3.2 安全性の評価

#### 5.3.2.1 Unlink-Verify 及び Link-Verify の完全性の証明

**定理 7.** *Unlink-Verify* は完全である。

**証明** *UA* による検証式 (5.3)、及び、*SA* による検証式 (5.4) がそれぞれ成立することを示せば十分である。

式 (5.3) が成立することは、以下のように確かめられる。

$$\begin{aligned} r'G_S &\overset{G_S}{=} \big(\omega(\pi(W)|c|*)\mu(k,*) + w' + w''\big)G_S \\ &\overset{G_S}{=} \omega(\pi(W)|c|*)\big(\sigma - (\sigma - \mu(k,*))\big)G_S + W \\ &\overset{G_S}{=} \omega(\pi(W)|c|*)\left(S - aid \cdot G_S\right) + W \end{aligned}$$

一方、式 (5.4) が成立することは、以下のように確かめられる。

$$\begin{aligned} rG_S &\overset{G_S}{=} \big(\omega(\pi(W)|c|*)(\mu(k,*) + \rho) + w' + w''\big)G_S \\ &\overset{G_S}{=} \omega(\pi(W)|c|*)\big(\sigma - (\sigma - \mu(k,*) - \rho\big)G_S + W \\ &\overset{G_S}{=} \omega(\pi(W)|c|*)\left(S - anm \cdot G_S\right) + W \end{aligned}$$

□

**定理 8.** *Link-Verify* は完全である。

**証明** 検証式 (5.5) 及び (5.6) が成立することは、定理 7 の場合と同様に示すことができるので、検証式 (5.7) が成立することを示せば十分である。

$$\begin{aligned} & x(S - anm \cdot G_S) + y \cdot \pi^{-1}(e_P \oplus \rho) \\ &\overset{G_S}{=} x(\mu(k,*) + \rho)G_S + y(\mu(k,*) + \rho)Q \\ &\overset{G_S}{=} (\mu(k,*) + \rho)(xG_S + yQ) \\ &\overset{G_S}{=} V \end{aligned}$$

□

#### 5.3.2.2 Unlink-Verify 及び Link-Verify の健全性の証明

**定理 9.** *Unlink-Verify* 及び *Link-Verify* は健全である。

**証明** Fiat-Shamir heuristic [34] に従えば、*UA* が出力する $(W,r)$ をメッセージ $c \mid *$ に対する電子署名と見做すことができ、同様に、*SpA* の出力も $c \mid *$ への署名となる。署名の検証は、公開鍵 $S - aid \cdot G_S$、或いは、公開鍵 $S - anm \cdot G_S$ による。このような見做しものとで、Unlink-Verify 及び Link-Verify の健全性は、以下の Claim で表現することができる。

**Claim .** *SA* が *UA* が提出した電子署名 $(W,r)$ の検証に成功するならば、無視しえる確率を除いて、*UA* にとって *Authentic* な *SpA* が少なくともひとつ存在して、同一のメッセージ $c \mid *$ に対して署名を生成している。

以下では、上記の Claim が、ベースとなる電子署名方式の Non-malleability に帰着することを示す。

問題の設定を一般化して、*UA* は、*SA* からメッセージ $m$ の入力を受け、

$$rG_S \overset{G_S}{=\!=} \omega(\pi(W) \mid m)\,(S - anm \cdot G_S) + W \tag{5.8}$$

を満足する $(anm, W, r)$ を出力するものとする。その際、*UA* は選択平文攻撃 (adaptive plaintext attack) を実行する能力を有すると仮定する。即ち、*UA* は、Authentic であるか否かを問わず任意の *SpA* に対して任意のメッセージ $m_*$ を問い合わせ、その答えとして $m_*$ への署名 $(W_*, r_*)$ を無条件に得る。

実際には、*UA* がアクセスする *SpA* は、全て *UA* にとって Authentic であると仮定してよい。

これは、*SP* が正しく *aid* の発行を行えば、*SpA* が計算する $\mu(k,*)$ は一様に分布する事実による。即ち、Authentic でない *SpA* へのアクセスは、等しい確率で $\mu(k,*)$ を選択することによって、*UA* 自身でシミュレートできるからである。

従って、*UA* は、*SpA* から受け取った $(W_*, r_*)$ に対して、下式を成立させる $aid_*$ を常に知っているとしてよい。

$$r_*G_S \overset{G_S}{=\!=} \omega(\pi(W_*) \mid m_*)\,(S - aid_* \cdot G_S) + W_* \tag{5.9}$$

さて、$\mathcal{O}$ を公開鍵ペア $(S,\sigma)$ に関する署名オラクルであるとする。即ち、$\mathcal{O}$ は、$\sigma$ を保持し、問い合わせ $m_*$ を受信すると、署名鍵 $\sigma$ による $m_*$ への署名 $(\bar{W}_*, \bar{r}_*)$ を無条件に計算して、返答する。$(\bar{W}_*, \bar{r}_*)$ は下式を満足する。

$$\bar{r}_*G_S \overset{G_S}{=\!=} \omega(\pi(\bar{W}_*) \mid m_*)S + \bar{W}_*. \tag{5.10}$$

*UA* と $\mathcal{O}$ とを利用して、署名スキームに選択平文攻撃を行う $\mathcal{A}$ を以下のように構成する。*UA* は、*SpA* に問い合わせを行う代わりに、$\mathcal{A}$ に問い合わせを行う。

- $\mathcal{A}$ は、メッセージ $m$ をランダムに生成し、*UA* に入力する。
- *UA* が $m_*$ を問い合わせると、$\mathcal{A}$ は $\mathcal{O}$ に $m_*$ を問い合わせる。更に、$\mathcal{A}$ は、得られた出力 $(\bar{W}_*, \bar{r}_*)$ と *UA* の知識 $(aid_*)$ から、下式により $(W_*, r_*)$ を計算して、*UA* に返す。

  $$W_* \overset{G_S}{=\!=} \bar{W}_*, \quad r_* = \bar{r}_* - \omega(\pi(\bar{W}_*) \mid m_*) \cdot aid_* \bmod n_S$$

  上式により計算した $(W_*, r_*)$ は、式 (5.9) を満足するのみではなく、$\bar{W}_*$ が一様に分布することから、*SpA* に問い合わせを行った場合の出力と同じ確率分布をとる。即ち、*UA* は、*SpA* と $\mathcal{A}$ とを区別できない。

- *UA* が検証式 (5.8) を満足する $(anm, W, r)$ を出力すると、$\mathcal{A}$ は下式に従って $\bar{W}$ 及び $\bar{r}$ を計算して、$(m, \bar{W}, \bar{r})$ を出力する。

  $$\bar{W} \stackrel{\mathcal{G}_S}{=} W, \quad \bar{r} = r + \omega(\pi(W) \mid m) \cdot anm$$

  $(\bar{W}, \bar{r})$ は、

  $$\begin{aligned} \bar{r}G_S &\stackrel{\mathcal{G}_S}{=} \omega(\pi(W) \mid m)(S - anm \cdot G_S) + W + \omega(\pi(W) \mid m) \cdot anm \cdot G_S \\ &\stackrel{\mathcal{G}_S}{=} \omega(\pi(W) \mid m)S + W \end{aligned}$$

  を満たすので、公開鍵 $S$ により検証可能な、$m$ への署名である。

このように、$\mathcal{A}$ は、$\mathcal{O}$ を $S$ に関する署名オラクルとして任意に利用しながら、署名を出力するが、ベースとなる署名スキームが Non-malleable であれば、$\mathcal{A}$ が $\mathcal{O}$ にメッセージ $m$ を問い合わせることなく $m$ への有効な署名 $(\bar{W}, \bar{r})$ を生成できる確率は無視しえるほど小さい。従って、$\mathcal{A}$ の定義により、*UA* がメッセージ $m$ を *SpA* に問い合わることなく有効な $(anm, W, r)$ を出力できる確率は無視できるほど小さい。 □

#### 5.3.2.3 Unlink-Verify の追跡不能性の証明

**定理 10.** *Unlink-Verify* は追跡不能である。

**証明** *UA* が式 (5.3) の検証に失敗した場合、*SA* は互いに $anm$ と $W$ を得るのみである。しかし、$anm$ と $W$ とは互いに独立な乱数 $\rho$ 及び $w''$ によって乱数化されているので、$(anm, W)$ は $[0, n_S) \times \mathcal{G}_S$ 上で一様な確率で分布する。

まず、次の Claim を証明する。

**Claim .** *UA* による *SA* への出力 $(anm, W, r)$ は、他のいかなる事象とも独立に、

$$D_{c,*} = \{(x, Y, z) \mid zG_S \stackrel{\mathcal{G}_S}{=} \omega(\pi(Y) \mid c \mid *)(S - xG_S) + Y\} \subset [0, n_S) \times \mathcal{G}_S \times [0, n_S)$$

上で一様に分布する。

**証明 (Claim)** $(anm, W)$ が定まれば、$D_{c,*}$ 上で、$r$ の値は一意に定まる。一方、$(anm, W)$ は $[0, n_S) \times \mathcal{G}_S$ 上で一様な確率で分布するので、Claim は証明された。 □

Claim から定理が導かれることは、以下のように分かる。$D_{c,*}$ の定義は $S$ と $(c, *)$ のみに依存し、*aid* には依存しない。即ち、*UA* が正しい処理を行う限りにおいては、*aid* の値にも認証イベントにも依存せずに、$(anm, W, r)$ は独立に一様に分布することから、$(anm, W, r)$ は一切の追跡情報を含み得ない。 □

#### 5.3.2.4 Link-Verify による Consensual Disclosure のサポートの証明

$(Q, e_P)$ は公開鍵 $S - anm \cdot G_S$ による $\rho$ の ElGamal 暗号であるので、$\sigma$ を知っている *SP* にのみ復号可能である。従って、Link-Verify が Consensual Disclosure をサポートしていることを示すためには、以下の 2 項目を示すことが必要かつ十分である。

- *SpA* が不正であっても、*UASP* に対して *aid* のみを開示し、他のエンティティには一切の追跡情報を開示しない。
- *UA* が不正であっても、*SP* は *UA* が認証に用いた *aid* を特定することが可能である。

上記の 2 項目は、以下に述べる定理 11 及び定理 12 によって、事実であることが証明される。

**定理 11.** *Link-Verify* において、*UA* は *SP* に対して *aid* のみを開示し、他のエンティティには一切の追跡情報を開示しない。

**証明** *UA* が検証式 (5.5) 乃至 (5.7) の検証に失敗すれば、*UA* が開示する情報は $[0, n_S) \times \mathcal{G}_S$ 上を一様に分布する $(anm, W)$ のみであるため、いかなる有用な情報をも含み得ない。

以下では、*UA* が検証式 (5.5) 乃至 (5.7) の検証に成功したと仮定して、まず、*SP* 以外のエンティティの視点からは、*UA* の出力が追跡不能であることを見る。

正しい *UA* の出力 $(anm, W, r, Q, s, e_P)$ において、$(anm, W, r, Q, s)$ は、$S$ にのみ依存して定義される曲面上を一様に分布する。従って、$(anm, W, r, Q, s)$ は、追跡情報を含まないことが、定理 10 と同様の議論で示される。

つまり、$e_P$ 及び $e_P$ と関連するデータのみが追跡性に影響するので、*UA* は $(anm, Q, e_P)$ のみを出力すると仮定しても一般性は失われない。

今、以下の性質を満足する多項式時間の攻撃者 $\mathcal{A}$ を考える。

- 事前に $(\mathcal{G}_S, G_S, n_S, S)$ の知識を有しているが、$\sigma$ に関する知識はない ($\mathcal{A}$ は *SP* と異なる)。
- 任意に *UA* にアクセスして出力を得ることができる。
- 入力 $(x, Y, z) \quad \in [0, n_S) \times \mathcal{G}_S \times \{0,1\}^k$ に対して、0 または 1 を出力する。

更に、公平なコインを振り、表がでれば *UA* にアクセスして

$$aid = anm + \pi(\sigma Y) \oplus z \bmod n_S$$

を満足する $(x, Y, z)$ を取得して $\mathcal{A}$ に入力し、裏がでれば一様にランダムに $(x, Y, z) \in_{\mathcal{R}} [0, n_S) \times \mathcal{G}_S \times \{0,1\}^k$ を生成して $\mathcal{A}$ に入力する。

*SP* 以外のエンティティに対する追跡不能性は、次の Claim から導かれる。

**Claim .** 式 *5.11* で定義される $\mathcal{A}$ の優位性 $\mathsf{Adv}^{\mathcal{A}}$ は無視しえる程小さい。

$$\mathsf{Adv}^{\mathcal{A}} = \Pr[\mathcal{A} \to 1 | (x, Y, z) \leftarrow UA] - \Pr[\mathcal{A} \to 1 | (x, Y, z) \in_{\mathcal{R}} [0, n_S) \times \mathcal{G}_S \times \{0,1\}^k] \tag{5.11}$$

以下では、Decisional Diffie-Hellman 問題の困難性に帰着させることにより、上記の Claim を証明する。

以下では、$\mathcal{A}$ を利用して、Decisional Diffie-Hellman 問題を解く多項式時間チューリング機械 $\tilde{\mathcal{P}}$ を構成する。$\tilde{\mathcal{P}}$ は、群定義 $(\mathcal{G}_S, G_S, n_S)$ と $(S, Q, R) \in \mathcal{G}_S^3$ とを与えられて、

$$\text{mult}_{G_S} S = \text{mult}_Q R \tag{5.12}$$

が成立するか否かを、以下のアルゴリズムに従って、判定する。

---

$\tilde{\mathcal{P}}$ のアルゴリズム

---

1. *aid* と $\rho \in [0, n_S)$ をランダムに生成し、$\mathcal{A}$ に $(aid - \rho \bmod n_S, Q, \pi(R) \oplus \rho)$ を入力する。
2. $\mathcal{A}$ から問い合わせを受ける都度、$x, y \in_{\mathcal{R}} [0, n_S)$ を独立にランダムに生成し、$(aid - x \bmod n_S, yG_S, \pi(yS) \oplus x)$ を $\mathcal{A}$ に返す。
3. $\mathcal{A}$ が 1 を出力した時には、1 を出力して停止し、0 を出力した時には、0 を出力して停止する。

---

$\tilde{\mathcal{P}}$ の優位性 $\mathsf{Adv}^{\tilde{\mathcal{P}}}$ を式 5.13 と定義する。

$$\mathsf{Adv}^{\tilde{\mathcal{P}}} = \Pr[\tilde{\mathcal{P}} \to 1 | \text{mult}_{G_S} S = \text{mult}_Q R] - \Pr[\tilde{\mathcal{P}} \to 1 | \text{mult}_{G_S} S \neq \text{mult}_Q R(x, Y, z)] \tag{5.13}$$

$\tilde{\mathcal{P}}$ は、$\mathcal{A}$ からの問い合わせに対し、*UA* と全く同じ分布の出力を返すので、

$$\mathsf{Adv}^{\mathcal{A}} = \mathsf{Adv}^{\tilde{\mathcal{P}}}$$

が成立する。

一方、Decisional Diffie-Hellman 問題の困難性の仮定より、$\mathsf{Adv}^{\tilde{\mathcal{P}}}$ は無視しえるほど小さく、従って、$\mathsf{Adv}^{\mathcal{A}}$ も無視しえるほど小さいので、Claim が証明された。

*SP* の視点から、*aid* 以外に追跡情報が開示されないことも、定理 10 と同様の議論で示される。

即ち、$e_P$ を復号できる *SP* にとって、$e_P$ は *aid* と同値である。従って、*aid* を除くと、*SP* が得る情報は $(anm, W, r, Q, s)$ となるが、先に述べたように、$(anm, W, r, Q, s)$ は $(S, c, *)$ にのみ依存する領域上を一様に分布するため、追跡情報を包含することはない。

□

**定理 12.** *Link-Verify* において、*SP* は正しい *aid* を復元する。

**証明** $(anm, Q, s)$ に対する署名スキームは $(anm, W, r)$ に対する署名スキームとは異なるが (ハッシュ計算における $\pi(Q)$ と $\pi(W)$ の挿入位置が異なる)、Non-malleability は同様に成立するので、定理 9 の証明と同様の議論により、*UA* にとって Authentic な *SpA* が $r \mid e_P$ に署名を行っていることが分かる。更に、この *SpA* は、Link-Verify の定義に従って、$r$ 及び $e_P$ の計算を行っていることも示される。

さて、与えられた $r$、$c$、及び、$*$ に対して、

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - anm \cdot G_S) + W$$

を満足する *anm* は一意に定まるので、それは当該 *SpA* が計算に用いた $\mu(k, *)$ 及び $\rho$ に対して、

$$anm = \sigma - \mu(k, *) - \rho \bmod n_S$$

を満足する。従って、*SP* が $r$ の検証に用いた *anm* に対して、$(anm, Q, e_P)$ から復号した $\rho$ を用いて計算した $aid = anm + \rho \bmod n_S$ は、*SpA* に対応する *aid* であることが示された。 □

### 5.3.3 計算量の評価

Unlink-Verify 及び Link-Verify の計算量を表 5.2 に示す。

表 5.2: Unlink-Verify 及び Link-Verify の計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Unlink-Verify | 2 | 3 | 2 | 7 |
| Link-Verify | 4 | 11 | 6 | 21 |

(楕円曲線上のスカラー倍演算の実行回数)

# 第6章　権限発行者と権限検証者を分離したプロトコルの提案

## 6.1 検証者認証機能

### 6.1.1 プロトコルの定義

この節では、Unlink-Verify 及び Link-Verify に、*SpA* が Diffie-Hellman 鍵交換アルゴリズムに従って *SA* を認証する機能を追加して、Unlink-Authenticate-SA 及び Link-Authenticate-SA を定義する。

以下では、$(A, \alpha)$ を *SA* の公開鍵ペアで、$A \overset{G_S}{=} \alpha G_S$ を満たすものとする。*SpA* は何らかの手段により $A$ を取得し、*SA* は $\alpha$ を保持しているものとし、下記の手順で認証を行う。

***SA* による鍵 *key* の計算** *SA* は、*UA* から Witness $W$ を受け取り、

$$key = \omega(\pi(\alpha W) \mid c)$$

により *key* を計算する。

***SpA* による鍵 *key* の計算** *SpA* は、

$$key = \omega(\pi((w' + w'')A) \mid c)$$

により *key* を計算する。但し、$w', w'' \in [0, n_S)$ は $W \overset{G_S}{=} (w' + w'')G_S$ を満足する乱数である。

***SpA* による *SA* の認証** *SA* は、*key* の暗号学的ハッシュ$e_1 = \omega(key)$ を計算し、*SpA* に送付する。*SpA* は自身が計算した *key* のハッシュが、$e_1$ に一致することを検証する。

#### 6.1.1.1 Unlink-Authenticate-SA の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w' \in_{\mathcal{R}} [0, n_S)$ |
| | $\xleftarrow{W'}$ | $W' \overset{G_S}{=} w' G_S$ |
| | $\rho \in_{\mathcal{R}} [0, n_S)$ | |
| | $anm = aid - \rho \bmod n_S$ | |
| | $w'' \in_{\mathcal{R}} [0, n_S)$ | |
| | $W \overset{G_S}{=} w'' G_S + W'$ | |
| | $\xleftarrow{W, anm}$ | |
| $key = \omega(\pi(\alpha W)\|c)$ | | |
| $e_1 = \omega(key)$ | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | | |
| | $\xrightarrow{c, *, e_1}$ | |
| | | $\xrightarrow{c, *, e_1, w''}$ |
| | | $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | Unless $e_1$ is valid, abort the session |
| | | $W \overset{G_S}{=} w'' G_S + W'$ |
| | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | $r' = a\mu(k, *) + w' + w'' \bmod n_S$ |
| | | $\xleftarrow{r}$ |
| | Unless $(W, r')$ is valid, abort the session. | |
| | $a = \omega(\pi(W) \mid c \mid *)$ | |
| | $r = r' + a\rho \bmod n_S$ | |
| Unless $(W, r)$ is valid, abort the session. | $\xleftarrow{r}$ | |

図 6.1: 検証者認証機能つき追跡不能権利認証プロトコル: Unlink-Authenticate-SA

#### 6.1.1.2 Link-Authenticate-SA の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w', q' \in_{\mathcal{R}} [0, n_S)$<br>$W' \overset{\mathcal{G}_S}{=} w'G_S$<br>$Q' \overset{\mathcal{G}_S}{=} q'G_S$ |
| | $\xleftarrow{W', U'}$ | |
| | $\rho \in_{\mathcal{R}} [0, n_S)$<br>$w'' \in_{\mathcal{R}} [0, n_S)$<br>$W \overset{\mathcal{G}_S}{=} w''G_S + W'$<br>$q'' \in_{\mathcal{R}} [0, n_S)$<br>$Q \overset{\mathcal{G}_S}{=} q''G_S + Q'$<br>$x, y \in_{\mathcal{R}} [0, n_S)$<br>$U \overset{\mathcal{G}_S}{=} xG_S + yQ$ | |
| $key = \omega(\pi(\alpha W)\|c)$<br>$e_1 = \omega(key)$ | $\xleftarrow{anm, W}$ | |
| $c \in_{\mathcal{R}} [0, n_S)$ | $\xrightarrow{c, *, e_1}$ | $\xrightarrow{c, *, e_1, w''}$ |
| | | $\xrightarrow{q'', U, \rho}$ |
| | | $key = \omega(\pi((w' + w'')A)\|c)$<br>Unless $e_1$ is valid, abort the session<br>$W \overset{\mathcal{G}_S}{=} w''G_S + W'$<br>$a = \omega(\pi(W) \mid c \mid *)$<br>$r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$<br>$Q \overset{\mathcal{G}_S}{=} q''G_S + Q'$<br>$e_P = \pi((\mu(k, *) + \rho)Q) \oplus \rho$<br>$V \overset{\mathcal{G}_S}{=} (\mu(k, *) + \rho)U$<br>$b = \omega(r \mid e_P \mid \pi(Q))$ |
| | | $\xleftarrow{r, s, e_P, V}$ |
| | | $s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$ |
| | Unless $W, r, Q, s, e_P, V$ are valid, abort the session. | |
| Unless $(W, r, Q, s, e_P)$ is valid, abort the session. | $\xleftarrow{r, Q, s, e_P}$ | |

図 6.2: 検証者認証機能つき追跡可能権利認証プロトコル: Link-Authenticate-SA

### 6.1.2 安全性の評価

権利認証としての、完全性、健全性、追跡不能性、及び、Consensual Disclosure のサポートに関する安全性は、Unlink-Verify 及び Link-Verify から継承される。

*SpA* による *SA* の認証の完全性と健全性は、Diffie-Hellman 鍵交換アルゴリズムの完全性と健全性に帰着される。

### 6.1.3 計算量の評価

Unlink-Authenticate-SA 及び Link-Authenticate-SA の計算量を表 5.2 に示す。

表 6.1: Unlink-Authenticate-SA 及び Link-Authenticate-SA の計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Unlink-Authenticate-SA | 3 | 3 | 3 | 9 |
| Link-Authenticate-SA | 5 | 11 | 7 | 23 |

(楕円曲線上のスカラー倍演算の実行回数)

## 6.2 検証者認証に基づく鍵転送

### 6.2.1 プロトコルの定義

鍵送付は、*SP* が *SA* に秘密の鍵を送付する機能である。*SA* は、権利認証プロトコルを成功ステータスで完了した場合に限り、該秘密鍵を取得する。即ち、*SpA* が $k$ を所有することを *SA* が検証し、かつ、*SpA* が *SA* を認証した場合に限り、*SA* はメッセージから秘密鍵を計算することが出来る。

$\kappa \in_{\mathcal{R}} [0, n_S)$ を *SP* が生成した乱数とし、$K \stackrel{G_S}{=} \kappa S$ を *SP* が *SA* に送付したい秘密鍵であるとする。鍵を送付するために、*SP* は *SA* に対し、$L \stackrel{G_S}{=} \kappa G_S$ を発行し、*SA* は**鍵送付プロトコル**により $K$ を再現する。

鍵送付プロトコルを実行するに当たり、*SA* は、乱数 $\lambda \in [0, n_S)$ を生成して $C \stackrel{G_S}{=} \lambda L$ を *SpA* に送付するように $L$ を乱数化するが、これは、通信を傍受する攻撃者に対して $K$ を保護することが目的である。

$K$ の利用方法の典型的な例は、*Service* のコンテンツを暗号化するための鍵として利用するものである。例えば、*SA* は楽曲の再生プレイヤであるとする。*SP* の立場からは、安全性が確認され、認定された再生プレイヤでのみ、暗号化された楽曲コンテンツが復号され、再生されるべきである。そのために、$K$ で楽曲コンテンツを暗号化しておけば、鍵送付プロトコルにより *SpA* が *SA* を認証した場合に限り、$K$ が再現され、楽曲コンテンツの復号が可能となる。

#### 6.2.1.1 Unlink-Key-Transfer の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w' \in_{\mathcal{R}} [0, n_S)$ |
| | $\xleftarrow{W'}$ | $W' \overset{\mathcal{G}_S}{=} w' G_S$ |
| | $\rho \in_{\mathcal{R}} [0, n_S)$ | |
| | $anm = aid - \rho \bmod n_S$ | |
| | $w'' \in_{\mathcal{R}} [0, n_S)$ | |
| | $W \overset{\mathcal{G}_S}{=} w'' G_S + W'$ | |
| | $x, z \in [0, n_S)$ | |
| | $U \overset{\mathcal{G}_S}{=} xG_S + zC$ | |
| | $\xleftarrow{W, anm}$ | |
| $key = \omega(\pi(\alpha W)\|c)$ | | |
| $e_1 = \omega(key)$ | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | | |
| | $\xrightarrow{c, C, *, e_1}$ | |
| | | $\xrightarrow{c, C, *, e_1, w''}$ |
| | | $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | Unless $e_1$ is valid, abort the session |
| | | $R' \overset{\mathcal{G}_S}{=} \mu(k, x)C$ |
| | | $V \overset{\mathcal{G}_S}{=} \mu(k, x)U$ |
| | | $W \overset{\mathcal{G}_S}{=} w'' G_S + W'$ |
| | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | $r' = a\mu(k, *) + w' + w'' \bmod n_S$ |
| | $\xleftarrow{r', R', V}$ | |
| | Unless $W, r', R', V$ are valid, abort the session. | |
| | $a = \omega(\pi(W) \mid c \mid *)$ | |
| | $r = r' + a\rho \bmod n_S$ | |
| | $R \overset{\mathcal{G}_S}{=} R' + \rho C$ | |
| | $\xleftarrow{r, R}$ | |
| Unless $(W, r)$ is valid, abort the session. | | |
| $K \overset{\mathcal{G}_S}{=} anm \cdot C + R$ | | |

図 6.3: 追跡不能鍵転送プロトコル: Unlink-Key-Transfer

***UA* による検証式**

$$r' G_S \overset{\mathcal{G}_S}{=} \omega(\pi(W) \mid c \mid *)(S - aid \cdot G_S) + W \tag{6.1}$$

$$V \overset{\mathcal{G}_S}{=} x(S - aid \cdot G_S) + zR' \tag{6.2}$$

***SA* による検証式**

$$rG_S \overset{\mathcal{G}_S}{=} \omega(\pi(W) \mid c \mid *)(S - anm \cdot G_S) + W \tag{6.3}$$

### 6.2.1.2 Link-Key-Transfer の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w', q' \in_{\mathcal{R}} [0, n_S)$ |
| | | $W' \overset{G_S}{=} w'G_S$ |
| | $\overset{W', U'}{\longleftarrow}$ | $Q' \overset{G_S}{=} q'G_S$ |
| | $\rho \in_{\mathcal{R}} [0, n_S)$ | |
| | $w'' \in_{\mathcal{R}} [0, n_S)$ | |
| | $W \overset{G_S}{=} w''G_S + W'$ | |
| | $q'' \in_{\mathcal{R}} [0, n_S)$ | |
| | $Q \overset{G_S}{=} q''G_S + Q'$ | |
| | $x, y, z \in_{\mathcal{R}} [0, n_S)$ | |
| | $U \overset{G_S}{=} xG_S + yQ + zC$ | |
| $key = \omega(\pi(\alpha W)\|c)$ | $\overset{anm, W}{\longleftarrow}$ | |
| $e_1 = \omega(key)$ | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | $\overset{c, C, *, e_1}{\longrightarrow}$ | $\overset{c, C, *, e_1, w''}{\longrightarrow}$ |
| | | $\overset{q'', U, \rho}{\longrightarrow}$ |
| | | $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | Unless $e_1$ is valid, abort the session |
| | | $W \overset{G_S}{=} w''G_S + W'$ |
| | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | $r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$ |
| | | $Q \overset{G_S}{=} q''G_S + Q'$ |
| | | $e_P = \pi((\mu(k, *) + \rho)Q) \oplus \rho$ |
| | | $R \overset{G_S}{=} (\mu(k, *) + \rho)C$ |
| | | $V \overset{G_S}{=} (\mu(k, *) + \rho)U$ |
| | | $b = \omega(r \mid e_P \mid \pi(Q))$ |
| | $\overset{r, R, s, e_P, V}{\longleftarrow}$ | $s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$ |
| | Unless $W, r, Q, s, e_P, R, V$ are valid, abort the session. | |
| Unless $(W, r, Q, s, e_P)$ is valid, abort the session. | $\overset{r, R, Q, s, e_P}{\longleftarrow}$ | |
| $K \overset{G_S}{=} anm \cdot C + R$ | | |

図 6.4: 追跡可能鍵転送プロトコル: Link-Key-Transfer

*UA* 及び *SA* による検証式は、以下のように与えられる。

**UA による検証式**

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - anm \cdot G_S) + W \tag{6.4}$$

$$sG_S \overset{G_S}{=} \omega(r \mid e_P \mid \pi(Q))(S - anm \cdot G_S) + Q \tag{6.5}$$

$$V \overset{G_S}{=} x(S - anm \cdot G_S) + y \cdot \pi^{-1}(e_P \oplus \rho) + zR \tag{6.6}$$

**SA による検証式** (6.4) 及び (6.5)

**SP による *aid* の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で *aid* を復号する。

1. 検証式 (6.4) 及び (6.5) を検査する。

2. 下式により $\rho$ 及び $aid$ を計算する。

$$\rho = \pi((\sigma - aaid)Q) \oplus e_P$$

$$aid = anm + \rho \bmod n_S$$

### 6.2.2 安全性の評価

権利認証としての完全性と健全性は、Unlink-Verify 及び Link-Verify から継承される。同様に、*SpA* による *SA* の認証に関する完全性と健全性は、Unlink-Authenticate-SA 及び Link-Authenticate-SA から継承される。

従って、以下では、Unlink-Key-Transfer の追跡不能性と、Link-Key-Transfer による Consensual Disclosure のサポートを証明する。

#### 6.2.2.1 Unlink-Key-Transfer の追跡不能性の証明

**定理 13.** *Unlink-Key-Transfer* は追跡不能である。

**証明** *UA* が検証式 (6.1) 及び (6.2) の検証に成功するケースのみを考えれば十分である。

$R' \overset{\mathcal{G}_S}{=} \mu(k,*)C$ が成り立てば、$R \overset{\mathcal{G}_S}{=} (\mu(k,*)+\rho)C$ が成り立つ。従って、$(anm, R)$ は

$$D_C = \{(anm, R) \mid anm \cdot C + R \overset{\mathcal{G}_S}{=} \sigma C\} \subset [0, n_S) \times \mathcal{G}_S$$

上で一様に分布するので、追跡に結びつく情報を含まない。

$U \overset{\mathcal{G}_S}{=} xG_S + zC$ より式 (6.7) が成り立つが、$(x,z)$ に関して *UA* に与えられる情報は $U$ のみであるため、*UA* は $n_S$ 組の $(x,y)$ を個別に識別することはできない。

$$\text{mult}_{G_S} U = x + z\text{mult}_{G_S} C \bmod n_S. \tag{6.7}$$

一方、検証式 (6.2) が成り立つので、以下が成立する。

$$\text{mult}_{G_S} V \equiv x\mu(k,*) + z\text{mult}_{G_S} R' \bmod n_S \tag{6.8}$$

式 (6.9) が成立しない場合、式 (6.7) と式 (6.8) を同時に満足する $(x,y) \in [0, n_S)^2$ は高々1つしか存在しない。

$$\text{mult}_{G_S} R' \equiv \mu(k,*)\text{mult}_{G_S} C \bmod n_S \tag{6.9}$$

即ち、式 (6.9) が成立しないにも拘わらず、検証式 (6.2) が成立する確率は高々 $\frac{1}{n_S}$ である。

従って、無視しえる確率を除いて、

$$R' \overset{\mathcal{G}_S}{=} \mu(k,*)C$$

が成り立つ。 □

#### 6.2.2.2 Link-Key-Transfer による Consensual Disclosure のサポートの証明

**定理 14.** *Link-Key-Transfer* において *UA* が開示する追跡情報は *aid* のみである。

**証明** 定理 13 の証明の場合と同様に、検証式 (6.4) 乃至 (6.6) を満足する時、

$$R \overset{G_S}{=} (\mu(k,*)+\rho)C$$

が成り立つことを示せば十分である。

*UA* が選択した $(x,y,z)$ は、式 (6.10) を満足する。

$$\text{mult}_{G_S} U = x + y\text{mult}_{G_S} Q + z\text{mult}_{G_S} C \bmod n_S \tag{6.10}$$

一方、検証式 (6.6) が成り立つので、以下が成立する。

$$\text{mult}_{G_S} V \equiv x(\mu(k,*)+\rho) + y\text{mult}_{G_S} \pi^{-1}(e_P \oplus \rho) + z\text{mult}_{G_S} R \bmod n_S \tag{6.11}$$

今、以下に示す式 (6.12) が成立しないとすると、方程式 (6.10) が定める平面と方程式 (6.11) が定める平面との交わりは高々1 次元である。

$$\text{mult}_{G_S} R = (\mu(k,*)+\rho)\text{mult}_{G_S} C \bmod n_S \tag{6.12}$$

即ち、交わりが、*UA* が選択した $(x,y,z)$ を含む確率は、高々 $\frac{1}{n_S}$ であり、無視しえるほど小さい。従って、無視しえる確率を除いて、$R \overset{G_S}{=} (\mu(k,*)+\rho)C$ が成立する。 □

### 6.2.3 計算量の評価

Unlink-Key-Transfer 及び Link-Key-Transfer の計算量を表 6.2 に示す。

表 6.2: Unlink-Key-Transfer 及び Link-Key-Transfer の計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Unlink-Key-Transfer | 4 | 9 | 5 | 18 |
| Link-Key-Transfer | 6 | 13 | 8 | 27 |

(楕円曲線上のスカラー倍演算の実行回数)