平成25年度

学位論文

消費者生成メディアサービスに適した著作権保護基盤技術に関する研究

指導教員

東京理科大学大学院 工学研究科

岩村 惠市 教授

学籍番号 4311701

氏名 稲村 勝樹

東京理科大学大学院

工学研究科電気工学専攻

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究背景

近年，インターネットや携帯電話網といったネットワークの発達により，音楽や動画といったマルチメディアコンテンツのネットワーク流通が盛んに行われるようになってきている．

2000 年代前半までのコンテンツ配信サービスでは，特定のコンテンツ提供者が消費者である一般ユーザに対しコンテンツを提供する商用サービスが主流であった．しかし，近年になって場所や時間の制限に囚われずに，複数の人がコンテンツファイルを作成・使用する場面が増加している．一例として，インターネットなどを活用して一般消費者がコンテンツを生成していく CGM（Consumer Generated Media:消費者生成メディア）という概念が発生し，2005 年に誕生した YouTube [1] などのように一般消費者が作成したコンテンツを流通させる，あるいは閲覧させることを目的とした CGM サービスが急速に広まってきている．

著作物とは「思想又は感情を創作的に表現したものであって，文芸，学術，美術又は音楽の範囲に属するもの」であるという定義があり [2]，著作者は自分の思想や感情を広く世間に訴えるために著作物を公開する．また，著作物は他者にとっての情報や知識となり，それを入手し見聞した新たな人が創作意欲を刺激され，新たな著作物が誕生する．したがって，本来は著作者の思想や感情の表現を破壊しない限り，著作物は自由に流通され，さらに引用・再利用され新たな著作物の要素となりうる物であると考えられる．

これまで，コンテンツに対する著作権保護技術として DRM（Digital Rights Management: デジタル著作権管理）技術があり，様々な方式が提案されてきた [3–5]．これら既存の技術で保護される対象は商用コンテンツであり，コンテンツ提供者がこのコンテンツから得られる（金銭的な意味での）利益を保護するために

視聴制御: 正規ユーザには視聴・閲覧を許可するが，不正ユーザには視聴を許可しない．

コピー制御: 許可された回数を超過したコピー作成を不可能とする．

といった機能を有している．これは，著作物から発生する利益が重要視されたことにより必要とされた技術である．

これに対し，CGM サービスは著作物を広く公開し，多くのユーザに主張を認識してもらうという発想で展開されており，コンテンツの自由な流通や再利用を行ってもらうことで元

のコンテンツの公開範囲を拡大し流通を促進させると同時に，再利用によりユーザによる新たなコンテンツの作成を容易にしている．実際にこのCGMサービスにおいて上記のような著作者・ユーザの行動を推奨すべく,「マッシュアップ」と呼ばれるコンテンツの二次利用，三次利用，... によるコンテンツ作成が行われており，マッシュアップのための表記法を規定したクリエイティブ・コモンズ [6] のような活動や，DRPC（Digital Rights Permission Code: 許諾コード） [7,8] のようなライセンス記述に関する技術規格の制定も始まっている．

本論文における研究目的は，著作物により発生した金銭的利益の保証を重要視した既存のDRM技術に代わり，CGMサービスによってもたらされつつある「『思想又は感情を創作的に表現』のために行われている著作者・ユーザの活動」の概念を基に，保護対象となっていなかった著作者人格権（氏名表示権や同一性保持権など）を保証する新たな著作権保護技術を実現することである．すなわち，コンテンツ流通や再利用による創作活動を許可し，その中において著作者の思想や感情の表現を保証するのに必要と考えられる要素技術を研究するのが目的である．

1.2 既存方式の課題

1.1節において，研究目的を説明した．これは，既存のDRM技術では1.1節で示した目的を達成できないことを意味している．

CGMサービスにおける著作者・ユーザの活動を保証する際に，既存技術ではどのような課題があるかを以下に示す.

コンテンツ引用過程の保証

CGMコンテンツでは様々な人が様々な形態でコンテンツ作成に関与するため，コンテンツの作成過程（CGMコンテンツへの貢献度）により各著作者の思想や感情の表現がどのようにそのコンテンツに影響しているかを見極める必要がある．これは，引用されたコンテンツの種類や順序が変わってしまうと，そのコンテンツにおける主張が変わってしまうことも予想される．したがって，コンテンツが引用されている順番が重要な情報となってくる．

マッシュアップコンテンツに対する著作者の権利主張の証拠，あるいはコンテンツの引用過程の表記・管理を行う方式については，梶らによって提案されている [9,10]．この方式は，W3CによってRDF（Resource Description Framework） [11] の記述方式を拡張し，コンテンツの構成（作成過程など）をメタデータとして詳細に記載できる．この著作権情報が表記されたメタデータの記載内容を基に上記で示したCGMサービスの展開を考えた場合，その記載内容について不正な改ざんや変更がない正当なものであることを保証する仕組みが必須と考えられる．

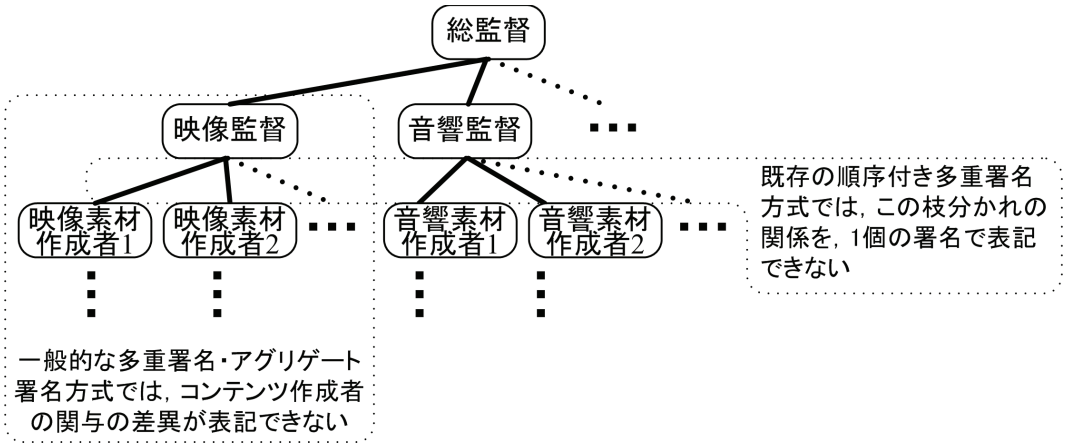

図 1.1: 一般的な多重署名／アグリゲート署名の課題

このデータの内容を保証する手法として電子署名の利用は有効な手段であり，上記の例で示した場面において，複数の署名者による電子署名の作成，およびその署名の検証を効率的に行うことが可能となる多重署名／アグリゲート署名方式 [12, 13] の適用は検討に値する．多重署名方式とは複数の署名者が同一のメッセージに署名を行い，その署名を集約し 1 つの電子署名で表す方式，アグリゲート署名方式とは複数の署名者がそれぞれ異なるメッセージに署名を行い，その署名を集約し 1 つの電子署名で表す方式である．しかし，各署名者間の関係，あるいは数人の署名者がある共通項（例:引用順，職位など）によって 1 つのグループとしてまとめられ，そのようなグループがいくつか存在した時のグループ間の関係の識別について，一般的な多重署名／アグリゲート署名方式では表現が難しいといった課題がある．CGM コンテンツ作成の場合で検討すると，マッシュアップのコンテンツ引用順，あるいは素材提供者や編集者といったグループ毎の区別が必要となることが想定されるが，上記の理由からこれらの署名方式をそのまま適用することは難しい．また，署名者の署名順序を規定できる順序付き多重署名方式 [14, 15] も存在するが，この方式は全署名者の署名順が一列である時にその順序を保証するものである．したがって，全署名者がいくつかのグループに分かれている時（例えば，映像編集，音響編集などのように提供する素材の種類によって編集者（映像監督や音響監督など）が複数いるような時）に，そのグループ間の関係を表現することは不可能である．上記で説明した既存の多重署名／アグリゲート署名方式での課題を図 1.1 に示す．

著作者による編集可否の設定

CGM コンテンツでは，すでに存在しているコンテンツに対し，さらに別なユーザがその既存コンテンツを独自に編集し，新たなコンテンツとして創出・流通させていくことが行われている．この場合，編集を含めコンテンツの作成者などに関する著作者情報が保証される仕組みや，そのコンテンツデータ（またはその一部）に対する編集の許可・不許可を著作者が指定できる仕組みが必要となる．すなわち，重視すべき著作権保護の概念を従来の DRM のような視聴制御やコピー制御ではなく，引用元の保証やユーザによるコンテンツ再利用操作の制御（本論文ではこれを「編集管理」と表現する）とし，コンテンツの著作者の意図に反しない編集を可能にする枠組みが求められている．

この編集管理に関して，編集過程の表記・管理を行う方式の研究 [16] が行われている．一方，編集制御を考慮したセキュリティ技術に関する考察は過去に行われている [17, 18] が，これを実現する具体的な方式の提案はほとんどなく，新たな編集管理技術の研究が必要とされている．ここで，編集制御とはコンテンツの指定された部分に対する変更・削除・追加といった操作を可能にし，指定部分以外は変更・削除・追加を不許可にする制御を指す.

特定値を含まない暗号化方式の実現

コンテンツを不正利用や著作者が許可しない編集をされないように制御するためには，暗号アルゴリズムを用いてコンテンツの機密性を保証することが有効である．

一般的に，コンテンツファイルとして扱われる符号化画像は，そのフォーマット特有の意味を定めた系列をマーカコード（以後，特定符号）としてヘッダ部に設定しており，この特定符号はデータ部に存在することが許されない場合が多い．ここで，符号化画像のデータ部のみを暗号化した場合を考えると，暗号化により特定符号と同じ符号が発生する可能性があり，このコンテンツを利用するプログラムを誤作動させる（すなわち，この特定符号により暗号化されたデータではなくヘッダ部と誤認する）要因となりうる．

この問題について，JPEG（Joint Photographic Experts Group）の後継として規格化された JPEG 2000 を例に具体的に説明する [19, 20]．JPEG 2000 では規格が複数のパートに分かれており，圧縮方式を規定するパートのみが実装必須の規格となっている．一方，圧縮以外の規格（セキュリティや動画対応など）の実装は実装者が必要に応じて決定できる．したがって，例えば PDF などのように簡易なデコーダを無料配布するなどといった場合，一般に普及するデコーダはセキュリティの規格に対応していないことが考えられる．ここで，セキュリティの規格に対応したエンコーダを用いて暗号化した符号化画像を作成・配布し，これをセキュリティ規格を実装していないデコーダで復号する場合を考える．JPEG 2000 においても，マーカ―コードにあたる特定符号はデータ部に存在することが許されていない．した

がって，セキュリティの規格に対応していないデコーダはその記載を暗号化されたデータとは理解せず通常のデータとして処理を行うため，データ部にマーカコードと同じ系列が存在すれば動作異常を起こし，セキュリティホールとなりうる可能性がある．

この問題に対して，いくつかの提案がされている．貴家らのグループがブロック暗号を用いた特定符号発生回避手法 [21] を，岩村らのグループがストリーム暗号を用いた手法 [22] を提案している．しかし，この 2 つの手法は暗号化を行いたいデータ部において非暗号化部分が一部存在するといった課題がある．また，貴家らのグループは別な手法の提案も行っている [23–25] が，それぞれ

- 安全性に問題がある Blowfish [26] に限定されている [23].
- 特定符号が発生したら暗号化鍵を変更しているが，その変更手順が示されていない [24].
- 安全性の評価を行っていない [25].

といった課題が残されている．

1.3 本論文における研究目的

CGM サービスを考慮した著作権保護技術に対し，既存の DRM 技術を適用するだけではその CGM サービスの特徴を活かすことができないことを 1.2 節に示した．

改めて既存の技術における最大の課題をまとめると，一度作成されたコンテンツはそのまま利用することが前提となっており，流通後のコンテンツについて編集を行ったり再利用したりすることを考慮されていないという点にある．そのため，既存の著作権保護の研究では，コンテンツ自体は全体を暗号化して配信すればよく，その復号鍵の配布を行うためにライセンス管理を行う方法に特化して議論されてきた．一方，CGM コンテンツの特徴は二次利用も含めた積極的な再利用にある．その際には部分的に引用を行ったり，既存のコンテンツを編集したりといった行為が行われる．これは既存の著作権保護技術では不正と扱われる行為であり，この CGM サービスにはそのまま適用できないことは自明である．

そこで，本論文ではコンテンツが再利用されてもその著作者の権利を主張できるような新たな著作権保護システムを提案し，新しいコンテンツ流通基盤モデルを構築することを最大の研究目的とする．そのモデルの概略図を図 1.2 に示すと同時に，そのモデルを実現する上で必要となる以下の方式を提案する．

(1) 引用過程を保証できる階層表記型多重署名／アグリゲート署名方式

これまで GDH（Gap-Diffie-Hellman）グループに基づく BLS 署名方式 [27]，およびこの方式に基づく多重署名方式 [28–30] を拡張し，署名者の立場を木構造に配置できる時に，そ

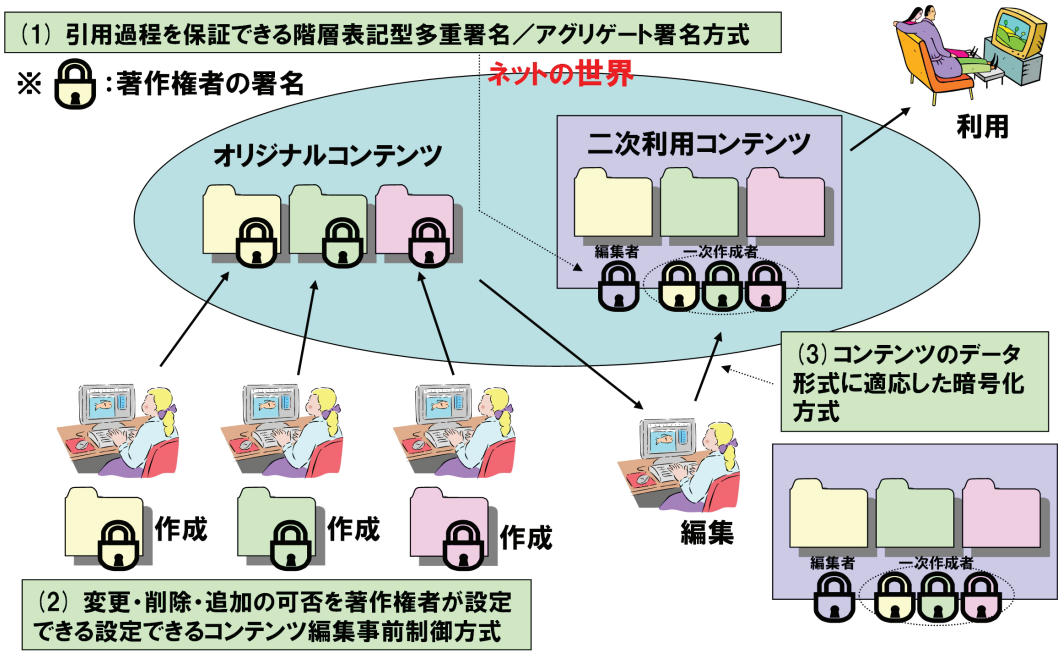

図 1.2: 目的とするコンテンツ流通基盤モデル

の関係も検証可能な木構造表記型多重署名方式が提案されている [31]．また，計算コストを削減する目的でグループ間の区別のみに特化した階層表記型多重署名方式も提案されている [32]．これらの方式では，階層の中間ノード，および最上位に位置するノードが，自分を含むグループ全員の署名者とその配下にいるグループ全員の署名者との関係性を示す中間鍵を生成する．署名検証時は，下位のノードに位置する署名者が作成した中間鍵から最上位のノードが作成した中間鍵まで，その正当性を順次証明し，最後に最上位ノードの中間鍵を用いて多重署名を検証する．これは，ノード間の関係を示す中間鍵の生成が必要となり，署名者が増加するとも中間鍵を含めた署名サイズが増加することを意味している．また，これらの方式では，署名対象となるメッセージが全ての署名者で同一である多重署名方式を拡張し，図 1.1 で示した内，枝分かれの関係を表現できるようにしたものであり，これにより著作者の関係を保証することが可能となる．一方，すでに流通しているコンテンツを引用・編集して新たなコンテンツを作成し，その引用者が追加で署名を行う，あるいは各制作者が内容を追加・変更していきながら共同でコンテンツ制作を行っていくような場面では，各制作者が署名対象とするデータの内容が異なるため，従来の研究では図 1.1 で示した内，コンテンツ作成者の関与の差異については完全に表現できない．したがって，署名者が増加しても署名サイズが増加しない階層表記型多重署名を考案すると同時に，署名対象のメッセージが署名者によって異なることを想定したアグリゲート署名 [13] を拡張した方式が必要となる．

そこで，署名の順番が前後になる 2 人の署名者について，後者が自分とその前者の関係を示す仕組みを基本とし，この関係を示す仕組みを順次合成していくことで署名順序をも検証可能とする順序付き多重署名／アグリゲート署名方式を考案する．さらにこの仕組みを 1 対多に拡張し，多段的に繰り返すことで木構造を含むあらゆる階層表記を実現した多重署名／アグリゲート署名方式を考案する．これにより，コンテンツの引用過程も含めた著作権表記の保証にも適応が可能となる．

(2) 変更・削除・追加の可否を著作者が設定できるコンテンツ編集事前制御方式

コンテンツ編集の事前制御を実現するために，電子署名を応用して変更・削除・追加の可否を著作者が設定できる方式を検討する．一般的な電子署名方式では，コンテンツに対する電子署名作成後にコンテンツ，あるいはその署名に変更が加えられると検証が行えなくなるため，電子署名作成後のコンテンツに対する編集作業を行うことが困難である．これに対し，署名作成後のコンテンツにおいて，その一部に対して秘匿・削除を行っても正真性が確保される墨塗り署名方式 [33–37] が提案されており，これらの方式は編集作業への適用を検討するに値する．しかし，従来の墨塗り署名方式は行政文書などにおける情報秘匿の観点での利用を想定しているため，データの一部変更（墨塗り）や削除のみを目的として提案されている．このため，編集制御と引用元保証の実現に必要な事前の変更・削除・追加に関する制御はほとんど考察されていない．

そこで本論文では，情報秘匿の観点ではなく，編集管理の観点から従来の墨塗り署名の手順を拡張し，編集者に対して部分的な編集を許可しつつ，コンテンツの引用元を保証することが可能な編集可否制御システムを提案する．コンテンツの編集を考慮する場合，データの変更・削除のみならず追加をも制御可能にすることが必要であり，さらに利便性を考慮してデータの変更・削除・追加は事前に制御可能であることが望ましい．ここで，事前制御とは著作者がコンテンツ作成時に変更・削除・追加の各制御を許可する部分をそのコンテンツデータに対する電子署名を作成することで限定し，編集者がそれに違反した処理を行った場合はそのコンテンツ全体の署名と整合が取れなくなるようにすることを意味する．この事前制御を行うことによって，一度の制御で著作者は自分の意図しない編集が行われることを防止し，編集者は著作者の許可範囲内で自分に使い勝手の良い，または嗜好にあったコンテンツへと編集することが可能になる．

(3) コンテンツのデータ形式に適応した暗号化方式

コンテンツのデータ形式に適応した暗号化について，ブロック暗号をベースに特定の方式に依存せず，以下の要件を満たす手法が必要となる．

- 部分暗号化したデータに特定符号を含まないことを保証する．すなわち，暗号化を解除せずに，部分暗号化したままのデータを再生器に入力しても誤作動を起こさないようにする．
- 暗号化する部分を選択でき，画質の制御を可能とする．
- 元々のファイルフォーマットが持つスケーラビリティを保持したまま，部分暗号化を可能とする．
- 少なくとも暗号文攻撃と既知／選択平文攻撃に対して安全性が評価されている．
- 非暗号化部分が従来法に比べて少ない．

この要件を満たすため,「一定の領域を除いた n ビット空間から，同じ領域を除いた n ビット空間への暗号化」を実現している Cycle-walking [38] の手法を拡張し，画像符号化に適した暗号化方式の提案がある [39]．この方式について，復号も含めて実装可能なアルゴリズムの整理を行い，同時にシミュレーション実装による実験結果と理論的検証による実現可能性についての考察を行う．

1.4 本論文の構成

本論文の構成は図 1.3 に示す．また，各章における概要は以下の通りである．

第 1 章 序論

研究背景と目的，および本論文の構成について示す．

第 2 章 基本理論

本研究を進める（あるいは理解する）にあたり，必要となる数論の知識，および基本となるセキュリティ要素について説明する．

第 3 章 引用過程を保証できる階層表記型多重署名／アグリゲート署名方式

BLS 署名に基づく多重署名／アグリゲート署名方式を改良し，署名順序が付けられる順序付き多重署名／アグリゲート署名方式が構成できることを示す．次に，この順序付き多重署名／アグリゲート署名方式の構成を拡張することで木構造表記型多重署名が構成できることを示す．さらにこの提案方式について証明可能安全性を持つことを示す．これにより，コンテンツの引用がどのような過程を経たとしても，全ての著作者をその引用順に合わせて表記できるようになる．

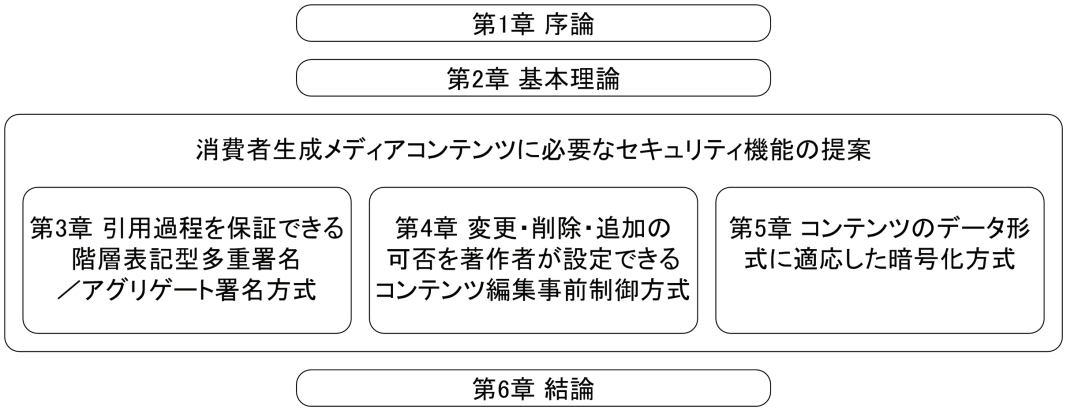

図 1.3: 論文の構成

第 4 章 変更・削除・追加の可否を著作者が設定できるコンテンツ編集事前制御方式

既存の墨塗り署名方式について説明し，これらの方式では変更・削除・追加の可否を設定する際に課題が生じることを示す．その上で，墨塗り署名を用いて変更・削除・追加用の個別署名を発行し，この個別署名で制御を行うコンテンツ編集事前制御方式を提案する．これにより，各制御の可否を事前に設定できる方式が実現できる．

さらに安全性や既存方式との比較に関する考察を行うと同時に，提案方式を n 次編集まで可能にする際に解決されなければならない課題について示す．

第 5 章 コンテンツのデータ形式に適応した暗号化方式

JPEG 2000 の特徴を示し，そこにデータ部に存在してはならないマーカコードについて説明する．その上で，暗号化してもこのマーカコードが出現しない Cycle-walking の拡張方式について説明する．これにより，マーカコードを出現させない暗号化方式が実現できる．

その方式について安全性や処理コストに関する考察を行うと同時に，方式の課題（特に利用モード適用時に生じる課題）について示す．

第 6 章 結論

本研究の主たる成果をまとめると同時に，今後の課題について示し，本論文全体の結論とする．

第2章　基本理論

2.1　数論

2.1.1　楕円曲線

楕円曲線とは $y^2 = x^3 + ax + b$ で表される曲線のことである．ここで，この楕円曲線上にある有限体 $\mathbb{F}_q$ の元の座標 (x, y) の集合に無限遠点 O を追加した集合を $\mathbb{E}(\mathbb{F}_q)$ とする．これを式で表すと以下の様になる．

$$\mathbb{E}(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q^2 : y^2 = x^3 + ax + b\} \cup \{O\} \tag{2.1}$$

この楕円曲線において，$\mathbb{E}(\mathbb{F}_q)$ の元による加算が定義される [40–42]．楕円曲線における加算，零元，逆元の定義を図 2.1 に示すと同時に，これらについての説明を以下に示す．

加算

楕円曲線上に異なる点 $P(x_1, y_1)$ と点 $Q(x_2, y_2)$（ただし，$x_1 \neq x_2$）が存在する時，この 2 点を通る直線の方程式は

$$y - y_1 = \frac{y_1 - y_2}{x_1 - x_2}(x - x_1) \tag{2.2}$$

であり，この直線と楕円曲線との交点（の x 座標）は 3 次方程式

$$\left(\frac{y_1 - y_2}{x_1 - x_2}(x - x1) + y_1\right)^2 = x^3 + ax + b \tag{2.3}$$

の解として表される．この 3 次方程式が 3 つの根 x_1, x_2, x_3 を持つ時，

$$(x - x_1)(x - x_2)(x - x_3) = 0 \tag{2.4}$$

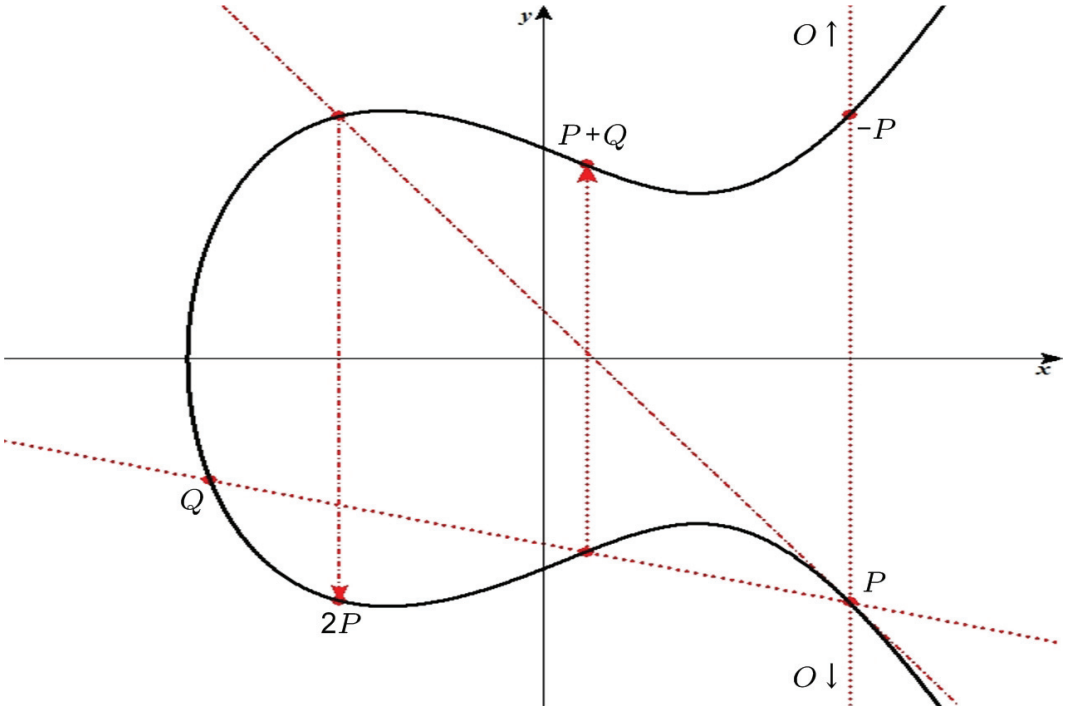

図 2.1: 楕円曲線における加算，零元，逆元の定義

と表されることから，$x_1 + x_2 + x_3$ の値は 3 次方程式の 2 次の係数と一致する．これを上記の直線と楕円曲線との交点を求める方程式と比較することにより，点 P, Q と異なる交点の座標

$$x_3 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)^2 - x_1 - x_2 \tag{2.5}$$

$$y_3 = y_1 + \frac{y_1 - y_2}{x_1 - x_2}(x_3 - x_1) \tag{2.6}$$

が得られる．この交点と x 軸対称となる位置にある楕円曲線上の点が $P + Q$ と定義され，$(x_3, -y_3)$ となる．

同一の点における加算（すなわち 2 倍算）については，2 点が極限的に一致したとみなし，その点における接線を求めてから異なる交点を求める作業を行う．例えば，点 $P(x_1, y_1)$ の 2 倍である $2P$ は以下の様に求める:
楕円曲線上の点 P における接線の方程式は

$$\begin{aligned} y - y_1 &= f'(x_1)(x - x_1) \\ &= \frac{3x_1^2 + a}{2y_1}(x - x_1) \end{aligned} \tag{2.7}$$

であり，この直線と楕円曲線との交点は（の x 座標）は 3 次方程式

$$\left(\frac{3x_1^2+a}{2y_1}(x-x_1)+y_1\right)^2 = x_3+ax+b \tag{2.8}$$

の解として表される．

この 3 次方程式が 3 つの根 x_1, x_2, x_3 を持つ時，この 3 次方程式が根 $x1, x3'$（ただし，x_1 は重根）を持つ時，

$$(x-x_1)^2(x-x_3') = 0 \tag{2.9}$$

と表されることから，$2x_1+x_3'$ の値は 3 次方程式の 2 次の係数と一致する．これを上記の直線と楕円曲線との交点を求める方程式と比較することにより，点 P と異なる交点の座標

$$x_3' = \left(\frac{3x_1^2+a}{2y_1}\right)^2 - 2x_1 \tag{2.10}$$

$$y_3' = y_1 + \frac{3x_1^2+a}{2y_1}(x_3'-x_1) \tag{2.11}$$

が得られる．この交点と x 軸対称となる位置にある楕円曲線上の点が $2P$ と定義され，$(x_3', -y_3')$ となる．

零元

零元とは，加法における単位元を意味する．したがって，楕円曲線上の零元 O を用いると

$$P+O=P \tag{2.12}$$

が成立する．

しかし，楕円曲線における加算の定義により，点 P における零元は，点 P とこの点 P に x 軸対称となる点とを結ぶ直線上にあるはずだが，この直線は点 P を通り y 軸と平行になる直線となり，この 2 点以外に楕円曲線との交点は存在しない．

そこで，便宜上この直線上にあるとみなした無限遠点を零元 O とする．

逆元

逆元とは，演算結果が零元となる元を意味し，これをマイナスの符号を付けて表記する（例えば，点 P における逆元は $-P$ とする）．

楕円曲線における加算と零元の定義より，点 P における逆元 $-P$ は，点 P を通り y 軸と平行になる直線上に存在する点 P 以外の楕円曲線との交点となる．したがって，点 P の逆元 $-P$ は，点 P と x 軸対称となる点となる．

2.1.2 ペアリング

ペアリングとは，後述する双線形性を持つ2入力1出力の関数のことである．ここでは，一般的な定義に基づくペアリング [43] と，具体的に演算が可能な楕円曲線上のペアリング [41] について説明する．

一般的な定義に基づくペアリング

$\mathbb{G}_1, \mathbb{G}_2$ をペアリングの演算が可能な位数 p の異なる巡回群とし，$\mathbb{G}_\tau$ を上記二つとは異なる位数 p のある有限体とする．さらに $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$ は $\mathbb{G}_1, \mathbb{G}_2$ の入力から $\mathbb{G}_\tau$ への写像を行う関数を表す．この関数において以下の特徴を持つ時，この関数をペアリングと定義する．

非縮退性 G_τ における単位元を i_τ，さらに $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ とする時，以下が成立する．

- 任意の P に対し，$e(P, Q) = i_\tau$ が成立するなら，Q は零元である．
- 任意の Q に対し，$e(P, Q) = i_\tau$ が成立するなら，P は零元である．

双線形性 以下の式が成立する．

- $P_1, P_2 \in \mathbb{G}_1, Q \in \mathbb{G}_2$ に対し，

$$e(P_1 P_2, Q) = e(P_1, Q) \cdot e(P_2, Q) \tag{2.13}$$

- $P \in \mathbb{G}_1, Q_1, Q_2 \in \mathbb{G}_2$ に対し，

$$e(P, Q_1 Q_2) = e(P, Q_1) \cdot e(P, Q_2) \tag{2.14}$$

- $a, b \in \mathbb{Z}_p^*$，および $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ に対し，

$$\begin{aligned} e(P^a, Q^b) = e(P^b, Q^a) = e(P^{ab}, Q) = e(P, Q^{ab}) \\ = e(P, Q)^{ab} \end{aligned} \tag{2.15}$$

楕円曲線上のペアリング

$\mathbb{G}_1, \mathbb{G}_2$ をペアリングの演算が可能な位数 p の異なる楕円曲線上の加法巡回群とし，$\mathbb{G}_\tau$ を上記二つとは異なる位数 p のある有限体とする．さらに $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$ は $\mathbb{G}_1, \mathbb{G}_2$ の入力から $\mathbb{G}_\tau$ への写像を行う関数を表す．この関数において以下の特徴を持つ時，この関数をペアリングと定義する．

非縮退性 G_τ における単位元を i_τ，さらに $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ とする時，以下が成立する．

- 任意の P に対し，$e(P, Q) = i_\tau$ が成立するなら，Q は零元である．
- 任意の Q に対し，$e(P, Q) = i_\tau$ が成立するなら，P は零元である．

双線形性 以下の式が成立する．

- $P_1, P_2 \in \mathbb{G}_1, Q \in \mathbb{G}_2$ に対し，

$$e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q) \tag{2.16}$$

- $P \in \mathbb{G}_1, Q_1, Q_2 \in \mathbb{G}_2$ に対し，

$$e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2) \tag{2.17}$$

- $a, b \in \mathbb{Z}_p^*$，および $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ に対し，

$$\begin{aligned} e(aP, bQ) = e(bP, aQ) = e(abP, Q) = e(P, abQ) \\ = e(P, Q)^{ab} \end{aligned} \tag{2.18}$$

このような性質を持つ関数として代表的なものに Tate ペアリングがある [44].

2.1.3 離散対数問題と DH 問題

Diffie と Hellman によって公開鍵暗号の概念が提案された時，離散対数問題に基づく DH（Diffie-Hellman）鍵配送が同時に提案されている [45]．ここで p を素数の冪乗で表される整数とすると，既約剰余類群 $(\mathbb{Z}/p\mathbb{Z})^\times$ は巡回群となる．位数 $\phi(p)$ の原始元 $g \in (\mathbb{Z}/p\mathbb{Z})^\times$ を生成元とすると，任意の $y \in (\mathbb{Z}/p\mathbb{Z})^\times$ は $y \equiv g^x$（ただし，$x \in \mathbb{Z}_p^*$）で一意に表され，この x（これを y の g に対する離散対数と呼ぶ）を求める問題を離散対数問題と呼ぶ [46].

この問題を解くアルゴリズムとしては Shanks の Baby-Step-Giant-Step アルゴリズム [47], Pollard の ρ 法 [48]，数体ふるい法 [49]，関数体ふるい法 [50] などがあるが，いずれも準指数

時間オーダのアルゴリズムであり，多項式時間で解くことができるアルゴリズムは見つかっていない．

ところで，DH 鍵配送は離散対数問題に基づいた鍵配送プロトコルであるが，厳密には離散対数問題を解かなくても共有する鍵を計算できる可能性が残されている．これを整理したものが DH 問題である．この DH 問題には CDH（Computational Diffie-Hellman）問題，および DDH（Decisional Diffie-Hellman）問題があり，以下の通りとなる．

CDH 問題 $a, b \in \mathbb{Z}_p^*$，および $g \in (\mathbb{Z}/p\mathbb{Z})^\times$ があり，(g, g^a, g^b) の組だけが既知の時，g^{ab} を求める問題．

DDH 問題 $a, b, c \in \mathbb{Z}_p^*$，および $g \in (\mathbb{Z}/p\mathbb{Z})^\times$ があり，(g, g^a, g^b, g^c) の組だけが既知の時，$c = ab$ であるかを判定する問題．

ここで，離散対数問題と CDH 問題を比較すると，離散対数問題が解ければ a, b の値は求まるので，g^{ab} を求めることは容易であることが分かる．したがって，離散対数問題は CDH 問題と同等以上に難しいと言える．また，CDH 問題と DDH 問題を比較すると，CDH 問題が解ければ g^{ab} の値は求まるので，この値と g^c の値を比較すれば容易に $c = ab$ であるかを判定できる．したがって，CDH 問題は DDH 問題と同等以上に難しいと言える．

これらの問題をさらに拡張し，co-CDH（Computational co-Diffie-Hellman）問題，および co-DDH（Decisional co-Diffie-Hellman）問題が定義される．$\mathbb{G}_1', \mathbb{G}_2'$ を位数 p の異なる巡回群とした時，それぞれの問題は以下の通りとなる．

co-CDH 問題 $a \in \mathbb{Z}_p^*$，および $g_1 \in \mathbb{G}_1', g_2 \in \mathbb{G}_2'$ があり，(g_1, g_2, g_1^a) の組だけが既知の時，g_2^a を求める問題．

co-DDH 問題 $a, b \in \mathbb{Z}_p^*$，および $g_1 \in \mathbb{G}_1', g_2 \in \mathbb{G}_2'$ があり，(g_1, g_2, g_1^a, g_2^b) の組だけが既知の時，$a = b$ であるかを判定する問題．

ここで，離散対数問題と co-CDH 問題を比較すると，離散対数問題が解ければ a の値は求まるので，g_2^a を求めることは容易であることが分かる．したがって，離散対数問題は co-CDH 問題と同等以上に難しいと言える．また，co-CDH 問題と co-DDH 問題を比較すると，co-CDH 問題が解ければ g_2^a の値は求まるので，この値と g_2^b の値を比較すれば容易に $a = b$ であるかを判定できる．したがって，co-CDH 問題は co-DDH 問題と同等以上に難しいと言える．

これらの問題を楕円曲線上に拡張して考えることができる．$g \in \mathbb{E}(\mathbb{F}_q)$ を生成元とすると，任意の $y \in \mathbb{E}(\mathbb{F}_q)$ は $y \equiv xg$（ただし，$x \in \mathbb{Z}_p^*$）で一意に表され，この x を求める問題を楕円離散対数問題と呼ぶ [46]．

次に，楕円曲線上の co-CDH 問題，および co-DDH 問題を定義する [27,43]．$\mathbb{G}_1'', \mathbb{G}_2''$ を位数 p の異なる楕円曲線上の加法巡回群とした時，それぞれの問題は以下の通りとなる．

楕円曲線上の co-CDH 問題 $a \in \mathbb{Z}_p^*$，および $g_1 \in \mathbb{G}_1'', g_2 \in \mathbb{G}_2''$ があり，(g_1, g_2, ag_1) の組だけが既知の時，ag_2 を求める問題．

楕円曲線上の co-DDH 問題 $a, b \in \mathbb{Z}_p^*$，および $g_1 \in \mathbb{G}_1'', g_2 \in \mathbb{G}_2''$ があり，(g_1, g_2, ag_1, bg_2) の組だけが既知の時，$a = b$ であるかを判定する問題．

ここで，楕円離散対数問題と楕円曲線上の co-CDH 問題を比較すると，離散対数問題が解ければ a の値は求まるので，ag_2 を求めることは容易であることが分かる．したがって，楕円離散対数問題は楕円曲線上の co-CDH 問題と同等以上に難しいと言える．また，楕円曲線上の co-CDH 問題と楕円曲線上の co-DDH 問題を比較すると，co-CDH 問題が解ければ ag_2 の値は求まるので，この値と bg_2 の値を比較すれば容易に $a = b$ であるかを判定できる．したがって，楕円曲線上の co-CDH 問題は楕円曲線上の co-DDH 問題と同等以上に難しいと言える．

2.2 共通鍵暗号

2.2.1 共通鍵暗号の基本概念

共通鍵暗号は，電子的に通信されるデータに対し，そのデータを秘匿し，内容が傍受されることを防止する目的として用いられる．共通鍵暗号は暗号化，復号の 2 つのアルゴリズムで構成される．そのアルゴリズムの概要を以下に示す [51].

暗号化 暗号化するメッセージ $m \in \{0,1\}^*$，および共通鍵 $K \in \{0,1\}^*$ を入力し，暗号文 c を出力する．

復号 暗号化アルゴリズムにより出力された暗号文 c，その暗号化の際に使用した共通鍵 K を入力し，元のメッセージ m を出力する．

上記で示した通り，暗号化と復号に同一の（共通の）鍵を用いることが特徴である．したがって，古典的な暗号方式はほぼ全てが共通鍵暗号方式と見なすこともできる．

この共通鍵暗号に対し，以下の要件を満たす必要がある．

秘匿性 (Confidentiality) 入手可能な情報から元のメッセージ求めることが困難であること．

頑強性 (Robustness) 暗号文を操作し，それに対応する平文に意図的な変更を行うことが困難であること [1].

また，暗号化や復号の処理の方法により，次の 2 種類に大別される．

[1]ただし，場合によっては頑強性をあえて必要としない方式もある．これは公開鍵暗号方式により多く見られる方式で，準同型暗号とも呼ばれる [52].

ブロック暗号 固定長のデータ（これをブロックと呼ぶ）を単位として処理を行う暗号．2013年現在は，ブロックサイズが 64 ビット，あるいは 128 ビット，鍵長は 128 ビット，192 ビット，256 ビットのいずれかの組み合わせで用いられるのが一般的である．また，ブロックサイズより大きいサイズのデータはブロック暗号アルゴリズムを繰り返し実行して暗号化を行うが，この繰り返しの方法には数種類あり，利用モードと呼ばれる（2.2.2 節参照）．代表的なものに AES（Advanced Encryption Standard） [53]，MISTY [54]，Camellia [55] がある．

ストリーム暗号 メッセージをビット単位あるいはバイト単位などで逐次暗号化する暗号である．一般的には鍵ストリーム生成部と呼ばれる一種の疑似乱数生成器により鍵ストリームを出力し，この鍵ストリームと平文の排他的論理和により暗号文が作成される．代表的なものに RC4 [56]，KCipher-2 [57] がある．

2.2.2 利用モード

ブロック暗号アルゴリズムは固定長のデータサイズを暗号化する．したがって，メッセージのデータサイズがこのブロックのサイズよりも大きい場合はこのアルゴリズムを繰り返し実行し暗号化を行う．単純な方法としてはメッセージをブロックサイズごとに分割し，順番に暗号化していく方法がある．しかし，この方法では分割したブロックの中に同じデータが存在したら，それを同じ共通鍵で暗号化すれば必ず同じ暗号文のブロックとして出力される．これにより，何らかの特定なメッセージが平文の中に繰り返し出現していることが判明し，攻撃者にヒントを与えることになる．

そこで，メッセージと共通鍵の他にもう一つ別な値を入力値とし，暗号化の処理を工夫することで上記の課題を解決する方法が提案されている．これをブロック暗号の利用モードという [51] [2]．

代表的な利用モードについて図 2.2 に示すと同時に，これらについての説明を以下に示す．

CBC（Cipher Block Chaining）モード i 番目のメッセージブロック m_i に対し，$i-1$ 番目の暗号化ブロック c_{i-1}（ただし，$i=1$ の時は IV（Initialization Vector:初期化ベクトル））との排他的論理和を計算し，その値を暗号化アルゴリズムに入力する．出力値を i 番目の暗号化ブロック c_i とする．

OFB（Output FeedBack）モード i 番目のメッセージブロック m_i に対し，IV を i 回繰り返し暗号化アルゴリズムに入力し得られた出力値との排他的論理和を計算する．その計算結果を i 番目の暗号化ブロック c_i とする．

[2] 単純にブロックを順番で暗号化する方法も一つの利用モードと考えられており，それを ECB（Electronic CodeBook）モードと呼ぶ．

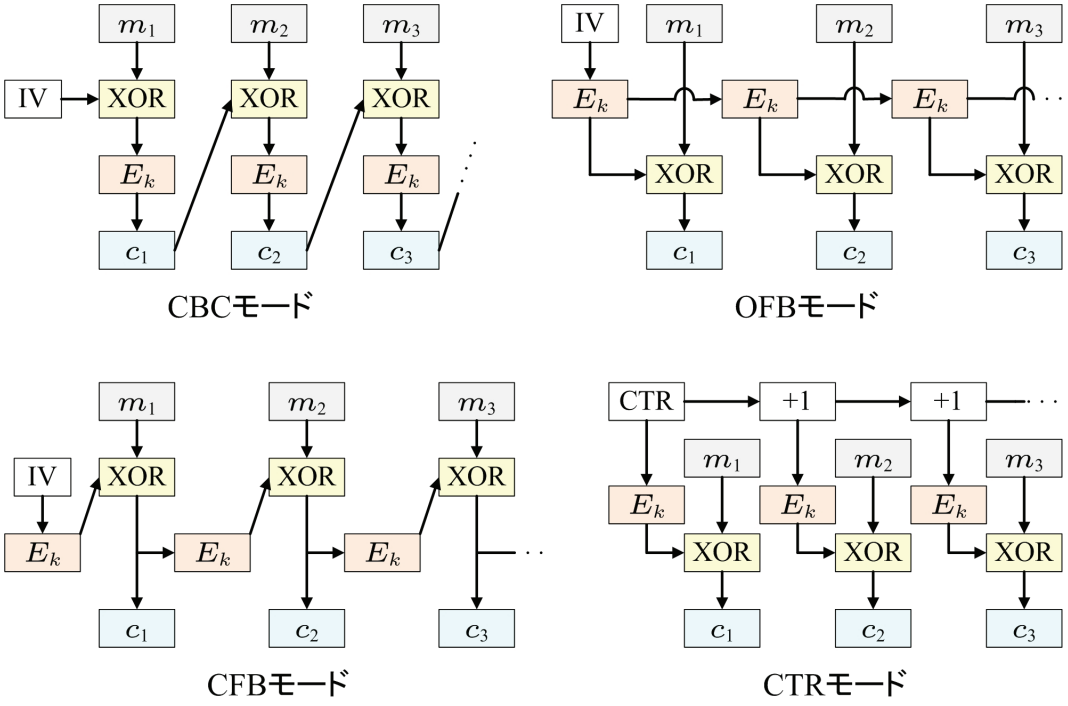

図 2.2: 代表的なブロック暗号の利用モード

CFB（Cipher FeedBack）モード i 番目のメッセージブロック m_i に対し，$i-1$ 番目の暗号化ブロック c_{i-1}（ただし，$i=1$ の時は IV）を暗号化アルゴリズムに入力し得られた出力値との排他的論理和を計算する．その計算結果を i 番目の暗号化ブロック c_i とする．

CTR（CounTeR）モード i 番目のメッセージブロック m_i に対し，カウンタと呼ばれる値（CTR）に $i-1$ 足した値を暗号化アルゴリズムに入力し得られた出力値との排他的論理和を計算する．その計算結果を i 番目の暗号化ブロック c_i とする．

2.2.3 Cycle-walking

Cycle-walking は，ブロック暗号においてブロック長以下のサイズの平文を暗号化する際に，暗号文がブロック長のサイズではなく平文と同サイズ以下となるようにするための手法である [38]．ここで $\mathbb{M}$ をブロック暗号のブロック長以下のサイズを持つ数値の集合とし，その $\mathbb{M}$ の要素の範囲をある整数 k を用いて $0 \leq M \leq k-1$ とする．さらに N を k 以下でもっとも最大となる 2 の冪乗数と定義し，n をその N のビット長（$n = \log_2 N$）とする．

ここで $E_K(\cdot)$ を一般的な共通鍵 K によるブロック暗号アルゴリズム，$m \in \mathbb{M}$ を暗号化を行いたい平文とした時，Cycle-walking による n ビット長の暗号化アルゴリズム $Cy_K(\cdot)$ を

Algorithm 1 に示す.

Algorithm 1 $Cy_K(m)$:Cycle-walking encryption procedure

```
a ← E_K(m)
if a ∈ M then
   return a
else
   return Cy_K(a)
end if
```

また，$E_K^{-1}(\cdot)$ を $E_K(\cdot)$ に対応する復号アルゴリズム，$c \in \mathbb{M}$ を復号を行いたい暗号文とした時，Cycle-walking による n ビット長の復号アルゴリズム $Cy_K^{-1}(\cdot)$ を Algorithm 2 に示す.

Algorithm 2 $Cy_K^{-1}(c)$:Cycle-walking decryption procedure

```
a ← E_K^{-1}(c)
if a ∈ M then
   return a
else
   return Cy_K^{-1}(a)
end if
```

この方式において，m の暗号化は集合 $\mathbb{M}$ へのランダムな置換と区別がつかず，安全であることが証明されている.

2.3 電子署名

2.3.1 電子署名の基本概念

電子署名は，電子的に通信されるデータに対し，そのデータの発信元の保証や偽造・改竄の防止を目的として用いられ，紙文書における印鑑やサイン（署名）に相当する役割を果たす．電子署名方式は Diffie と Hellman によって概念が提案され [45]，代表的なものには Rivest，Shamir，Adleman によって初めて実現された RSA 署名 [58] や，El Gamal によって提案された El Gamal 署名 [59]，およびその改良型である DSA（Digital Signature Algorithm）[60] がある.

一般的な電子署名方式は鍵生成，署名作成，署名検証の 3 つのアルゴリズムで構成される．そのアルゴリズムの概要を以下に示す [46, 51].

鍵生成 セキュリティパラメータ k を設定し，ユーザごとに異なる値（通常は乱数）を入力し，それにしたがい署名鍵と検証鍵の鍵ペア (x, v) を出力する．セキュリティパラメー

タとは鍵長など電子署名の安全性の強度を決定するものであり，乱数などの値によりユーザごとに異なる鍵ペアが生成される．また，この時に署名対象となるメッセージの空間 $\mathbb{M}$ が決定される．

署名作成 署名対象となるメッセージ $m \in \mathbb{M}$，および署名者の署名鍵 x を入力し，電子署名 σ を出力する．

署名検証 署名対象となるメッセージ $m \in \mathbb{M}$，その電子署名 σ，および署名者の検証鍵 v を入力し，署名がメッセージに対応しているものかどうかを判定した結果，True または False を出力する．

なお，通常検証鍵を公開する際には，信頼できる第三者機関を介して公開することが望ましいとされ，そのための仕組みとして PKI（Public Key Infrastructure:公開鍵基盤）がある [61]．

この電子署名に対し，以下の要件を満たす必要がある．

正当性 (Correctness) 正当な署名者が作った署名は，その署名者の検証鍵による検証結果が True になること．

安全性 (Security) 検証を通過するのは正当な署名者が作った署名文に限り，それ以外のひとが作ったいかなる偽造署名もその署名者の検証鍵による検証結果が False となること．

この要件を厳密に満たすため，電子署名アルゴリズムに対し適応的選択文書攻撃に対する存在的偽造不能性が証明されることが要求される．

2.3.2 GDH グループに基づく電子署名

2.1.3 節で，DH 問題に関して難易度の異なる問題が定義されることを示した．ここで，もし CDH 問題は難しい問題である一方，DDH 問題は簡単な問題であるという条件が満たされる位数 p の有限体 $\mathbb{G}_{\mathrm{GDH}}$ が存在する場合，この $\mathbb{G}_{\mathrm{GDH}}$ は GDH グループと定義される [62]．

この定義された GDH グループに基づく様々な研究が行われ [63, 64]，セキュリティ技術への応用の提案も行われた [43, 65]．さらに以下に示す電子署名方式が実現できることも示されている [62]．

鍵生成 $g \in \mathbb{G}_{\mathrm{GDH}}$ を生成元とする．$x \in \mathbb{Z}_p^*$ を選び，$v = g^x$ を計算する．x を署名鍵，v を検証鍵とする．

署名作成 一方向性ハッシュ関数 $H : \{0,1\}^* \to \mathbb{G}_{\mathrm{GDH}}$ を定義する．$m \in \{0,1\}^*$ を署名対象となる平文とした時，署名者は $h = H(m)$ を計算し，$\sigma = h^x$ を生成する．σ を m に対する電子署名とする．

署名検証 検証者に g, v, m, σ が与えられた時，その検証者は $h = H(m)$ を計算し，DDH 問題を解くアルゴリズムを利用して以下の判定を行う．

$$\log_g v = \log_h \sigma \bmod p \tag{2.19}$$

この時，電子署名が正しく作成されているならば，検証式の左辺は

$$\log_g v = \log_g g^x = x \tag{2.20}$$

となり，右辺は

$$\log_h \sigma = \log_h h^x = x \tag{2.21}$$

となり，同じ値となる．

2.3.3 BLS 署名方式

Boneh，Lynn，Shacham により， 2.1.2 節で示したペアリングを利用することで 2.3.2 節で示した署名方式が実現できることが示され，BLS 署名方式の提案が行われた [27]．BLS 署名方式では， 2.1.3 節で示した co-DDH 問題がペアリングにより簡単な問題になることを利用する． 2.1.2 節で定義された記号を基に，この署名方式は以下の通り構成される．

鍵生成 $g \in \mathbb{G}_1$ を生成元とする．$x \in \mathbb{Z}_p^*$ を選び，$v = g^x$ を計算する．x を署名鍵，v を検証鍵とする．

署名作成 一方向性ハッシュ関数 $H : \{0,1\}^* \to \mathbb{G}_2$ を定義する．$m \in \{0,1\}^*$ を署名対象となる平文とした時，署名者は $h = H(m)$ を計算し，$\sigma = h^x$ を生成する．σ を m に対する電子署名とする．

署名検証: 検証者に g, v, m, σ が与えられた時，その検証者は $h = H(m)$ を計算し，ペアリングを用いて以下の判定を行う．

$$e(g, \sigma) = e(v, h) \tag{2.22}$$

上記で定義されている一方向性ハッシュ関数としては *MapToGroup* がある [27, 43]．

この時，電子署名が正しく作成されているならば，署名検証における判定式の左辺は

$$e(g, \sigma) = e(g, h^x) = e(g, h)^x \tag{2.23}$$

となり，右辺は

$$e(v,h) = e(g^x,h) = e(g,h)^x \tag{2.24}$$

となり，同じ値になる．さらにこの方式について存在的偽造不可能性が証明されている．

実際には，この BLS 署名方式は楕円曲線上の演算において実現される． 2.1.2 節で定義された記号を基に，楕円曲線上の BLS 署名方式は以下の通り構成される．

鍵生成 $g \in \mathbb{G}_1$ を生成元とする．$x \in \mathbb{Z}_p^*$ を選び，$v = xg$ を計算する．x を署名鍵，v を検証鍵とする．

署名作成 一方向性ハッシュ関数 $H : \{0,1\}^* \to \mathbb{G}_2$ を定義する．$m \in \{0,1\}^*$ を署名対象となる平文とした時，署名者は $h = H(m)$ を計算し，$\sigma = xh$ を生成する．σ を m に対する電子署名とする．

署名検証: 検証者に g, v, m, σ が与えられた時，その検証者は $h = H(m)$ を計算し，ペアリングを用いて以下の判定を行う．

$$e(g,\sigma) = e(v,h) \tag{2.25}$$

この時，電子署名が正しく作成されているならば，署名検証における判定式の左辺は

$$e(g,\sigma) = e(g,xh) = e(g,h)^x \tag{2.26}$$

となり，右辺は

$$e(v,h) = e(xg,h) = e(g,h)^x \tag{2.27}$$

となり，同じ値になる．

この BLS 署名を基に，さまざまな拡張署名方式が提案されている [13, 28–30, 66–74].

2.3.4 多重署名方式

2.3.3 節で示した BLS 署名を基に，署名対象となるメッセージが各署名者で共通な電子署名を集約し，署名サイズが署名者数に依存せずに一定値以下とすることを可能とした多重署名方式が提案されている [28, 29].

新たに $\mathbb{U} = \{u_1, \cdots, u_n\}$ を署名作成が可能な署名者のグループとして定義し，さらに $\mathbb{L} = \{u_{i_1}, \cdots, u_{i_l}\} \subseteq \mathbb{U}$ を実際に多重署名作成に参加した署名者のグループと定義する．さら

に $\mathbb{J} = \{i_1, \cdots, i_l\}$ を，この多重署名へ参加した署名者を示す符号（u_k の k に相当）の全員分の集合とする．この時，多重署名は以下の通りに構成される．

鍵生成 $g \in \mathbb{G}_1$ を生成元とする．署名者 $u_i \in \mathbb{U}$ について $x_i \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_i = g^{x_i}$（楕円曲線上なら $v_i = x_i g$）を計算する．x_i を署名者 u_i の署名鍵，v_i を署名者 u_i の検証鍵とする．

署名作成 一方向性ハッシュ関数 $H : \{0,1\}^* \to \mathbb{G}_2$ を定義する．$m \in \{0,1\}^*$ を多重署名作成に参加する署名者 $u_{i_\alpha} \in \mathbb{L}$ の署名対象となるメッセージとした時，署名者 $u_{i_\alpha} \in \mathbb{L}$ は $h = H(m)$ を計算し，

$$\sigma_{i_\alpha} = h^{x_{i_\alpha}} \tag{2.28}$$

を生成する．
（楕円曲線上なら

$$\sigma_{i_\alpha} = x_{i_\alpha} h \tag{2.29}$$

を生成する．）
σ_{i_α} を m に対する署名者 u_{i_α} の電子署名とする．

多重化 多重署名作成に参加する全ての署名者の電子署名 σ_{i_α} を集め，

$$\sigma = \prod_{j \in \mathbb{J}} \sigma_j \tag{2.30}$$

を計算する．
（楕円曲線上なら

$$\sigma = \sum_{j \in \mathbb{J}} \sigma_j \tag{2.31}$$

を計算する．）
$(m, \mathbb{L}, \sigma)$ をメッセージと多重署名の組とする．

署名検証 検証者に $(m, \mathbb{L}, \sigma)$, および g が与えられ，さらに $\mathbb{L}$ から検証に必要となる全ての検証鍵 v_j（$j \in \mathbb{J}$）から

$$v = \prod_{j \in \mathbb{J}} v_j \tag{2.32}$$

を計算する．

（楕円曲線上なら

$$v = \sum_{j \in \mathbb{J}} v_j \tag{2.33}$$

を計算する.）

さらに検証者は m から $h = H(m)$ を計算し，ペアリングを用いて以下の判定を行う.

$$e(g, \sigma) = e(v, h) \tag{2.34}$$

この時，電子署名が正しく作成されているのであれば，署名検証における判定式の左辺は

$$e(g, \sigma) = e(g, \prod_{j \in \mathbb{J}} \sigma_j) = e(g, h^{\sum_{j \in \mathbb{J}} x_j}) = e(g, h)^{\sum_{j \in \mathbb{J}} x_j} \tag{2.35}$$

あるいは楕円曲線上なら

$$e(g, \sigma) = e(g, \sum_{j \in \mathbb{J}} \sigma_j) = e(g, \sum_{j \in \mathbb{J}} x_j h) = e(g, h)^{\sum_{j \in \mathbb{J}} x_j} \tag{2.36}$$

となり，右辺は

$$e(v, h) = e(\prod_{j \in \mathbb{J}} v_j, h) = e(g^{\sum_{j \in \mathbb{J}} x_j}, h) = e(g, h)^{\sum_{j \in \mathbb{J}} x_j} \tag{2.37}$$

あるいは楕円曲線上なら

$$e(v, h) = e(\sum_{j \in \mathbb{J}} v_j, h) = e(\sum_{j \in \mathbb{J}} x_j g, h) = e(g, h)^{\sum_{j \in \mathbb{J}} x_j} \tag{2.38}$$

となり，同じ値になる．さらにこの方式について存在的偽造不能性が証明されている.

2.3.5 アグリゲート署名方式

2.3.3 節で示した BLS 署名を基に，署名対象となるメッセージが各署名者で全て異なっている電子署名を集約し，署名サイズが署名者数に依存せずに一定値以下とすることを可能としたアグリゲート署名方式が提案されている [13, 27].

2.3.4 節と同様に $\mathbb{U} = \{u_1, \cdots, u_n\}$ を署名作成が可能な署名者のグループとして定義し，さらに $\mathbb{L} = \{u_{i_1}, \cdots, u_{i_l}\} \subseteq \mathbb{U}$ を実際にアグリゲート署名作成に参加した署名者のグループと定義する．さらに $\mathbb{J} = \{i_1, \cdots, i_l\}$ を，このアグリゲート署名へ参加した署名者を示す符号（u_k の k に相当）の全員分の集合とする．この時，アグリゲート署名は以下の通りに構成される.

鍵生成: $g \in \mathbb{G}_1$ を生成元とする．署名者 $u_i \in \mathbb{U}$ について $x_i \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_i = g^{x_i}$（楕円曲線上なら $v_i = x_i g$）を計算する．x_i を署名者 u_i の署名鍵，v_i を署名者 u_i の検証鍵とする．

署名作成: 一方向性ハッシュ関数 $H : \{0,1\}^* \to \mathbb{G}_2$ を定義する．m_{i_α} をアグリゲート署名作成に参加する署名者 $u_{i_\alpha} \in \mathbb{L}$ の署名対象となるメッセージとした時，署名者 $u_{i_\alpha} \in \mathbb{L}$ は $h_{i_\alpha} = H(m_{i_\alpha})$ を計算し，

$$\sigma_{i_\alpha} = h_{i_\alpha}^{x_{i_\alpha}} \tag{2.39}$$

を生成する．
（楕円曲線上なら

$$\sigma_{i_\alpha} = x_{i_\alpha} h_{i_\alpha} \tag{2.40}$$

を生成する．）
σ_{i_α} を m_{i_α} に対する署名者 u_{i_α} の電子署名とする．

アグリゲート: アグリゲート署名作成に参加する全ての署名者の電子署名 σ_{i_α} を集め，

$$\sigma = \prod_{j \in \mathbb{J}} \sigma_j \tag{2.41}$$

を計算する．
（楕円曲線上なら

$$\sigma = \sum_{j \in \mathbb{J}} \sigma_j \tag{2.42}$$

を計算する．）
$(m_{i_1}, \ldots, m_{i_l}, \mathbb{L}, \sigma)$ をメッセージとアグリゲート署名の組とする．

署名検証: 検証者に $(m_{i_1}, \ldots, m_{i_l}, \mathbb{L}, \sigma)$, および g が与えられ，さらに $\mathbb{L}$ から検証に必要となる全ての検証鍵 v_j（$j \in \mathbb{J}$）が得られた時，検証者は全ての m_{i_α} から $h_{i_\alpha} = H(m_{i_\alpha})$ を計算し，ペアリングを用いて以下の判定を行う．

$$e(g, \sigma) = \prod_{j \in \mathbb{J}} e(v_j, h_j) \tag{2.43}$$

この時，電子署名が正しく作成されているのであれば，署名検証における判定式の左辺は

$$e(g,\sigma) = e(g, \prod_{j\in\mathbb{J}} h_j^{x_j}) = \prod_{j\in\mathbb{J}} e(g, h_j^{x_j}) = \prod_{j\in\mathbb{J}} e(g,h_j)^{x_j} \tag{2.44}$$

あるいは楕円曲線上なら

$$e(g,\sigma) = e(g, \sum_{j\in\mathbb{J}} x_j h_j) = \prod_{j\in\mathbb{J}} e(g, x_j h_j) = \prod_{j\in\mathbb{J}} e(g,h_j)^{x_j} \tag{2.45}$$

となり，右辺は

$$\prod_{j\in\mathbb{J}} e(v_j,h_j) = \prod_{j\in\mathbb{J}} e(g^{x_j},h_j) = \prod_{j\in\mathbb{J}} e(g,h_j)^{x_j} \tag{2.46}$$

あるいは楕円曲線上なら

$$\prod_{j\in\mathbb{J}} e(v_j,h_j) = \prod_{j\in\mathbb{J}} e(x_j g,h_j) = \prod_{j\in\mathbb{J}} e(g,h_j)^{x_j} \tag{2.47}$$

となり，同じ値になる．さらにこの方式について存在的偽造不能性が証明されている [13].

第3章　引用過程を保証できる階層表記型多重署名／アグリゲート署名方式

3.1　階層を考慮したセキュリティプロトコル

鍵生成の効率化を目的として階層を考慮したセキュリティプロトコルとしては，階層的 ID ベース暗号と呼ばれる方式がある [75]．ID ベース暗号とは，ユーザの ID 情報から公開鍵が生成可能となることで，証明書の発行を必要としない公開鍵暗号方式である [76]．この方式では，PKG（Private Key Generator:秘密鍵生成センター）と呼ばれる 1 つの信頼できる機関を設定する．PKG は master secret と呼ばれる秘密鍵を生成し，ユーザの秘密鍵を生成するのに使用される．全てのユーザの秘密鍵を，同じ master secret を利用して 1 つの PKG が生成するため，ユーザ数の増加と共に PKG の負担が増大するという課題がある．そこで，複数の PKG を階層的に配置し，ルートに対応する PKG は自らの秘密鍵 (master secret に対応) を用いて，子供となる PKG の秘密鍵を生成することで，下位の PKG への秘密鍵生成の委任が可能となる．この方式は秘密鍵生成の効率化のために PKG を階層化しており，ユーザが階層化されているものではない．

復号処理の効率化を目的としたものとしては，階層的暗号化と呼ばれる方式がある．これは主に JPEG 2000 など，データが階層的に記録され，1 つのファイルで様々な品質での利用が可能となっているコンテンツデータなどに適応されるものである．復号鍵は階層的に導出可能な構造になっており，許可されている品質以上での利用を防止すると同時に，後から高品質での利用を求めるユーザに対し，追加利用する階層までの復号鍵の差分を発行することで利用を許可する方式となっている [77, 78]．

電子署名方式においては，多重署名における階層化表現のアイデアが示されている [30]．しかし，一部に具体的な演算手法が定義されていない数式が示されており，実現性に課題がある．

実現可能なアグリゲート署名における階層表現については，山本と尾形によって提案されている方式 [79] がある．これは，自身以前の接続関係の情報に自身の識別情報とメッセージを一つにしたデータを作成し，これを署名対象とすることでそれまでの署名者との関係を表している．したがって，署名のプロトコルとして順序性を保証しているものではなく，メッセージの一部に階層情報を記載しておくという手法となっている．一方，この提案方式は署

名作成時には過去の署名参加者が作成したアグリゲート署名を全て集める必要がある．このため，本稿の目的である「コンテンツを引用して新たなコンテンツを作成していく過程を保証すること」を考えた場合，過去にコンテンツ作成にかかわった人の情報を全て集めてくる必要があり，編集者の負荷が徐々に増えるといった課題がある．上記とは別に，白川らによる提案方式 [80] もある．この方式では，ある階層以下でメッセージの改変が必要になった際に，その階層以下の署名だけを入れ替えることで新たなアグリゲート署名が作成可能となるといった利点がある．この利点により，例えば編集コンテンツの著作権保護システムに適応させることで，作成コンテンツの一部改修などの編集作業を可能にしている．一方，階層を識別するための中間鍵が必要となり，階層の深さに依存して中間鍵の個数が増加することに伴う署名サイズの増加といった課題が残されている．

3.2 階層表記型多重署名方式

3.2.1 順序付き多重署名方式

記号，前提条件，および要求条件

使用される記号，前提条件，セキュリティの要求条件について以下に示す．

記号:

$\mathbb{G}_1, \mathbb{G}_2$**:** ペアリングの演算が可能な楕円曲線上の点の集合（$\mathbb{G}_1$ の要素をペアリング関数の第 1 引数，$\mathbb{G}_2$ の要素をペアリング関数の第 2 引数とする）．

g_1**:** $\mathbb{G}_1$ の要素である生成元．

g_2**:** $\mathbb{G}_2$ の要素である生成元．

e**:** ペアリング関数．

u_y**:** 第 y 署名者（ただし，$u_y \neq \text{null}$）．

x_y, v_y, v'_y**:** u_y の署名鍵（$x_y \in \mathbb{Z}_p^*$），および検証鍵（$v_y = x_y g_1, v'_y = x_y g_2$）．

$\mathbb{L}_y$**:** u_1 から u_y までの署名者の署名順序情報．

V_y**:** 中間鍵．

m_y**:** u_y が署名対象とするメッセージ．

H**:** $\{0,1\}^* \to \mathbb{G}_2$ となる一方向性ハッシュ関数（例:*MapToGroup* [27, 43]）．

σ'_y**:** u_y の BLS 署名．

σ_y**:** u_1 から u_y までの順序付き多重署名．

前提条件

- 公開鍵基盤は整備されており，認証局により全ての署名者の署名鍵，および検証鍵のペアが正当に発行されている．
- 署名者に発行されている鍵ペア以外に，新たな鍵ペアの発行は行わない．
- 署名者は署名作成時において，定められた手順により正当に署名作成を行う．
- 多重署名作成中において，署名者間の通信は安全に行われ，作成中の中間情報を第三者が入手することは不可能である．

セキュリティの要求条件

電子署名のセキュリティ要求条件については，駒野らが提唱した多重署名方式におけるセキュリティモデルの構築手法 [81,82] などが提唱されている．本論文で提案する方式は「引用過程を保証できる」ことが目的であり，オリジナルコンテンツ作成者や編集者の著作権を正当に主張できること，またコンテンツ作成に関する責任を明確にすることに着目し，以下に示す要求条件を満たすことにする．

署名順序偽造不能性 多重署名に参加しない第三者が全ての公開情報を入手しても，多重署名の署名順序を勝手に変更したりすることが困難であること．

存在的偽造不能性 多重署名に参加しない第三者が全ての公開情報を入手しても，多重署名を偽造することが困難であること．

参加否認不能性 多重署名に参加している署名者が，結託して多重署名の参加を否認することが困難であること．

偽装参加不能性 多重署名に参加している署名者が，多重署名に参加している署名者かどうかに関わらず，第三者を勝手に署名者として追加することが困難であること．

プロトコル

鍵生成 $g_1 \in \mathbb{G}_1$，および $g_2 \in \mathbb{G}_2$ を生成元とする．署名者 u_i について $x_i \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_i = x_i g_1, v_i' = x_i g_2$ を計算する．x_i を署名者 u_i の署名鍵，v_i, v_i' を署名者 u_i の検証鍵とする．

署名作成 以下の手順で，多重署名が作成される．

1. 第 1 署名者 u_1 は，以下の計算を行う．

$$h = H(m) \tag{3.1}$$
$$\sigma'_1 = x_1 h \tag{3.2}$$
$$\sigma_1 = \sigma'_1 \tag{3.3}$$
$$\mathbb{L}_1 = \{(0, u_1)\} \tag{3.4}$$
$$V_1 = v_1 \tag{3.5}$$

署名者 u_1 は $(v_1, \sigma'_1, \sigma_1, \mathbb{L}_1, V_1)$ を第 2 署名者 u_2 に送信する．

2. 第 i 署名者 u_i は，第 $i-1$ 署名者 u_{i-1} から受信した $(v_{i-1}, \sigma'_{i-1}, \sigma_{i-1}, \mathbb{L}_{i-1}, V_{i-1})$ を用いて，以下の計算を行う．

$$h = H(m) \tag{3.6}$$
$$\sigma'_i = x_i h \tag{3.7}$$
$$\sigma_i = \sigma_{i-1} + (x_i - 1)\sigma'_{i-1} + \sigma'_i \tag{3.8}$$
$$\mathbb{L}_i = \mathbb{L}_{i-1} + \{(u_{i-1}, u_i)\} \tag{3.9}$$
$$V_i = V_{i-1} + (x_i - 1)v_{i-1} + v_i \tag{3.10}$$

署名者 u_i は $(v_i, \sigma'_i, \sigma_i, \mathbb{L}_i, V_i)$ を第 $i+1$ 署名者 u_{i+1} に送信する．

3. 最後の署名者 u_n は，受信した $(v_{n-1}, \sigma'_{n-1}, \sigma_{n-1}, \mathbb{L}_{n-1}, V_{n-1})$ を用いて，以下の計算を行う．

$$h = H(m) \tag{3.11}$$
$$\sigma'_n = x_n h \tag{3.12}$$
$$\sigma_n = \sigma_{n-1} + (x_n - 1)\sigma'_{n-1} + \sigma'_n \tag{3.13}$$
$$\mathbb{L}_n = \mathbb{L}_{n-1} + \{(u_{n-1}, u_n)\} \tag{3.14}$$
$$V_n = V_{n-1} + (x_n - 1)v_{n-1} + v_n \tag{3.15}$$

この σ_n，$\mathbb{L}_n$，および V_n を，署名者 $u_1, \ldots, u_n$ の，署名対象 m に対する多重署名として公開する．

署名検証 以下の手順で，多重署名の検証を行う．

1. 検証者は，$\mathbb{L}_n$ に示されている全ての署名者の検証鍵 $v_1, \ldots, v_n$，$v'_1, \ldots, v'_n$ および署名対象となる全てのメッセージ m を集める．

2. 検証者はペアリングを用いて以下の判定を行う．

$$e(V_n, g_2) = e(v_1, g_2) \cdot \prod_{j=2}^{n} e(v_{j-1}, v'_j) \tag{3.16}$$

署名の順序は，$\mathbb{L}_n$ の要素の内第 1 項が null の要素の第 2 項に示されている署名者が最初の署名者，その署名者が第 1 項に示されている要素の第 2 項に示されている署名者が第 2 署名者というように決定する．
もし正しく中間鍵が作成されていれば，左辺は

$$\begin{aligned} e(V_n, g_2) =& e((x_1 + \sum_{j=2}^{n}(x_{j-1}x_j))g_1, g_2) \\ =& e(g_1, g_2)^{x_1 + \sum_{j=2}^{n}(x_{j-1}x_j)} \\ =& e(g_1, g_2)^{x_1} \cdot \prod_{j=2}^{n} e(g_1, g_2)^{x_{j-1}x_j} \\ =& e(x_1 g_1, g_2) \cdot \prod_{j=2}^{n} e(x_{j-1}g_1, x_j g_2) \\ =& e(v_1, g_2) \cdot \prod_{j=2}^{n} e(v_{j-1}, v'_j) \end{aligned} \tag{3.17}$$

となり，右辺と一致する．

3. 検証者はメッセージから $h = H(m)$ を計算し，ペアリングを用いて以下の判定を行う．

$$e(g, \sigma_n) = e(V_n, h) \tag{3.18}$$

もし正しく署名が作成されていれば，

$$\begin{aligned} e(g_1, \sigma_n) =& e(g_1, (x_1 + \sum_{j=2}^{n}(x_{j-1}x_j))h) \\ =& e((x_1 + \sum_{j=2}^{n}(x_{j-1}x_j))g_1, h) \\ =& e(V_n, h) \end{aligned} \tag{3.19}$$

となり，右辺と一致する．

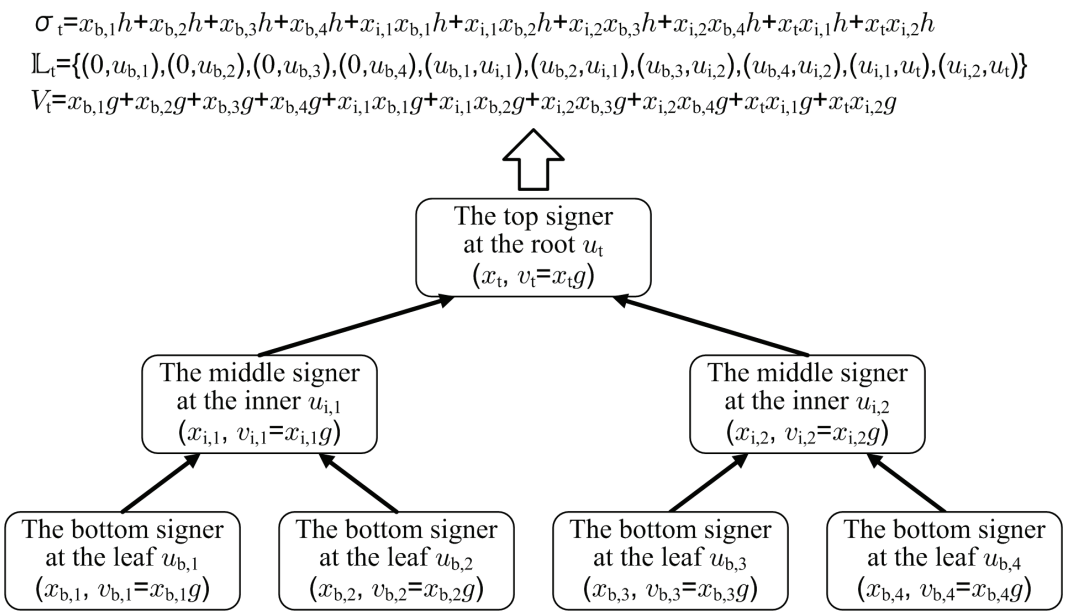

図 3.1: 2 分木 3 階層の木構造表記型多重署名

3.2.2 木構造表記型多重署名方式

順序付き多重署名方式からの拡張概要

3.2.1 節で示した順序付き多重署名方式は，署名者が一列方向で順番に署名を作成する際に，前後の署名者間の関係を示す演算を導入し，その手順を繰り返すことで，署名者の順序を規定している．この手順を隣接する署名者に対応して 1 対多に拡張し，多段的に繰り返すことで，木構造表記型多重署名方式を構成することが可能となる．

なお， 節では図 3.1 に示す 2 分木 3 階層の木構造を例に説明する．

記号，前提条件，および要求条件

使用される記号について， 3.3.1 節で定義したものに加えて，下記を新たに定義する．

$u_{\mathbf{t}}$**:** ルートノード署名者（1 人）．

$u_{\mathbf{i},o_\mathbf{i}}$**:** 中間ノード署名者 ($o_\mathrm{i}$ は 2 人いる中間ノード署名者の固有識別番号).

$u_{\mathbf{b},o_\mathbf{b}}$**:** リーフノード署名者 ($o_\mathrm{b}$ は 4 人いるリーフノード署名者の固有識別番号).

$x_{\mathbf{t}}, v_{\mathbf{t}}, v'_{\mathbf{t}}$**:** ルートノード署名者の署名鍵，および検証鍵（$v_\mathrm{t} = x_\mathrm{t} g_1, v_\mathrm{t} = x_\mathrm{t} g_2$）．

$x_{\mathbf{i},o_\mathbf{i}}, v_{\mathbf{i},o_\mathbf{i}}, v'_{\mathbf{i},o_\mathbf{i}}$**:** 中間ノード署名者の署名鍵，および検証鍵（$v_{\mathrm{i},o_\mathrm{i}} = x_{\mathrm{i},o_\mathrm{i}} g_1, v'_{\mathrm{i},o_\mathrm{i}} = x_{\mathrm{i},o_\mathrm{i}} g_2$）．

$x_{\mathbf{b},o_\mathrm{b}}, v_{\mathbf{b},o_\mathrm{b}}, v'_{\mathbf{b},o_\mathrm{b}}$**:** リーフノード署名者の署名鍵，および検証鍵（$v_{\mathrm{b},o_\mathrm{b}} = x_{\mathrm{b},o_\mathrm{b}} g_1, v'_{\mathrm{b},o_\mathrm{b}} = x_{\mathrm{b},o_\mathrm{b}} g_2$）.

$\mathbb{L}_\mathbf{t}$**:** リーフノード署名者からルートノード署名者までの全署名者の位置情報（関係を示した接続情報）.

$\mathbb{L}_{\mathbf{i},o_\mathrm{i}}$**:** リーフノード署名者から中間ノード署名者 $u_{\mathrm{i},o_\mathrm{i}}$ までの署名者の位置情報（関係を示した接続情報）.

$\mathbb{L}_{\mathbf{b},o_\mathrm{b}}$**:** リーフノード署名者 $u_{\mathrm{b},o_\mathrm{b}}$ の位置情報.

$V_\mathbf{t}$**:** 多重署名を検証する最終中間鍵.

$V_{\mathbf{i},o_\mathrm{i}}, V_{\mathbf{b},o_\mathrm{b}}$**:** V_t を求めるためのサブ中間鍵.

$\sigma'_{\mathbf{i},o_\mathrm{i}}, \sigma'_{\mathbf{b},o_\mathrm{b}}$**:** BLS 署名方式による $u_{\mathrm{i},o_\mathrm{i}}$，あるいは $u_{\mathrm{b},o_\mathrm{b}}$ の m への署名.

$\sigma_\mathbf{t}$**:** 全署名者の署名を集約した木構造表記型多重署名.

$\sigma_{\mathbf{i},o_\mathrm{i}}, \sigma_{\mathbf{b},o_\mathrm{b}}$**:** σ_t を求めるための中間ノード署名者 $u_{\mathrm{i},o_\mathrm{i}}$，あるいはリーフノード署名者 $u_{\mathrm{b},o_\mathrm{b}}$ までのサブ木構造表記型多重署名.

また，前提条件，およびセキュリティの要求条件については 3.2.1 節で示したものと同じである.

プロトコル

鍵生成 $g \in \mathbb{G}_1$ を生成元とする．署名者 $u_{q,r(q)_w}$ について，$x_{q,r(q)_w} \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_{q,r(q)_w} = x_{q,r(q)_w} g_1, v'_{q,r(q)_w} = x_{q,r(q)_w} g_2$ を計算する．$x_{q,r(q)_w}$ を署名者 $u_{q,r(q)_w}$ の署名鍵，$v_{q,r(q)_w}, v'_{q,r(q)_w}$ を署名者 $u_{q,r(q)_w}$ の検証鍵とする.

署名作成 署名作成手順を以下に示す.

1. リーフノード署名者 $u_{\mathrm{b},1}$ は

$$h = H(m) \tag{3.20}$$

$$\sigma'_{\mathrm{b},1} = x_{\mathrm{b},1} h \tag{3.21}$$

$$\sigma_{\mathrm{b},1} = \sigma_{\mathrm{b},1} \tag{3.22}$$

$$\mathbb{L}_{\mathrm{b},1} = \{(0, u_{\mathrm{b},1})\} \tag{3.23}$$

$$V_{\mathrm{b},1} = v_{\mathrm{b},1} \tag{3.24}$$

を計算し，$(u_{\mathrm{b},1}, v_{\mathrm{b},1}, \sigma'_{\mathrm{b},1}, \sigma_{\mathrm{b},1}, \mathbb{L}_{\mathrm{b},1}, V_{\mathrm{b},})$ を署名者 $u_{\mathrm{b},1}$ の署名セットとして，上位の中間ノード署名者 $u_{\mathrm{i},1}$ に送信する．

他のリーフノード署名者も同様の手順により，署名者 $u_{\mathrm{b},2}$ は $u_{\mathrm{i},1}$ に，署名者 $u_{\mathrm{b},3}, u_{\mathrm{b},4}$ は $u_{\mathrm{i},2}$ に送信する．

2. 中間ノード署名者 $u_{\mathrm{i},1}$ は

$$h = H(m) \tag{3.25}$$
$$\sigma'_{\mathrm{i},1} = x_{\mathrm{i},1} h \tag{3.26}$$
$$\sigma_{\mathrm{i},1} = S_{\mathrm{b},1} + S_{\mathrm{b},2} \tag{3.27}$$
$$+ (x_{\mathrm{i},1} - 1)\sigma_{\mathrm{b},1} + (x_{\mathrm{i},1} - 1)\sigma_{\mathrm{b},2} + \sigma_{\mathrm{i},1} \tag{3.28}$$
$$\mathbb{L}_{\mathrm{i},1} = \mathbb{L}_{\mathrm{b},1} + \mathbb{L}_{\mathrm{b},2} + \{(u_{\mathrm{b},1}, u_{\mathrm{i},1}) + \{(u_{\mathrm{b},2}, u_{\mathrm{i},1})\} \tag{3.29}$$
$$V_{\mathrm{i},1} = V_{\mathrm{b},1} + V_{\mathrm{b},2} \tag{3.30}$$
$$+ (x_{\mathrm{i},1} - 1)v_{\mathrm{b},1} + (x_{\mathrm{i},1} - 1)v_{\mathrm{b},2} + v_{\mathrm{i},1} \tag{3.31}$$

を計算し，$(u_{\mathrm{i},1}, v_{\mathrm{i},1}, \sigma'_{\mathrm{i},1}, \sigma_{\mathrm{i},1}, \mathbb{L}_{\mathrm{i},1}, V_{\mathrm{i},1})$ を署名者 $u_{\mathrm{i},1}$ の署名セットとして，上位のルートノード署名者 u_t に送信する．

もう 1 人の中間ノード署名者 $u_{\mathrm{i},1}$ も同様の手順により，ルートノード署名者 u_t に送信する．

3. ルートノード署名者 u_{t} は

$$h = H(m) \tag{3.32}$$
$$\sigma'_{\mathrm{t}} = x_{\mathrm{t}} h \tag{3.33}$$
$$\sigma_{\mathrm{t}} = S_{\mathrm{i},1} + S_{\mathrm{i},2} + (x_{\mathrm{t}} - 1)\sigma_{\mathrm{i},1} + (x_{\mathrm{t}} - 1)\sigma_{\mathrm{i},2} + \sigma_{\mathrm{t}} \tag{3.34}$$
$$\mathbb{L}_{\mathrm{t}} = \mathbb{L}_{\mathrm{i},1} + \mathbb{L}_{\mathrm{i},2} + \{(u_{\mathrm{i},1}, u_{\mathrm{t}}) + \{(u_{\mathrm{i},2}, u_{\mathrm{t}})\} \tag{3.35}$$
$$V_{\mathrm{t}} = V_{\mathrm{i},1} + V_{\mathrm{i},2} + (x_{\mathrm{t}} - 1)v_{\mathrm{i},1} + (x_{\mathrm{t}} - 1)v_{\mathrm{i},2} + v_{\mathrm{t}} \tag{3.36}$$

を計算し，$(\sigma_{\mathrm{t}}, \mathbb{L}_{\mathrm{t}}, V_{\mathrm{t}})$ を木構造表記型多重署名セットとして公開する．

署名検証 署名検証手順を以下に示す．

1. 検証者は $\mathbb{L}_{\mathrm{t}}$ に示されている署名者の情報から，全ての検証鍵 $v_{\mathrm{t}}, v_{\mathrm{i},o_{\mathrm{i}}}, v_{\mathrm{b},o_{\mathrm{b}}}$ を集める．
2. 検証者は $\mathbb{L}_{\mathrm{t}}$ の情報から署名者の構造を確認し，ペアリングを用いて以下の判定

を行う．

$$
\begin{aligned}
&e(V_{\mathrm{t}}, g_2) \\
&= \prod_{j=1}^{4} e(v_{\mathrm{b},j}, g_2) \cdot \prod_{k=1}^{2} e(v_{\mathrm{i},1}, v'_{\mathrm{b},k}) \cdot \prod_{n=3}^{4} e(v_{\mathrm{i},2}, v'_{\mathrm{b},n}) \cdot \prod_{q=1}^{2} e(v_{\mathrm{t}}, v'_{\mathrm{i},q})
\end{aligned} \tag{3.37}
$$

もし正しく中間鍵が作成されていれば，左辺は

$$
\begin{aligned}
e(V_{\mathrm{t}}, g_2) =& e\left(\left(\sum_{j=1}^{4} x_{\mathrm{b},j} + \sum_{k=1}^{2}(x_{\mathrm{i},1}x_{\mathrm{b},k}) + \sum_{n=3}^{4}(x_{\mathrm{i},2}x_{\mathrm{b},n}) + \sum_{q=1}^{2}(x_{\mathrm{t}}x_{\mathrm{i},q})\right) g_1, g_2\right) \\
=& e(g_1, g_2)^{\sum_{j=1}^{4} x_{\mathrm{b},j} + \sum_{k=1}^{2}(x_{\mathrm{i},1}x_{\mathrm{b},k}) + \sum_{n=3}^{4}(x_{\mathrm{i},2}x_{\mathrm{b},n}) + \sum_{q=1}^{2}(x_{\mathrm{t}}x_{\mathrm{i},q})} \\
=& e(g_1, g_2)^{\sum_{j=1}^{4} x_{\mathrm{b},j}} \cdot e(g_1, g_2)^{\sum_{k=1}^{2}(x_{\mathrm{i},1}x_{\mathrm{b},k})} \\
& \cdot e(g_1, g_2)^{\sum_{n=3}^{4}(x_{\mathrm{i},2}x_{\mathrm{b},n})} \cdot e(g_1, g_2)^{\sum_{q=1}^{2}(x_{\mathrm{t}}x_{\mathrm{i},q})} \\
=& \prod_{j=1}^{4} e(x_{\mathrm{b},j}g_1, g_2) \cdot \prod_{k=1}^{2} e(x_{\mathrm{i},1}g_1, x_{\mathrm{b},k}g_2) \\
& \cdot \prod_{n=3}^{4} e(x_{\mathrm{i},2}g_1, x_{\mathrm{b},n}g_2) \cdot \prod_{q=1}^{2} e(x_{\mathrm{t}}g_1, x_{\mathrm{i},q}g_2) \\
=& \prod_{j=1}^{4} e(v_{\mathrm{b},j}, g_2) \cdot \prod_{k=1}^{2} e(v_{\mathrm{i},1}, v'_{\mathrm{b},k}) \cdot \prod_{n=3}^{4} e(v_{\mathrm{i},2}, v'_{\mathrm{b},n}) \cdot \prod_{q=1}^{2} e(v_{\mathrm{t}}, v'_{\mathrm{i},q})
\end{aligned} \tag{3.38}
$$

となり，右辺と一致する．

3. 検証者はメッセージから $h = H(m)$ を計算し，ペアリングを用いて以下の判定を行う．

$$
e(g_1, \sigma_{\mathrm{t}}) = e(V_{\mathrm{t}}, h) \tag{3.39}
$$

もし正しく署名が作成されていれば，左辺は

$$
\begin{aligned}
e(g_1, \sigma_{\mathrm{t}}) =& e\left(g_1, \left(\sum_{j=1}^{4} x_{\mathrm{b},j} + \sum_{k=1}^{2}(x_{\mathrm{i},1}x_{\mathrm{b},k}) + \sum_{n=3}^{4}(x_{\mathrm{i},2}x_{\mathrm{b},n}) + \sum_{q=1}^{2}(x_{\mathrm{t}}x_{\mathrm{i},q})\right) g_2\right) \\
=& e\left(\left(\sum_{j=1}^{4} x_{\mathrm{b},j} + \sum_{k=1}^{2}(x_{\mathrm{i},1}x_{\mathrm{b},k}) + \sum_{n=3}^{4}(x_{\mathrm{i},2}x_{\mathrm{b},n}) + \sum_{q=1}^{2}(x_{\mathrm{t}}x_{\mathrm{i},q})\right) g_1, g_2\right) \\
=& e(V_{\mathrm{t}}, h)
\end{aligned} \tag{3.40}
$$

となり，右辺と一致する．

3.2.3 安全性の考察

3.2.1 節で定義したセキュリティの要求事項について考察する．

署名順序偽造不能性 2 人の署名者 u_{i-1}, u_i がそれぞれの署名鍵 x_{i-1}, x_i，および署名対象となるメッセージ m を用いて $x_{i-1}x_iH(m)$ の演算を行い，各署名者間におけるこの演算結果の総和が多重署名となる．さらにこの多重署名を検証するために，2 人の署名者 u_{i-1}, u_i はそれぞれの署名鍵 x_{i-1}, x_i から $x_{i-1}x_ig_1$ の演算を行い，各署名者間におけるこの演算結果の総和となる中間鍵 V_n を作成する．上記における署名鍵の積の値である $x_{i-1}x_i$ が，2 人の署名者の順序が隣接していることを保証する．したがって，この V_n の偽造が困難で正当性が保証されれば，署名順序の正当性が保証される．

ここでは，Boldyreba が行った手法 [28,29] を用いて，V_n の安全性を証明する．

定理 1 ランダムオラクルモデルにおいて，V_n の偽造困難性と *CDH* 問題の困難性は等価である．

証明 1 A を V_n を偽造しようとする攻撃者とし，B を *CDH* 問題を解く攻撃者とする．B の攻撃が成功するなら A の攻撃が成功することは自明であるため，A の攻撃が成功するなら B の攻撃が成功することを示すことで，両者の攻撃が等価であることを示す．

$n = 1$ の場合： V_n は生成されない．

$n = 2$ の場合： *2* 人の署名者による順序付き多重署名となる．B は v_1, v_2 の検証鍵を保持しており，ランダムオラクルへの応答を行う．この v_1, v_2 に対し，$\exists x_1, \exists x_2 \in \mathbb{Z}_p^*$ を用いて x_1g, x_2g と表すことが可能となるが，B には x_1, x_2 の値は未知であるものとする．

B は A を *Honest Player* として実行する．まず，B は A に v_1, v_2 を与え，A は V_2 を出力する．B はこの出力値と保持している v_2 から

$$x_1x_2g_1 = V_2 - v_2 \tag{3.41}$$

を計算することにより，x_1g_1, x_2g_1 から $x_1x_2g_1$ が得られ，B の攻撃が成功する．

$n = z (z \geq 3)$ の場合： *3* 人以上の署名者による順序付き多重署名となる．さらに $n = z-1$ において，定理 *1* が証明されていると仮定する．B は v_{z-1}, v_z の検証鍵を保持しており，ランダムオラクルへの応答を行う．この v_{z-1}, v_z に対し，$\exists x_{z-1}, \exists x_z \in \mathbb{Z}_p^*$ を用いて $x_{z-1}g, x_zg$ と表すことが可能となるが，B には x_{z-1}, x_z の値は未知であるものとする．

B は A を *Honest Player* として実行する．まず，B は A に v_{z-1}, v_z を与え，A は V_{z-1}, V_z を出力する．B はこの出力値と保持している v_{z-1}, v_z から

$$x_{z-1}x_z g_1 = V_z - V_{z-1} + v_{z-1} - v_z \tag{3.42}$$

を計算することにより，$x_{z-1}g_1, x_z g_1$ から $x_{z-1}x_z g_1$ が得られ，B の攻撃が成功する．

以上，数学的帰納法により，定理 1 が成立する． ■

存在的偽造不能性 署名順序の正当性が保証された中間鍵 V_n によって検証される多重署名とは，順序付き多重署名セットに含まれる σ_n のことである．したがって，この σ_n の偽造が困難であれば，多重署名の正当性が保証される．

ここでは，Boldyreba が行った手法 [28,29] を用いて，σ_n の安全性を証明する．

定理 2 ランダムオラクルモデルにおいて，σ_n の偽造困難性と *BLS* 署名の偽造困難性は等価である．

証明 2 A を σ_n を偽造しようとする攻撃者とし，B を *BLS* 署名を偽造しようとする攻撃者とする．B の攻撃が成功するなら A の攻撃が成功することは自明であるため，A の攻撃が成功するなら B の攻撃が成功することを示すことで，両者の攻撃が等価であることを示す．

B は v_n の検証鍵を保持しており，ランダムオラクル，および署名オラクルへの応答を行う．この v_n に対し，$\exists x_n \in \mathbb{Z}_p^*$ を用いて $x_n g$ と表すことが可能となるが，B には x_n の値は未知であるものとする．

B は A を *Honest Player* として実行する．まず，B は A に v_n を与え，A は V_n，および最後から l 人前までの全ての署名鍵 $x_1, \ldots, x_{n-1}$ を出力する．さらに A はランダムオラクルと署名オラクルへの応答により，署名対象のメッセージ m とその署名 σ_{i_n} を求め，B に返答する．B はこの出力値から

$$x_n H(m) = (x_{n-1} - 1)^{p-1} \left(\sigma_n - \sum_{j=2}^{n-1} \left(x_{j-1} x_j H(m) \right) \right) \tag{3.43}$$

を計算することにより，$x_n H(m)$ が得られ，B の攻撃が成功する．

以上により，定理 2 が成立する． ■

参加否認不能性 提案方式において，隣接する 2 人の署名者 u_{i-1}, u_i の関係を，σ_n や V_n の式内における各々の署名鍵の積 $x_{i-1}x_i$ で表現している．したがって，ある署名鍵 x'_{i-1}, x'_i

を用いて $x'_{i-1}x'_i = x_{i-1}x_i$ が成り立つなら，u_{i-1}, u_i は自分らが署名に参加したのではなく x'_{i-1}, x'_i を署名鍵として保有している署名者が参加したという主張を許すこととなり，署名参加の否認が可能となる．

しかし，3.2.1 節の鍵生成の項目より，$x'_{i-1}x'_i = x_{i-1}x_i$ となる x'_{i-1}, x'_i を得ることは不可能である．したがって，結託攻撃による署名参加の否認は不可能である．

偽装参加不能性 提案方式において，多重署名は

$$\sigma_n = x_1x_2h + \cdots + x_{i-1}x_ih + x_ix_{i+1}h + \cdots + x_{n-1}x_nh + x_nh \tag{3.44}$$

となる．ここで，ある署名者 u_i が署名に参加していない $u_{i'}$ を署名者として勝手に追加しようと考えたとする．まず，署名者 u_i は $u_{i'}$ の検証鍵 $v_{i'}$ を入手し，$v_d = v_i + v_{i'}$ を計算する．署名者 u_i が $v_d = x_dg_1$ となる x_d が得られるなら，

$$\begin{aligned}\sigma_{n'} &= x_1x_2h + \cdots + x_{i-1}(x_d - x_i)h + (x_d - x_i)x_{i+1}h + \cdots + x_{n-1}x_nh + x_nh \\ &= x_1x_2h + \cdots + x_{i-1}x_{i'}h + x_{i'}x_{i+1}h + \cdots + x_{n-1}x_nh + x_nh)\end{aligned} \tag{3.45}$$

を計算することにより，$u_{i'}$ を署名者として追加させることが可能となる．

しかし，$v_d = x_dg_1$ から x_d を直接得ることは離散対数問題により困難である．したがって，実際には $u_{i'}$ を署名者として追加することは困難である．

なお木構造表記型多重署名方式は，順序付き多重署名方式と同様に隣接する署名者間の関係を繰り返すことで構成されており，中間鍵の構造，および多重署名の構造は同質のものである．したがって，安全性については同等の議論が成立し，木構造表記型多重署名は 3.2.1 節に示したセキュリティの要求事項を満たしている．

3.3 階層表記型アグリゲート署名方式

3.3.1 順序付きアグリゲート署名方式

記号，前提条件，および要求条件

使用される記号，前提条件，セキュリティの要求条件について以下に示す．

記号:

$\mathbb{G}_1, \mathbb{G}_2$**:** ペアリングの演算が可能な楕円曲線上の点の集合（$\mathbb{G}_1$ の要素をペアリング関数の第 1 引数，$\mathbb{G}_2$ の要素をペアリング関数の第 2 引数とする）．

g: $\mathbb{G}_1$ の要素である生成元.

e: ペアリング関数.

u_y: 第 y 署名者（ただし，$u_y \neq \text{null}$）.

x_y, v_y: u_y の署名鍵（$x_y \in \mathbb{Z}_p^*$），および検証鍵（$v_y = x_y g$）.

$\mathbb{L}_y$: u_1 から u_y までの署名者の署名順序情報.

m_y: u_y が署名対象とするメッセージ.

H: $\{0,1\}^* \to \mathbb{G}_2$ となる一方向性ハッシュ関数（例:*MapToGroup* [27, 43]）.

σ_y: u_1 から u_y までの順序付きアグリゲート署名.

前提条件

- 公開鍵基盤は整備されており，認証局により全ての署名者の署名鍵，および検証鍵のペアが正当に発行されている.
- 署名者に発行されている鍵ペア以外に，新たな鍵ペアの発行は行わない.
- 署名者は署名作成時において，定められた手順により正当に署名作成を行う.
- アグリゲート署名作成中において，署名者間の通信は安全に行われ，作成中の中間情報を第三者が入手することは不可能である.

セキュリティの要求条件

電子署名のセキュリティ要求条件については，駒野らが提唱した多重署名方式におけるセキュリティモデルの構築手法 [81, 82] などが提唱されている．本論文で提案する方式は「引用過程を保証できる」ことが目的であり，オリジナルコンテンツ作成者や編集者の著作権を正当に主張できること，またコンテンツ作成に関する責任を明確にすることに着目し，以下に示す要求条件を満たすことにする.

アグリゲート署名正当性（存在的偽造不能性） アグリゲート署名に参加しない第三者が全ての公開情報を入手しても，アグリゲート署名を偽造したり，すでに作成されたアグリゲート署名の署名順序を勝手に変更したりすることが困難であること.

参加否認不能性 アグリゲート署名に参加している署名者が，結託してアグリゲート署名の参加を否認することが困難であること.

偽装参加不能性 アグリゲート署名に参加している署名者が，アグリゲート署名に参加している署名者かどうかに関わらず，第三者を勝手に署名者として追加することが困難であること.

プロトコル

鍵生成 $g \in \mathbb{G}_1$ を生成元とする．署名者 u_i について $x_i \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_i = x_i g$ を計算する．x_i を署名者 u_i の署名鍵，v_i を署名者 u_i の検証鍵とする．

署名作成 以下の手順で，アグリゲート署名が作成される．

1. 第 1 署名者 u_1 は，メッセージ m_1 から $h_1 = H(m_1)$ を求め， 2.3.3 節で説明した BLS 署名と同様な署名作成処理を行うことで

$$\sigma_1 = x_1 h_1 \tag{3.46}$$

を計算する．また，

$$\mathbb{L}_1 = \{(\text{null}, u_1)\} \tag{3.47}$$

を作成する．この σ_1，$\mathbb{L}_1$，および m_1（または h_1）を第 2 署名者 u_2 に送信する．

2. 第 i 署名者 u_i は，第 $i-1$ 署名者 u_{i-1} から受信した m_{i-1} を用いて $h_{i-1} = H(m_{i-1})$ を求める（または h_{i-1} を受信したら，それを利用する）．さらに署名者 u_i は，自分が本来署名したいメッセージ m_i から $h_i = H(m_i)$ を求める．これと署名者 u_{i-1} から受信した σ_{i-1} を用いて

$$\begin{aligned}\sigma_i &= \sigma_{i-1} + x_i h_{i-1} + x_i h_i \\ &= x_1 h_1 + \sum_{j=2}^{i} (x_j h_{j-1} + x_j h_j)\end{aligned} \tag{3.48}$$

を計算する．また，受信した $\mathbb{L}_{i-1}$ を用いて

$$\begin{aligned}\mathbb{L}_i &= \mathbb{L}_{i-1} + \{(u_{i-1}, u_i)\} \\ &= \{(\text{null}, u_1), (u_1, u_2), \ldots, (u_{i-1}, u_i)\}\end{aligned} \tag{3.49}$$

を作成する．この σ_i，$\mathbb{L}_i$，および m_i（または h_i）を第 $i+1$ 署名者 u_{i+1} に送信する．この手順を最後から一人手前の署名者まで再帰的に行う．

3. 最後の署名者 u_n は，直前の署名者 u_{n-1} から受信した m_{n-1} を用いて $h_{n-1} = H(m_{n-1})$ を求める（または h_{n-1} を受信したら，それを利用する）．さらに署名者 u_n は，自分が本来署名したいメッセージ m_n から $h_n = H(m_n)$ を求める．これと署名者 u_{n-1}

から受信した σ_{n-1} を用いて

$$\begin{aligned}\sigma_n &= \sigma_{n-1} + x_n h_{n-1} + x_n h_n \\ &= x_1 h_1 + \sum_{j=2}^{n} (x_j h_{j-1} + x_j h_j)\end{aligned} \tag{3.50}$$

を計算する．また，受信した $\mathbb{L}_{n-1}$ を用いて

$$\begin{aligned}\mathbb{L}_n &= \mathbb{L}_{n-1} + \{(u_{n-1}, u_n)\} \\ &= \{(\text{null}, u_1), (u_1, u_2), \ldots, (u_{n-1}, u_n)\}\end{aligned} \tag{3.51}$$

を作成する．この σ_n，および $\mathbb{L}_n$ を，署名者 $u_1, \ldots, u_n$ の，署名対象 $m_1, \ldots, m_n$ に対するアグリゲート署名として公開する．

署名検証 以下の手順で，アグリゲート署名の検証を行う．

1. 検証者は，$\mathbb{L}_n$ に示されている全ての署名者の検証鍵 $v_1, \ldots, v_n$，および署名対象となる全てのメッセージ $m_1, \ldots, m_n$ を集める．署名の順序は，$\mathbb{L}_n$ の要素の内第 1 項が null の要素の第 2 項に示されている署名者が最初の署名者，その署名者が第 1 項に示されている要素の第 2 項に示されている署名者が第 2 署名者というように決定する．
2. 検証者は，集めたメッセージから $h_i = H(m_i)$ を求める．
3. 検証者はペアリングを用いて以下の判定を行う．

$$e(g, \sigma_n) = e(v_1, h_1) \cdot \prod_{j=2}^{n} e(v_j, h_{j-1} + h_j) \tag{3.52}$$

もし正しく署名が作成されていれば，左辺は

$$\begin{aligned}e(g, \sigma_n) =& e(g, x_1 h_1 + \sum_{j=2}^{n} (x_j h_{j-1} + x_j h_j)) \\ =& e(g, x_1 h_1) \cdot e(g, x_2(h_1 + h_2)) \cdot \ldots e(g, x_n(h_{n-1} + h_n)) \\ =& e(x_1 g, h_1) \cdot e(x_2 g, h_1 + h_2) \cdot \ldots e(x_n g, h_{n-1} + h_n) \\ =& e(v_1, h_1) \cdot \prod_{j=2}^{n} e(v_j, h_{j-1} + h_j)\end{aligned} \tag{3.53}$$

となり，右辺と一致する．

アグリゲート署名への新たな署名者の追加 最後の署名者 u_n は σ_n，および $\mathbb{L}_n$ を署名対象 $m_1, \ldots, m_n$ に対するアグリゲート署名として公開している．

ここで，新たな署名者 u_{n+1} が自分を追加して新たなアグリゲート署名を作成する場合を考える．この時，署名者 u_n は作成したアグリゲート署名（$\sigma_n, \mathbb{L}_n$），および m_n（または h_n）を新たな署名者 u_{n+1} に送信し，署名者 u_{n+1} が署名作成の 3 の手順を行う．すなわち署名者 u_{n+1} は

$$\begin{aligned}\sigma_{n+1} &= \sigma_n + x_{n+1}h_n + x_{n+1}h_{n+1} \\ &= x_1h_1 + \sum_{j=2}^{n+1}(x_jh_{j-1} + x_jh_j)\end{aligned} \tag{3.54}$$

を計算し，

$$\begin{aligned}\mathbb{L}_{n+1} &= \mathbb{L}_n + \{(u_n, u_{n+1})\} \\ &= \{(\mathrm{null}, u_1), (u_1, u_2), \ldots, (u_n, u_{n+1})\}\end{aligned} \tag{3.55}$$

を作成する．この σ_{n+1}，および $\mathbb{L}_{n+1}$ を新たなアグリゲート署名として公開する．

この手順で示したように，新たな署名者が公開されているアグリゲート署名に自分を追加した新たなアグリゲート署名を作成する場合，それまでの最終署名者は新たな処理を必要とせず，公開されている情報を新たな署名者に送信するだけで良い．

3.3.2 木構造表記型アグリゲート署名方式

順序付きアグリゲート署名方式からの拡張概要

3.3.1 節で示した順序付きアグリゲート署名方式は，署名者が一列方向で順番に署名を作成する際に，前後の署名者間の関係を示す演算を導入し，その手順を繰り返すことで，署名者の順序を規定している．この手順を隣接する署名者に対応して 1 対多に拡張し，多段的に繰り返すことで，図 3.2 に示すような署名者の関係を表記できる木構造表記型アグリゲート署名方式を構成することが可能となる．

記号，前提条件，および要求条件

使用される記号について， 3.3.1 節で定義したものに加えて，下記を新たに定義する．

$u_{q,r(q)_w}$**：** コンテンツの作成者，あるいは編集者であり，アグリゲート署名作成に参加した署名者（各署名者は木構造におけるノードに配置され，q は，ルートノードを階層 1 とし

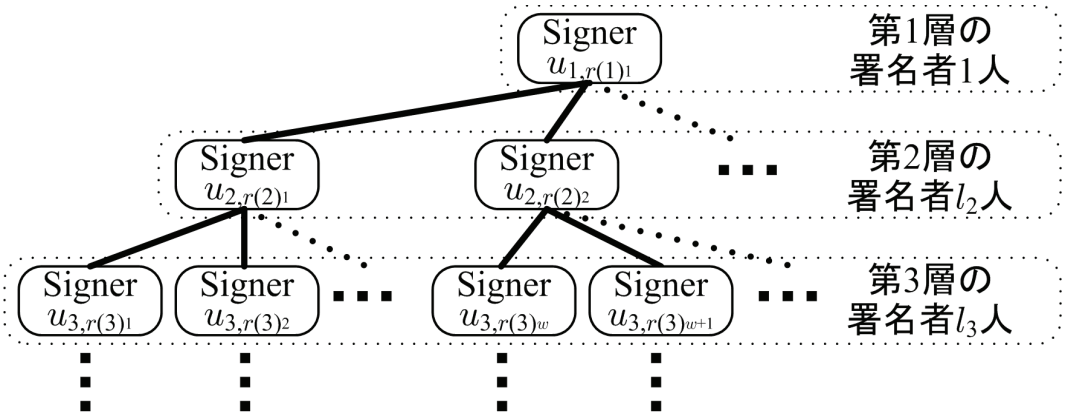

図 3.2: 木構造表記型アグリゲート署名における署名者の関係

た時の，その作成者・編集者が位置するノードの階層を示し（$1 \leq q \leq n$），$r(q)_w$ は階層 q に位置する署名者の識別符号（l_q を q 階層にいる全署名者数とした時，$1 \leq w \leq l_q$）とする．但し $u_{q,r(q)_w} \neq \text{null}$）．

$x_{q,r(q)_w}, v_{q,r(q)_w}$：$u_{q,r(q)_w}$ の署名鍵，および検証鍵．

$\sigma_{q,r(q)_w}$：$u_{q,r(q)_w}$ までの木構造表記型アグリゲート署名．

$\mathbb{L}_{q,r(q)_w}$：木構造で表現される $u_{q,r(q)_w}$ 以下の署名者の関係を示した接続情報
（例えば図 3.3 のような署名者の関係の場合は

$$\begin{aligned}\mathbb{L}_{q,r(q)_w} = \{&(u_{q,r(q)_w}, \{u_{q+1,r(q+1)_\alpha}, \dots\}),\\ &(u_{q+1,r(q+1)_\alpha}, \{u_{q+2,r(q+2)_\beta}, \dots\}), \dots,\\ &(u_{i-1,r(i-1)_\gamma}, \{u_{i,r(i)_\delta}, \dots\}), \dots\\ &(\text{null}, \{u_{i,r(i)_\epsilon}\}), \dots\}\end{aligned} \tag{3.56}$$

但し，他の部分木における添字の符号との関係で，$\alpha, \dots, \epsilon$ は途中の数字からの数字となる場合がある）．

$m_{q,r(q)_w}$：$u_{q,r(q)_w}$ が署名対象とするメッセージ（コンテンツのハッシュ値やメタ情報など）．

また，前提条件，およびセキュリティの要求条件については 3.3.1 節で示したものと同じである．

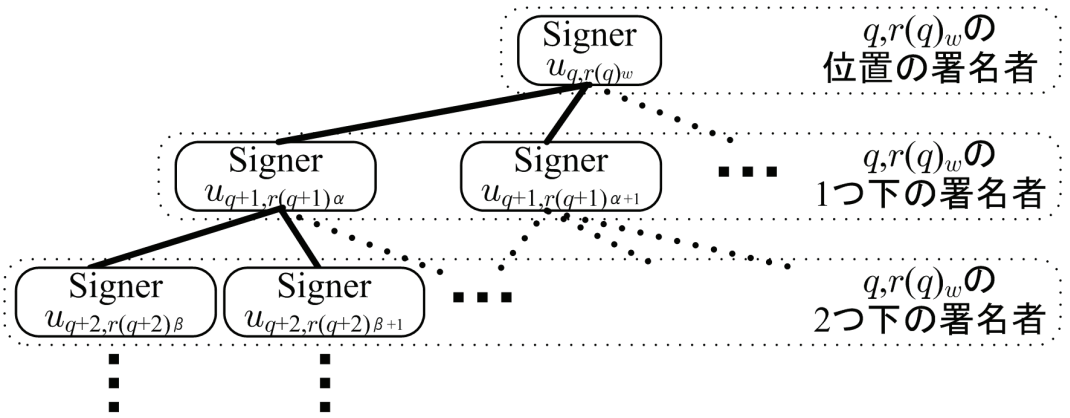

図 3.3: $\mathbb{L}_{q,r(q)_w}$ が示す署名者の関係

プロトコル

鍵生成 $g \in \mathbb{G}_1$ を生成元とする．署名者 $u_{q,r(q)_w}$ について，

$x_{q,r(q)_w} \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_{q,r(q)_w} = x_{q,r(q)_w} g$ を計算する．$x_{q,r(q)_w}$ を署名者 $u_{q,r(q)_w}$ の署名鍵，$v_{q,r(q)_w}$ を署名者 $u_{q,r(q)_w}$ の検証鍵とする．

署名作成 以下の手順で，アグリゲート署名が作成される．

1. リーフノードに位置する署名者 $u_{l,r(l)_{\alpha 0}}$ は，メッセージ $m_{l,r(l)_{\alpha 0}}$ から $h_{l,r(l)_{\alpha 0}} = H(m_{l,r(l)_{\alpha 0}})$ を求め， 2.3.3 節で説明した BLS 署名と同様な署名作成処理を行うことで

$$\sigma_{l,r(l)_{\alpha 0}} = x_{l,r(l)_{\alpha 0}} h_{l,r(l)_{\alpha 0}} \tag{3.57}$$

を計算する．また，

$$\mathbb{L}_{l,r(l)_{\alpha 0}} = (\mathrm{null}, \{u_{l,r(l)_{\alpha 0}}\}) \tag{3.58}$$

を作成する．この $\sigma_{l,r(l)_{\alpha 0}}$，$\mathbb{L}_{l,r(l)_{\alpha 0}}$，および $m_{l,r(l)_{\alpha 0}}$ (または $h_{l,r(l)_{\alpha 0}}$) を自分の親ノードに位置する署名者 $u_{l-1,r(l-1)_{\beta 0}}$ に送信する．

2. 中間ノードに位置する（リーフノードとルート以外の）署名者 $u_{s,r(s)_\beta}$ は，自分の子ノードに位置する署名者 $u_{s+1,r(s+1)_{\gamma_1}}, \ldots, u_{s+1,r(s+1)_{\gamma_{k'}}}$ から受信した全ての $m_{s+1,r(s+1)_i}$ (但し $\gamma_1 \le i \le \gamma_{k'}$) を用いて $h_{s+1,r(s+1)_i} = H(m_{s+1,r(s+1)_i})$ を求める（または $h_{s+1,r(s+1)_i}$ を受信したら，それを利用する）．さらに署名者 $u_{s,r(s)_\beta}$ は，自分が本来署名したいメッセージ $m_{s,r(s)_\beta}$ から $h_{s,r(s)_\beta} = H(m_{s,r(s)_\beta})$ を求める．これと全ての子ノードの

署名者 $u_{s+1,r(s+1)_i}$ から受信した $\sigma_{s+1,r(s+1)_i}$ を用いて

$$\sigma_{s,r(s)_\beta} = \sum_{i=\gamma_1}^{\gamma_{k'}} (\sigma_{s+1,r(s+1)_i} + x_{s,r(s)_\beta} h_{s+1,r(s+1)_i}) + x_{s,r(s)_\beta} h_{s,r(s)_\beta} \tag{3.59}$$

を計算する．また，

$$\mathbb{L}_{s,r(s)_\beta} = \sum_{i=\gamma_1}^{\gamma_{k'}} \mathbb{L}_{s+1,r(s+1)_i} + \{(u_{s,r(s)_\beta}, \{u_{s+1,r(s+1)_1}, \ldots, u_{s+1,r(s+1)_{k'}}\})\} \tag{3.60}$$

を作成する．この $\sigma_{s,r(s)_\beta}$，$\mathbb{L}_{s,r(s)_\beta}$，および $m_{s,r(s)_\beta}$(または $h_{s,r(s)_\beta}$) を自分の親ノードに位置する署名者 $u_{s-1,r(s-1)_{\beta'}}$ に送信する．この手順をルートの子ノードに位置する署名者まで再帰的に行う．

3. ルートに位置する署名者 $u_{1,r(1)_1}$ は，自分の子ノードに位置する署名者 $u_{2,r(2)_1}, \ldots, u_{2,r(2)_\delta}$ から受信した全ての $m_{2,r(2)_i}$（但し $1 \leq i \leq \delta$）を用いて $h_{2,r(2)_i} = H(m_{2,r(2)_i})$ を求める（または $h_{2,r(2)_i}$ を受信したら，それを利用する）．さらに署名者 $u_{1,r(1)_1}$ は，自分が本来署名したいメッセージ $m_{1,r(1)_1}$ から $h_{1,r(1)_1} = H(m_{1,r(1)_1})$ を求める．これと全ての子ノードの署名者 $u_{2,r(2)_i}$ から受信した $\sigma_{2,r(2)_i}$ を用いて

$$\sigma_{1,r(1)_1} = \sum_{i=1}^{\delta} (\sigma_{2,r(2)_i} + x_{1,r(1)_1} h_{2,r(2)_i}) + x_{1,r(1)_1} h_{1,r(1)_1} \tag{3.61}$$

を計算する．また，

$$\mathbb{L}_{1,r(1)_1} = \sum_{i=1}^{\delta} \mathbb{L}_{2,r(2)_i} + \{(u_{1,r(1)_1}, \{u_{2,r(2)_1}, \ldots, u_{2,r(2)_\delta}\})\} \tag{3.62}$$

を作成する．この $\sigma_{1,r(1)_1}$，および $\mathbb{L}_{1,r(1)_1}$ を，全署名者 $u_{q,r(q)_w}$ の，全署名対象 $m_{q,r(q)_w}$ に対するアグリゲート署名として公開する．

署名検証 以下の手順で，アグリゲート署名の検証を行う．

1. 検証者は，$\mathbb{L}_{1,r(1)_1}$ に示されている全ての署名者の検証鍵 $v_{q,r(q)_w}$，および署名対象となる全てのメッセージ $m_{q,r(q)_w}$ を集める．
2. 検証者は，集めたメッセージから $h_{q,r(q)_w} = H(m_{q,r(q)_w})$ を求める．

3. 検証者はペアリングを用いて以下の判定を行う.

$$\begin{aligned} & e(g, \sigma_{1,r(1)_1}) \\ =& e(v_{1,r(1)_1}, h_{1,r(1)_1}) \cdot \prod_{q,i,j \in \mathbf{All}} e(v_{q,r(q)_i} + v_{q+1,r(q+1)_j}, h_{q+1,r(q+1)_j}) \end{aligned} \tag{3.63}$$

（但し，**All** とはアグリゲート署名における木構造の直接の親子関係になっている箇所全て（の署名者）を示す.）

もし正しく署名が作成されていれば，左辺は

$$\begin{aligned} e(g, \sigma_{1,r(1)_1}) =& e(g, \sum_{i=1}^{\delta} (\sigma_{2,r(2)_i} + x_{1,r(1)_1} h_{2,r(2)_i}) + x_{1,r(1)_1} h_{1,r(1)_1}) \\ =& e(g, x_{1,r(1)_1} h_{1,r(1)_1}) \cdot e(g, \sum_{i=1}^{\delta} (\sigma_{2,r(2)_i} + x_{1,r(1)_1} h_{2,r(2)_i})) \\ & \cdots \\ =& e(g, x_{1,r(1)_1} h_{1,r(1)_1}) \cdot e(g, \sum_{q,i,j \in \mathbf{All}} ((x_{q,r(q)_i} + x_{q+1,r(q+1)_j} h_{q+1,r(q+1)_j})) \\ =& e(x_{1,r(1)_1} g, h_{1,r(1)_1}) \cdot \prod_{q,i,j \in \mathbf{All}} e((x_{q,r(q)_i} + x_{q+1,r(q+1)_j}) g, h_{q+1,r(q+1)_j}) \end{aligned} \tag{3.64}$$

となり，右辺と一致する.

アグリゲート署名への新たな署名者の追加 ルートの署名者 $u_{1,r(1)_1}$ は $\sigma_{1,r(1)_1}$，および $\mathbb{L}_{1,r(1)_1}$ を全署名対象 $m_{q,r(q)_w}$ に対するアグリゲート署名として公開している.

ここで，新たな署名者が自分を追加して新たなアグリゲート署名を作成する場合を考える．ここでは便宜上，新たな署名者を $u_{0,r(0)_1}$，この署名者 $u_{0,r(0)_1}$ が自分自身を追加させるアグリゲート署名の最終署名者を $u_{1,r(1)_1}, \ldots, u_{1,r(1)_\epsilon}$ とする．すなわち，署名者 $u_{0,r(0)_1}$ は $\sigma_{1,r(1)_1}, \ldots, \sigma_{1,r(1)_\epsilon}$，および $\mathbb{L}_{1,r(1)_1}, \ldots, \mathbb{L}_{1,r(1)_\epsilon}$ に対し，自分自身を追加した新たなアグリゲート署名を作成する.

この時，署名者 $u_{1,r(1)_i}$（但し $1 \leq i \leq \epsilon$）は作成したアグリゲート署名（$\sigma_{1,r(1)_i}, \mathbb{L}_{1,r(1)_i}$），および $m_{1,r(1)_i}$（または $h_{1,r(1)_i}$）を新たな署名者 $u_{0,r(0)_1}$ に送信し，署名者 $u_{0,r(0)_1}$ が署名作成の 3 の手順を行う．すなわち署名者 $u_{0,r(0)_1}$ は

$$\sigma_{0,r(0)_1} = \sum_{i=1}^{\epsilon} (\sigma_{1,r(1)_i} + x_{0,r(0)_1} h_{1,r(1)_i}) + x_{0,r(0)_1} h_{0,r(0)_1} \tag{3.65}$$

を計算し，

$$\mathbb{L}_{0,r(0)_1} = \sum_{i=1}^{\epsilon} \mathbb{L}_{1,r(1)_i} + \{(u_{0,r(0)_1}, \{u_{1,r(1)_1}, \ldots, u_{1,r(1)_\epsilon}\})\} \tag{3.66}$$

を作成する．この $\sigma_{0,r(0)_1}$，および $\mathbb{L}_{0,r(0)_1}$ を，新たなアグリゲート署名として公開する．

この手順で示したように，新たな署名者が公開されているアグリゲート署名に自分を追加した新たなアグリゲート署名を作成する場合，それまでのルートの署名者は新たな処理を必要とせず，公開されている情報を新たな署名者に送信するだけで良い．

3.3.3 木構造表記型アグリゲート署名を用いた引用過程表記

引用過程表記の概要

3.3.2 節で，任意の構造に対応した木構造表記型アグリゲート署名プロトコルについて説明した．

ここで，CGM サービスでマッシュアップによるコンテンツの引用を行った際に，XML を拡張したものや，クリエイティブ・コモンズで使用されている RDF（正確には RDF を拡張した梶らによる方式 [9,10]）といったメタデータの記載手法を用いて，著作権情報（ライセンス情報）の 1 つとしてそのコンテンツの引用過程を記載し，引用・編集されたコンテンツと併せて提供することになる．このメタデータに対し，そのメタデータに対応するコンテンツ（のハッシュ値）と併せて 1 つのメッセージとみなし，それに署名を付けることでメタデータの内容を保証することができる．この保証について，マッシュアップされるごとにその引用過程を木構造表記型アグリゲート署名として表記し，その電子署名の検証によりコンテンツの引用過程を確認できるようにする目的で，提案方式を適用する．

3.3.3 節では，このアグリゲート署名を用いたコンテンツ引用過程表記の手順について，図 3.4 に示す 2 分木 3 階層の場合を例に取り説明する．

プロトコル

鍵生成 3.3.2 節の手順で $u_{1,r(1)_1}, u_{2,r(2)_1}, u_{2,r(2)_2}, u_{3,r(3)_1}, u_{3,r(3)_2}, u_{3,r(3)_3}, u_{3,r(3)_4}$ の署名鍵，および検証鍵を生成する．

署名作成 以下の手順で，アグリゲート署名が作成される．

1. オリジナルコンテンツ作成者（第 3 層の署名者）$u_{3,r(3)_1}$ は，メッセージ $m_{3,r(3)_1}$ か

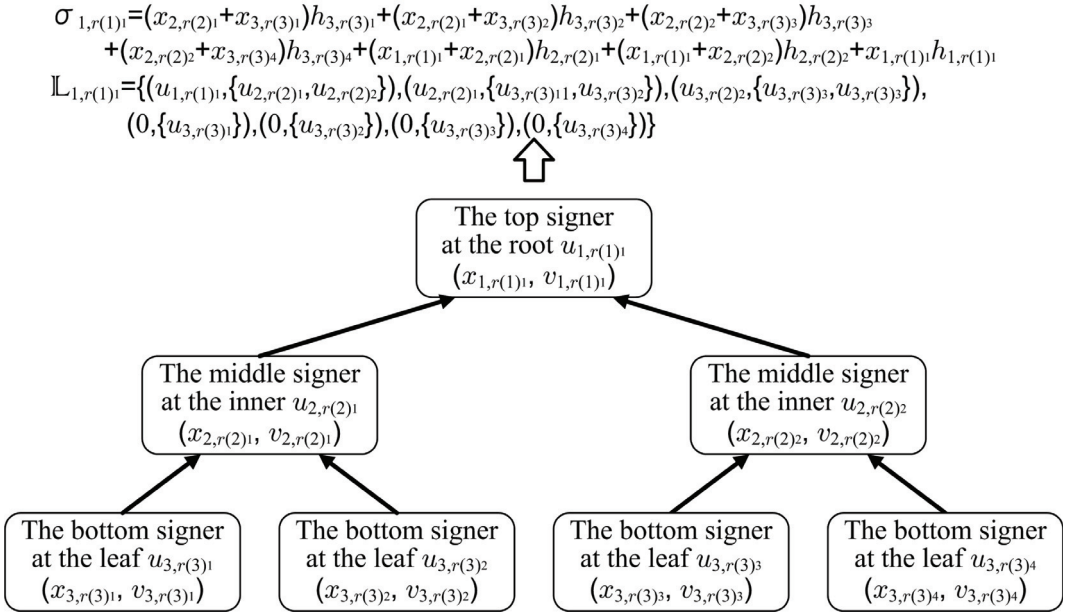

図 3.4: 2 分木 3 階層におけるコンテンツ引用過程

ら $h_{3,r(3)_1} = H(m_{3,r(3)_1})$ を求め，

$$\sigma_{3,r(3)_1} = x_{3.r(3)_1} h_{3.r(3)_1} \tag{3.67}$$

を計算する．また，

$$\mathbb{L}_{3,r(3)_1} = (\text{null}, \{u_{3,r(3)_1}\}) \tag{3.68}$$

を作成する．この $\sigma_{3,r(3)_1}$，$\mathbb{L}_{3,r(3)_1}$，および $m_{3,r(3)_1}$ (または $h_{3,r(3)_1}$) を自分の親ノードに位置する中間編集者 $u_{2,r(2)_1}$ に送信する．第 3 層の他の作成者も同様の手順により署名を作成し，$u_{3,r(3)_2}$ は $u_{2,r(2)_1}$ へ，$u_{3,r(3)_3}$，および $u_{3,r(3)_4}$ は $u_{2,r(2)_2}$ へ送信する．

2. 中間編集者（第 2 層の署名者）$u_{2,r(2)_1}$ は，自分の子ノードに位置する 2 人のオリジナルコンテンツ作成者 $u_{3,r(3)_1}, u_{3,r(3)_2}$ から受信した $m_{3,r(3)_1}, m_{3,r(3)_2}$ を用いて $h_{3,r(3)_1} = H(m_{3,r(3)_1}), h_{3,r(3)_2} = H(m_{3,r(3)_2})$ を求める（または $h_{3,r(3)_1}, h_{3,r(3)_2}$ を受信したら，それを利用する）．さらに中間編集者 $u_{2,r(2)_1}$ は，自分が本来署名したいメッセージ $m_{2,r(2)_1}$ から $h_{2,r(2)_1} = H(m_{2,r(2)_1})$ を求める．これと $u_{3,r(3)_1}$，および $u_{3,r(3)_2}$ から受信した $\sigma_{3,r(3)_1}$，および $\sigma_{3,r(3)_2}$ を用いて

$$\sigma_{2,r(2)_1} = \sigma_{3,r(3)_1} + x_{2,r(2)_1} h_{3.r(3)_1} + \sigma_{3,r(3)_2} + x_{2,r(2)_1} h_{3,r(3)_2} + x_{2,r(2)_1} h_{2,r(2)_1} \tag{3.69}$$

を計算する．また，

$$\mathbb{L}_{2,r(2)_1} = \{(u_{2,r(2)_1}, \{u_{3,r(3)_1}, u_{3,r(3)_2}\}), (\text{null}, \{u_{3,r(3)_1}\}), (\text{null}, \{u_{3,r(3)_2}\})\} \tag{3.70}$$

を作成する．この $\sigma_{2,r(2)_1}$，$\mathbb{L}_{2,r(2)_1}$，および $m_{2,r(2)_1}$ (または $h_{2,r(2)_1}$) をルートに位置する最終編集者 $u_{1,r(1)_1}$ に送信する．第 2 層のもう 1 人の中間編集者 $u_{2,r(2)_2}$ も同様の手順により署名と付随情報を作成し，$u_{1,r(2)_1}$ へ送信する．

3. 最終編集者（ルートに位置する署名者）$u_{1,r(1)_1}$ は，自分の子ノードに位置する 2 人の中間編集者 $u_{2,r(2)_1}, u_{2,r(2)_2}$ から受信した $m_{2,r(2)_1}, m_{2,r(2)_2}$ を用いて $h_{2,r(2)_1} = H(m_{2,r(2)_1}), h_{2,r(2)_2} = H(m_{2,r(2)_2})$ を求める（または $h_{2,r(2)_1}, h_{2,r(2)_2}$ を受信したら，それを利用する）．さらに最終編集者 $u_{1,r(1)_1}$ は，自分が本来署名したいメッセージ $m_{1,r(1)_1}$ から $h_{1,r(1)_1} = H(m_{1,r(1)_1})$ を求める．これと中間編集者 $u_{2,r(2)_1}$，および $u_{2,r(2)_2}$ から受信した $\sigma_{2,r(2)_1}$，および $\sigma_{2,r(2)_2}$ を用いて

$$\sigma_{1,r(1)_1} = \sigma_{2,r(2)_1} + x_{1,r(1)_1} h_{2,r(2)_1} + \sigma_{2,r(2)_2} + x_{1,r(1)_1} h_{2,r(2)_2} + x_{1,r(1)_1} h_{1,r(1)_1} \tag{3.71}$$

を計算する．また，

$$\begin{aligned}\mathbb{L}_{1,r(1)_1} = \{&(u_{1,r(1)_1}, \{u_{2,r(2)_1}, u_{2,r(2)_2}\}), (u_{2,r(2)_1}, \{u_{3,r(3)_1}, u_{3,r(3)_2}\}),\\ &(u_{2,r(2)_2}, \{u_{3,r(3)_3}, u_{3,r(3)_4}\}), (\text{null}, \{u_{3,r(3)_1}\}), (\text{null}, \{u_{3,r(3)_2}\}),\\ &(\text{null}, \{u_{3,r(3)_3}\}), (\text{null}, \{u_{3,r(3)_4}\})\}\end{aligned} \tag{3.72}$$

を作成する．この $\sigma_{1,r(1)_1}$，および $\mathbb{L}_{1,r(1)_1}$ を，コンテンツ作成に関与した全員のアグリゲート署名として公開する．

署名検証 以下の手順で，アグリゲート署名の検証を行う．

1. 検証者は，$\mathbb{L}_{1,r(1)_1}$ に示されている全ての署名者の検証鍵 $v_{1,r(1)_1}, v_{2,r(2)_1} v_{2,r(2)_2}, v_{3,r(3)_1}, v_{3,r(3)_2}, v_{3,r(3)_3}, v_{3,r(3)_4}$，および署名対象となる全てのメッセージ $m_{1,r(1)_1}, m_{2,r(2)_1}, m_{2,r(2)_2}, m_{3,r(3)_1}, m_{3,r(3)_2}, m_{3,r(3)_3}, m_{3,r(3)_4}$ を集める．
2. 検証者は，集めたメッセージから $h_{1,r(1)_1}, h_{2,r(2)_1}, h_{2,r(2)_2}, h_{3,r(3)_1}, h_{3,r(3)_2}, h_{3,r(3)_3}, h_{3,r(3)_4}$ を求める．

3. 検証者はペアリングを用いて以下の判定を行う．

$$
\begin{aligned}
& e(g, \sigma_{1,r(1)_1}) \\
=& e(v_{1,r(1)_1}, h_{1,r(1)_1}) \cdot e(v_{1,r(1)_1} + v_{2,r(2)_1}, h_{2,r(2)_1}) \\
& \cdot e(v_{1,r(1)_1} + v_{2,r(2)_2}, h_{2,r(2)_2}) \cdot e(v_{2,r(2)_1} + v_{3,r(3)_1}, h_{3,r(3)_1}) \\
& \cdot e(v_{2,r(2)_1} + v_{3,r(3)_2}, h_{3,r(3)_2}) \cdot e(v_{2,r(2)_2} + v_{3,r(3)_3}, h_{3,r(3)_3}) \\
& \cdot e(v_{2,r(2)_2} + v_{3,r(3)_4}, h_{3,r(3)_4})
\end{aligned} \tag{3.73}
$$

もし正しく署名が作成されていれば，3.3.2 節における署名検証の手順 3 と同様に，左辺は右辺と一致する．

3.3.4 安全性の考察

アグリゲート署名正当性（存在的偽造不能性） 提案方式では 3.3.2 節で示したように，アグリゲート署名のセット（$\sigma_{1,r(1)_1}$ と $\mathbb{L}_{1,r(1)_1}$ のセット）を出力する．このセットの中の $\mathbb{L}_{1,r(1)_1}$ によって隣接する署名者の関係が示され，それに従い検証鍵と対応するメッセージ（のハッシュ値）をペアリング関数に順次入力していくことで，アグリゲート署名 $\sigma_{1,r(1)_1}$ を検証する手順となっている．この時のアグリゲート署名の安全性について考察する．

3.3.2 節で提案した方式において，攻撃者 A は偽装を行うある一人の（リーフノードの）署名者の検証鍵 $v'_{l_\alpha,r(l_\alpha)_1}$ を入力値とした時に，想定している木構造に応じて残りのノードの署名鍵と検証鍵のペアの出力を行うものとする．A の攻撃が成功するとは，上記の入力値，および出力した鍵ペアに対し，想定した木構造に対応するアグリゲート署名 $\sigma'_{1,r(1)_1}$ を出力できること，すなわち $\sigma'_{1,r(1)_1}$ の偽造に成功することを意味する．ここでは，Boldyreba が行った手法 [28,29] を用いて，$\sigma'_{1,r(1)_1}$ の安全性を証明する．

定理 3 ランダムオラクルモデルにおいて，*3.3.2* 節で提案した方式における $\sigma'_{1,r(1)_1}$ の偽造可能性と *BLS* 署名の偽造可能性は等価である．

証明 3 A を $\sigma'_{1,r(1)_1}$ を偽造しようとする攻撃者とする．B を *BLS* 署名を偽造する攻撃者とした時，A の攻撃が成功するなら，B の攻撃が成功することを示す（B の攻撃が成功するなら，A の攻撃が成功することは自明であるため省略する）．

B は一つの検証鍵 $v'_{l_\alpha,r(l_\alpha)_1}$ を持っており，ランダムオラクル，および署名オラクルへの応答を行う．

B は A を *Honest Player* として実行する．まず B は A に $v'_{l_\alpha,r(l_\alpha)_1}$ を与え，A はそれ以外のリーフノードの署名鍵と検証鍵のペア $(x'_{l_\alpha,r(l_\alpha)_\beta}, v'_{l_\alpha,r(l_\alpha)_\beta})$，およびリーフノード以外のノードの署名鍵と検証鍵のペア $(x'_{q,r(q)_w}, v'_{q,r(q)_w})$ を出力する．また，A はランダムオラクルと署名オラクルへの応答により $\sigma'_{1,r(1)_1}$ とその署名の対象となるメッセージ $m'_{q,r(q)_w}$（偽造ノードを含む全ノード分）を求め，B に返答する．B は $\sigma'_{1,r(1)_1}$ を用いて，

$$\begin{aligned}&\sigma'_{1,r(1)_1} - \sum_{i,j,k\in \textbf{\textit{All'}}} \left((x'_{q_i,r(q_i)_j} + x'_{q_i,r(q_i)_j})H(m'_{q_i+1,r(q_i+1)_k})\right) \\ &\qquad - x'_{1,r(1)_1}H(m'_{1,r(1)_1}) - x'_{l_\alpha+1,r(l_\alpha+1)_\gamma}H(m'_{l_\alpha,r(l_\alpha)_1}) \\ =&x'_{l_\alpha,r(l_\alpha)_1}H(m'_{l_\alpha,r(l_\alpha)_1}) \end{aligned} \tag{3.74}$$

（但し，***All'*** とはアグリゲート署名における木構造の直接の親子関係になっている箇所全て（の署名者）のうち，$x'_{l_\alpha,r(l_\alpha)_1}$ が含まれている箇所を除外したものを示す）
の計算により，$v'_{l_\alpha,r(l_\alpha)_1}$ に対応する *BLS* 署名を作成することができ，B の攻撃が成功する．

以上により，定理 3 が成立する． ■

参加否認不能性 隣接する 2 人の署名者 $u_{q_i,r(q_i)_\alpha}, u_{q_i+1,r(q_i+1)_\beta}$ の関係を示すために，前者が署名対象にしているメッセージのハッシュ値 $H(m_{q_i+1,r(q_i+1)_\beta})$ に対し，この 2 人の署名者が自分の署名鍵 $x_{q_i,r(q_i)_\alpha}, x_{q_i+1,r(q_i+1)_\beta}$ を用いて $(x_{q_i,r(q_i)_\alpha}+x_{q_i+1,r(q_i+1)_\beta})H(m_{q_i+1,r(q_i+1)_\beta})$ を計算している．この時，この 2 人の署名者のものではない異なる署名鍵 $x'_{q_i,r(q_i)_\alpha}, x'_{q_i+1,r(q_i+1)_\beta}$ を用いて $x_{q_i,r(q_i)_\alpha}+x_{q_i+1,r(q_i+1)_\beta} = x'_{q_i,r(q_i)_\alpha}+x'_{q_i+1,r(q_i+1)_\beta}$ が成り立つなら，$u_{q_i,r(q_i)_\alpha}, u_{q_i+1,r(q_i+1)_\beta}$ は自分らが署名に参加したのではなく $x'_{q_i,r(q_i)_\alpha}, x'_{q_i+1,r(q_i+1)_\beta}$ を署名鍵として保有している署名者が参加したという主張を許すことになり，署名参加の否認が可能となる．

上記の結託攻撃に対しては，いずれの署名鍵も $\mathbb{Z}_p^*$ の要素であることから，どの 2 個の要素を選んでも和の値が異なるような鍵空間 $\mathbb{K} \subset \mathbb{Z}_p^*$ を準備し，署名鍵が $\mathbb{K}$ の要素となるようにすることで，回避することが可能となる．このような $\mathbb{K}$ を構成する単純な例としては，全署名者の署名鍵をその値が小さい順に並べた時，超増加列（各数がそれまでの整数の和よりも大きくなる数列）になっている場合が考えられる．

超増加列が $\mathbb{K}$ を構成できることの証明 $\mathbb{K}' \subset \mathbb{Z}_p^*$ となる $\mathbb{K}'$ の要素を小さい順から並べると，超増加列になっているものとする．任意の 2 個の鍵 $k_m, k_n \in \mathbb{K}'$（但し $k_m < k_n$）を選んだ時，この鍵の和 $k_m + k_n$ と，少なくともどちらか 1 個が違う鍵の時の和の値とを比較する．比較対象の鍵の値が 2 個とも k_n より小さい場合，超増加列の「k_n より小さい全ての鍵の値の総和を求めても k_n より小さい」という性質から，k_n より小さい 2 個の鍵の和は $k_m + k_n$ より小さく，値は一致しない．また，

表 3.1: 署名のデータサイズコスト

	署名サイズ	中間鍵サイズ
一般的な順序付き多重署名	$w_o Q_s$	0
提案方式（多重署名）	Q_s	Q_v
提案方式（アグリゲート署名）	Q_s	0

1 個の鍵が k_n より大きい場合（仮に $k_o > k_n$ と置く），超増加列の「k_o より小さい全ての鍵の値の総和を求めても k_o より小さい」という性質から，$k_m + k_n < k_o$ が成り立つため，$k_m + k_n$ は k_o を含んだ 2 個の鍵の和の値とも一致しない．したがって，$\mathbb{K}'$ は鍵空間 $\mathbb{K}$ の条件を満たす． ■

さらに巡回群において効率よく $\mathbb{K}$ を構成できる（$\mathbb{K}$ の要素数を，単純な超増加列の時よりも多くすることができる）値の組み合わせに関する理論が提唱されている [83].

偽装参加不能性 偽装を行おうとする第三者の署名鍵を $x_{q_{\mathrm{not}},r(q_{\mathrm{not}})_\gamma}$ とする．この時，ある署名参加者 $u_{q_i,r(q_i)_\alpha}$ が任意のメッセージ $m_{q_{\mathrm{not}},r(q_{\mathrm{not}})_\gamma}$，およびこのメッセージに対する第三者の BLS 署名 $x_{q_{\mathrm{not}},r(q_{\mathrm{not}})_\gamma} H(m_{q_{\mathrm{not}},r(q_{\mathrm{not}})_\gamma})$ を入手し，ルートに位置する署名者 $u_{1,r(1)_1}$ までの全ての手順が終了した後に $(x_{q_i,r(q_i)_\alpha} + x_{q_{\mathrm{not}},r(q_{\mathrm{not}})_\gamma}) H(m_{q_{\mathrm{not}},r(q_{\mathrm{not}})_\gamma})$ を計算すれば，$\sigma_{1,r(1)_1}$ に加算することでこの第三者を勝手に署名者として追加する，すなわち成りすますことが可能となる．

上記の攻撃に対しては，各署名者が自分より下位の署名者の署名対象のメッセージ（の一部）を自分の署名対象のメッセージに追記しておくといった回避策で対抗することが可能となる．

3.4 コストの考察

3.4.1 署名のデータサイズコスト

3.2 節で提案した階層表記型多重署名と 3.3 節で提案した階層表記型アグリゲート署名について，データサイズコストの評価を行った．その評価結果を表 3.1 に示す．

ここで，以下の値を定義する．

w_o**:** リーフノードの総数.

Q_s**:** ベースとなる電子署名のデータサイズ.

Q_v**:** 電子署名における検証鍵のデータサイズ.

表 3.2: 計算コスト

	署名作成（1 ユーザあたり）	署名検証
提案方式（多重署名）	$4w_lC_a + (2w_l + 1)C_b$	$(w_s + 3)C_c$
提案方式（アグリゲート署名）	$2w_lC_a + (w_l + 1)C_b$	$(w_s + 1)C_c$

既存の順序付き多重署名方式は，一列で構成される署名者の関係のみが表記できる．したがって，木構造のように枝分かれしている，あるいは表記する必要がある順序列が複数存在する場合，そのルートが構成される（木構造ならリーフノードの）数だけ多重署名の数も増加する．

一方，本研究における提案方式では，どのような構造であっても作成される電子署名や中間鍵は高々一つである．したがって，署名者の数に関係なくデータサイズは一定値（階層表記型多重署名の場合は中間鍵が発行されるので $Q_s + Q_v$，階層表記型アグリゲート署名の場合は中間鍵が発行されるないので Q_s のデータサイズ）となる．

以上により，データサイズコストについては，提案方式は既存方式より優位であると言える．

3.4.2 計算コスト

3.2 節で提案した階層表記型多重署名と 3.3 節で提案した階層表記型アグリゲート署名について，計算コストの評価を行った．その評価結果を表 3.2 に示す．ただし，一般的な順序付き多重署名方式は RSA 署名方式や ElGamal 署名方式をベースにしており，楕円曲線上の演算やペアリングを用いる BLS 署名方式とは異なっているため，3.4.1 節で行ったような単純なコスト比較は行えないしたがって，既存の順序付き多重署名方式との比較は行わない．

ここで，以下の値を定義する．

w_l**:** 2 人の署名者の接続を表すリンク数.

w_s**:** 全署名者数.

C_a**:** 楕円曲線上における加算の計算コスト.

C_b**:** 楕円曲線上におけるスカラー倍の計算コスト.

C_c**:** ペアリングの計算コスト.

署名作成時においては，階層表記型多重署名方式は階層表記型アグリゲート署名方式の約 2 倍の計算コストとなっている．これは，階層表記型多重署名方式は署名の作成と中間鍵の

作成を行っており，この二つは似た構造を持つことから約 2 倍の計算コストが必要となるためである．

一方，署名検証時においては，全署名者数に関係なく階層表記型多重署名方式は階層表記型アグリゲート署名方式より 2 回だけペアリングの計算コストが多い．これは，階層表記型多重署名方式における中間鍵の正当性検証の演算コストと階層表記型アグリゲート署名方式におけるアグリゲート署名検証の演算コストが同じであり，階層表記型多重署名方式における多重署名検証の演算コストを行うのに 2 回のペアリングが必要となるためである．

第4章　変更・削除・追加の可否を著作者が設定できるコンテンツ編集事前制御方式

4.1　編集制御の概要

本章で示す研究成果の目的は，コンテンツの編集操作に関する可否の制御を著作者が事前に設定し，引用して編集する人はその著作者に問い合わせることなく正当な編集操作のみが行えるような方式を提案することである．

ここで，図 4.1 に編集操作の例を示す．本章が対象とする編集操作はいかの 3 種類である．

変更: 存在しているデータの内容を変える．

削除: 存在しているデータを消去する．

追加: データが存在しない箇所にデータを追加する．

また，事前制御とは著作者があらかじめ上記編集操作が可能な箇所を設定しておくことを意味する．

4.2　墨塗り署名

4.2.1　墨塗り署名の概要

墨塗り・削除の事前制御が可能な署名方式として，2.3.5 節で説明したアグリゲート署名に基づいて，宮崎らによる SUMI-6 方式 [37] や泉らによる墨塗り・削除署名方式 (IIKO 方式) [34] が提案されている．しかし，上記の方式は追加の制御までは考慮されていない．一方，齊藤らによる墨塗り署名方式 [33] において，変更・削除・追加のリアルタイムでの制御が可能になっている．この方式では，編集者は自由に更新した部分データを著作者に送り，著作者は受け取ったデータを審査し，署名を付与することによってデータの変更・削除・追加の制御を可能にしている．しかし，著作者が編集のたびに編集制御に介在しているため，事前制御にはなっていない．さらに伊豆らの提案による PIAT 署名方式 [35] では，各更新者がデータ更新部分の情報を付加していくことにより変更・削除・追加制御を可能にしているが，この方式も事前制御は実現されていない．

図 4.1: 編集操作の例

提案方式との比較のため，泉らによる 4.2.2 節で IIKO 方式 [34]，および 4.2.3 節で伊豆らによる PIAT 署名 [35] について説明する．

4.2.2 IIKO 方式

IIKO 方式では，それぞれの部分文書において，以下の制御状態を設定している．

- 墨塗り可能・削除可能（SADA）．
- 墨塗り可能・削除禁止（SADP）．
- 墨塗り禁止・削除可能（SPDA）．
- 墨塗り禁止・削除禁止（SPDP）．
- 墨塗り状態・将来削除可能（SDA）．
- 墨塗り状態・将来削除禁止（SDP）．
- 削除状態（D）．

以下，それぞれの状態における制御について，図 4.2 を用いて説明する．

M_α^* を全体の文書の中の部分文書とし，それぞれにある制御状態が規定されているものとする．この M_α^* のハッシュ値を h_α とする．また，SDA，SDP の状態時には部分文書が墨塗りされているため，対応する M_α^* が明示されておらず，ハッシュ値のみが存在することにな

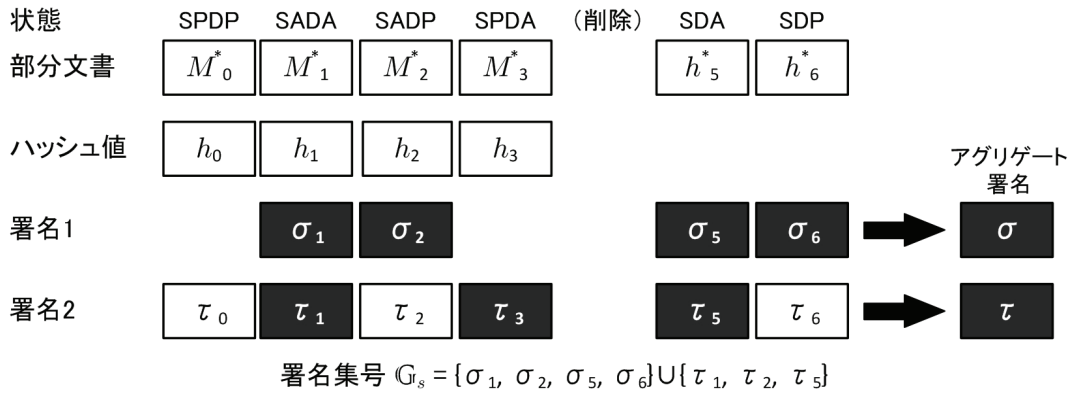

図 4.2: IIKO 方式

り，これを h^*_β で表現する．この M^*_α について，墨塗り可能な，あるいは墨塗り済の部分文書に対する署名を σ_i とし，これを重畳させたアグリゲート署名を σ とする．この時，墨塗りが禁止されている部分文書に対する σ_i は存在せず，また σ にも重畳されていないため，この墨塗りが禁止されている部分文書に不正に墨塗りを行ったとしても，σ によって不正が検出される．

一方，全ての部分文書に対して署名 τ_j を生成し（ただし，$\sigma_{j'} \neq \tau_{j'}$ とする），これを重畳させたアグリゲート署名を τ とする．削除が可能な部分文書に対する τ_{j_0}（図 4.2 における反転部分）は別途提示し，削除が禁止されている部分文書に対する τ_{j_1} は公開しないものとする．対象となる部分文書が削除された場合は，τ からこの τ_{j_0} の値を演算により削除して，演算結果を新たなアグリゲート署名とする．この時，削除が禁止されている部分文書を不正に削除しても，対応する τ_{j_1} が公開されていないため τ から削除することが不可能であり，不正が検出される．

上記の制御を行うのに必要な情報として，墨塗りが可能な部分文書の署名 σ_i，および削除が可能な部分文書の署名 τ_j を要素とする署名集合 $\mathbb{G}_s$ を生成し，これを公開する．

4.2.3 PIAT 署名方式

PIAT 署名方式では，墨塗りを行う者が署名を作成することで，墨塗り情報としてランダムなデータ列の上書きを行うことが可能になり，情報の秘匿のみならず任意のデータ変更が可能になる．以下，RCS 型 PIAT 署名について図 4.3 を用いて説明する．

墨塗りを行う者は複数人いるものとし，(i, j, h_i) はそれぞれ i 番目の部分データが j 番目に墨塗りを行う者によって墨塗りされ，墨塗り以前の部分データのハッシュ値は h_i であったことを意味する．j 番目に墨塗りを行う者は，墨塗り後のデータのハッシュ値の集合 $h^{(j)}_0$，墨

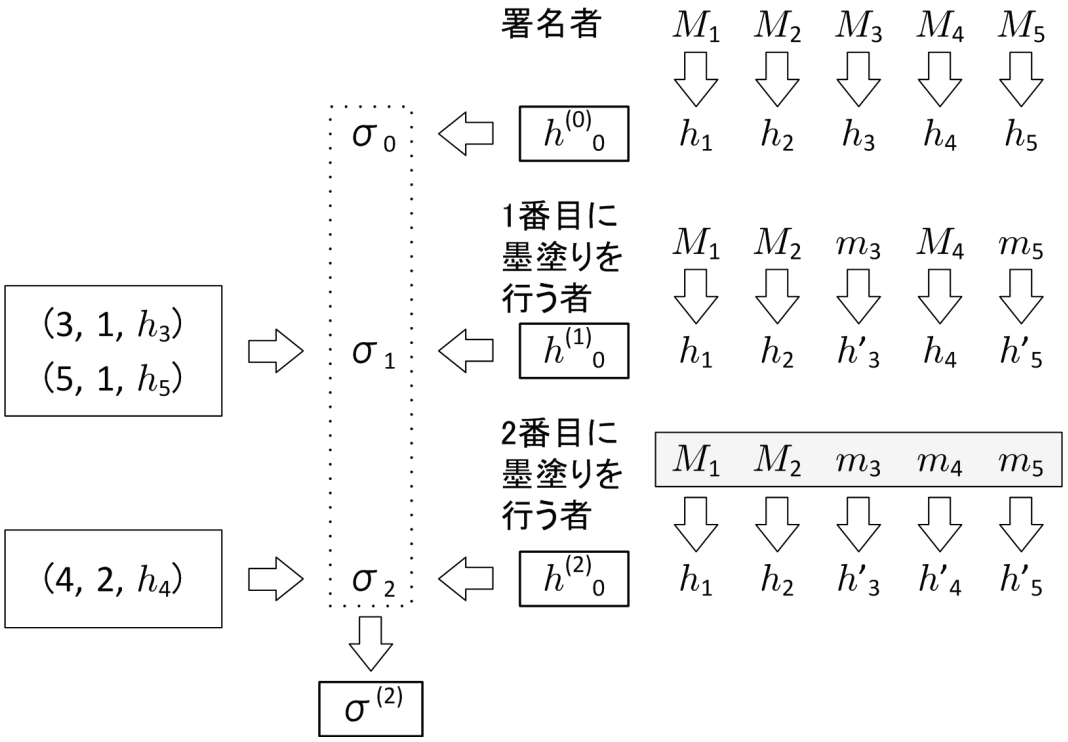

図 4.3: RCS 型 PIAT 署名方式

塗り情報 $S_j = \{(i_1, j, h_{i_1}), \ldots, (i_k, j, h_{i_k})\}$，$j-1$ 番目以前に墨塗りを行った者のアグリゲート署名 $\sigma^{(j-1)}$，$h_0^{(j)}$ と S_j の署名を重畳したアグリゲート署名 $\sigma^{(j)}$，墨塗り後のデータ列を検証者に出力する．検証者は，墨塗り後のデータ列と墨塗り情報を用いて，墨塗りとは逆順に墨塗り情報を復元し，墨塗り後のデータの真正性を検証する．

4.2.4 既存の墨塗り署名の課題

追加制御

データの編集制御において，変更や削除は実在する既存の部分データに対して行われる操作であり，その部分データに対して変更，および削除に関する個別署名を直接設定する．

しかし，追加制御とは部分データ間に新規のデータを追加することに対する許可・禁止を設定する制御のことであり，追加するデータは既存の部分データ中には存在しない．このため，事前に個別署名を設定することが不可能である．もし追加したい位置の前後の部分データに追加の可否を示す制御情報を持たせるとした場合，その前後のデータのどちらか一方（あるいは両方）が削除されてしまうと，その制御情報も削除されてしまうことになり，追加に関する制御が不可能になる．

変更制御

従来の墨塗り署名では，墨塗りの対象となる部分データをそのデータのハッシュ値に変更することで情報の秘匿を行い，さらにそのハッシュ値を署名検証に用いることでデータの正真性を示すこととなる．

しかし，変更制御を考慮した場合，対象とする部分データがそのデータのハッシュ値と対応しないデータへ変更されることになり，署名検証に用いるハッシュ値が保持されていない．このため，変更後に署名の検証を行うことが不可能になる．このことは，CGM サービスを考慮してコンテンツの部分データの変更を想定している編集管理システムへ適用することがが困難であることを意味し，データの変更を考慮したシステムが必要となる．

4.3 コンテンツ編集事前制御方式

4.3.1 提案方式の概要

4.2.4 節において，従来方式では編集管理に必要なコンテンツの部分変更（墨塗りでない任意の変更）や追加までを含む編集作業の事前制御が困難であることを示した．

そこで，従来の墨塗り署名方式を拡張して，変更・削除・追加の事前制御を実現する編集事前制御方式を提案した．また，e-Learning において教育用に用いる動画コンテンツを編集することを想定した，図 4.4 に示す編集制御システムを構築した．図 4.4 において，それぞれのウィンドウは以下を示している．

(1) コンテンツに対する電子署名の作成ボタン（sign,edit,reset）.

(2) 変更・削除に関する部分データの状態設定.

(3) 追加に関する部分データの状態設定.

(4) 作成・編集操作用ウィンドウ.

(5) 編集を行う前のコンテンツデータ.

(6) 追加用のコンテンツデータ.

(7) 編集後のコンテンツデータ.

(8) 変更用の個別署名.

(9) 削除用の個別署名.

(10) 追加用の個別署名.

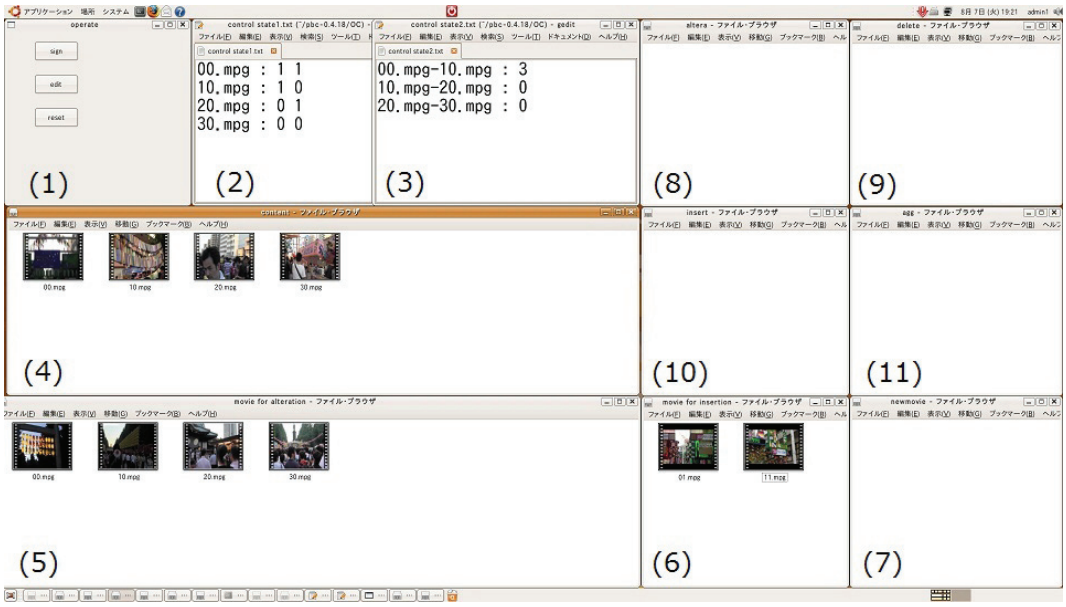

図 4.4: 提案方式の画面

(11) 変更・削除・追加のアグリゲート署名署名.

このシステムにおいて (5)，(6) にあるコンテンツデータを (4) にドラッグ・アンド・ドロップする，あるいは (4) にあるコンテンツデータの消去を行うことで，オリジナルコンテンツの作成，あるいはオリジナルコンテンツへの変更・削除・追加といった各編集作業を行う．さらに上段左のボタンにより作成，あるいは編集されたコンテンツに対する電子署名を作成するシステムとなっている．詳細な操作説明は 4.3.4 節で行う．

本章では，この提案方式について説明する．

4.3.2 準備

エンティティ

提案方式では以下のエンティティを想定する．

著作者（作成者） オリジナルコンテンツを作成し，その編集許諾制御を設定する．

編集者 コンテンツの編集権限を持ち，コンテンツの編集を行う．

検証者 正当な署名が付与されているコンテンツであることを検証する．

著作者は自らの署名鍵と検証鍵を保有し，コンテンツに対する署名を作成することで自分が引用元であることを保証すると同時に，各部に分割された部分データのうち編集（変更・削除・追加）に関する許可・禁止の制御を事前に設定する．コンテンツの編集権限を持つ編集者も自らの署名鍵と検証鍵を保有し，許可された範囲内でデータの編集を行う．検証者は署名検証に必要な著作者と編集者の公開鍵を持ち，検証を行う．検証者の機能をコンテンツ再生機器などに実装し，その再生機器は正当な署名を持たないコンテンツを再生しないものとする．

部分データの制御状態

提案方式では，部分データの制御について以下の状態を定義する．

(a) 変更可能・削除可能（CADA）．

(b) 変更可能・削除禁止（CADP）．

(c) 変更禁止・削除可能（CPDA）．

(d) 変更禁止・削除禁止（CPDP）．

(e) 削除状態（D）．

(f) 追加可能（AA）．

(g) 追加禁止（AP）．

部分データは，既存データと空データの 2 種類を考える．

既存データとは実在する部分データであり，再生機器が再生対象とするコンテンツの部分を構成する．既存データは状態 (a)〜(e) を持つ．ここで， 4.2.2 節に示された状態との比較を行うと，墨塗り状態に対応する変更済の状態が存在しない．これは，一度墨塗りを行うとそれ以上の墨塗り処理が行えず再変更が困難になる従来方式と異なり，提案方式では一度変更した部分データの再変更も考慮したシステムを実現するためである．このことは，編集作業により変更されたデータは，次の編集作業において新たな既存データとして，設定された状態にしたがって再変更が行えることを意味している．ただし，変更したデータを変更済として再変更を禁止する場合は，状態 (c)，あるいは (d) を設定することで対応する．

一方，空データは再生機器が再生対象としないデータのことであり，データの追加を制御するための制御データとして考える．この時，空データは状態 (f)，あるいは (g) を持つ．追加可能状態の空データは編集者による追加データと置き換えることが可能であり，追加されたデータは以後新たな既存データとなり，状態 (a)〜(e) が設定される．

記号

2.3.5 節で説明したアグリゲート署名を前提とし，本章，および付録で使用している記号を以下の通りとする．

M : オリジナルコンテンツデータ（変更前）．

M_i : オリジナルコンテンツの部分データ．

M^* : 配布用コンテンツデータ．

M_i^* : 配布用コンテンツデータの部分データ．

B_j : 追加制御用空データ（j は追加する位置の直前の部分データと直後の部分データの間の位置を示す）．

B_j^* : 空データに対する配布用データ（j は追加する位置の直前の部分データと直後の部分データの間の位置を示す）．

d : 空データ用，および削除したコンテンツの部分データ用のダミーデータ．

N_i : 変更後のコンテンツの部分データ．

N_j : 追加後のコンテンツの部分データ（j は追加する位置の直前の部分データと直後の部分データの間の位置を示す）．

N_i^* : 変更後の再配布用コンテンツの部分データ．

N_j^* : 追加後の再配布用コンテンツの部分データ（j は追加する位置の直前の部分データと直後の部分データの間の位置を示す）．

J : 空データが存在している位置 j の集合（編集後にデータが追加された位置については，J から削除される）．

0^c : 0 の c ビット列（0 が c 個並んだデータ）．

1^c : 1 の c ビット列（1 が c 個並んだデータ）．

σ_i : 変更可能な部分データの変更制御用の個別署名．

τ_i : 削除可能な部分データの削除制御用の個別署名．

ζ_j : 追加可能な空データの追加制御用の個別署名．

ϵ_i : 変更前の部分データと変更後の部分データとの差分データに対する個別署名．

σ : 全ての σ_i を重畳したアグリゲート署名.

τ : 全ての τ_i を重畳したアグリゲート署名.

ζ : 全ての ζ_j を重畳したアグリゲート署名.

ϵ : 全ての ϵ_i を重畳したアグリゲート署名.

ID : コンテンツデータの識別子.

ID_i, ID_j : 部分データの識別子（情報の一部に，データの構成順を示す数値 i，あるいは j を含む）.

$H(\cdot)$: 一方向性ハッシュ関数.

h_i : 配布用部分データのハッシュ値.

h_j : 空データに対する配布用データのハッシュ値.

$||$: 前後の値の連接.

g : 生成元.

s_x : 署名鍵（$x = 0$ なら著作者，$x = 1$ なら編集を行った編集者を示す）.

v_x : 検証鍵（$x = 0$ なら著作者，$x = 1$ なら編集を行った編集者を示す）.

$/$: 除算.

$\backslash$: 差集合（$P \setminus Q$ は，集合 P から集合 Q の要素を削除することを意味する）.

G_d : 削除に対応する署名集合.

G_c : 変更に対応する署名集合.

D_b : 変更前後におけるデータのハッシュ値の差分値の集合（(b) からの変更制御時）.

D_c : 削除前後におけるデータのハッシュ値の差分値の集合（(c) からの変更制御時）.

I_O : 著作者が作成したコンテンツの部分データが存在している位置情報 i の集合.

I_E : 編集者が変更，あるいは追加を行ったコンテンツの部分データの位置情報 i の集合.

I_{D_b} : D_b に出力を行った位置情報 i の集合.

I_{D_c} : D_c に出力を行った位置情報 i の集合.

G_a： 追加に対応する署名集合.

r： 編集履歴データ.

σ_r： 編集履歴データの署名.

$e(\cdot,\cdot)$： ペアリング.

x_k： 検証者が検証時に計算する変更用のハッシュ値.

y_k： 検証者が検証時に計算する削除用のハッシュ値.

X_α： x_k を著作者，および編集者毎に相乗した数値（$\alpha = 0$ なら著作者，$\alpha = 1$ なら編集者）.

Y_α： y_k を著作者，および編集者毎に相乗した数値（$\alpha = 0$ なら著作者，$\alpha = 1$ なら編集者）.

前提条件

提案方式における前提条件を，以下に示す.

- 電子署名の利用に伴い，PKI（Public Key Infrastructure:公開鍵基盤）に基づいたシステムが構築されており，著作者・編集者が持つ署名鍵・検証鍵に対して CA（Certification Authority:認証局）により正当に認証されている（公開鍵証明書が発行されている）.
- 正当な第三者機関によって，コンテンツデータの識別子と付随する情報（著作者情報，作品情報，登録日時など）が管理されている.
- 全てのソフトウェアは，例えば難読化 [84] などの技術によりプログラムの解析は不可能である．すなわち，コンテンツの作成時・編集時における署名作成処理，および検証者による署名作成処理において，全ての処理はソフトウェアの内部で行われ，著作者・編集者・検証者は出力された数値以外の情報の入手は不可能である.
- 追加時に追加できるデータの数は，追加可能な位置 1 つに対し 1 個のみとし，複数のデータは追加できない.

4.3.3 署名作成の概略

図 4.5 に提案方式の部分データに対する署名作成の概略を示す．ここで，図 4.5 の個別署名において実線の枠で囲まれ記号が書かれているものは署名集合に出力される個別署名を表し，記号のないものはアグリゲート署名には含まれるが署名集合には含まれない個別署名を表す．ただし，ϵ_j は署名集合には含まれず，その位置のハッシュ値の差分値が集合に含まれ

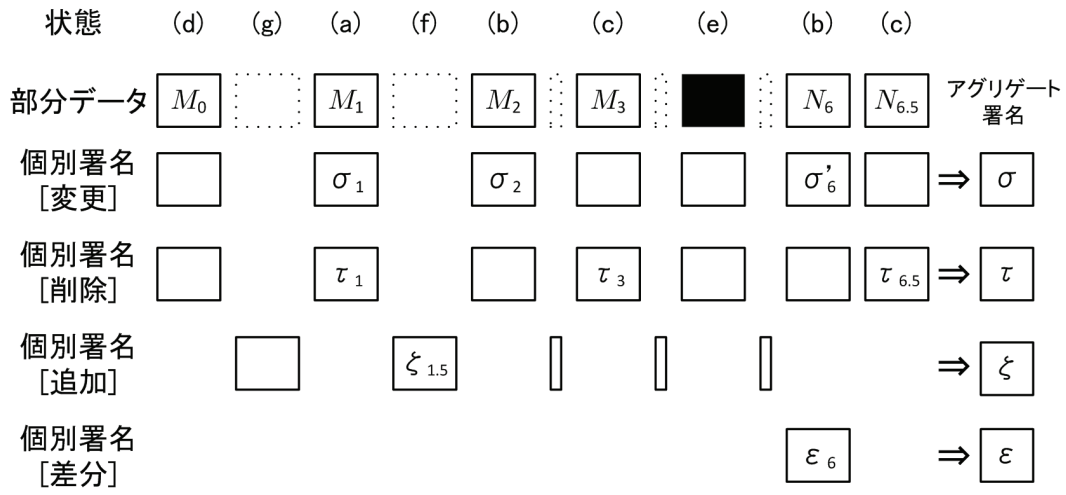

図 4.5: 部分データに対する署名作成の概略

る．また，点線の枠で囲まれた部分は空データを，黒く塗りつぶされた部分は削除されたコンテンツの部分データを表す.

著作者

著作者はコンテンツ M を作成し，同時にコンテンツを編集制御したい領域毎に n 個に分割する．分割された部分データ M_i に対してコンテンツ識別子 ID，部分データ識別子 IDi を連接させ，$h_i = H(ID||ID_i||M_i)$ を計算する．この h_i を 2 つの識別子に連接させ，さらに変更制御用の個別署名には 0 の c ビット列を連接させ $\sigma_i = (H(ID||ID_i||h_i||0^c))^{s_0}$ を，削除制御用の個別署名には 1 の c ビット列を連接させ $\tau_i = (H(ID||ID_i||h_i||1^c))^{s_0}$ をそれぞれ生成する．なお，コンテンツ識別子 ID，および部分データ識別子 ID_i は部分データの繰返し，順序交換，あるいは同一データの署名などの対策として用い，c は公開パラメータとして用いる．

さらに G_d，G_c を用意し，編集を許可する部分データの個別署名だけを各集合に出力する．また，変更・削除に対するアグリゲート署名 σ，および τ を計算する．ただし，σ は変更制御用の個別署名を，τ は削除署名用の個別署名を全て乗算することにより計算される．この処理により，σ，および τ に重畳された個別署名に対応する部分データは，署名集合に個別署名が含まれていなければ除算できないため変更・削除が不可能になる．ただし，このままでは以下のような攻撃が可能になる．例えば，2 つの部分データからなる場合を考え，1 つが変更可，もう 1 つが変更不可とする．また，出力されているアグリゲート署名がこの 2 つの部分データの個別署名の積だけによって構成されているとする．この場合，アグリゲート署名を変更可の部分データの個別署名で除算すると変更不可の部分データに対する個別署名

が計算され，変更不可の部分データの変更が可能になる．この攻撃をさらに一般化すると，単純にコンテンツの部分データからのみで個別署名やアグリゲート署名を構成した場合，G_c に出力された個別署名を全て重畳した値を変更後の新たなアグリゲート署名として，あるいは G_d に出力された個別署名を全て重畳した値を削除後の新たなアグリゲート署名として出力すると，その出力値は個別署名が出力されていない値で σ，および τ を除算した結果と一致し，個別署名が出力されていない部分データの変更・削除を行う不正が可能となる．これを防ぐため，コンテンツ全体を示す ID を基に個別署名 σ_0，τ_0 を作成し，σ，τ にそれぞれ重畳しておく（ただし，σ_0，τ_0 は G_c，G_d には出力しない）．これにより，上記の攻撃があった時にも変更不可の部分データに対する個別署名が露呈しない．また，強引にアグリゲート署名を残った署名で除算すれば σ_0 や τ_0 も除算されていることになり，不正があったことが検出できる．

また，提案方式では各既存データ間には空データがあると想定する．著作者は，各空データを識別できる部分データ識別子を割り振る．ここでは説明のため，ID_i と ID_{i+1} の既存データ間の空データの識別子を $ID_{i+0.5}$ と表示することにする（すなわち，$j = i + 0.5$ とする）．さらに d を連接させて $h_{i+0.5} = H(ID\|ID_{i+0.5}\|d)$ を計算し，$\zeta_{i+0.5} = (H(ID\|ID_{i+0.5}\|h_{i+0.5}\|0^c))^{s_0}$ を生成する．著作者は，その個別署名を乗算し，追加に対するアグリゲート署名 ζ を計算し，追加を許可する空データの位置の個別署名を G_a の要素として出力する．ただし，G_a に出力された個別署名を全て重畳した値で ζ の除算を行うと，追加不可の空データの位置の個別署名を全て重畳した値（仮に ζ' とする）が生成される．これにより，4.3.3 節で説明している追加処理時に追加した位置の個別署名で ζ の除算を行う処理の代わりに，この ζ' で除算することで，追加不可な空データ全ての位置に同時に不正なコンテンツを追加するという攻撃が可能になる．これを防ぐため，仮想的に誰もが追加不可な空データの個別署名 $\zeta_{-0.5}$ を作成し，ζ に重畳しておく．これにより，上記の攻撃があった時には，ζ' と同時に $\zeta_{-0.5}$ の値も除算されていることになり，不正があったことを検出できる．

編集者

編集者が状態 (a) の部分データに対して変更・削除を行う場合を考える．まず，変更に対して編集者は G_c から該当部分の（編集前の）個別署名を削除し，その変更データの再変更を許可する場合は，この編集者が生成した個別署名を追加する．すなわち，編集者は G_c から該当する部分の個別署名を用いてアグリゲート署名 σ に対して除算を行い（この結果をここでは仮に σ' と置く），4.3.3 節に示した手順で変更データに対する個別署名を作成し，この値と σ' との乗算を計算する．ただし，編集者による新たな個別署名の作成には編集者の署名鍵（例えば 1 番目に編集を行った編集者なら s_1）を用いる（以後も同様）．さらに削除に対しては，編集者は G_d から該当部分の（編集前の）個別署名を削除し，その変更データ

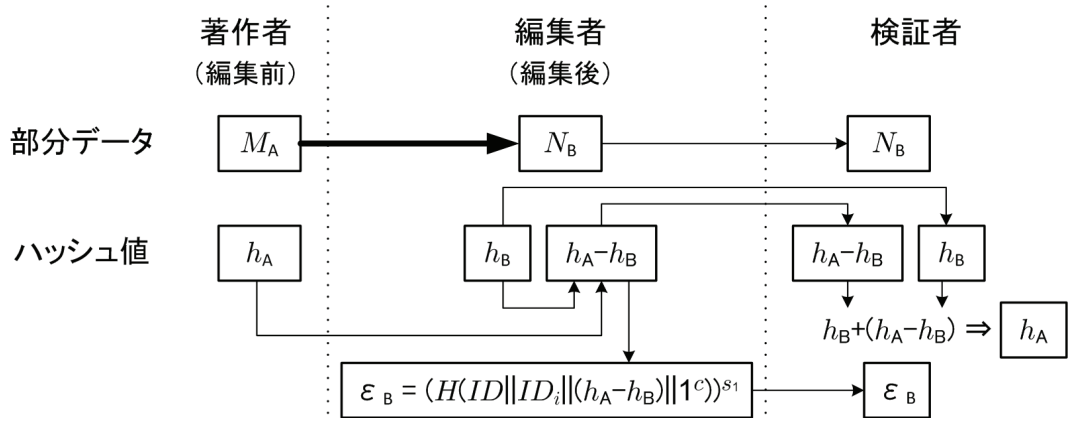

図 4.6: 変更制御時のハッシュ値の差分

の削除を許可する場合は，この編集者が生成した削除制御用の個別署名を追加する．すなわち G_d から該当する部分の削除制御用の個別署名を用いてアグリゲート署名 τ に対して除算を行い（この結果をここでは仮に τ' と置く），4.3.3 節に示した手順で変更データに対する新たな削除制御用の個別署名を作成し，この値と τ' との乗算を計算する．したがって，変更データに対して再変更・削除を許可する場合，このデータは再び状態 (a) になる．また，編集者が状態 (a) の部分データを削除する場合，G_c，および G_d からそれぞれ対象データの個別署名を用いてアグリゲート署名 σ，および τ に対して除算を行うことで，該当部分の個別署名を削除する．また，部分データを変更せずに状態のみを変更する場合は，希望の状態に応じて署名集合 G_c，及び/または G_d から該当する対象データの個別署名を削除する．以上により，状態 (a) から (a)〜(e) への推移が可能になる．

次に，編集者が状態 (b) の部分データに対して変更を行う場合を考える．変更制御に関する処理については先に示した (a) と同様の手順で行う．ただし，削除制御に関しては，この部分データに該当する削除制御用の個別署名が G_d に存在しないため，変更前の部分データと変更後の部分データからそれぞれハッシュ値をとり，その差分値に署名作成を行うことでデータ更新の正当性を示す．この時，このハッシュ値の差分値は D_b に保持する．この様子を図 4.6 に示す．以上により，検証時に新規データのハッシュ値との加算によって元データのハッシュ値を復元し，アグリゲート署名の検証に用いることが可能になる．したがって，状態 (b) から (b)，または (d) への推移が可能になる．

次に，編集者が状態 (c) の部分データに対して削除を行う場合を考える．編集者は G_d から該当する部分の削除制御用の個別署名を用いてアグリゲート署名 τ に対して除算を行い，該当する対象データの個別署名を削除する．この時，この部分データに該当する変更制御用の個別署名が G_c に存在しないため，変更前の部分データと変更後の部分データからそれぞれハッシュ値をとり，その差分値に署名作成を行うことでデータ更新の正当性を示す（ただし

削除の場合はデータは存在しないため，ハッシュ値を 0 とみなして差分値に署名を行う，すなわち変更前のハッシュ値に対する署名となる）．この時，このハッシュ値の差分値は D_c に保持する．以上により，検証時に新規データのハッシュ値との加算によって元データのハッシュ値を復元し，アグリゲート署名の検証に用いることが可能になる．また，状態のみの変更を行う場合は，G_d から該当する個別署名を削除する．したがって，状態 (c) から (d)，または (e) への推移が可能になる．

最後に，編集者が状態 (f) の空データからの差し替えにより，新規データの追加を行う場合を考える．編集者は G_d から該当する部分の空データに対する個別署名を用いてアグリゲート署名 ζ に対して除算を行い，該当する対象データの個別署名を削除する．また，編集者の署名鍵を用いて追加データに対する新たな変更制御用と削除制御用の個別署名を作成し，それぞれ σ，および τ との乗算を計算する．追加データに対する変更・削除を許可する場合，それぞれ作成した個別署名を G_c，および G_d に出力する．ここで，追加データに対するコンテンツ識別子は小数点を用いて表す（例えば $ID_{i+0.5}$ とする）が， 4.3.2 節に記載した通り，追加可能な位置 1 つに対し 1 個とする．また，追加可能な空データを追加不能にする場合，G_a から該当する個別署名を削除するのみでよい．以上により，状態 (f) から (a)〜(d)，および (g) への推移が可能になる．

なお，状態 (d)，および (g) にある部分データについては，編集作業に必要な個別署名が出力されておらず，アグリゲート署名から削除することが不可能であることから，状態の変更が不可能であり，全ての編集制御が不可能である．また，提案方式において便宜上状態 (e) が定義されているが，実際には該当する部分データが存在せず，その削除した位置に対する識別子とダミーデータを関連付けしたものに対する変更用，追加用の各個別署名を作成し，アグリゲート署名に重畳させた上で，各個別署名を出力させないようにすることで，その部分に対するデータの変更・削除が不可能となる．また，追加の個別署名も存在していないため，追加も行えない．したがって，状態 (d) は (a)〜(c)，および (g) から遷移するのみ，状態 (e) は (a)，および (c) から遷移するのみ，状態 (g) は (f) から遷移するのみとなる．

上記で説明した状態遷移を図 4.7 に示す．

検証者

検証者は，最初に追加データが正当に追加されたものであるかを検証する．追加データが存在する位置は空データが削除された位置に該当し，小数点以下の数値を用いた識別子により追加データとしての認識が可能である．したがって，追加データが存在しない位置の空データに対して， 4.3.3 節で示した手順で空データに対するハッシュ値 $H(ID\|ID_{i+0.5}\|h_{i+0.5}\|0^c)$ を計算し，アグリゲート署名 ζ の検証を行う．

この検証に成功したら，次に追加データを既存データに追加し，ハッシュ値の差分値が

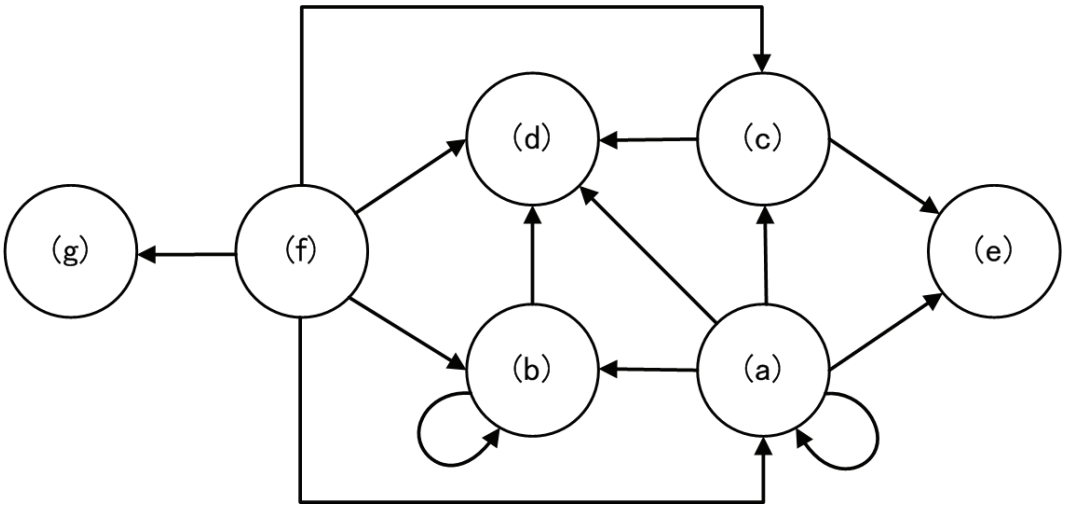

図 4.7: 状態遷移

存在するデータに関して，その正当性を既存データとハッシュ値・差分値の組合せ，および D_b, D_c を用いて ϵ_i の検証を行う．

この検証に成功したら，最後にこのハッシュ値の差分値から変更制御用と削除制御用に分けて元のハッシュ値を復元する．さらに検証者はこれ以外の既存データについてそのハッシュ値をその既存データから計算し，アグリゲート署名 σ，および τ の検証を行う．

以上の検証手順が成功したら，全てのデータの正当性が保証される．

4.3.4 システム操作手順

著作者による署名作成

提案システムにおいて，図 4.4 の (5) にあるコンテンツデータから (4) のウィンドウにドラッグ・アンド・ドロップを行うことで，オリジナルコンテンツを作成する．また，その際に 4.3.2 節で示した制御状態を図 4.4 の (2)，(3) の画面で設定する．次に，図 4.4 の (1) にある「Sign」ボタンを押すことにより，各制御状態に合わせて (8)，(9)，(10) に個別署名が，さらに事前制御用のアグリゲート署名が (11) に作成される署名が作成される．提案システムにおいてオリジナルコンテンツの署名作成までを行った時の様子を図 4.8 に示す．

編集者によるデータ更新

図 4.4 の (2)，(3) で示されている制御状態にしたがって，(5)，(6) で示されているデータを (4) にドラッグ・アンド・ドロップを行う，あるいは (4) のコンテンツデータを削除する

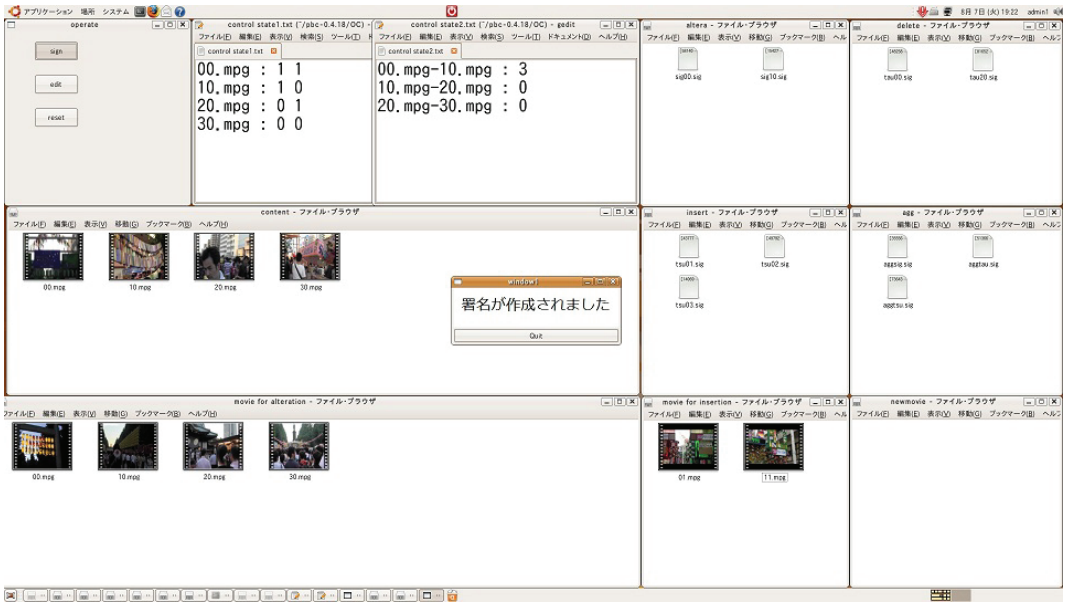

図 4.8: 署名作成

ことで編集を行う．いずれの場合も，制御状態にしたがって，正当なデータ更新を行った場合は，正しい動作を行うことが確認されている．その様子を図 4.9 に示す．

4.4 アルゴリズム

4.4.1 構成

提案した編集事前制御方式について，

1. 著作者と編集者の署名鍵・検証鍵を生成する．
2. 1 人の著作者が，コンテンツを作成する．
3. 1 人の編集者が，コンテンツを編集する．
4. 検証者は，編集されたコンテンツに対する署名の検証を行う．

のシナリオを想定した際のアルゴリズムは下記の通りである．

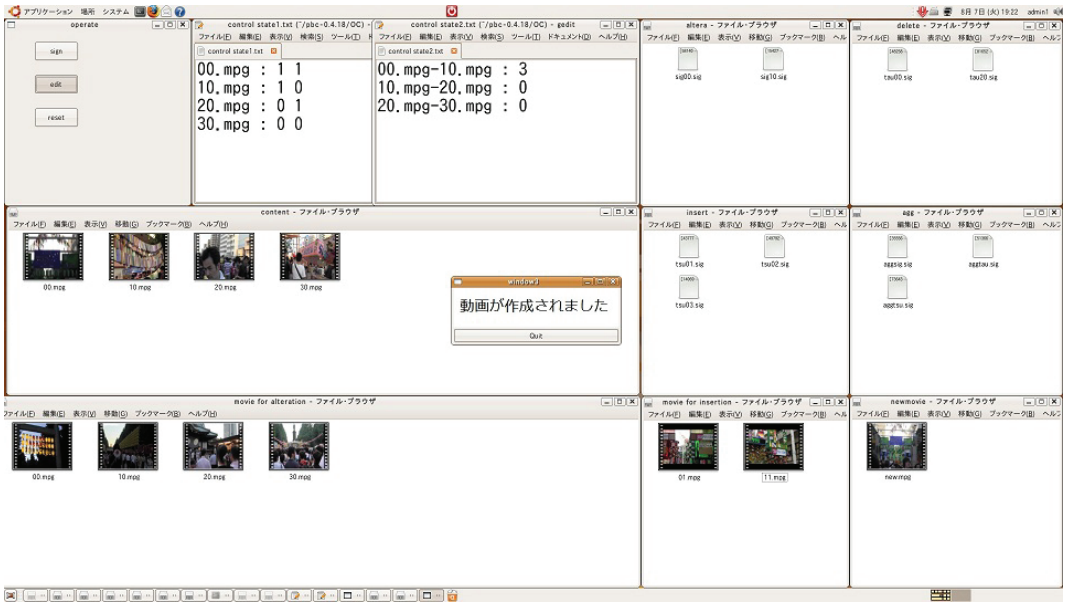

図 4.9: コンテンツ編集

4.4.2 鍵生成

著作者は署名鍵 s_0 を生成し，対応する検証鍵を以下のように計算する．

$$v_0 \leftarrow g^{s_0} \tag{4.1}$$

同様に，編集者は署名鍵 s_1 を生成し，対応する検証鍵を以下のように計算する．

$$v_1 \leftarrow g^{s_1} \tag{4.2}$$

4.4.3 著作者による署名作成

以下に著作者によるコンテンツ作成時に行われる署名作成の手順を示す．

S1. オリジナルコンテンツ M を領域毎に分割する．

$$M \rightarrow M_1, M_2, \ldots, M_n \tag{4.3}$$

S2. コンテンツデータ識別子 ID，および部分データ識別子 ID_i を生成し，以下のように設

定する．

$$M_i^* \leftarrow ID\ (i = 0)$$
$$\text{または } ID\|ID_i\|M_i\ (i \geq 1) \tag{4.4}$$

（ただし，不正変更・削除防止用として，M_0^* を変更・削除不可な部分データとする．）

S3. 分割されたデータ間に空データ B_j が存在すると仮定し，既定のダミーデータ d を用いて以下のように設定する．

$$B_j^* \leftarrow ID\|ID_j\|d \tag{4.5}$$

（ただし，不正追加防止用として，変更不可な空データ $B_{-0.5}$ が存在すると仮定し，$B_{-0.5}^*$ も作成する．）

S4. それぞれの部分データに状態 (a)〜(e)，空データに状態 (f)〜(g) のうちの一つを設定する．

S5. それぞれの部分データ，およびからデータからハッシュ値を計算する．

$$h_i \leftarrow H(M_i^*) \tag{4.6}$$
$$h_j \leftarrow H(B_j^*) \tag{4.7}$$

S6. コンテンツの部分データが存在する全ての位置 i を I_O に，空データが存在する全ての位置 j を集合 J にそれぞれ出力し．さらに全ての $i \in I_O$，および全ての $j \in J$ について以下の変更・削除・追加制御用の個別署名を作成する．

$$\text{変更用: } \sigma_i \leftarrow (H(ID\|ID_i\|h_i\|0^c))^{s_0} \tag{4.8}$$
$$\text{削除用: } \tau_i \leftarrow (H(ID\|ID_i\|h_i\|1^c))^{s_0} \tag{4.9}$$
$$\text{追加用: } \zeta_j \leftarrow (H(ID\|ID_j\|h_j\|0^c))^{s_0} \tag{4.10}$$

S7. S6 で作成した個別署名を重畳し，アグリゲート署名を作成する．

$$\sigma \leftarrow \prod_{i \in I_O} \sigma_i \tag{4.11}$$
$$\tau \leftarrow \prod_{i \in I_O} \tau_i \tag{4.12}$$
$$\zeta \leftarrow \prod_{j \in J} \zeta_j \tag{4.13}$$

また，変更・削除・追加を許可する個別署名をそれぞれの署名集合 G_c, G_d, G_a に出力する．

S8. 配布用コンテンツ $M^* = \{M_0^*, M_1^*, \ldots, M_n^*\}$，署名集合 G_c，G_d，G_a，アグリゲート署名 σ，τ，ζ，著作者が作成したコンテンツの部分データの位置情報 i の集合 I_O，部分データの間の空データが存在する位置情報の集合 J をセットにして配布する．

4.4.4 編集者によるデータ更新

編集者が編集を行う時には，以下で示す P1〜P5 のいずれかの処理を基準にデータ更新が行われる．

P1. 以下，6 個の更新．

$$M_i^* \leftarrow N_i^* \tag{4.14}$$
$$h_i' \leftarrow H(N_i^*) \tag{4.15}$$
$$\sigma \leftarrow \sigma/\sigma_i \tag{4.16}$$
$$G_c \leftarrow G_c \setminus \{\sigma_i\} \tag{4.17}$$
$$\sigma_i' \leftarrow (H(ID\|ID_i\|h_i'\|0^c))^{s_1} \tag{4.18}$$
$$\sigma \leftarrow \sigma \times \sigma_i' \tag{4.19}$$

P2. 以下，4 個の更新．

$$\tau \leftarrow \tau/\tau_i \tag{4.20}$$
$$G_d \leftarrow G_d \setminus \{\tau_i\} \tag{4.21}$$
$$\tau_i' \leftarrow (H(ID\|ID_i\|h_i'\|1^c))^{s_1} \tag{4.22}$$
$$\tau \leftarrow \tau \times \tau_i' \tag{4.23}$$

P3. 以下，1 個の更新．

$$\epsilon_i \leftarrow (H(ID\|ID_i\|(h_i - h_i')\|1^c))^{s_1} \tag{4.24}$$

P4. 以下，6 個の更新.

$$B_j^* \leftarrow N_{j'}^* \tag{4.25}$$
$$h'_{j'} \leftarrow H(N_{j'}^*) \tag{4.26}$$
$$\zeta \leftarrow \zeta/\zeta_j \tag{4.27}$$
$$G_a \leftarrow G_a \setminus \{\zeta_j\} \tag{4.28}$$
$$\sigma'_{j'} \leftarrow (H(ID\|ID_{j'}\|h'_{j'}\|0^c))^{s_1} \tag{4.29}$$
$$\sigma \leftarrow \sigma \times \sigma'_{j'} \tag{4.30}$$

P5. 以下，2 個の更新.

$$\tau'_{j'} \leftarrow (H(ID\|ID_{j'}\|h'_{j'}\|1^c))^{s_1} \tag{4.31}$$
$$\tau \leftarrow \tau \times \tau'_{j'} \tag{4.32}$$

この処理に基づき，以下に編集者によるコンテンツ編集時に行われる署名データ更新の手順を示す.

E1. M^* から編集する部分データ M_i^*，あるいは追加する空データ B_j^* を決定する.

E2. 変更，あるいは削除したい部分データについて，以下のように設定する.

$$N_i^* \leftarrow ID\|ID_i\|N_i \tag{4.33}$$

（ただし，削除の場合は

$$N_i \leftarrow d \tag{4.34}$$

とする.）
また，追加したい部分データについて，以下のように設定する.

$$N_j^* \leftarrow ID\|ID_j\|N_j \tag{4.35}$$

E3. 編集による状態の推移に合わせて，以下の処理を行う.

(a)→(a): P1，P2，および以下の 4 個の更新.

$$G_c \leftarrow G_c \cup \{\sigma_i'\} \tag{4.36}$$
$$G_d \leftarrow G_d \cup \{\tau_i'\} \tag{4.37}$$
$$I_O \leftarrow I_O \setminus \{i\} \tag{4.38}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.39}$$

(a)→(b): 状態のみ変更の場合は，以下の 1 個の更新.

$$G_d \leftarrow G_d \setminus \{\tau_i\} \tag{4.40}$$

データの変更がある場合は，P1，P2，および以下の 3 個の更新.

$$G_c \leftarrow G_c \cup \{\sigma_i'\} \tag{4.41}$$
$$I_O \leftarrow I_O \setminus \{i\} \tag{4.42}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.43}$$

(a)→(c): 状態のみ変更の場合は，以下の 1 個の更新.

$$G_c \leftarrow G_c \setminus \{\sigma_i\} \tag{4.44}$$

データの変更がある場合は P1，P2，および以下の 3 個の更新.

$$G_d \leftarrow G_d \cup \{\tau_i'\} \tag{4.45}$$
$$I_O \leftarrow I_O \setminus \{i\} \tag{4.46}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.47}$$

(a)→(d): 状態のみ変更の場合は，以下の 2 個の更新.

$$G_d \leftarrow G_d \setminus \{\tau_i\} \tag{4.48}$$
$$G_c \leftarrow G_c \setminus \{\sigma_i\} \tag{4.49}$$

データの変更がある場合は P1，P2，および以下の 2 個の更新.

$$I_O \leftarrow I_O \setminus \{i\} \tag{4.50}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.51}$$

(a)→(e): P1（ただし，$M_i^* \leftarrow N_i^*$ は行わず，代わりに $M^* \leftarrow M^* \setminus M_i^*$ を行う），P2，および以下の 2 個の更新．

$$I_O \leftarrow I_O \setminus \{i\} \tag{4.52}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.53}$$

(b)→(b): P1，P3，および以下の 5 個の更新．

$$G_c \leftarrow G_c \cup \{\sigma_i'\} \tag{4.54}$$
$$D_b \leftarrow D_b \cup \{(h_i - h_i')\} \tag{4.55}$$
$$I_{D_b} \leftarrow I_{D_b} \cup \{i\} \tag{4.56}$$
$$I_O \leftarrow I_O \setminus \{i\} \tag{4.57}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.58}$$

(b)→(d): 状態のみ変更の場合は，以下の 1 個の更新．

$$G_c \leftarrow G_c \setminus \{\sigma_i\} \tag{4.59}$$

データの変更がある場合は P1，P3，および以下の 4 個の更新．

$$D_b \leftarrow D_b \cup \{(h_i - h_i')\} \tag{4.60}$$
$$I_{D_b} \leftarrow I_{D_b} \cup \{i\} \tag{4.61}$$
$$I_O \leftarrow I_O \setminus \{i\} \tag{4.62}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.63}$$

(c)→(d): 以下の 1 個の更新．

$$G_d \leftarrow G_d \setminus \{\tau_i\} \tag{4.64}$$

(c)→(e): P1（ただし，$M_i^* \leftarrow N_i^*$ は行わず，代わりに $M^* \leftarrow M^* \setminus M_i^*$ を行う），P2，および以下の 2 個の更新．

$$I_O \leftarrow I_O \setminus \{i\} \tag{4.65}$$
$$I_E \leftarrow I_E \cup \{i\} \tag{4.66}$$

(f)→(a): P4，P5，および以下の 4 個の更新.

$$G_c \leftarrow G_c \cup \{\sigma'_{j'}\} \tag{4.67}$$
$$G_d \leftarrow G_d \cup \{\tau'_{j'}\} \tag{4.68}$$
$$J \leftarrow J \setminus \{j'\} \tag{4.69}$$
$$I_E \leftarrow I_E \cup \{j'\} \tag{4.70}$$

(f)→(b): P4，P5，および以下の 3 個の更新.

$$G_c \leftarrow G_c \cup \{\sigma'_{j'}\} \tag{4.71}$$
$$J \leftarrow J \setminus \{j'\} \tag{4.72}$$
$$I_E \leftarrow I_E \cup \{j'\} \tag{4.73}$$

(f)→(c): P4，P5，および以下の 3 個の更新.

$$G_d \leftarrow G_d \cup \{\tau'_{j'}\} \tag{4.74}$$
$$J \leftarrow J \setminus \{j'\} \tag{4.75}$$
$$I_E \leftarrow I_E \cup \{j'\} \tag{4.76}$$

(f)→(d): P4，P5，および以下の 2 個の更新.

$$J \leftarrow J \setminus \{j'\} \tag{4.77}$$
$$I_E \leftarrow I_E \cup \{j'\} \tag{4.78}$$

(f)→(g): 以下の 1 個の更新.

$$G_a \leftarrow G_a \setminus \{\zeta_j\} \tag{4.79}$$

E4. 計算された ϵ_i を重畳する.

$$\epsilon \leftarrow \prod_{i \in I_D} \epsilon_i \quad （ただし $I_D = I_{D_b} \cup I_{D_c}$） \tag{4.80}$$

また，編集履歴データ r とその署名 σ_r を作成する.

E5. 更新したコンテンツ M^*，署名集合 G_c，G_d，G_a，アグリゲート署名 σ，τ，ζ，ϵ，空データが存在する位置情報の集合 J，D_b に出力を行った位置情報 i の集合 I_{D_b}，D_c に出力を行った位置情報 i の集合 I_{D_c}，著作者が作成したコンテンツの部分データが存在

している位置情報 i の集合 I_O，編集者が変更・追加を行ったコンテンツの部分データの位置情報 i の集合 I_E，ハッシュ値の差分値の集合 D_b，D_c，編集履歴データ r とその署名 σ_r をセットにして再配布する．

4.4.5 署名検証

以下に検証者による署名検証の手順を示す．

V1. 各部分データに連接されている ID と M_0^* が一致しているか確認する．

V2. データが追加されていない位置のデータ B_j^* から以下の検証を行う．

$$e(\zeta, g) = e(\prod_{j \in J} H(ID \| ID_j \| H(B_j^*) \| 0^c), v_0) \tag{4.81}$$

V3. V2 の検証が成功したら，M^* の要素である部分データ M_i^*, N_i^*，および追加データ $N_{j'}^*$ の部分データ識別子 $ID_i, ID_{j'}$ すべての順序が正しいことを確認する（以後，i, j' を含めて k として表す）．

V4. D_b にハッシュ値の差分値を持つ部分データが存在すること，および D_c にハッシュ値の差分値を持つ部分データが存在しないことを確認し，以下の検証を行う．

$$e(\epsilon, g) = e(\prod_{k \in I_D} H(ID \| ID_k \| (h_k - h'_k) \| 1^c), v_1) \quad （ただし\ I_D = I_{D_b} \cup I_{D_c}） \tag{4.82}$$

V5. V4 の検証が成功したら，すべての $k \in I_D$（ただし $I_D = I_{D_b} \cup I_{D_c}$）において $h'_k = H(M_k^*)$ を計算し，以下の計算を行う．

$$h'_k + (h_k - h'_k) \quad (= h_k) \tag{4.83}$$

V6. D_b にハッシュ値の差分値がある場合は削除の検証用，D_c にハッシュ値の差分値がある場合は変更の検証用に V5 で求めたハッシュ値を使用し，これ以外の検証に対しては既存の部分データ（部分データが存在しない場合はダミーデータ）のハッシュ値を使用して，以下の計算を行う．
変更用:すべての $k \in I$ において

$$x_k \leftarrow H(ID \| ID_k \| h_k \| 0^c) \tag{4.84}$$

削除用:すべての $k' \in I$ において

$$y_{k'} \leftarrow H(ID \| ID_{k'} \| h_{k'} \| 1^c) \quad （ただし $I = I_O \cup I_E$）$$ (4.85)

V7. r から

$$著作者用: X_0 \leftarrow \prod_{k \in I_O} x_k$$ (4.86)

$$Y_0 \leftarrow \prod_{k \in I_O} y_k$$ (4.87)

$$編集者用: X_1 \leftarrow \prod_{k \in I_E} x_k$$ (4.88)

$$Y_1 \leftarrow \prod_{k \in I_E} y_k$$ (4.89)

(4.90)

を計算し，以下の検証を行う．

$$e(\sigma, g) = e(X_0, v_0) \cdot e(X_1, v_1)$$ (4.91)

$$e(\tau, g) = e(Y_0, v_0) \cdot e(Y_1, v_1)$$ (4.92)

4.5 考察

4.5.1 安全性

禁止処理の検出

提案方式において，許可されていない編集を行った禁止処理に対して，どのように不正が検証されるかを以下に示す．

まず，状態 (g) における空データへの追加について考える．追加データの識別子は既存データの識別子と異なる（ID_i に対し ID_j である）ため，両者は識別可能である．また，ζ にはこの部分の空データに対する個別署名が重畳されているにもかかわらず，G_a の要素として個別署名が存在しないことから，除算を行うことが不可能である．したがって，署名検証が失敗する．

次に，状態 (b) における既存データの削除を考える．この場合，G_c の要素として存在する不正箇所の個別署名で σ の除算を行い，さらに ϵ_i を作って ϵ に重畳し，D_b にそのハッシュ値の差分値を追加することで整合性が保たれる．しかし，状態 (b) における正当な変更処理の場合，D_b に追加したハッシュ値の差分値に該当する部分データが必ず存在するが，削除の場合

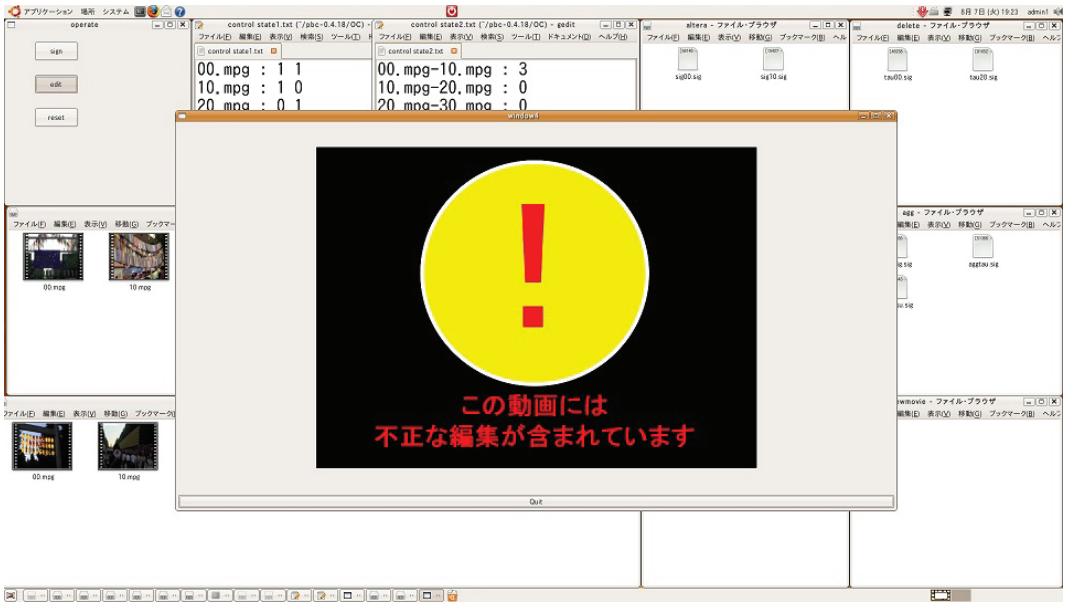

図 4.10: 不正編集時

はその部分データが存在しないため，データが削除されたことが判明する．また，ハッシュ値の差分値を D_c に追加した場合も正しいハッシュ値が復元されず，署名検証が失敗する．

次に，状態 (c) における既存データの変更を考える．この場合，G_d の要素として存在する不正箇所の個別署名で τ の除算を行い，変更したデータによる新たな署名を τ に重畳し，さらに ϵ_i を作って ϵ に重畳し，D_c にそのハッシュ値の差分値を追加することで整合性が保たれる．しかし，状態 (c) における正当な削除処理の場合，どのような部分データも存在しないが，変更の場合はこの存在しないはずの部分データが存在することになり，データが変更されたことが判明する．したがって，正しいハッシュ値が復元されず，署名検証が失敗する．

最後に，状態 (d) における既存データの変更，および削除を考える．この場合，D_b，および D_c にハッシュ値の差分値を追加することで整合性が保たれる．しかし，変更に対しては D_b のハッシュ値の差分値と既存データとの組合せが，削除に対しては D_c のハッシュ値の差分値と既存データの組合せが上記と同様になるので，正しいハッシュ値が復元されず，署名検証が失敗する．

図 4.10 は不正処理を検出した場合を示す．作成したシステムでは，前述したいずれの場合においても不正な編集を検出できることが検証済みである．

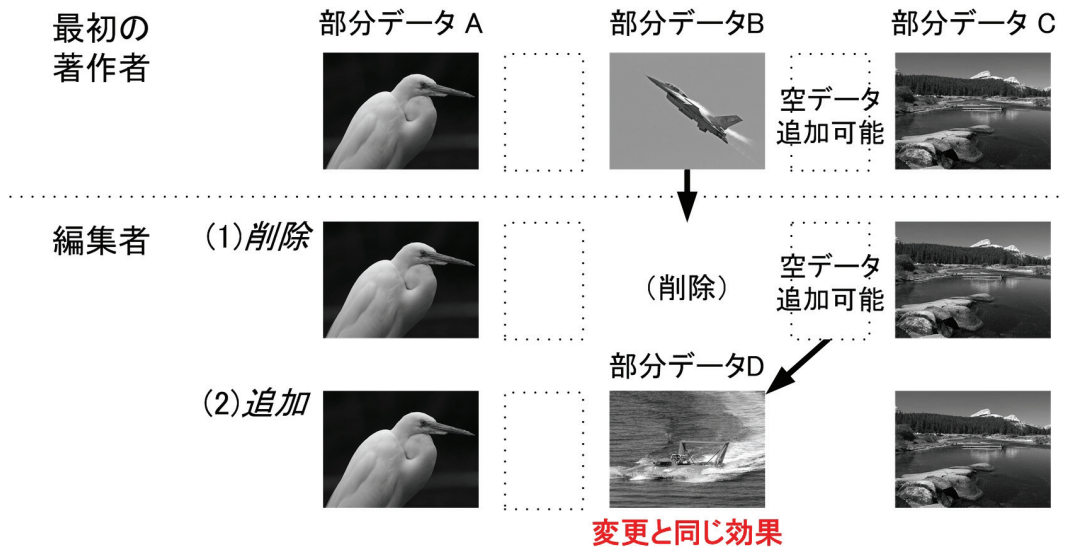

図 4.11: 状態推移における留意点

状態推移における留意点

各部分データ，および空データに制御状態を定めており，下記の通りの署名の組み合わせが存在する．

状態 (a) 既存の部分データ，σ_i，τ_i.

状態 (b) 既存の部分データ—，σ_i.

状態 (c) 既存の部分データ，τ_i.

状態 (d) 既存の部分データ.

状態 (e) なし（ただし，ハッシュ値の差分値が存在する可能性あり）.

状態 (f) 空データ，ζ_j.

状態 (g) 空データ.

この時，特定の制御状態が連続した場合に新規データの追加を考慮すると，図 4.11 に示す留意点がある．以下にこの留意点と対応について説明する．

図 4.11 において，画像 A，B，C を「変更禁止・削除可能」な部分データ，点線を「追加可能」な部分データの間を示すものとする．この場合，画像 B を削除して「追加可能」な部分データ間に画像 D を追加すると, 画像 B から画像 D に変更したことになる（『画像 A→ 画

表 4.1: 提案方式と既存方式との比較

	変更	削除	追加	署名作成コスト（n：分割数）	署名検証コスト（ペアリング演算の回数）
IIKO	△	○	×	$2n$	2
PIAT	△	△	△	1	1
提案方式	○	○	○	$3n + 1$	4

像 B→ 画像 C』⇒『画像 A→ 画像 D→ 画像 C』）．このように，連続する 2 つの部分データの制御状態が「変更禁止・削除可能」であり，この 2 つの部分データ間が「追加可能」の状態を定めていた場合，連続する 2 つの部分データのどちらか一つを削除し,「追加可能」部分に削除した部分データに代わるデータを追加することで，事実上データの変更を行うことが可能になる．

したがって，提案方式において著作者はコンテンツを分割する際に，

- 「変更禁止・削除可能」の部分データが連続しないようにする．
- 「変更禁止・削除可能」の部分データが連続する場合は，この 2 つの部分データ間は「追加禁止」と設定する．

のいずれかの対応を行う必要がある．

位置情報における留意点

空データの位置，状態推移以外の編集が行われなかった部分データの位置，変更・追加が行われた部分データの位置，変更前後におけるハッシュ値の差分値を求めた位置，削除前後におけるハッシュ値の差分値を求めた位置をそれぞれ J，I_O，I_E，I_{D_b}，I_{D_c} と定義している．この位置情報が正しいとして検証を行っているが，その正当性を保証するためには，それぞれの位置情報に対し編集者が署名を作成する（例えば $(H(J\|I_O\|I_E\|I_{D_b}\|I_{D_c}))^{s_1}$ など）といった対応を行う必要がある．

4.5.2 従来方式との比較

4.2 節で説明した従来の墨塗り署名方式を含めて，変更・削除・追加における事前制御の可否，および各署名方式における事前署名数・検証回数の比較を行った．その比較結果を表 4.1 に示す．なお，事前制御が可能ならば○（ただし，ハッシュ値以外の変更が不可ならば△とする），事前制御ではなく不正更新の検知が可能ならば△，どちらも満たさない場合を×とする．

表 4.1 で示す通り，提案方式のみが変更・削除・追加のいずれについても事前制御を可能にしており，制限がある従来方式に対して優位性がある．

一方，提案方式は全ての制御において各個別署名を生成しているため，従来方式に比べ署名回数や検証回数が多い．特に，著作者は分割データ数に対して約 3 倍の署名作成が必要となる．しかし，これはコンテンツ作成時に 1 度だけ事前に行っておく処理であり，コンテンツ配布以降に署名作成を行う必要はないため，実用上はほとんど問題ないと考えられる．

また，変更禁止または削除禁止の既存データの編集に関し，ハッシュ値の差分値を保持する必要がある．しかし，この場合は個別署名 ϵ_i を保持せず，ハッシュ値の差分値を保持する部分データは σ_i，あるいは τ_i のどちらかを保持することはない．したがって，各既存データは個別署名，およびハッシュ値の差分値の 2 種類のデータの両方を持ち，空データはどちらか 1 つを持つことになる．部分データがパケットにより構成される場合はこのパケットのヘッダ部などにもつことが可能であり，部分データ自体のサイズに比べると十分小さく，実用上問題ないと考えられる．

以上により，提案方式は変更・削除・追加に対する全ての事前制御を可能にし，かつ処理負荷の面でも実用上問題のないシステムであると言える．

4.5.3 *n* 次編集に対する課題

4.3.2 節において，提案方式は著作者が作成したコンテンツに対する編集（これを「1 次編集」と表現する）について説明しており，編集者が作成したコンテンツに対する編集（これを「*n* 次編集」と表現する）についても，同様な処理を行うことを想定している．

この提案方式において，*n* 次編集における基本的な編集制御は保護される．一方，*n* 次編集の特性を考慮した場合，さらに以下の項目を検討する必要がある．

編集順序の保証

一般的な著作物の場合，オリジナルの著作物と，これを引用・編集して作成された二次的著作物は，どちらの著者にも著作権が生じることが規定されている [2]．この時，コンテンツの部分データに対して，誰がどの段階で編集を行ったかを確認する仕組みが必要となる場合が考えられる．例えば，ある段階までは変更許可だった部分データが変更不可に制御状態が変えられているコンテンツがあり，その部分データを編集したい編集者が編集制御許可の状態でかつ最も新しく編集されたコンテンツで編集を行うことを希望する場合があげられる．この場合では，最初の著作者が変更許可だったとしても，ある編集者が編集不可にしていれば，その部分データを変更することは不正と見なされる．それを回避する手段として変

更が許可されているコンテンツで編集を行うといった方法があり，そのためにはどの段階で制御状態が変えられたかを確認する必要があることが想定される．

したがって，*n* 次編集においてどの段階の編集者が，どのような編集を行ったかの識別を行う仕組みが必要となる．しかし，提案方式において上記の判断が行える情報は 4.3.2 節で示した時間情報のみであり，ここが詐称可能な場合は編集過程を保証できず，確認ができなくなる可能性がある．

上記に対し，編集を行った際に変更・追加される署名について，その順序を保証する仕組みを導入するといった対策が考えられる．例えば，提案方式のベースとなっている 3.3 節で提案したアグリゲート署名方式に基づく順序付きアグリゲート署名について，同じ署名方式をベースとしていることから，提案方式に対する導入に関して検討の余地がある．

削除制御時の有効ダミーデータの選択

分割された部分データ間には追加制御用のダミーデータが存在する．1 回編集のみが認められている場合は，そのダミーデータが使用される機会は 1 度のみであり，何らかの編集制御が行われた後はそのダミーデータに対する制御状態を考慮する必要はない．

しかし，*n* 次編集においては次の課題が生じる．削除が可能な部分データの前後に異なる制御状態が設定されたダミーデータ（状態 (f) と状態 (g)）が存在し，1 次の編集において部分データを削除した時，異なる制御状態のダミーデータが 2 個残されることになり，*n* 次以降の編集においてどちらが有効かを判定できない．

上記に対し，有効となるダミーデータの制御をどのように規定するか，その制御を保証する手順はどのように行うかを検討する必要がある．

著作者以外による署名の付け替え

提案方式においては，著作者が変更・削除・追加可能とした部分データに対し，その後の編集者がその署名を全て入れ替えることが可能である．

このような部分データは著作者から見ればどのように扱われてもよいデータであると解釈でき，編集管理の観点から編集者が上記の変更を行っても問題とは考えない．しかし，そのデータが流通する場合は著作権法の観点で問題が生じる可能性があること，さらに *n* 次編集により作成された著作物の場合は途中の編集者の署名が消去されてしまう可能性があることから，再考の必要がある．

複数のコンテンツデータの追加制御

提案方式においては，4.3.2 節で示した通り，コンテンツの部分データ間で追加できるコンテンツデータの数は最大 1 つとなっている．しかし，動画などにおいて複雑な編集を行っていく時に，複数のデータをつなぎ合わせて挿入するといった処理が行われることがあり，2 つ以上のデータを追加できることが望ましい．

しかし，これを n 次編集により作成される著作物での実現を考えると様々な課題が発生することが予想される．その一例として，最初は複数のデータが追加可能となっていた場合でも，ある編集者が追加した後にそのデータの前後で追加不可にしたいといった要望が出た時に，どのように制御の設定を行うかといった課題がある．

そのため，データ追加後においてその前後での追加制御の設定をどのように行うか，また，その設定のための個別署名の出力をどのように行うかなど，想定される課題を整理して対応する必要がある．

第5章　コンテンツのデータ形式に適応した暗号化方式

5.1　JPEG 2000

JPEG 2000 符号化 [19, 20] を理解するために，図 5.1 に示した JPEG 2000 エンコーダの各処理を説明する．

入力画像は，まずウェーブレット変換（DWT）によりサブバンド分解される．この時，小さい解像度レベルに属する係数ほど低い周波数の情報を含むことになる．分解された入力画像は，その後量子化される．量子化されたウェーブレット係数は，EBCOT アルゴリズムにより符号化される．このアルゴリズムを，コードブロック分割，係数モデリング，算術符号化とレート制御，レイヤ形成，パケット生成の 5 つの部分に分けて説明する．

コードブロック分割: 各サブバンドは，コードブロックと呼ばれる矩形のブロック（例えば 64×64 など）に分割される．これらのコードブロックは，それぞれ独立に符号化される．

係数モデリング: 各コードブロックのウェーブレット係数列に対し，ビットプレーンに基づく係数モデリングを行う．これにより，係数ビットが重要度順に並んだエンベデッド符号列を生成する．MSB から LSB までの全てのビットプレーンは，三つのサブビットプレーン（パス）分解される．各サブビットプレーンの境界は打切り点と呼ばれ，後にデータを切り捨てる際の最小の分割単位となる．

算術符号化／レート制御: 係数モデリングにより生成されたエンベデッド符号列に対して，適応算術符号化を施す．また，目標とする符号語に合せるため，符号語の一部を切り捨てる操作を行う．どの符号語を切り捨てるかで画質が変動する．

レイヤ形成: SNR スケーラブルが必要である時に行う．各レイヤはそれぞれ，各コードブロックのエンベデッド符号の一部を含む．高いレイヤほど画像の再生にあたって重要な成分を含むことになる．

パケット生成: 各レイヤを，複数のボディと呼ばれる単位に分解し，それぞれにヘッダ情報を付加する．この操作により生成されたパケットは，パケットヘッダ（データ長や符号化パスの情報などを含む）とパケット本体（ボディ）の 2 つから構成されている．

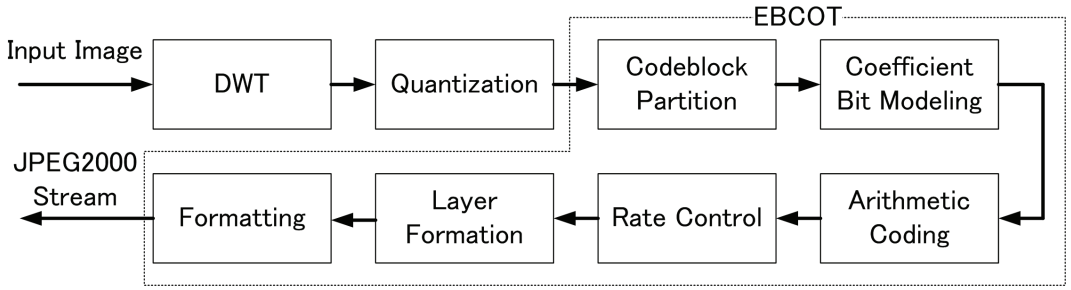

図 5.1: JPEG 2000 エンコーダ

ここで，各ボディは簡単のために解像度とレイヤ以外のパラメータを 1 とすると，それぞれ対応するレイヤと解像度レベルの情報を持つ．この結果，生成されるパケットの総数はレイヤ数と解像度レベル数の積となる．また，ヘッダ情報には各コードブロックの算術符号列の長さ，サブビットプレーンの個数などの情報が含まれる．上記のアルゴリズムを経て生成された全てのパケットをまとめ，グローバルヘッダ情報を付加したものが，最終的な JPEG 2000 の符号列となる（ただし，以上のような各種ヘッダ情報やデータ分割の最小単位であるサブビットプレーンは，1 バイトの整数倍のサイズとすることを規定している）．

この時，高解像度部分に相当する符号列のみを部分暗号化し，低解像度部分に相当する符号列は非暗号部分とすることにより，低解像度の画像は開示するが，高解像度画像は保護することが可能となる．

JPEG 2000 における前述の特定符号としては，0xFF90～FFFF の値を有するマーカ（定義情報を格納するコード），およびマーカセグメントコードがあげられる．これらは 2 バイトで表され，先頭の 1 バイトは 0xFF である．一方，マーカセグメントは 1 つのマーカとそれに追随するパラメータから構成される．

JPEG 2000 では，特に 0xFF90～FFFF の範囲のマーカに 2 つの特別な意味を持たせている．1 つは，これらのマーカがコードストリームの区切りを意味している．これにより，パケットおよびパケットヘッダを位置づけることが可能となる．他の 1 つは，これらのマーカが圧縮データ自身の中に存在しないことを意味している．したがって，暗号化対象データはボディ部であるため，JPEG 2000 の部分暗号化において回避したいのは 0xFF90～FFFF のマーカコードの生成である．

セキュリティに関する規格に対応した復号器が部分暗号化部分をどのように識別するかについては，メタデータ（画像情報の復号に依存しない形で任意の情報を符号列の中に挿入出来るデータ）というオプションを用いて，部分暗号化部分を復号器に知らせることを想定する．

5.2 ブロックベース Cycle-walking

5.2.1 Cycle-walking の課題

2.2.3 節において，特定のサイズ以下である平文・暗号文の空間に合わせて暗号アルゴリズムを適用させる Cycle-walking について説明した．

しかし，Cycle-walking における暗号化対象はブロック長以下のサイズの平文であるが，本論文では特定のビット列を含まない平文として適用する．また，Cycle-walking はそのブロック長が規定されていない一般的な手法であるが，これを AES のような一定のブロック長を持つブロック暗号に用いて，そのブロック長より長い平文を暗号化しようとする場合を考える．この時，以下の問題が発生する．

ここでは平文を JPEG 2000 データのボディ部として説明する．Algorithm 1 に示した Cycle-walking のアルゴリズムに従うと，例えば平文が 1280 ビットである場合，AES は 128 ビットを 1 ブロックとして 10 ブロック分暗号化を行う．その結果が指定された集合 $\mathbb{M}$ に属すかどうかを検査する，すなわち，暗号化結果に 0xFF90〜FFFF の特定符号が含まれるかどうかを検査する．もし 1 つでも含まれていれば，全 10 ブロックを再暗号化を行う．この時，まず全平文である 10 ブロックを保持するメモリが暗号処理中に必要になる．さらに 10 ブロック中 1 ブロックにのみ特定符号が含まれていても全 10 ブロックを再暗号化する必要があり，効率的でない．

そこで，Cycle-walking を 1 ブロック単位で適用することを考える．この場合，暗号化した 1 ブロック中に特定符号が発生したら，そのブロックだけを再暗号化すればよく，メモリも 1 ブロックで良いので効率的である．しかし，ブロック間の繋ぎ目に特定符号が発生した場合（0xFF90〜FFFF の特定符号は 2 バイトあるため，前ブロックの最後のバイトが 0xFF で，後バイトの最初のバイトが 0x90〜FF の場合），Algorithm 1 に示すアルゴリズムでは対応できない．

5.2.2 ブロックベース Cycle-walking の概要

5.2 節で示した課題に対し，Stutz と Uhl によってブロックベース Cycle-walking [39] が提案されている．これは，複数ブロックを暗号化する際に，ブロック内に特定符号が存在しなくなるまで再暗号化した後，更に一つ手前の暗号化ブロックとの繋ぎ目に特定符号が存在するかを検査し，存在する場合にはさらに再暗号化を行うといったものである．したがって，そのブロックの暗号化が完了する条件は

- ブロック内に特定符号が存在しないこと．
- 一つ手前のブロックとの繋ぎ目で特定符号が存在しないこと．

の両方を満たした時である [1].

5.2.3 暗号化アルゴリズム

$E_K(\cdot)$ を一般的な共通鍵 K によるブロック暗号アルゴリズム（AES など），m を暗号化を行いたい JPEG 2000 のファイルデータとした時，ブロックベース Cycle-walking による暗号化アルゴリズム $Cp_K(\cdot)$ を Algorithm 3 に示す.

Algorithm 3 $Cp_K(m)$:Encryption procedure of our scheme

```
Divid m into P(0)...P(n − 1)
{The size of each P(k) is the block size of E_K(·).}
x ← null
for i = 0 to n − 1 do
  repeat
    bool ← 1
    a ← E_K(P(i))
    if x == null then
      r ← a
    else
      r ← x||a
    end if
    for j = 0xFF90 to 0xFFFF do
      if Code j is within r then
        bool ← 0
      end if
    end for
    P(i) ← a
  until bool == 1
  bot ← The bottom Byte of P(i)
  if bot == 0xFF then
    x ← bot
  else
    x ← null
  end if
end for
return P(0)||...||P(n − 1)
```

[1] JPEG 2000 の場合，サブビットプレーンは 1 バイトの整数倍のサイズとすることが規定されているため，例えば 1 ブロック目の最後の 1 バイトが 0x1F，2 ブロック目の最初の 1 バイトが 0xF9 となる場合などは，特定符号と解釈されない．このような場合は，特に回避策を必要としないものとする．

このアルゴリズムの概要を説明すると，以下の通りである．

1. JPEG 2000 のファイルデータを，$E_K(\cdot)$ で処理が行えるサイズ毎のブロックに分割する．
2. 以下の (a)〜(e) の手順を，最初のブロックから最後のブロックまで行う．
   (a) ブロックを $E_K(\cdot)$ で（再）暗号化する．
   (b) もし一つ前のブロックから記憶させたバイトデータがある場合は，そのデータを（再）暗号化したブロックに連接し，存在しない場合はそのままのデータで，次の確認手順に進む．
   (c) 特定符号が含まれているか確認し，含まれている場合はこの暗号化データ（連接したデータは除く）を入力値として (a) に戻る．含まれていない場合は次に進む．
   (d) 暗号化ブロックの最後のバイト値が 0xFF の場合は，0xFF をバイトデータとして記憶させ，ブロックの最後のバイト値が 0xFF 以外の場合は何も記憶させず，次に進む．
   (e) このブロックの暗号化データとする．
3. 全ての暗号化データを連接し，これを JPEG 2000 のファイルデータに対する暗号化データとして出力する．

ここで，2-(b),(c),(d) の手順について説明する．前述したように複数ブロックの暗号化を行う場合，ブロックの繋ぎ目において特定符号が発生する可能性がある．そのために，一つ手前のブロックにおける暗号化結果の最後のバイトが 0xFF である場合，(b) においてそれを次のブロックと連接し，さらに (c) においてその連接ブロックの検査により特定符号が検出された場合は再暗号化を行う．

5.2.4 復号アルゴリズム

5.2.3 節において，ブロックベース Cycle-walking の暗号化アルゴリズムについて説明した．復号する場合はこの暗号化に合わせるように，ブロック内および一つ手前の暗号化ブロックとの繋ぎ目において特定符号が含まれているか確認し，いずれの場合においても特定符号が含まれなく．なるまで再復号を行う．この手順を全ブロックで行うことにより，ファイルデータが復号される．

しかし，Stutz と Uhl による提案では，暗号化については言及しているが，復号については言及されていない [39]．通常のブロック暗号と異なり，判定の際に必要な情報を保存するという手順が必要であり，復号アルゴリズムにおいてそれがどのように行われるかを正しく記載する必要がある．

そこで，$E_K^{-1}(\cdot)$ を $E_K(\cdot)$ に対応する復号アルゴリズム，c を暗号化された JPEG 2000 のファイルデータとした時，ブロックベース Cycle-walking による復号アルゴリズム $Cp_K^{-1}(\cdot)$ を Algorithm 4 に示す.

このアルゴリズムの概要を説明すると，以下の通りである.

Algorithm 4 $Cp_K^{-1}(c)$:Decryption procedure of our scheme

```
Divide c into P(0)...P(n − 1)
{The size of each P(k) is the block size of E_K^{-1}(·).}
x ← null
for i = 0 to n − 1 do
  bot ←The Bottom Byte of P(i)
  repeat
    bool ← 1
    a ← E_K^{-1}(P(i))
    if x == null then
      r ← a
    else
      r ← a||x
    end if
    for j = 0xFF90 to 0xFFFF do
      if Code j is within r then
        bool ← 0
      end if
    end for
    P(i) ← a
  until bool == 1
  if bot == 0xFF then
    x ← bot
  else
    x ← null
  end if
end for
return P(0)||...||P(n − 1)
```

1. JPEG 2000 のファイルデータを，$E_K^{-1}(\cdot)$ で処理が行えるサイズ毎のブロックに分割する.
2. 以下の (a)〜(f) の手順を，最初のブロックから最後のブロックまで行う.
   (a) 入力ブロックの最後のバイト値を記憶する.
   (b) 入力ブロックを $E_K^{-1}(\cdot)$ で復号する（手順 (d) から戻ったデータは再復号する）.

(c) もし一つ前のブロックの復号手順 (e) で記憶させたバイトデータがある場合は，そのデータを手順 (b) で復号されたブロックの前に連接し，存在しない場合は連接せずに，次の確認手順に進む．

(d) 手順 (c) から来たブロックに特定符号が含まれているか確認し，含まれている場合はこのブロック（手順 (b) で連接したデータは除く）を入力値として手順 (a) に戻る．含まれていない場合は次に進む．

(e) 手順 (a) で記憶したバイト値が 0xFF の場合は，0xFF をバイトデータとして記憶させ，バイト値が 0xFF 以外の場合は何も記憶させず，次に進む．

(f) 手順 (e) から来たブロックを，手順 (a) に入力されたブロックの復号データとする．

3. すべての復号データを連接し，これを復号された JPEG 2000 のファイルデータとして出力する．

ここで，2-(d) の手順は 5.2.4 節で示した暗号化の 2-(c) の手順に対応するものである．

5.3 考察

5.3.1 安全性

Algorithm 1 と Algorithm 3 において，実際に行っている処理は同じで，$\mathbb{M}$ の空間の違いのみである．Algorithm 1 における $\mathbb{M}$ は 0 以上 $k-1$ の数値であるのに対し，Algorithm 3 における $\mathbb{M}$ は 0xFF90〜FFFF が含まれていないデータ（数値）となる．したがって，安全性に関する議論は Cycle-walking と同じであり，ブロックベース Cycle-walking との安全性の差は $\mathbb{M}$ の要素数に依存する計算コストの差で表すことができる．これは，もし二つの要素数が同じであるなら，その安全性は等価であることを意味している．

5.3.2 処理コスト

計算コスト

ブロックベース Cycle-walking は，ブロックに特定符号が含まれなくなるまで繰り返し暗号化および復号を行うが，この繰り返しの回数により実用性が大きく変化する．この回数について考察を行う．ただし，ここでは簡単のため暗号化を完全なランダム化とみなし，注目する平文のバイトもランダムだと考える．

ここで，通常のブロック暗号（暗号ブロックサイズは t バイトとする）において，1 ブロックにおける暗号化 1 回分を 1 ステップと定義する．あるブロックを B，B の一つ前にある暗

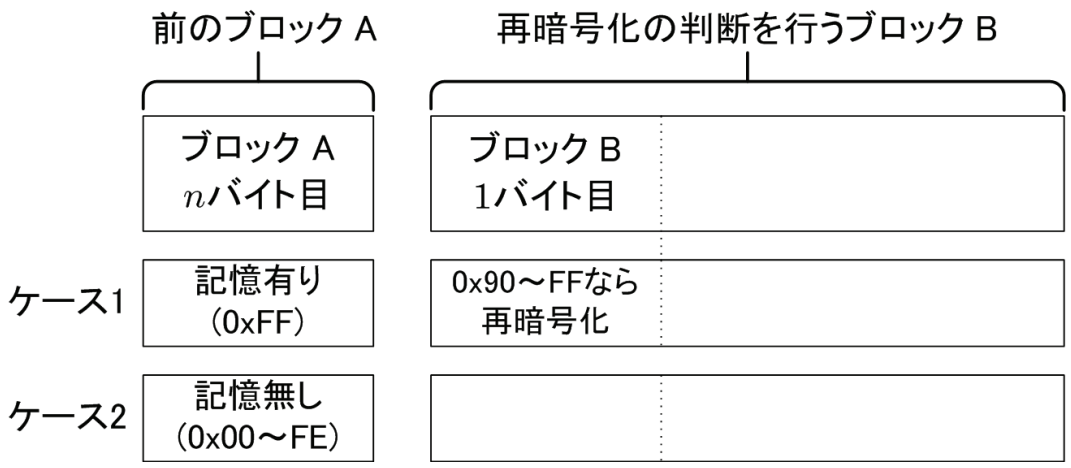

図 5.2: ブロック状態の場合分け

号化が終了したブロックを A と置き，図 5.2 に示す 2 種類のケースに分け，それぞれのケースで暗号化したブロック B を再び暗号化する条件を考える．

ケース 1 A の n バイト目の記憶データが存在する（値が 0xFF）．

ケース 2 A の n バイト目の記憶データが存在しない（値が 0x00～FE）．

ケース 1 の時，再暗号化が行われる条件は B の 1 バイト目が 0x90～FF （1 バイト列が 0x90～FF となる確率は $\frac{7\times16}{2^8}$），1 バイト目から n バイト目までの間で連続する 2 バイトが 0xFF90～FFFF （1 つの 2 バイト列あたり，0xFF90～FFFF となる確率は $\frac{7\times16}{2^{16}}$）のいずれかの状態が発生した時となる．この条件が起こる確率は，上記のいずれも発生しない（すなわち，再暗号化が行われない）確率の値である

$$\tilde{P}_1 = \left(1 - \frac{7\times16}{2^8}\right)\left(1 - \frac{7\times16}{2^{16}}\right)^{t-1} \tag{5.1}$$

を求めて，1 からその確率を引くことで求められる．その値は

$$\begin{aligned} P_1 &= 1 - \tilde{P}_1 \\ &= 1 - \left(1 - \frac{7\times16}{2^8}\right)\left(1 - \frac{7\times16}{2^{16}}\right)^{t-1} \end{aligned} \tag{5.2}$$

となる．

ケース 2 の時，再暗号化が行われる条件は B の 1 バイト目から n バイト目までの間で連続する 2 バイトが 0xFF90～FFFF の状態が発生した時となる．この条件が起こる確率は，上記

が発生しない（すなわち，再暗号化が行われない）確率の値である

$$\tilde{P}_2 = \left(1 - \frac{7 \times 16}{2^{16}}\right)^{t-1} \tag{5.3}$$

を求めて，1 からその確率を引くことで求められる．その値は

$$P_2 = 1 - \tilde{P}_2 = 1 - \left(1 - \frac{7 \times 16}{2^{16}}\right)^{t-1} \tag{5.4}$$

となる．

ブロック A において n バイト目のブロックが 0xFF となる確率は $\frac{1}{2^8}$ である．したがって，ケース 1，2 それぞれの発生確率 Q_1，Q_2 は

$$Q_1 = \frac{1}{2^8} \tag{5.5}$$

$$Q_2 = \left(1 - \frac{1}{2^8}\right) \tag{5.6}$$

となる．

これらの値を基に，1 ブロックあたりにおける暗号化のステップ数の期待値 E を計算する．ケース x において，k ステップ（$k \geq 1$）で暗号化処理が終了する確率は $P_x^{k-1}\tilde{P}_x$ となる．したがって，期待値は以下のようになる．

$$\begin{aligned} E =& Q_1 \left(\sum_{k=1}^{\infty} k P_1^{k-1} \tilde{P}_1\right) + Q_2 \left(\sum_{k=1}^{\infty} k P_2^{k-1} \tilde{P}_2\right) \\ =& Q_1 \tilde{P}_1 \left(\sum_{k=1}^{\infty} k P_1^{k-1}\right) + Q_2 \tilde{P}_2 \left(\sum_{k=1}^{\infty} k P_2^{k-1}\right) \end{aligned} \tag{5.7}$$

ただし，B が先頭ブロックの時は，A に相当するブロックが存在しないため，ケース 2 は発生しない．したがって，先頭ブロックにおける暗号化のステップ数の期待値を計算すると，以下の通りとなる．

$$E' = \tilde{P}_2 \left(\sum_{k=1}^{\infty} k P_3^{k-1}\right) \tag{5.8}$$

上記の式から，16 バイト（128 ビット）ブロック暗号を用いた時の，それぞれのブロックにおける暗号化処理ステップ数の期待値を計算すると，以下の通りとなる．

先頭ブロック　　$E' \simeq 1.025989$

先頭以外のブロック　$E \simeq 1.029106$

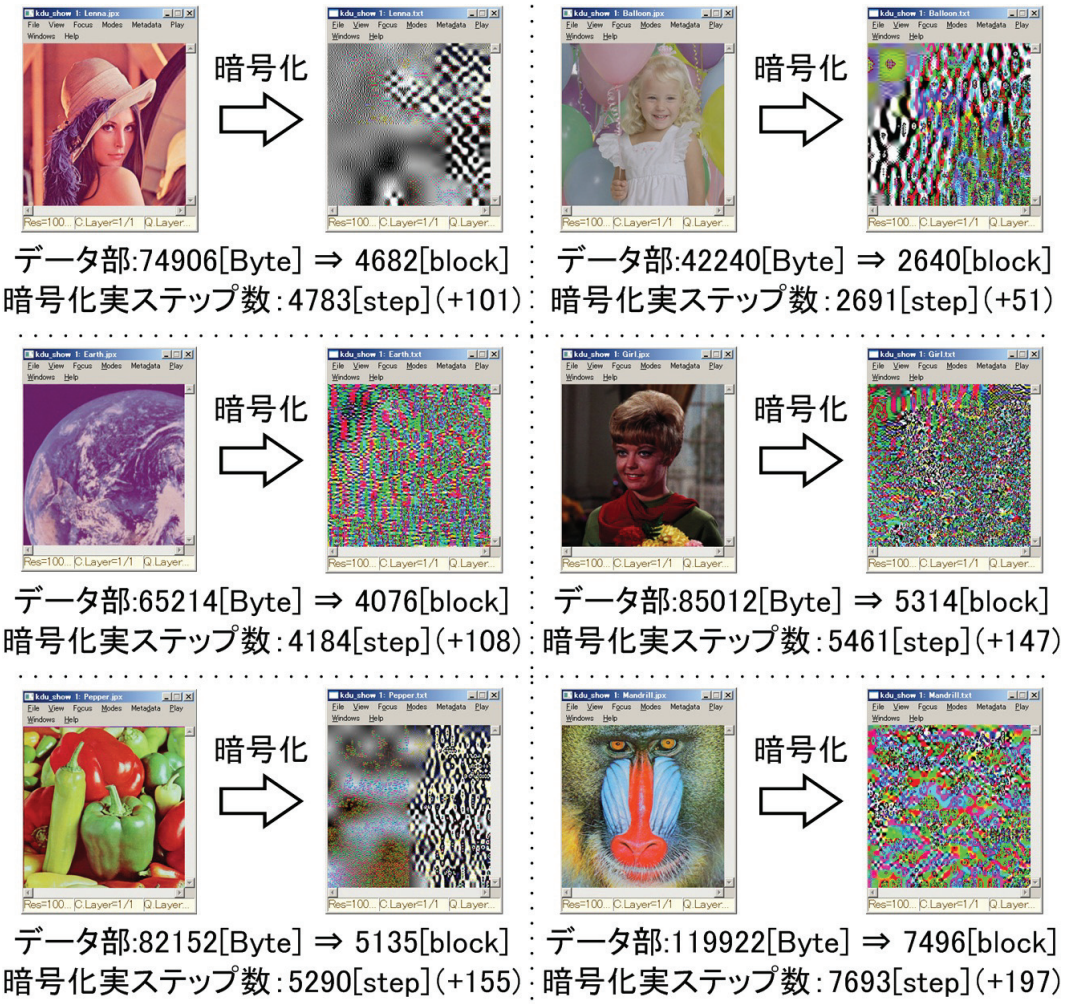

図 5.3: ブロックベース Cycle-walking 実装実験の結果

したがって，いずれのブロックにおいてもブロックベース Cycle-walking におけるステップ数と通常のブロック暗号におけるステップ数との間にほとんど差がなく，実用的であると判断できる．

なお，実際の画像を用いて簡易的に実装実験を行った．その結果を図 5.3 に示す．実験は，レイヤ構造を持たない JPEG 2000 フォーマットの画像ファイル（256×256[pixel]）を 6 種類用意し，データ部に対してブロックベース Cycle-walking による暗号化を行い，暗号化のステップ数を計測した，また，データ部にマーカコードがあるとエラーを発生する画像ビューワを用いて，データ部が暗号化されたファイルを閲覧した際の画像ビューアの動作を観測した．

実験の結果，いずれの画像ファイルにおいても，ブロック数に対する再暗号化処理の発

生回数（すなわち，通常のブロック暗号におけるステップ数に対するブロックベース Cycle-walking におけるステップ数の増加分）の比率は 2〜3 %に収まっていることを確認した．また，データ部が暗号化されたいずれの画像ファイルを画像ビューワで閲覧した場合，エラーを発生せずにサンドストーム状態の画像が表示されたことを確認した．

今後は，さらに多くの画像を用いると同時に，レイヤ毎の暗号化など様々な条件下での詳細な評価実験を行う予定である．

メモリサイズ

2.2.3 節で示した通り，本来の Cycle-walking は暗号化のために全平文に相当するメモリサイズが必要である．

一方， 5.2.3 節で示したアルゴリズム Cp_k では，用いるブロック暗号の 1 ブロック分のメモリに加え，一つ手前の暗号化されたブロックの最後の 1 バイトを記憶する必要がある．さらに 5.2.4 節で示したアルゴリズム Cp_k^{-1} では，用いるブロック暗号の 1 ブロック分のメモリに加え，事前に記憶しておく一つ手前の暗号化されたブロックの最後の 1 バイト，および次の復号処理で用いるための暗号化された現ブロックの最後の 1 バイトを記憶する必要がある．そのため必要となる最大メモリサイズはブロック暗号のブロックサイズ+2 バイトとなる [2]．ブロック暗号として AES を使用する場合，Cp_k，Cp_k^{-1} では AES のブロック長である 16 バイトに 2 バイトを加えた 18 バイトで良い．現在のデコーダに実装されるメモリ量から考えると，これは十分許容範囲と考えられる．

ここで，JPEG 2000 において特定符号は必ず 0xFF で始まっているため，ブロック間の繋ぎ目の処理について「ブロック末尾のバイトが 0xFF になったら再暗号化する」とすれば，少なくとも暗号化後のデータにおいてブロック間の繋ぎ目に特定符号は含まれず，ブロックの先頭や最終バイトを記憶するメモリを削減できる．また，ステップ数の期待値も 1.030012 と，ブロックベース Cycle-walking とほとんど変わらないため，問題が無いように見受けられる．

しかし，上記の場合は繋ぎ目の部分における暗号化後のデータが 0xFF00〜FF89 であっても再暗号化されることになり，平文と暗号文の空間が異なる（暗号文の方が小さい）サイズとなる．これは異なる平文から同一の暗号文が出力される可能性があることを意味しており，そのような暗号文を復号する際に必ずしも正しい平文に戻らないことになる．したがって，この方法では暗号アルゴリズムとして正しく機能しない．

[2] JPEG 2000 以外のフォーマットにおいて，一般的にはフォーマットで規定されている特定符号によって必要なメモリサイズはそれぞれ異なるが，必要となる追加のメモリサイズは特定符号のデータサイズの 2 倍未満となる．

5.3.3 操作モードにおけるブロックベース Cycle-walking の留意点

5.2 節において説明したブロックベース Cycle-walking を用いれば，任意のブロック暗号を用いて全ての JPEG 2000 のファイルデータを暗号化することが可能となる．しかし，このアルゴリズムのままでは，同じ平文ブロックが複数ある場合，これらは全て同じ暗号化結果になってしまう可能性がある．

このような問題に対し，通常のブロック暗号の場合は操作モードを工夫することで解決できることが知られている．しかし，ブロックベース Cycle-walking を様々な操作モードに適用させる際には，アルゴリズムの処理を抜けた箇所で起こる可能性がある特定符号の発生を考慮しなければならない．

例えば，5.2 節における Algorithm 4 を，一般的に利用される操作モードの一つである CBC モードへ適用させた場合，Algorithm 5 で示した処理となる．

Algorithm 5 $Cc_K(m)$:CBC mode of $Cp_K(m)$

Divid m into $P(0)\ldots P(n-1)$
{The size of each $P(k)$ is the block size of $E_K(\cdot)$.}
IV is set in random value.
for $i = 0$ to $n-1$ **do**
  if $i == 0$ **then**
    $P(i) \leftarrow P(i) \oplus \mathrm{IV}$
  else
    $P(i) \leftarrow P(i) \oplus P(i-1)$
  end if
  $P(i) \leftarrow Cp_K(P(i))$
end for
return $P(0)\|\ldots\|P(n-1)$

この Algorithm 5 の方法では，次の課題があることが分かっている．

Algorithm 5 における課題 Algorithm 4 はブロックの継ぎ目も含めて入力値に特定符号が含まれないことが前提であるが，Algorithm 5 はブロック毎に一度 Algorithm 4 の処理を抜け，次のブロックは直前のブロックの暗号化データとの排他的論理和を計算したものを Algorithm 4 への入力値としている．そのため，その出力値の継ぎ目に特定符号は含まれていないことが保証できない．

この課題に対し，暗号化する際において継ぎ目に特定符号は含まれていないかを判定する処理をブロックベース Cycle-walking の処理の外側にも改めて組込み，その特定符号が含ま

れている場合は除去する処理，および判定の処理を行っても最初のブロックから逐次復号できる処理を追加する必要がある．

第6章　結論

6.1　研究成果

本研究において得られた成果を以下の通りである．

第 3 章の成果 BLS 署名方式を拡張することで，特定のサーバーの関与を必要とせずに，既存の署名鍵のみで署名者の前後において，後者が自分とその前者のメッセージに署名し，この署名を順次合成していくことで署名者の署名順序や関係を保証できる多重署名／アグリゲート署名の作成が可能であることを示した．また，その作成した署名は既存の検証鍵のみでの検証が可能であることを示した．その結果，順序付き多重署名／アグリゲート署名方式，およびその合成手順を 1 対多に拡張することで構成された木構造表記型多重署名／アグリゲート署名方式が実現できることを示し，コンテンツの引用過程を表現できる著作権保証方式が実現可能であることを示した．

第 4 章の成果 墨塗り署名を拡張し，コンテンツ編集に対して変更・削除・追加に関する事前制御を可能にするシステムを提案した．提案方式では各制御に必要となる個別署名を作成し重畳させたアグリゲート署名を設定することで，各制御の可否に対応している．また，既存の墨塗り署名と比較し，部分データの署名回数や付属データ数は増加を実用上問題ない程度に抑えつつ，部分データ間へのデータの追加制御や墨塗り以外の変更制御を新たに実現し，変更・削除・追加全ての事前制御を可能にしていることを示した．

第 5 章の成果 ファイルフォーマットが定義され，かつそのフォーマットに特定符号（マーカコード）を持つデータに対して，次世代画像フォーマットである JPEG 2000 を例に，そのマーカコードを考慮したブロックベース Cycle-walking について評価を行った．その結果，以下の条件に対応することが可能となり，実用性の高い暗号化方式であることを示した．

(a) 部分暗号化したデータに特定符号を含まないことが保証される．すなわち，暗号化を解除せずに，部分暗号化したままのデータを再生器に入力しても誤作動を起こさない．

(b) 暗号化する部分を選択でき，画質の制御が可能である．

(c) 元々のファイルフォーマットが持つスケーラビリティを保持したまま，部分暗号化を行うことができる．

(d) ブロックベース Cycle-walking（Algorithm 3）は Cycle-walking と同等の安全性が評価されている．

(e) 非暗号化部分が従来方式と比較して少ない．

6.2 コンテンツ流通基盤の統括システム構築の課題

6.1 節において，本研究における成果を簡単にまとめた．この成果から， 1.3 節で示した流通基盤モデルに対する適用形態について図 6.1 に示す．

第 3 章において，階層表記型多重署名／アグリゲート署名方式を提案した．これを基に，コンテンツを編集する際に，それまでのコンテンツ作成に関与した著作者の情報を，その引用・編集過程も含めて保持しつつ，新たに編集者の署名が最新の引用・編集過程情報に更新されて追加される署名作成機能が実現される．また，その作成された署名について，全ての著作者がどのような関係によりコンテンツが作成されたかを正当に評価できる署名検証機能が実現される．

第 4 章において，コンテンツ編集時全制御方式を提案した．これを基に，コンテンツ作成時に変更・削除・追加の可否について設定し，その情報をコンテンツのメタ情報の一部として付加することが可能となる事前制御機能付き編集機能が実現される．この機能により，許可された（正当な）編集については編集後のコンテンツにおいて対応する電子署名に更新されるが，許可されていない（不正な）編集については編集後のコンテンツにおいて対応する電子署名に更新できない，これにより，電子署名の検証を行うことにより，作成されたコンテンツが正当なものであるかどうかを評価できる．

第 5 章において，特定符号を考慮した暗号化方式を提案した．これを基に，ヘッダ部や公開するコンテンツデータ部は暗号化を行わず，特別な制御や保護を行いたいコンテンツデータ部のみを暗号化できる暗号化機能が実現される．この機能により，例えばサムネイルを公開しコンテンツデータの概要を知ることができつつ，高画質なデータについては制御状況に応じて非公開にするといった細かな制御が可能となる．

本研究では，各課題に対する対応を検討し，個別に方式提案を行った．そのため，図 6.1 で示されているように，各課題に対する機能が独立して存在している．しかし，いずれの機能も要素技術であり，これらの機能を一つのアプリやサービス，流通基盤に同時に実装することも可能である．また，既存のアプリやサービスに対しても，拡張機能やプラグインとして実装することで，これまでの資産を活用するような方式として提案することも検討できる．

したがって，コンテンツ流通基盤として以下の研究を行い，統括システムを構築することが今後の目的となる．

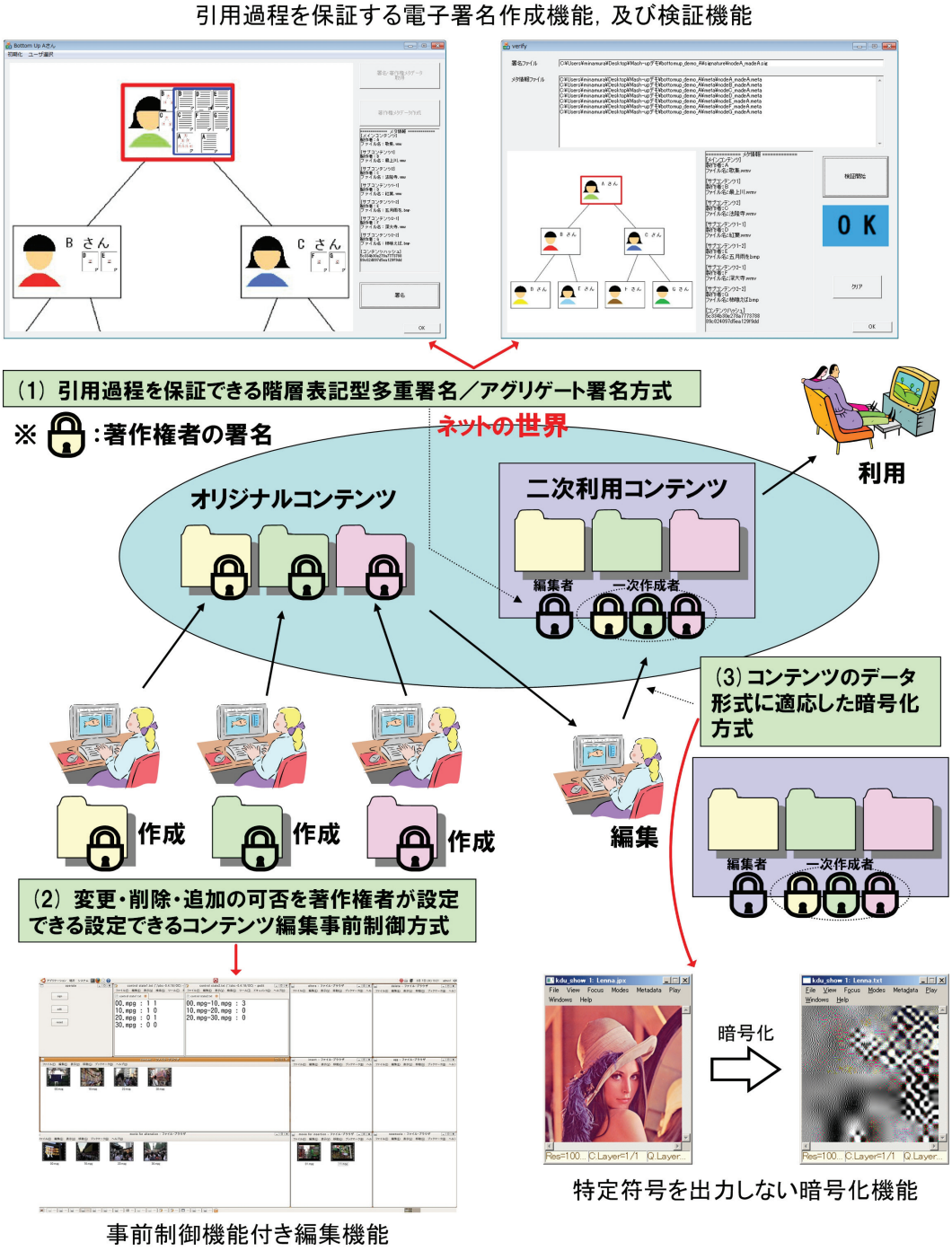

図 6.1: 研究成果のコンテンツ流通基盤への適用

- 提案方式各々が連携して機能する統括コンテンツ流通基盤の検討.
  - 連携して機能するためのプロトコルの設計.
  - 統括流通基盤としての安全性の定義，およびセキュリティ機能の構築.
- 機能実装手段に関する検討.
  - トータルな設計モデルの構築.
  - 既存のアプリやサービスへの機能拡張.
  - 実装における脆弱性の確認，およびその対策手法の提案.

6.3 今後の予定

今後について，下記の課題についての検討，および開発を行う予定である.

1. 階層表記型多重署名／アグリゲート署名方式の安全性について，コンテンツ引用過程表記手法の観点から議論を行った．今後，一般的なアグリゲート署名の安全性の議論に基づき，提案した署名方式の安全性の考察を行い，必要であればさらなる改良を検討する予定である.
2. 編集事前制御方式について， 4.5.3 節で示した n 次編集に対する課題の検討，および識別子を含む情報の管理を行う機関の設計を行う.
3. 暗号化方式について，様々な操作モード適用時における特定符号除去手順を検討する．また，データフォーマットに依存する方式であるため，どのようなデータに対し提案方式が有効であるかの評価を行う.
4. 6.2 節で示した提案方式のそれぞれの機能が連携する統括コンテンツ流通基盤についての検討を行い，モデル構築を行う．また，既存のアプリやサービスに対し，このモデルにしたがった機能拡張の可否について検討する.
5. 現在株式会社電通，および株式会社セルシスと共同で，CGM コンテンツ流通サービス基盤のプロトタイプを検討中である．セルシスではユーザが作成したコンテンツを流通させるサービス [85] を展開しているが，このサービスに許諾コード [7, 8] と本研究の成果を組み合わせて，既存のコンテンツを二次利用して新たなコンテンツとして提供できる仕組みと，その著作権を正当に表記・制御できる機能を実現し，実用化試験を行う.

謝辞

本論文をまとめるにあたり，社会人学生であり研究室に伺う時間が変則的になるなか，長期にわたり多大なる御指導・御支援をいただきました岩村惠市教授に心より感謝いたします.

博士後期課程在籍中におきましては，研究内容に関する熱心な議論や実装に関する様々なアドバイス・サポートをいただいた前助教の柿崎淑郎様，および本論文執筆時における現助教の姜玄浩様に深く感謝いたします.

本論文の審査におきましては，適切な御指導，御助言をいただいた半谷精一郎教授，村口正弘教授，浜本隆之教授，長谷川幹雄准教授に深く感謝いたします.

勤務先であった株式会社 KDDI 研究所に入社してから異動になるまでの間の研究活動に対する御支援，ならびに異動による KDDI 株式会社に帰任後の大学での研究活動に対する御理解をいただき，ご多忙にもかかわらず外部審査委員をお引き受けいただいた株式会社 KDDI 研究所執行役員の田中俊昭様に心より感謝いたします.

研究活動を行う上で様々な事務手続きを行っていただき学外発表の活動を支えていただいたのみならず，時にはストレス発散など個人的な事情にもお付き合いいただき学生生活を支えていただいた元秘書の山内かほる様，ならびに研究や大学生活において相談に乗っていただいたキヤノン株式会社の金田北洋様に深く感謝いたします.

本研究テーマを推進するにあたり，同じグループのメンバーとして研究に対する議論や実装評価のとりまとめなどいろいろと協力いただいた修了生の白川瑞樹様，佐野達彦様，本論文執筆時における現役学生の伊佐仁様，岩崎友哉様，小倉圭司様，古賀克磨様，中村政秀様に感謝の意を表すと同時に，3 年間の学生生活において年齢差を超えてお付き合いいただいた岩村研究室の卒業生，修了生，現役の学生の方々に感謝いたします.

研究を進めるにあたり，株式会社電通様，株式会社セルシス様，ガラット株式会社様にはアプリケーションソフトやコンテンツの提供をいただきました．心より感謝いたします.

最後に，私の我が儘で博士を取得するため東京理科大学大学院の博士後期課程に入学したにもかかわらず，様々な形でサポートしていただいた父，母，姉，弟に感謝いたします.

参考文献

[1] YouTube: http://www.youtube.com/.

[2] 著作権法: 1970.

[3] 電波産業会: “デジタル放送におけるアクセス制御方式,” ARIB STD-B25 6.3 版, http://www.arib.or.jp/english/html/overview/doc/2-STD-B25v6_3.pdf, 2013.

[4] 4C Entity: “Content Protection for Recordable Media,” http://www.4centity.com/.

[5] AACS LA: “Advanced Access Content System,” http://www.aacsla.com/home.

[6] Creative Commons: http://creativecommons.org/.

[7] 中西康浩, 木下信幸: “ブロードバンド時代に臨む MPEG 標準化動向とコンテンツ管理技術:4.「許諾コード」方式 - 利用許諾符号フレームワークとその体系 -,” 情報処理, Vol.48, No.10, pp.1128–1136, 情報処理学会, 2007.

[8] International Electrotechnical Commission: “Multimedia Home Server Systems - Digital Rights Permission Code,” International Standard IEC 62227 ed1.0, 2008.

[9] 梶克彦, 長尾確: “Annphony:メタコンテンツ処理のためのプラットフォーム,” 情報科学技術レターズ –FIT 2006, Vol.5, pp.381–384, 電子情報通信学会, 2006.

[10] 梶克彦, 長尾確: “部分引用の管理に基づく web コンテンツのマッシュアップ,” 情報処理学会第 69 回全国大会, 5D-1, 2007.

[11] W3C: “Resource Description Framework (RDF)”, http://www.w3.org/RDF/, 1999.

[12] Itakura, K., and Nakamura, K.: “A Public-key Cryptosystem Suitable for Digital Multisignatures,” NEC Research&Development, Vol.71, pp.1–8, 1983.

[13] Boneh, D., Gentry, C., Lynn, B., and Shacham, H.: “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps,” Advances in Cryptology –*EUROCRYPT 2003*, LNCS 2656, pp.416–432, Springer, 2003.

[14] 齊藤泰一: “順序指定可能な多重署名,” 暗号と情報セキュリティシンポジウム –*SCIS ’97*, 33A, 1997.

[15] Mitsuru, T.: “An Order-Specified Multisignature Scheme Secure against Active Insider Attacks,” Information Security and Privacy –*ACISP 2002*, LNCS 2384, pp.328-345, Springer, 2002.

[16] 藤田邦彦, 塚田恭章: “コンテンツ循環に対応できるデジタル著作権管理記述言語,” コンピュータセキュリティシンポジウム –*CSS 2008*, C6-2, 2008.

[17] 藤田邦彦, 塚田恭章: “クリエイティブ・コモンズ利用許諾の形式意味論,” 情報処理学会論文誌, Vol.49, No.9, pp.3165–3179, 2008.

[18] 秦野康生, 宮崎邦彦, 金子敏信: “暗号技術を用いたディジタルコンテンツの閲覧・編集制御に関する一考察,” コンピュータセキュリティシンポジウム –*CSS 2008*, C6-5, 2008.

[19] Taubman, D. S., and Marcellin, M. W.: “JPEG2000: Image Compression Fundamentals, Standards and Practice,” Springer, 2001.

[20] ISO/IEC FDIS15444-1: “Information Technology - JPEG 2000 Image Coding System - Part-1: Core Coding System,” ISO/IEC JTC1/SC29/WG1, 2001.

[21] 貴家仁志, 今泉祥子, 渡邊修: “マーカコードの発生を考慮した JPEG2000 符号化画像の情報開示法,” 電子情報通信学会論文誌, Vol.J86-D-II, No.11, pp.1628–1636, 2003.

[22] 岩村惠市, 林淳一: “JPEG2000 符号化画像のマーカコード発生を回避できる暗号化方式,” 電子情報通信学会論文誌, Vol.J90-A, No.11, pp.839–850, 2007.

[23] 安藤勝俊, 渡邊修, 貴家仁志: “JPEG2000 符号化画像の情報半開示法,” 電子情報通信学会論文誌, vol.J85-D-II, No.2, pp.282–290, 2002.

[24] 安藤勝俊, 貴家仁志: “レイヤ構造を利用した JPEG2000 符号化画像の暗号化法,” 電子情報通信学会論文誌, vol.J85-A, No.10, pp.1091–1099, 2002.

[25] Takagi, A., and Kiya, H.: “Scrambling of MPEG Video by Exchanging Motion Vectors,” IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E87-A, No.8, pp.2054–2057, 2004.

[26] Schneier, B.: “Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish),” Fast Software Encryption –*FSE ’93*, LNCS 809, pp.191–204, Springer, 1994.

[27] Boneh, D., Lynn, B., and Shacham, H.: “Short Signatures from the Weil Pairing,” Advances in Cryptology –*ASIACRYPT 2001*, LNCS 2248, pp.514–532, Springer, 2001.

[28] Boldyreva, A.: “Efficient threshold signature, multisignature and blind signature schemes based on the Gap-Diffie-Hellman-group signature scheme,” Cryptology ePrint Archive, Report 2002/118, http://eprint.iacr.org/2002/118, 2002.

[29] Boldyreva, A.: “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme,” Public Key Cryptography –*PKC 2003*, LNCS 2567, pp.31–46, Springer, 2003.

[30] Lin, C. Y., Wu, T. C., and Zhang, F.: “A Structured Multisignature Scheme from the Gap Diffie-Hellman Group,” Cryptology ePrint Archive, Report 2003/090, http://eprint.iacr.org/2003/090, 2003.

[31] 稲村勝樹, 田中俊昭: “コンテンツの二次利用を実現する著作権保証方式,” 暗号と情報セキュリティシンポジウム –*SCIS 2009*, 1B2-4, 2009.

[32] 稲村勝樹, 渡辺龍, 田中俊昭: “回覧文書閲覧確認に適した階層表記型多重署名方式の提案と実装評価,” 電子情報通信学会論文誌, Vol.J93-B, No.10, pp.1378–1387, 2010.

[33] 齊藤旭, 山田裕也, 岩村惠市: “編集可能コンテンツに対する墨塗り署名を用いた電子署名システムの提案,” 情報処理学会研究報告, コンピュータセキュリティ研究会 –*CSEC*, 2007-CSEC-39, Vol.2007, No.126, pp.49–54, 2007.

[34] 泉雅巳, 伊豆哲也, 國廣昇, 太田和夫: “墨塗り・削除署名の拡張,” 情報処理学会研究報告, コンピュータセキュリティ研究会 –*CSEC*, 2007-CSEC-38, Vol.2007, No.71, pp.355–361, 2007.

[35] Izu, T., Kunihiro, N., Ohta, K., Takenaka, M., and Yoshioka, T.: “A Sanitizable Signature Scheme with Aggregation,” Information Security Practice and Experience –*ISPEC 2007*, LNCS 4464, pp.51–64, Springer, 2007.

[36] 伊豆哲也, 佐野誠, 國廣昇, 太田和夫, 武仲正彦: “Aggregate 署名を用いた墨塗り署名方式,” 暗号と情報セキュリティシンポジウム –*SCIS 2007*, 2C4-3, 2007.

[37] Miyazaki, K., Hanaoka, G., and Imai, H.: “Invisibly Sanitizable Digital Signature Scheme,” IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E91-A, No.1, pp.392–402, 2008.

[38] Black, J., and Rogaway, P.: “Ciphers with Arbitrary Finite Domains,” The Cryptographer’s Track at the RSA Conference –*CT-RSA 2002*, LNCS 2271, pp.114–130, Springer, 2002.

[39] Stutz, T., and Uhl, A.: “Efficient Format-Compliant Encryption of Regular Languages: Block-Based Cycle-Walking,” Communications and Multimedia Security, LNCS 6109, pp.81–92, 2010.

[40] Washington, L. C.: “Elliptic Curves: Number Theory and Cryptography Second Edition,” Discrete Mathematics and Its Applications, Chapman and Hall/CRC, 2008.

[41] 岡本栄司, 岡本健, 金山直樹: “ペアリングに関する最近の研究動向,” IEICE Fundamentals Review, Vol.1, No.1, pp.51–60, 2007.

[42] Silverman, J. H., and Tate, J.: “Rational Points on Elliptic Curves,” Undergraduate Texts in Mathematics, Springer, 1994.

[43] Boneh, D., and Franklin, M. K.: “Identity-Based Encryption from the Weil Pairing,” Advances in Cryptology –*CRYPTO 2001*, LNCS 2139, pp.213–229, Springer, 2001.

[44] Frey, G., and Rück, H. G.: “A Remark Concerning m-divisibility and the Discrete Logarithm in the Divisor Class Group of Curves,” Math. Comp., Vol.62, No.206, pp.865–874, 1994.

[45] Diffie, W., and Hellman, M. E.: “New Directions in Cryptography,” IEEE Trans. on Information Theory, Vol.22, Issue 6, pp.644–654, 1976.

[46] 森山大輔, 西巻陵, 岡本龍明: “公開鍵暗号の数理,” シリーズ応用数理, Vol.2, 共立出版, 2011.

[47] Shanks, D.: “Class Number; a Theory of Factorization and Genera,” Symposia in Pure Mathematics, Vol.20, pp.415–440, American Mathematical Society, 1971.

[48] Pollard, J. M.: “A Monte Carlo Method for Factorization,” BIT Numerical Mathematics, Vol.15, Issue 3, pp.331–334, 1975.

[49] Schirokauer, O.: “Using Number Fields to Compute Logarithms in Finite Fields,” Mathematics of Computation, Vol.69, No.231, pp.1267–1283, American Mathematical Society, 2000.

[50] Adleman, L. M.: “The Function Field Sieve,” Algorithmic Number Theory Symposium –*ANTS*-I, LNCS 877, pp.108–121, Springer, 1994.

[51] 岡本栄司: “暗号理論入門 (第 2 版),” 共立出版, 2002.

[52] Gentry, C.: “Computing Arbitrary Functions of Encrypted Data,” Communications of the ACM, Vol.53, No.3, pp.97–105, 2010.

[53] National Institute of Standards and Technology: “Announcing the ADVANCED ENCRYPTION STANDARD (AES),” FIPS 197, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[54] 松井充: “ブロック暗号アルゴリズム MISTY,” 電子情報通信学会技術研究報告, 情報セキュリティ研究会 –*ISEC*, Vol.96, NO.167, pp.35–48, 1996.

[55] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., and Tokita, T.: “Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis,” Selected Areas in Cryptography –*SAC 2000*, LNCS 2012, pp.39–56, Springer, 2000.

[56] Rivest, R. L.: “The RC4 Encryption Algorithm,” RSA Data Security, Inc., 1992.

[57] Kiyomoto, S., Tanaka, T., and Sakurai, K.: “K2: A Stream Cipher Algorithm using Dynamic Feedback Control,” Security and Cryptography –*SECRYPT 2007*, pp.204-213, INSTICC Press, 2007.

[58] Rivest, R. L., Shamir, A., and Adleman, L.M.: “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Communications of the ACM, Vol.21, No.2, pp.120–126, 1978.

[59] El Gamal, T.: “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” Advances in Cryptology –*CRYPTO ’84*, LNCS 196, pp.10–18, Springer, 1984.

[60] Romine, C.: “Digital signature standard (DSS),” Federal Information Processing Standards Publication, FIPS PUB 186–4, National Institute of Standards and Technology, 2013.

[61] Nash, A., Duane, W., Joseph, C., and Brink, D.: “PKI: Implementing and Managing E-Security,” RSA Press. Mcgraw-Hill Osborne Media, 2001.

[62] Okamoto, T., and Pointcheval, D.: “The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes,” Public Key Cryptography –*PKC 2001*, LNCS 1992, pp.104–118, Springer, 2001.

[63] Joux, A., and Nguyen, K.: “Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups,” Cryptology ePrint Archive, Report 2001/003, http://eprint.iacr.org/2001/003, 2001.

[64] Joux, A., and Nguyen, K.: "Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups," Springer J. of Cryptology, Vol.16, No.4 pp.239–247, 2003.

[65] Joux, A.: "A One Round Protocol for Tripartite Diffie-Hellman," Algorithmic Number Theory Symposium –*ANTS*-IV, LNCS 1838, pp.385–394, Springer, 2000.

[66] Lysyanskaya, A.: "Unique Signatures and Verifiable Random Functions from the DH-DDH Separation," Advances in Cryptology –*CRYPTO 2002*, LNCS 2442, pp.597–612, Springer, 2002.

[67] Cha, J. C., and Cheon, J. H.: "An Identity-Based Signature from Gap Diffie-Hellman Groups," Public Key Cryptography –*PKC 2003*, LNCS 2567, pp.18–30, Springer, 2003.

[68] Nicolosi, A., and Mazières, D.: "Secure Acknowledgment of Multicast Messages in Open Peer-to-Peer Networks," Peer-to-Peer Systems III –*IPTPS 2004*, LNCS 3279, pp.259–268, Springer, 2004.

[69] Bender, A., Katz, J., and Morselli, R.: "Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles," Theory of Cryptography Conference –*TCC 2006*, LNCS 3876, pp.60–79, Springer, 2006.

[70] Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., and Waters, B.: "Sequential Aggregate Signatures and Multisignatures Without Random Oracles," Advances in Cryptology –*EUROCRYPT 2006*, LNCS 4004, pp.465–485, Springer, 2006.

[71] Shacham, H., and Waters, B.: "Efficient Ring Signatures Without Random Oracles," Public Key Cryptography –*PKC 2007*, LNCS 4450, pp.166–180, Springer, 2007.

[72] Law, L., and Matt, B. J.: "Finding Invalid Signatures in Pairing-Based Batches," IMA International Conference on Cryptography and Coding, LNCS 4887, pp.34–53, Springer, 2007.

[73] Boldyreva, A., Gentry, C., O'Neill, A., and Yum, D. H.: "Ordered Multisignatures and Identity-Based Sequential Aggregate Signatures, with Applications to Secure Routing," ACM Conference on Computer and Communications Security –*CCS 2007*, pp.276–285, ACM-PRESS, 2007.

[74] Matt, B. J.: "Identification of Multiple Invalid Signatures in Pairing-Based Batched Signatures," Public Key Cryptography –*PKC 2009*, LNCS 5443, pp.337–356, Springer, 2009.

[75] Gentry, C., and Silverberg, A.: “Hierarchical ID-based Cryptography,” Advances in Cryptology –*ASIACRYPT 2002*, LNCS 2501, pp.548–566, Springer, 2002.

[76] Shamir, A.: “Identity-Based Cryptosystems and Signature Schemes,” Advances in Cryptology –*CRYPTO ’84*, LNCS 196, pp.47–53, Springer, 1984.

[77] 今泉祥子, 藤吉正明, 渡邊修, 貴家仁志: “結託攻撃耐性を有する JPEG 2000 の階層性を考慮したアクセス制御型暗号化法,” 暗号と情報セキュリティシンポジウム –*SCIS 2006*, 4F1-5, 2006.

[78] 今泉祥子, 藤吉正明, 貴家仁志: “再帰型ハッシュ連鎖を用いた暗号鍵生成方式と多次元階層的アクセス制御への適用,” 映像情報メディア学会誌, Vol.65, No.2, pp.193–202, 2011.

[79] Yamamoto, D. and Ogata, W: “Structured Aggregate Signatures,” Symposium on Cryptography and Information Security –*SCIS 2006*, 3A4-3, 2006.

[80] 白川瑞樹, 岩村惠市: “ボトムアップ型デジタルコンテンツ編集システム,” 情報処理学会研究報告, コンピュータセキュリティ研究会 –*CSEC*, Vol.2010-CSEC-49, No.7, pp.1–6, 2010.

[81] Komano, Y., Ohta, K., Shimbo, A., and Kawamura, S.: “On the Security of Probabilistic Multisignature Schemes and Their Optimality,” Cryptology in Malaysia –*Mycrypt 2005*, LNCS 3715, pp.132–150, Springer, 2005.

[82] Komano, Y., Ohta, K., Shimbo, A., and Kawamura, S.: “Provably Secure Multisignatures in Formal Security Model and Their Optimality,” IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E91-A, No.1, pp.107–118, 2008.

[83] Erdös, P.: “Some Applications of Ramsey’s Theorem to Additive Number Theory,” Europ. J. Combinatorics, Vol.1, pp.43–46, 1980.

[84] Fukushima, K., Kiyomoto, S., Tanaka, T., and Sakurai, K.: “Analysis of Program Obfuscation Schemes with Variable Encoding Technique,” IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E91-A, No.1, pp.316–329, 2008.

[85] 創作活動応援サイト CLIP: `http://www.clip-studio.com/clip_site/`.

発表文献／技術展示

【学会論文誌 (査読付)】

1. 稲村勝樹, 岩村惠市: “新しい階層表記型アグリゲート署名を用いたコンテンツ引用過程表記手法,” 情報処理学会論文誌, Vol.53, No.9, pp.2267–2278, 2012.
2. 稲村勝樹, 齊藤旭, 岩村惠市: “拡張墨塗り署名を用いたコンテンツ編集事前制御システム,” 電気学会論文誌 C, Vol.133, No.4, pp.802–815, 2013.
3. Inamura, M., and Iwamura, K.: “Content Approval Systems with Expansions of a New Pair-connected-structured Aggregate Signature,” IGI Global International J. of E-Entrepreneurship and Innovation, Vol.4, Issue 2, pp.15–37, 2013.

【国際会議予稿集 (査読付)】

（筆頭著者）

1. Inamura, M., Iwamura, K., Watanabe, R., Nishikawa, M., and Tanaka, T.: “A New Tree-structure-specified Multisignature Scheme for a Document Circulation System,” Security and Cryptography –*SECRYPT 2011*, pp.362–369, SciTePress, 2011.
2. Inamura, M., and Iwamura, K.: “A Structured Aggregate Signature Scheme with Pairing-based Cryptography,” International Conference on Digital Information Processing and Communications –*ICDIPC2013*, pp.64–73, SDIWC Digital Library, 2013.

（共著）

1. Sano, T., Kakizaki, Y., Inamura, M., Echizen, I., and Iwamura, K.: “A Digital Signature Scheme Enabling Pre-control of Content Editing for Secondary Use,” International Conference on Digital Information Processing and Communications –*ICDIPC2013*, pp106–113, SDIWC Digital Library, 2013.

【国内会議予稿集 (査読無)】

（筆頭著者）

1. 稲村勝樹, 岩村惠市: “Cycle-walking の拡張に関する考察,” コンピュータセキュリティシンポジウム *–CSS 2013*, 2C1-4, 2013.

（共著）

1. 佐野達彦, 柿崎淑郎, 稲村勝樹, 岩村惠市: “コンテンツ二次利用に適した編集制御可能な電子署名システム,” コンピュータセキュリティシンポジウム *–CSS 2011*, 3D2-3, 2011.
2. 白川瑞樹, 稲村勝樹: “編集順序を表現できる事前制御可能なコンテンツ編集システム,” 暗号と情報セキュリティシンポジウム *–SCIS 2012*, 4F2-3, 2012.
3. 伊佐仁, 小出雅史, 稲村勝樹, 岩村惠市: “二次利用コンテンツの構成集合を識別可能とするデジタル署名方式,” 第 11 回情報科学技術フォーラム *–FIT 2012*, L-004, 2012.
4. 伊佐仁, 小出雅史, 稲村勝樹, 岩村惠市: “コンテンツの再編集を許可する木構造表記型多重署名方式,” コンピュータセキュリティシンポジウム *–CSS 2012*, 2C2-2, 2012.
5. 岩崎友哉, 稲村勝樹, 岩村惠市: “ID ベース署名の順序付きアグリゲート署名への拡張に関する検討,” 暗号と情報セキュリティシンポジウム *–SCIS 2013*, 2A2-1, 2013.
6. 伊佐仁, 稲村勝樹, 岩村惠市: “グループ化を用いた木構造表記型多重署名方式,” 暗号と情報セキュリティシンポジウム *–SCIS 2013*, 2C3-3, 2013.
7. 佐野達彦, 岩村惠市, 稲村勝樹, 柿崎淑郎: “コンテンツ循環に適した事前制御可能なコンテンツ編集システムの実装と評価,” 暗号と情報セキュリティシンポジウム *–SCIS 2013*, 2C4-1, 2013.
8. 小倉圭司, 佐野達彦, 稲村勝樹, 岩村惠市: “編集順序を保証するコンテンツ編集アプリケーションの開発,” 電子情報通信学会技術研究報告, マルチメディア情報ハイディング・エンリッチメント研究会 *–EMM*, Vol.113, No.66, pp.13–18, 2013.
9. 岩崎友哉, 稲村勝樹, 岩村惠市: 情報処理学会研究報告, コンピュータセキュリティ研究会 *–CSEC*, Vol.2013-CSEC-62, No.14, pp.1–6, 2013.

【技術展示】

1. 佐野達彦, 伊佐仁, 稲村勝樹, 岩村惠市: “コンテンツ二次利用に適した事前編集制御システム,” 暗号と情報セキュリティシンポジウム *–SCIS 2012*, 2012.

付 録A　階層表記型アグリゲート署名方式の拡張

A.1　拡張の概要

3.3.2 節において順序付きアグリゲート署名方式，および 3.3.3 節においてその拡張である木構造表記型アグリゲート署名方式について提案した．この提案方式は 2 人の署名者の関係を示す署名を組み込むことで 2 人が連続していることを示し，これを順序方向，あるいは木構造方向に拡張することで構造を表記している．

本論文のではコンテンツの引用関係を示す目的から順序付きアグリゲート署名方式と木構造表記型アグリゲート署名方式の 2 種類についてのみ 3.3 節で説明したが，2 人の関係の組み合わせで示すことができればどのような構造でも表記できる．そこで，他の代表的な構造 [30] へ拡張したプロトコルを説明する．なお，その構造については図 A.1 に示す.

A.2　記号，前提条件，および要求条件

使用される記号，前提条件，セキュリティの要求条件について以下に示す.

記号:

$\mathbb{G}_1, \mathbb{G}_2$**:** ペアリングの演算が可能な楕円曲線上の点の集合（$\mathbb{G}_1$ の要素をペアリング関数の第 1 引数，$\mathbb{G}_2$ の要素をペアリング関数の第 2 引数とする）.

g**:** $\mathbb{G}_1$ の要素である生成元.

e**:** ペアリング関数.

u_y**:** 第 y 署名者（ただし，$u_y \neq \text{null}$）.

x_y, v_y**:** u_y の署名鍵（$x_y \in \mathbb{Z}_p^*$），および検証鍵（$v_y = x_y g$）.

$\mathbb{L}_y$**:** u_0 から u_y までの署名者の関係を示した接続情報

m_y**:** u_y が署名対象とするメッセージ.

H**:** $\{0,1\}^* \rightarrow \mathbb{G}_2$ となる一方向性ハッシュ関数（例:*MapToGroup* [27, 43]）.

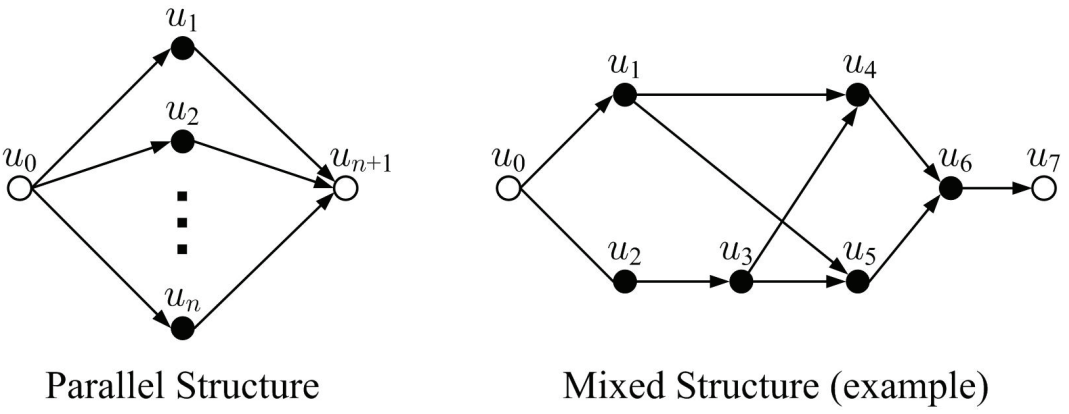

図 A.1: 代表的な構造

σ_y**:** u_0 から u_y までの順序付きアグリゲート署名.

また，前提条件，およびセキュリティの要求条件については 3.3.1 節で示したものと同じである.

A.3 Parallel Structure を表記するアグリゲート署名方式

A.3.1 鍵生成

$g \in \mathbb{G}_1$ を生成元とする．署名者 u_i について $x_i \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_i = x_i g$ を計算する．x_i を署名者 u_i の署名鍵，v_i を署名者 u_i の検証鍵とする.

A.3.2 署名作成

PS1. 最初の署名者 u_0 は，メッセージ m_0 から $h_0 = H(m_0)$ を求め， 2.3.3 節で説明した BLS 署名と同様な署名作成処理を行うことで

$$\sigma_0 = x_0 h_0 \tag{A.1}$$

を計算する．また，

$$\mathbb{L}_0 = \{(\text{null}, u_0)\} \tag{A.2}$$

を作成する．この σ_0，$\mathbb{L}_0$，および m_0（または h_0）を署名者 $u_1, \ldots, u_n$ に送信する.

PS2. 署名者 u_i（$1 \leq i \leq n$）は，署名者 u_0 から受信した m_0 を用いて $h_0 = H(m_0)$ を求める（または h_0 を受信したら，それを利用する）．さらに署名者 u_i は，自分が本来署名したいメッセージ m_i から $h_i = H(m_i)$ を求める．これと署名者 u_0 から受信した σ_0 を用いて

$$\begin{aligned}\sigma_i &= \sigma_0 + x_i h_0 + x_i h_i \\ &= x_0 h_0 + x_i h_0 + x_i h_i\end{aligned} \tag{A.3}$$

を計算する．また，受信した $\mathbb{L}_0$ を用いて

$$\begin{aligned}\mathbb{L}_i &= \mathbb{L}_0 + \{(u_0, u_i)\} \\ &= \{(\text{null}, u_0), (u_0, u_i), \}\end{aligned} \tag{A.4}$$

を作成する．この σ_i，$\mathbb{L}_i$，および m_i（または h_i）を最後の署名者 u_{n+1} に送信する．

PS3. 最後の署名者 u_{n+1} は，直前の署名者 $u_1, \ldots, u_n$ から受信した $m_1, \ldots, u_n$ を用いて $h_1 = H(m_1), \ldots, h_n = H(m_n)$ を求める（または h_i を受信したら，それを利用する）．さらに署名者 u_{n+1} は，自分が本来署名したいメッセージ m_{n+1} から $h_{n+1} = H(m_{n+1})$ を求める．これと署名者 $u_1, \ldots, u_n$ から受信した $\sigma_1, \ldots, \sigma_n$ を用いて

$$\begin{aligned}\sigma_{n+1} &= \sum_{j=1}^{n} \left(\sigma_j + x_{n+1} h_j\right) - (n-1)\sigma_1 + x_{n+1} h_{n+1} \\ &= x_0 h_0 + \sum_{j=1}^{n} \left(x_j h_0 + x_j h_j + x_{n+1} h_j\right) + x_{n+1} h_{n+1}\end{aligned} \tag{A.5}$$

を計算する．また，受信した $\mathbb{L}_1, \ldots, \mathbb{L}_n$ を用いて

$$\begin{aligned}\mathbb{L}_n &= \sum_{j=1}^{n} \mathbb{L}_j + \{(u_j, u_{n+1})\} \\ &= \{(\text{null}, u_0), (u_0, u_1), \ldots, (u_0, u_n), (u_1, u_{n+1}), \ldots, (u_1, u_{n+1})\}\end{aligned} \tag{A.6}$$

を作成する．この σ_{n+1}，および $\mathbb{L}_{n+1}$ を，署名者 $u_0, \ldots, u_{n+1}$ の，署名対象 $m_0, \ldots, m_{n+1}$ に対するアグリゲート署名として公開する．

A.3.3 署名検証

PV1. 検証者は，$\mathbb{L}_{n+1}$ に示されている全ての署名者の検証鍵 $v, \ldots, v_n$，および署名対象となる全てのメッセージ $m_0, \ldots, m_{n+1}$ を集める．

PV2. 検証者は，集めたメッセージから $h_0 = H(m_0), \ldots, h_{n+1} = H(m_{n+1})$ を求める．

PV3. 検証者はペアリングを用いて以下の判定を行う．

$$e(g, \sigma_{n+1}) = e(v_0, h_0) \cdot \prod_{j=1}^{n} e(v_j, h_0 + h_j) \cdot \prod_{j=1}^{n} e(v_{n+1}, h_j) \cdot e(v_{n+1}, h_{n+1}) \tag{A.7}$$

もし正しく署名が作成されていれば，左辺は

$$\begin{aligned} e(g, \sigma_n) =& e(g, x_0 h_0 + \sum_{j=1}^{n} (x_j h_0 + x_j h_j + x_{n+1} h_j) + x_{n+1} h_{n+1}) + x_{n+1} h_{n+1}) \\ =& e(g, x_0 h_0) \cdot \prod_{j=1}^{n} e(g, x_j (h_0 + h_j)) \cdot \prod_{j=1}^{n} e(g, x_{n+1} h_j) \cdot e(g, x_{n+1} h_{n+1}) \\ =& e(x_0 g, h_0) \cdot \prod_{j=1}^{n} e(x_j g, h_0 + h_j) \cdot \prod_{j=1}^{n} e(x_{n+1} g, h_j) \cdot e(x_{n+1} g, h_{n+1}) \\ =& e(v_0, h_0) \cdot \prod_{j=1}^{n} e(v_j, h_0 + h_j) \end{aligned} \tag{A.8}$$

となり，右辺と一致する．

A.4 Mixed Structure を表記するアグリゲート署名方式

A.4.1 鍵生成

$g \in \mathbb{G}_1$ を生成元とする．署名者 u_i について $x_i \in \mathbb{Z}_p^*$ を選び（全ての署名者の署名鍵は各々異なるものとする），$v_i = x_i g$ を計算する．x_i を署名者 u_i の署名鍵，v_i を署名者 u_i の検証鍵とする．

A.4.2 署名作成

図 A.1 で示した構造を例に説明する．

MS1. 最初の署名者 u_0 は，メッセージ m_0 から $h_0 = H(m_0)$ を求め，2.3.3 節で説明した BLS 署名と同様な署名作成処理を行うことで

$$\sigma_0 = x_0 h_0 \tag{A.9}$$

を計算する．また，

$$\mathbb{L}_0 = \{(\text{null}, u_0)\} \tag{A.10}$$

を作成する．この σ_0，$\mathbb{L}_0$，および m_0（または h_0）を署名者 u_1，および u_2 に送信する．

MS2. 署名者 u_1 は，署名者 u_0 から受信した m_0 を用いて $h_0 = H(m_0)$ を求める（または h_0 を受信したら，それを利用する）．さらに署名者 u_1 は，自分が本来署名したいメッセージ m_1 から $h_1 = H(m_1)$ を求める．これと署名者 u_0 から受信した σ_0 を用いて

$$\sigma'_1 = x_1 h_1 \tag{A.11}$$

$$\sigma_1 = \sigma_0 + x_1 h_0 + x_1 h_1 \tag{A.12}$$

を計算する．また，受信した $\mathbb{L}_0$ を用いて

$$\mathbb{L}_0 = \{(\text{null}, u_0), (u_0, u_1)\} \tag{A.13}$$

を作成する．この σ_1，$\mathbb{L}_1$，および m_1（または h_1）を署名者 u_4，および u_5 に送信する．

MS3. 署名者 u_2 は，署名者 u_0 から受信した m_0 を用いて $h_0 = H(m_0)$ を求める（または h_0 を受信したら，それを利用する）．さらに署名者 u_1 は，自分が本来署名したいメッセージ m_2 から $h_2 = H(m_2)$ を求める．これと署名者 u_0 から受信した σ_0 を用いて

$$\sigma'_2 = x_2 h_2 \tag{A.14}$$

$$\sigma_2 = \sigma_0 + x_2 h_0 + x_2 h_2 \tag{A.15}$$

を計算する．また，受信した $\mathbb{L}_0$ を用いて

$$\mathbb{L}_0 = \{(\text{null}, u_0), (u_0, u_2)\} \tag{A.16}$$

を作成する．この σ_2，$\mathbb{L}_2$，および m_2（または h_2）を署名者 u_3 に送信する．

MS4. 署名者 u_3 は，署名者 u_2 から受信した m_2 を用いて $h_2 = H(m_2)$ を求める（または h_2 を受信したら，それを利用する）．さらに署名者 u_3 は，自分が本来署名したいメッセージ m_3 から $h_3 = H(m_3)$ を求める．これと署名者 u_2 から受信した σ_2 を用いて

$$\sigma'_3 = x_3 h_3 \tag{A.17}$$

$$\sigma_3 = \sigma_2 + x_3 h_2 + x_3 h_3 \tag{A.18}$$

を計算する．また，受信した $\mathbb{L}_0$ を用いて

$$\mathbb{L}_3 = \mathbb{L}_2 + \{(u_2, u_3)\} \tag{A.19}$$

を作成する．この σ_3，$\mathbb{L}_3$，および m_3（または h_3）を署名者 u_4，および u_5 に送信する．

MS5. 署名者 u_4 は，署名者 u_1，u_3 から受信した m_1，m_3 を用いて $h_1 = H(m_1)$，$h_3 = H(m_3)$ を求める（または h_1，h_3 を受信したら，それを利用する）．さらに署名者 u_4 は，自分が本来署名したいメッセージ m_4 から $h_4 = H(m_4)$ を求める．これと署名者 u_1，u_3 から受信した σ_1，σ_3 を用いて

$$\sigma'_4 = x_4 h_4 \tag{A.20}$$

$$\sigma_4 = \sigma_1 + \sigma_3 - \sigma_0 + x_4 h_1 + x_4 h_3 + x_4 h_4 \tag{A.21}$$

を計算する．また，受信した $\mathbb{L}_1$，$\mathbb{L}_3$ を用いて

$$\mathbb{L}_4 = \mathbb{L}_1 + \mathbb{L}_3 + \{(u_1, u_4), (u_3, u_4)\} \tag{A.22}$$

を作成する．この σ_4，$\mathbb{L}_4$，および m_4（または h_4）を署名者 u_6 に送信する．

MS6. 署名者 u_5 は，署名者 u_1，u_3 から受信した m_1，m_3 を用いて $h_1 = H(m_1)$，$h_3 = H(m_3)$ を求める（または h_1，h_3 を受信したら，それを利用する）．さらに署名者 u_5 は，自分が本来署名したいメッセージ m_5 から $h_5 = H(m_5)$ を求める．これと署名者 u_1，u_3 から受信した σ_1，σ_3 を用いて

$$\sigma'_5 = x_5 h_5 \tag{A.23}$$

$$\sigma_5 = \sigma_1 + \sigma_3 - \sigma_0 + x_5 h_1 + x_5 h_3 + x_5 h_5 \tag{A.24}$$

を計算する．また，受信した $\mathbb{L}_1$，$\mathbb{L}_3$ を用いて

$$\mathbb{L}_5 = \mathbb{L}_1 + \mathbb{L}_3 + \{(u_1, u_5), (u_3, u_5)\} \tag{A.25}$$

を作成する．この σ_5，$\mathbb{L}_5$，および m_5（または h_5）を署名者 u_6 に送信する．

MS7. 署名者 u_6 は，署名者 u_4，u_5 から受信した m_4，m_5 を用いて $h_4 = H(m_4)$，$h_5 = H(m_5)$ を求める（または h_4，h_5 を受信したら，それを利用する）．さらに署名者 u_6 は，自分が本来署名したいメッセージ m_6 から $h_6 = H(m_6)$ を求める．これと署名者 u_4，u_5 から

受信した σ_4，σ_5 を用いて

$$\sigma'_6 = x_6 h_6 \tag{A.26}$$

$$\sigma_6 = \sigma_4 + \sigma_5 - \sigma_0 - \sigma' 1 - \sigma' 3 + x_6 h_4 + x_6 h_5 + x_6 h_6 \tag{A.27}$$

を計算する．また，受信した $\mathbb{L}_4$，$\mathbb{L}_5$ を用いて

$$\mathbb{L}_6 = \mathbb{L}_4 + \mathbb{L}_5 + \{(u_4, u_6), (u_5, u_6)\} \tag{A.28}$$

を作成する．この σ_6，$\mathbb{L}_6$，および m_6（または h_6）を署名者 u_7 に送信する．

MS8. 最後の署名者 u_7 は，署名者 u_6 から受信した m_6 を用いて $h_6 = H(m_6)$ を求める（または h_6 を受信したら，それを利用する）．さらに署名者 u_7 は，自分が本来署名したいメッセージ m_7 から $h_7 = H(m_7)$ を求める．これと署名者 u_6 から受信した σ_6 を用いて

$$\sigma_7 = \sigma_6 + x_7 h_6 + x_7 h_7 \tag{A.29}$$

を計算する．また，受信した $\mathbb{L}_6$ を用いて

$$\mathbb{L}_7 = \mathbb{L}_6 + \{(u_6, u_7)\} \tag{A.30}$$

を作成する．この σ_7，および $\mathbb{L}_7$ を，署名者 $u_0, \ldots, u_7$ の，署名対象 $m_0, \ldots, m_7$ に対するアグリゲート署名として公開する．

A.4.3 署名検証

PV1. 検証者は，$\mathbb{L}_7$ に示されている全ての署名者の検証鍵 $v_, \ldots, v_7$，および署名対象となる全てのメッセージ $m_0, \ldots, m_7$ を集める．

PV2. 検証者は，集めたメッセージから $h_0 = H(m_0), \ldots, h_7 = H(m_7)$ を求める．

PV3. 検証者はペアリングを用いて以下の判定を行う．

$$\begin{aligned} & e(g, \sigma_7) \\ = \; & e(v_0, h_0) \cdot e(v_1, h_0 + h_1) \cdot e(v_2, h_0 + h_2) \cdot e(v_3, h_2 + h_3) \cdot e(v_4, h_1 + h_3 + h_4) \\ & \cdot e(v_5, h_1 + h_3 + h_5) \cdot e(v_6, h_4 + h_5 + h_6) \cdot e(v_7, h_6 + h_7) \end{aligned} \tag{A.31}$$

もし正しく署名が作成されていれば，左辺は

$$
\begin{aligned}
e(g,\sigma_7) =& e(g,\sigma_0+\sum_{j=1}^{6}\sigma'_j + x_7h_7 + x_1h_0 + x_2h_0 + x_3h_2 + x4(h_1+h_3) \\
&\qquad + x_5(h_1+h_3) + x_5(h_4+h_5) + x_7h_6) \\
=& e(g,x_0h_0)\cdot e(g,x_1(h_0+h_1))\cdot e(g,x_2(h_0+h_2))\cdot e(g,x_3(h_2+h_3)) \\
&\cdot e(g,x_4(h_1+h_3+h_4))\cdot e(g,x_5(h_1+h_3+h_5)) \\
&\cdot e(g,x_6(h_4+h_5+h_6))\cdot e(g,x_7(h_6+h_7)) \\
=& e(x_0g,h_0)\cdot e(x_1g,h_0+h_1)\cdot e(x_2g,h_0+h_2) \\
&\cdot e(x_3g,h_2+h_3)\cdot e(x_4g,h_1+h_3+h_4)\cdot e(x_5g,h_1+h_3+h_5) \\
&\cdot e(x_6g,h_4+h_5+h_6)\cdot e(x_7g,h_6+h_7) \qquad \text{(A.32)}
\end{aligned}
$$

となり，右辺と一致する．