Zeutro LLC

Encryption & Data Security

OpenABE デザイン・ドキュメント

バージョン 1.0

内容

要旨

本書は、OpenABE ライブラリのソフトウェア設計説明書です。本書は、Zeutro, LLC によって作成され、Zeutro, LLC の所有物です。

属性ベース暗号化（Attribute-Based Encryption：ABE）は、暗号学的に（つまり数学的に）強制される柔軟なポリシー・ベースのアクセス制御を可能にする次世代の公開鍵暗号化である。

OpenABE は C/C++ソフトウェア・ライブラリで、いくつかの属性ベースの暗号化スキームと、認証された共通鍵暗号化、公開鍵暗号化、デジタル署名、X.509 証明書の取り扱い、鍵導出関数、擬似ランダム生成器などの他のコア暗号機能を実装しています。

この文書の目的は 2 つある。第一に、この文書では属性ベースの暗号化を紹介し、ABE が対処する問題の種類（今日のクラウド環境で一般的なものなど）を説明し、従来の暗号化アプローチと比較した場合の利点を詳述する。次に、ABE とその他の暗号化ツールのサポートを含む OpenABE ライブラリについて説明します。このドキュメントの使用目的は、ABE と OpenABE ライブラリの暗号設計を理解することである。

第 1 章

はじめに

この章では、属性ベースの暗号化（ABE）を紹介し、先行する暗号化システムの文脈に位置づける。次の章では、この技術が現在どのようにして広く利用できるようになったかを取り上げる。

1.1 先行暗号化システムの背景

暗号化の必要性 今日、機密データは多くの場合、アクセスを仲介するストレージ・サーバーによって保護されている。これらのサーバーは、許可されたユーザにはデータへのアクセスを許可し、許可されていないユーザにはアクセスを拒否するよう任されている。

このモデルのシンプルさは魅力的だが、ストレージ保護を信頼できるシステムだけに頼るのは、多くの理由から今日の環境では賢明ではない：

- 第一に、サードパーティの商用サイトにデータを保存することへの経済的圧力が高まっている。しかし、サードパーティのサーバーに単にクリアな状態で情報を保存することは、明らかに望ましくない。
- 第二に、機密データは物理的な危険にさらされやすいデバイスやマシンに保存されることが多い。例えば、ノートパソコンは簡単に紛失したり盗まれたりする可能性がある。ネットワーク帯域幅が制限されていたり、単に利用できなかったりする状況では、これらのデバイスは機密データをオフロードする好機が来るまで保存しておく必要がある。デバイスがキャプチャされた場合、機密情報が漏洩する可能性がある。
- 最後に、たとえデータが物理的に安全な場所に保存されていたとしても、漏洩する可能性はある。政府のネットワークや企業に対する標的型攻撃の数は、近年劇的に増加している。情報技術の専門家の一般的な前提は、組織のネットワークは、ある時点で不正な者によって侵入されると考えるのが妥当であるということである。

上記の問題を考えると、データ・アクセスを暗号化によって保護することが重要である。大雑把に言えば、暗号化は、ユーザ（またはデバイス）がデータを復号化できる適切な秘密鍵にアクセスしなければ理解できないようにデータを符号化する方法を提供する。この秘密鍵がなければ、攻撃者は暗号化されたデータの元の内容を知ることができない。このようにして、*基礎となるストレージやストレージ・サービスが侵害されても*、機密情報を非公開に保つことができる。

従来の暗号化システムの限界 秘密鍵暗号化（対称鍵暗号化とも呼ばれる）は、事前に共有された秘密を持つ 2 人のユーザが安全にデータを暗号化・復号化することを可能にする。現在使用されている秘密鍵暗号化方式の代表例は、Advanced Encryption Standard (AES) [10]である。秘密鍵方式は多くの場合、非常に優れた効率を提供するが、多くのアプリケーションには不十分である。これらのシステムの主な欠点は、安全に通信する前に、両

方のユーザがこの共有秘密を持っていなければならないことであり、今日の多くのオンライン設定では非現実的である。*事前の*共有秘密なしで安全に情報を交換することが、公開鍵暗号化が取り組む主な課題である。

公開鍵暗号化（非対称鍵暗号化とも呼ばれる）システムでは、暗号化されたデータは 1 人の*既知の*ユーザによって復号化される。暗号化されたデータを受け取りたいユーザは、公開鍵と秘密鍵のペアを生成する。彼女は公開鍵を公開し、これは誰でも彼女に暗号化するために使うことができ、秘密鍵は秘密にしておく。現在使用されている公開鍵暗号化方式の代表的な例は、RSA [17]（発明者である Rivest、Shamir、Adleman にちなんで命名された）である。この機能は、暗号化された電子メールや安全なウェブセッションの確立などの用途には便利ですが、より高度なデータ共有に必要な表現力には欠けています。

例えば、システム内のユーザの資格情報や役割に基づいてデータを共有したいとする。エンジニア" **AND** "レベル 5 アクセス") **OR** "監査部 "のポリシーでデータを暗号化したいユーザがいるとする。従来の暗号化技術を使用すると、そのユーザは、このポリシーに一致するすべてのユーザを検索し、列挙し、そして暗号化する必要があります。(今日の実務では、RSA を使って各ユーザに AES キーを暗号化し、この AES キーの下でデータを一度暗号化することができるだろう)。多くのポリシーの場合、このリストが長かったり、データ所有者が利用できなかったりすることを考えると、これは大変な作業である。さらに、担当者は頻繁に役割や責任を変更する。従来の公開鍵暗号化を使ってポリシーに従って情報を共有しようとすると、複数の障害にぶつかる。

- ポリシーに合致するすべてのユーザを列挙するデータベースを提供するのは複雑な作業である。データベースは継続的に更新されなければならず、データを暗号化したいユーザは接続を維持しなければならない。
- 複数のユーザがポリシーに一致する場合、それぞれのユーザに対して個別にデータを暗号化しなければならない。暗号化にかかる時間と暗号文のサイズ（およびその結果として発生するストレージや帯域幅のコスト）は、たとえポリシー自体が小さくても、ポリシーに合致するユーザ数に応じて線形に増大する。
- この方法は、さまざまな人が新しい役割に就くような状況ではうまく機能しない。あるポリシーに合致するすべてのユーザに対してデータを暗号化して保存したとする。あるユーザが、そのデータへのアクセスを許可するクレデンシャルを後で取得した場合、どうなるでしょうか？
- 最後に、どのユーザが特定のクレデンシャルを持っているかという情報自体が制限されることが多い。アフガニスタンの特殊部隊の情報を暗号化する任務を負った兵士が、このポリシーに合致する他のすべての兵士を列挙できるとは限らない。

ABE は、これらの障害を克服できる、より強力な公開鍵暗号化である。

1.2 属性ベースの暗号化入門

ここ 10 年の間に、属性ベース暗号化（Attribute-Based Encryption：ABE）と呼ばれる、データ暗号化の新しいビジョンが登場した。属性ベースの暗号化システムでは、個々のユーザに対して暗号化する代わりに、暗号文自体に*任意の*アクセス制御ポリシーを埋め込むことができる。このため、アクセス制御は暗号の難しい計算によって実行されるようになり、信頼できるサーバーに依存する必要がなくなった。

ABE システムでは、属性は「女性」、「上級生」、「CS256」、「2014」のような文字列または数字とすることができる。アクセス制御ポリシーは、"2014" **AND** "female" **AND** ("CS128" **OR** "CS256") **AND** ("freshman"

OR "sophomore")のように、これらの属性に対する任意のブーリアン式にすることができます。(ブール式を超える複雑さのポリシーについては、このサブセクションの最後で説明します)。

基本的な ABE システムでは、(暗号化に使われる) 公開パラメータを公開し、(復号化のための) 秘密鍵をユーザに配布する権威が 1 つ存在する。公開パラメータを持つ人は誰でも、適切な権限を持つユーザだけがデータを復元できるようにデータを暗号化できる。ABE には主に 3 つのタイプがある。

1. **役割ベースのアクセス制御：暗号文-ポリシーABE。**CP-ABE システムでは、属性はユーザと関連付けられ、ポリシーは暗号文と関連付けられる。属性がポリシーを満たす場合に限り、ユーザは暗号文を復号できる。

   例アリスは "女性"、"看護師"、"3 階"、"呼吸器専門医 "という属性を持つ。ボブはポリシー("doctor" **or** "nurse") **AND** ("floor 3" **or** "floor4")にマッチするスタッフのために患者の医療ファイルを暗号化する。アリスはこの患者のファイルを開くことができる。

   このシステムは、技術報告書、人事文書、医療記録など、暗号化ポリシーがあらかじめわかっている場合にうまく機能する。

2. **コンテンツベースのアクセス制御：キーポリシーABE。**KP-ABE システムでは、ポリシーはユーザ (つまりその秘密鍵) に関連付けられ、属性は暗号文に関連付けられる。ユーザが暗号文を復号できるのは、その属性が自分の (秘密鍵の) ポリシーを満たす場合のみである。

   例 Y 社は電子メールの記録を管理しており、各電子メールは、その電子メールのメタデータに関連する一連の属性 (例えば、宛先、送信元、件名、日付、IP アドレスなど) と共に暗号化されている。その後、2014 年 3 月の最初の 2 週間に、従業員エドワードによって機密情報が流出したことが判明する。Y 社は、監査人がポリシーに一致するメールを開くことができるキーを生成することができる (From：From: Edward AND (Dated: March 1-14, 2014))に合致するメールを開くことができるキーを生成することができる。これにより、監査人はメールデータベース全体を閲覧することなく、必要な情報の断片にアクセスすることができます。

   このシステムは、例えば、アナリストや監査人への情報公開、召喚状への対応、電子メールやクラウド、その他のビッグデータアプリケーションでの検索など、データのスライスへのアクセスを許可するのに適している。

3. **複数組織の役割ベースのアクセス制御：多権限 ABE。**MA-ABE システムは、複数の独立した権威をサポートできる CP-ABE システムである。つまり、複数の異なる権威がそれぞれの属性セットに基づいてユーザに秘密鍵を付与し、誰でも複数の権威からの属性に対するポリシーを使ってデータを暗号化することができる。

   例ボブの属性は「男性」、「生年月日：1960 年 10 月 14 日」、「米国パスポート」、「シニア・サイエンティスト」、「製品 Z グループ」、「雇用主：Capstone」などである。キャロルは、(「DoJ: Secret Clearance」**または**「DoD: Secret Clearance」) **AND** (「Capstone: Product Z Group」**AND**「US Passport: over the age of 35」) のような暗号化ポリシーを生成することができます。

   複数のオーソリティをサポートする機能により、大規模な組織内でも局所的な制御が可能になり (例えば、組織の各部門が独自の属性を認証できる)、信頼の境界を越えた情報の共有が容易になる。MA-ABE 機能は、Zeutro 社により特許取得済み (米国特許第 8880875 号)。MA-ABE 機能は OpenABE には含まれていませんが、Zeutro's Toolkit (ZTK)と呼ばれるより包括的なライブラリの一部として提供されており、商用ライセンスで利用可能です。

OpenABE でこれらの ABE システムをどのように実装するかについての正確な説明は、第 2 章を参照されたい。

共謀への耐性。これらすべてのシステムの基礎となるのは、*結託*攻撃や複数の鍵にアクセスする攻撃者に対するセキュリティの考え方である。求められる保証は、結託したユーザ・グループが、それぞれが個別にアクセスできるデータの合計よりも多くのデータにアクセスできないことである。つまり、特権をエスカレートさせて、誰も見ることを許可されていないデータを回復することはできない。

ポリシー（"Navy Officer" **AND** "PACOM"）**または** "TRANSCOM "に一致する人に情報を共有する例を考えてみましょう。攻撃者が 2 人の異なるユーザを侵害したとする：海軍将校」のクレデンシャルを持つキーを持つ海軍将校 Smith と、「PACOM」のクレデンシャルを持つ別のキーを持つ陸軍上等兵 Jones である。安全なシステムは、攻撃者が暗号化されたデータを復号化する能力を否定しなければならない。この攻撃に対して安全なシステムを構築することが、安全な ABE システム構築における中核的な技術的課題である。これは一般的に、ABE を技術として、同様の機能を主張する従来の公開鍵暗号化や共通鍵暗号化から「設計」されたシステムと区別するものである。

属性ベースおよび機能ベースの暗号化の背景。2005 年、ブレント・ウォーターズ博士（Zeutro のチーフ・サイエンティスト）とアミット・サハイ教授は、属性ベースの暗号化の概念的基礎を築き、ABE システムの最初の（限定的な）実現を発表した[18]。それ以来、ABE に関する 100 を超える査読付き出版物が発表され、機能性、効率性、セキュリティ分析、信頼の分散化など、基盤となる暗号機能に関する大きな進展が見られます。

OpenABE の初期リリースでは、アクセス制御ポリシーがブール式である ABE システムに焦点を当てている。

しかし、アクセス制御ポリシーは正規表現でもよい（Zeutro 米国特許第 8566601 号）。あるいは、ある程度の効率を犠牲にしても、回路で表現できるポリシーであれば何でもよい。

時を経て、ABE の概念は*機能的暗号化（Functional Encryption）*という概念に広がったが、ここでは ABE との関係を明確にするために*機能的暗号*化を定義する。ABE システムでは、アクセス制御ポリシーに基づいて、暗号文のメッセージ全体を復元するかしないかを決めることができる。しかし、機能的暗号化システムでは、ユーザのアクセス許可に基づいて、メッセージの異なる機能が返される可能性がある。例えば、暗号化されたメッセージが写真である場合、クリアランスレベルの高いユーザは画像全体を見ることができるかもしれないが、クリアランスレベルの低いユーザは部分的に冗長化された画像しか見ることができないかもしれない。機能的な暗号化スキームには、どのような種類の機能でも可能にする候補の構成が存在するが、これらのシステムはまだ実用に十分な効率性を持っていない。関数型暗号化の概要については、[9]を参照されたい。現代の公開鍵暗号化の状況を要約するために、図 1.2 を示します。

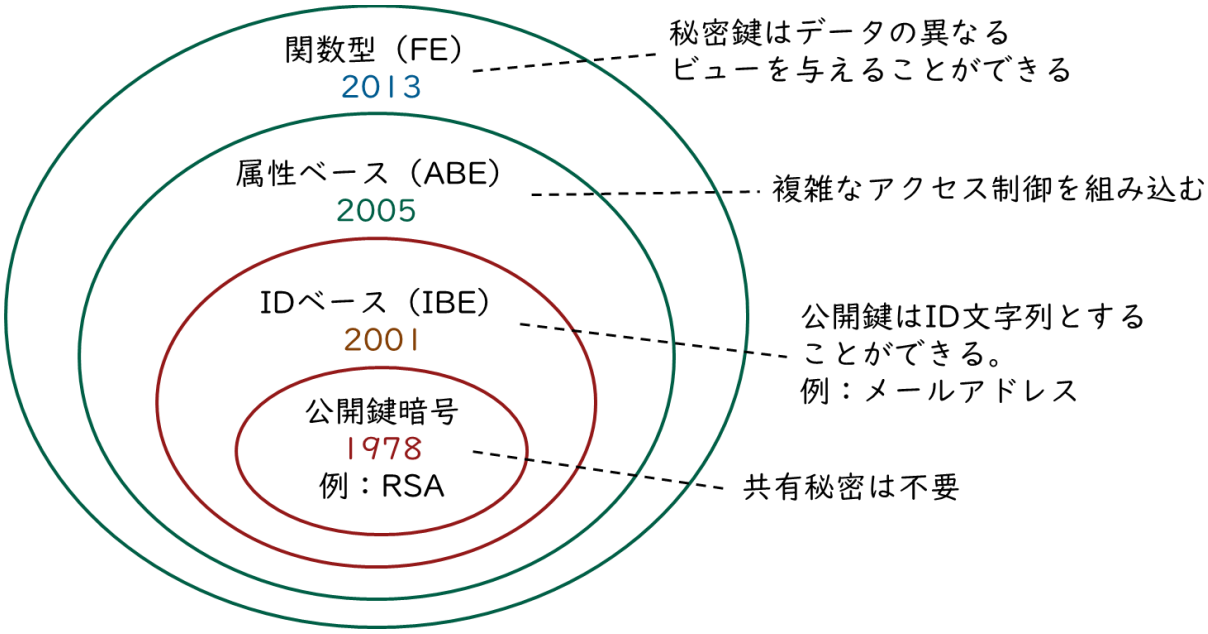

図 1.1：公開鍵暗号化の進化の概要。

1.3 OpenABE の目的

OpenABE ライブラリ OpenABE は、最先端の暗号アルゴリズム、業界標準の暗号関数とツール、直感的なアプリケーションプログラミングインタフェース（API）を組み込んだ暗号ライブラリです。OpenABE は、開発者が機密データの保護とアクセス制御のために ABE のメリットを享受できるアプリケーションに ABE 技術をシームレスに組み込めるようにすることを目的としています。OpenABE は使いやすく設計されており、開発者が暗号化の専門家である必要はありません。

OpenABE の目的は、ABE のパワーを使いやすい暗号ライブラリとして広く利用できるようにすることです。私たちは、この技術がデータセキュリティ業界に革命をもたらすと信じています。

ABE を実用的な展開に対応させるには、多くの顕著な障害があった。ABE の基礎となる暗号は、改ざん攻撃に対するセキュリティ、トラスト・ドメイン間での共有、効率性、失効などを同時に実現するために高度化する必要がありました。Zeutro の暗号技術者チームは、主要な ABE システムとソリューションのスイートをまとめ、複数の米国特許を取得しました。これらの機能の一部は OpenABE に含まれており、その他は商用ライセンスで利用可能です。

第 2 章

OpenABE ライブラリ

2.1 概要

OpenABE ライブラリは、さまざまなアプリケーションで使用できるいくつかの属性ベース暗号化（ABE）方式（[1]）を実装した C++ライブラリです。このライブラリの目標は、アーキテクチャ・レベルでセキュリティ保証を組み込んだ効率的な実装を開発することである。OpenABE は、アプリケーション・ロジックを更新することなく、ある暗号方式を別の暗号方式に置き換えることができるという意味でモジュール的であり、一般的な暗号タスクを実行するのに必要なルーチンをサポートしているという意味で包括的であり、比較的少ない労力でいくつかの高度な暗号方式を追加サポートできるという意味で拡張可能である。さらに、OpenABE は暗号化スキーム設計のベストプラクティスを取り入れており、選択暗号文攻撃に対するセキュリティ、共通鍵の伝送のためのシンプルなインターフェース、大きなデータオブジェクトの暗号化の実行をサポートしています。

まずセクション 2.2 でライブラリのハイレベルな機能について説明し、残りのセクションで図 2.1 に示す OpenABE アーキテクチャの構成要素について詳しく説明する。

2.2 ハイレベルな機能

公開鍵暗号用の汎用アプリケーション・プログラミング・インターフェース（API）。伝統的な公開鍵暗号と、属性ベース暗号（Attribute-Based Encryption：ABE）や関数型暗号（Functional Encryption：FE）のような高度な暗号の両方をサポートする暗号化ライブラリを開発する上での課題は、それらのインターフェースの違いと、文献に数多く存在するスキームの違いである。例えば、属性ベースの暗号化方式には、暗号文ポリシーや鍵ポリシーなど、さまざまな種類があります。ABE の各フレーバーの中には、いくつかの代替スキームがあり、それぞれが機能、パフォーマンス、セキュリティの異なるトレードオフを提供しています。

これまでの学術的な取り組みでは、その結果、複数の異なる（そして互換性のない）実装が行われてきた。これは急速に進歩している分野であるため、我々は（非 FE 型を含む）多くのスキームをサポートできる単一の包括的なライブラリを実装することにした。詳細はセクション 2.3 を参照のこと。

これを実現するために、OpenABE は一般化された API を提供し、多くの異なるスキームに共通する要素を特定することで暗号化プロセスを簡素化します。この結果、任意の数のスキームに対して単一のインターフェイスが実現され、コードをほとんど変更することなく、アプリケーションをあるスキームから別のスキームに移行することが可能

[1]暗号化スキームは、2 者間のすべての通信を監視できる盗聴者の存在下で、2 者が秘密裏に通信することを可能にする一連のアルゴリズムで構成される。

になります。OpenABE API は、基礎となるスキームの種類が異なる場合（特定のアルゴリズムへの入力など）にのみ、直感的な方法で異なるオプションを提供します。

モジュラー基礎数学ライブラリ。以前の高度な暗号化プロトタイプの限界は、楕円曲線演算やバイリニア写像（またはペアリング）などの重要な演算を実行するために必要な、単一の基礎となる数学ライブラリに大きく依存していることです。このため、RELIC のような最新の数学ライブラリが提供する性能向上など、技術の進歩を利用するためにコードを更新することは非常に困難です。

このライブラリーの設計における我々の目標は、基本的な楕円曲線とバイリニア演算をサポートする中立的な数学 API を提供することであった。その結果、我々の API は 1 つ以上の特定の数学ライブラリーに接続し、実装することができる。このアプローチの利点は、一元化された数学 API により、新しいライブラリを利用するためのライブラリのアップデートが比較的容易であること、あるいは、異なる実装における特定の強みを利用する複数のバージョンの OpenABE を展開することも可能であることです。

この基礎となる数学 API を Zeutro Math Library（ZML）と呼び（図 2.1 の下部）、OpenABE 全体でこの抽象化を使用しています。現在の実装では、RELIC ライブラリ[11]で利用可能な最も高速なペアリング実装の 1 つを使用しており、以前の取り組みよりもパフォーマンスが劇的に向上しています。

暗号文改ざん攻撃に対する保護。暗号化システムが耐えるべき攻撃には、確立されたさまざまなカテゴリーがある。一般的に、最も基本的なセキュリティは、敵が暗号化されたメッセージを見ることを許される、*選択された平文攻撃に対するセキュリティ*（CPA-secure）と呼ばれるものです。実世界の実装では、敵対者が暗号文を改ざんし、その後のシステムの挙動を観察することを許される、*選択された暗号文攻撃*（CCA-secure）に対する*セキュリティ*と呼ばれる、より高いレベルのセキュリティが要求される。多くの暗号システムでは、クライアントがエラーを投げるかどうかを観察するだけで、平文を知ることができる。有名な例としては、Bleichenbacher 氏[8]が RSA PKCS#1 暗号化標準にそのような弱点があることを実証したことが挙げられる。したがって、CCA のセキュリティは、一般的に実用的な配備に必要であると見なされている。

OpenABE は、すべての ABE（および非 ABE）暗号化システムの CCA セキュア実装を使用し、このセキュリティレベルはデフォルトでオンになっています。ABE でこれを実現するために、CPA セキュアなスキームを CCA セキュアなスキームに変換する一般的なアプローチを開発した。その詳細はセクション 2.3.8 で説明する。

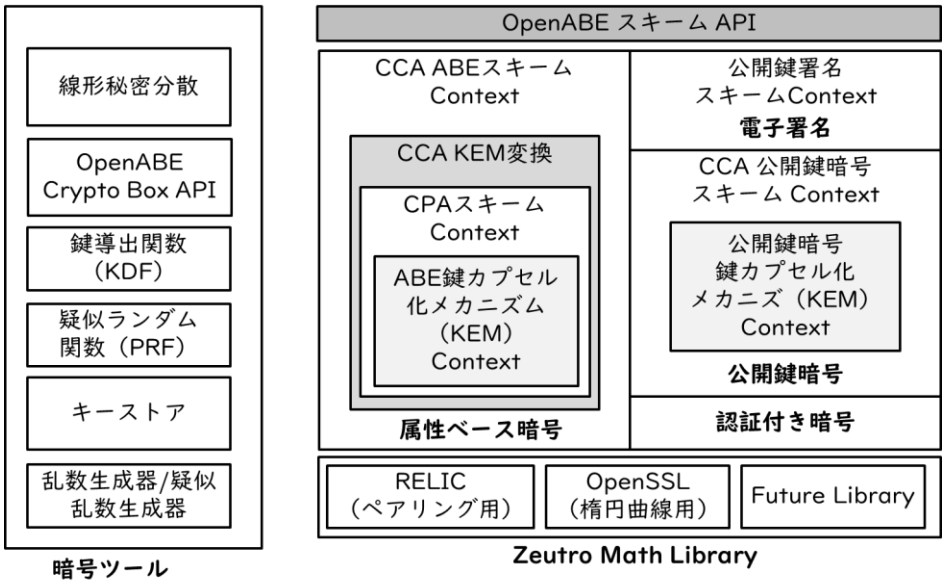

図 2.1：OpenABE ライブラリのアーキテクチャ。

2.3 コア暗号アルゴリズム

このセクションでは、実装したコア暗号アルゴリズムについて説明する。具体的には、OpenABE は以下を提供する：

- キーポリシーABE や暗号文ポリシーABE を含む、複数のタイプの属性ベース暗号化（ABE）キーカプセル化（KEM）スキームをサポート。各タイプの ABE KEM スキームに対して、選択暗号文のセキュリティも提供します。
- 選択暗号文セキュリティ付き公開鍵暗号化、デジタル署名、認証済み共通鍵暗号化をサポート。
- 線形秘密分散方式（LSSS）、鍵導出関数（KDF）、擬似乱数関数（PRF）を含む、いくつかの一般的な暗号化ツールのサポート。また、擬似乱数生成器（PRG）とモジュール的に交換可能な乱数生成器（RNG）のサポートも含まれています。

まず、このセクションの残りの部分で使用するいくつかの用語の背景を説明し、次に各暗号アルゴリズムと実装の選択肢について説明する。

2.3.1 代数的設定

双線形群。G1、G2、GT を素数次数 p の群とする。写像 e：G1 G2× → GT は、次の 3 つの性質を満たすとき、許容双線形写像（またはペアリング）である：

1. *双線形性*：すべての $g_1 \in$ G1、$g_2 \in$ G2、および $a,b \in$ Z_p について、$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ が成立する。
2. *非属性*：g_1 と g_2 がそれぞれ G1 と G2 の生成子である場合、$e(g_1,g_{(2)})$ は GT の生成子である。
3. *効率*：任意の $g_1 \in$ G1 と $g_2 \in$ G2 が与えられたとき、$e(g_1,g_{(2)})$ を計算する効率的に計算可能な関数が存在する。

許容双線形写像生成器 BSetup は、セキュリティ・パラメータ 1^τ を入力として、双線形写像群 $(p,g_1,g_2,G1,G2,GT,e)$ のパラメータを出力するアルゴリズムであり、p は $\Theta(2^\tau)$ の素数であり、G1,G2,GT は次数 p の群であり、$g_{(1)}$ は G1 を生成し、$g_{(2)}$ は G2 を生成し、e は ：G1 G2×→ GT は許容双線形写像である。上記の双線形写像は*非対称写像*と呼ばれ、実装はこの非常に効率的な設定を用いる。OpenABE の曲線の選択についてはセクション 2.5 で後述する。

2.3.2 属性

属性は任意のバイナリ文字列であり、システムの初期化時に制限されたり固定されたりすることはない。属性のセットは、あなたが望むものであれば何でもよく、あなたのアプリケーションに固有のものである可能性が高いことを強調します。

ここでは、属性を表現するための特別なアプローチを説明する。OpenABE では、属性は 2 つの部分に分かれています：「type}:{*value*}」です。表現される属性のタイプを示すタイプ部分と、実際の属性文字列を示す*値*部分です。この*タイプ*フィールドは、アトリビュートの名前空間またはドメインをキャプチャする方法として機能します。例えば、"*system* : *attribute* I "は、特定の *attribute*I がシステム専用型の属性であることを示すことができます。

ある種の特徴では、数値を属性として表現することが望ましい。OpenABE では、Bethencourt *ら*[7]と同様のアプローチで属性を拡張し、数値をサポートしています。具体的には、ある *n ビット*整数 *k* に対して数値属性 "*a*= k" を表現するために、比較を*バイナリ*表現に変換し、*k* の各ビット（最大 32 ビット）の値を指定する *n 個の*（非数値）属性を生成します。最適化として、"*a*= *k#b*"（b は *k* を表現するために使用するビット数）のような単純なエンコーディングで、値 *k* を表現するために実際に使用するビット数を指定することをサポートします。[2]

2.3.3 アクセス制御構造

この節では、ABE の説明を通じて使用するアクセス制御構造（略してアクセス構造）の概念を定義する。

定義 2.3.1 アクセス構造。$P_1,P_2,...,P_n\}$を*当事者の集合とする。コレクション*$A \subseteq 2^{\{P_1,P_2,...,P_n(\})}$*は、*$\forall\ B,C:\ B \in\ A$ *and* $B \subseteq\ C$ *then* $C \in\ A$ *のとき、単調である。アクセス構造（それぞれ、単調アクセス構造）は、*$\{P_1,P_2,...,P_n\}$*の空でない部分集合のコレクション（それぞれ、単調コレクション）*A *である、*$A \subseteq 2^{\{P_1,P_2,...,P_n\}}\ \phi\ \}$である。A に*含まれる集合を認可集合、*A に*含まれない集合を未認可集合と呼ぶ。*

OpenABE では、アクセス構造はツリーとして表現されるブール式である。[3]ツリーの非リーフ・ノードはそれぞれ閾値ゲートを表し、子ノードと閾値によって記述される。num_x をノード x の子の数、k_x をその閾値とすると、$0 < k_x \leq num_x$ となる。$k_x =$ I のとき、閾値ゲートは OR ゲートであり、$k_x =\ num_{(x)の}$ とき、それは AND ゲートである。木の各葉ノード x は、属性と閾値 k_x=I によって記述される。

ここで、ABE スキームの記述で使用する木に関するいくつかの用語を定義する。木におけるノード x の親を **parent**(x)とする。関数 **att**(x)は x が葉ノードの場合のみ定義され、木の葉ノード x に関連する属性を示す。アクセス・ツリーT は、各ノードの子ノード間の順序も定義する。つまり、ノードの子には I から *num までの*番号が付けられる。**index**(x) は、ノード x に関連付けられたそのような番号を返します。インデックス値は、ツリーが構築されるときに、アクセス構造内のノードに一意に割り当てられることに注意してください。

セクション 2.3.2 で説明したように、我々は数値属性を「ビットの袋」表現 [7]として表現する。整数比較（*例:* $<,\leq\ ,>,\geq\ ,=$）をサポートするために、非数値属性表現に AND ゲートと OR ゲートを使用する。$a < k$ "のような比較では、定数 k のビットとゲートの選択に直接的な関係がある。他の演算子 $\leq,>,\geq\ ,=$ についても同様であり、定数 k *に応じて*最大 n ゲート、または場合によってはそれ以下のゲートを使用する。この技法により、アクセス構造で日付を表現することで、ABE 復号鍵の失効を実装できることに注意。セクション 2.3.10 を参照のこと。

[2] k がすべて b ビットで表現できることを確認するためにサニティチェックを行うことに注意。

[3]学術文献では、秘密共有の定義は通常、行列として表現されるアクセス構造上で記述される[6]。同様に、アクセス構造は正規言語（正規表現など）を表現するために拡張することもできる。

アクセス構造を満たす。T をルート r を持つ木とする。ノード x をルートとする T の部分木を $T_{(x)と}$する。属性の集合 γ がアクセス構造 T_x を満たす場合、これを $T_x(\gamma)=1$ と表す。OpenABE では、$T_x(\gamma)$ を計算する **ScanTree という**再帰的アルゴリズムを以下のように実装している：

- x が非リーフノードの場合、ノード x のすべての子 x^0 に対して $T_{x^0}(\gamma)$ を評価する。$T_x(\gamma)$ は、少なくとも $k_{(x)の}$ 子も 1 を返す場合にのみ 1 を返す。
- x がリーフノードである場合、$T_x(\gamma)$ は **att**$(x) \in \gamma$ の場合に限り 1 を返す。
- 実装では、アルゴリズムは、T を満たすのに必要な γ の属性の最小部分集合も返す。

2.3.4 秘密の共有

ABE の構築は通常、秘密分散スキームを中核的な構成要素として利用するので、このセクションでそれを説明する。OpenABE はアクセス構造をツリー表現で表現するため、通常の秘密共有の定義[6]をこの特殊なデータ構造に若干適応させています。秘密分散スキームは 2 つのアルゴリズムから構成される：

ComputeSecretShare$(s,T) \rightarrow \lambda$. このアルゴリズムは、秘密 $s \in Z_p$、アクセス構造 T を入力とし、T_r の構造に従って s の秘密共有を計算する。まず、アクセス構造の各非リーフノード x に対して多項式 q_x を選択する。これはルートノード r から再帰的に行う。

木の各ノード x について、多項式 q_x の次数 $d_{(x)を}$、そのノードの閾値 k_x より 1 小さい値に設定する（すなわち、$d_x = k_{(x)} - 1$）。ルート・ノードについては、$q_r(0) = s$ とし、多項式 q_r の他の点を d_r ランダムに設定して完全に定義する。他のノード x については、$q_x(0) = q_{親(x)}$(**index**(x))とし、$q_{(x)を}$ 完全に定義するために $d_{(x)の}$ 他の点をランダムに選ぶ。

上記のように定義された多項式 q を用いて、T のリーフノード x に到達するために、$\lambda_x = q_{(x)}(0)$ を設定する。したがって、$\{\lambda_x\}$、を s の秘密共有の集合とする。` は T のリーフノードの数である。アルゴリズムは λ を出力する。

すべての秘密共有方式が再構成特性を持つことは文献[6]で証明されている。ここでは、アクセス構造 T の構造に従う再構成アルゴリズムを定義する。

RecoverCoefficient$(T, \gamma) \rightarrow (\omega, T)$。このアルゴリズムは、アクセス構造 T と属性集合 γ を入力とする。このアルゴリズムは、アクセス構造 T と属性の集合 γ を入力とする。属性の集合 γ がアクセス構造 T を満たす場合（すなわち、$T(\gamma)=1$）、次のように処理を進める。ラグランジュ係数 $\Delta_{i,S}(x)$ を $i \in$ Zp と Z_p の要素の集合 S に対して定義する：

$$\triangle_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

まず **ScanTree** アルゴリズムを実行し、アクセス構造 T を満たす部分集合 $T \subseteq \gamma$ を選ぶ。次に、各ノード x について、$\omega_x = \omega_{parent(x)} - \Delta_{i,S_x}(0)$ を設定する。アルゴリズムはラグランジュ係数 ω と属性 T の集合を出力する。

2.3.5 キーポリシー属性ベースの暗号化

このセクションでは、Goyal-Pandey-SahaiWaters Large Universe システム[11, Section 5]、[15,16]の変種に対する Key-Policy ABE Key Encapsulation スキームについて説明する。KEM の変種は、[15, セクション 2.2.3]の最適化も取り入れている。構築は 4 つのアルゴリズムからなる。セットアップアルゴリズ ムは、1 つの権威に対して公開パラメータとマスターシークレットを生成する。オーソリティは keygen を実行し、ユーザにきめ細かなアク セスを許可するアクセス構造を持つユーザの秘密鍵を生成することができる。暗号化アルゴリズムは属性の記述セットを入力とし、対称鍵と暗号文を出力する。復号アルゴリズムは、認可されたユーザが暗号文を復号するために使用する。

セットアップ(τ,n) → PK,MSK.セットアップアルゴリズムは、セキュリティパラメータ τ を入力とし、BSetup(1^{τ})→(p,g_1,g_2,G1,G2,GT,e)を実行し、ランダムな指数 $y \in Z_p$ を選択し、$Y = e(g_1,g_2)^y$ を計算する。さらに、ランダムオラクルとしてモデル化した耐衝突ハッシュ関数 $H_1 : GT \rightarrow \{0,1\}^n$ を使用する[4]このアルゴリズムは、公開パラメータ PK とマスターシークレット MSK を以下のように出力する:

$$PK = \{g_{(1)}, g_{(2)}, e(g_{(1)},g_{(2)})^y\}, \quad MSK = y$$

Keygen(PK,MSK,T) → SK。T: $\{0,1\}^* \rightarrow$ G1 をランダムオラクルとしてモデル化する関数とする。[5]

鍵生成アルゴリズムは、T (γ)=1 の場合に限り、属性 γ の集合の下で暗号化された メッセージの復号を可能にする秘密鍵を出力する。アルゴリズムは、アクセス構造 T に従って y のランダム化シェアを計算する(例えば、computeSecretShares(y,T) → λ)。ツリーの各リーフノード x に対して、以下の秘密値が渡される:

$$D_x = g_1^{\lambda_x} \cdot T(i)^{r_x} \text{ここで } i = \mathbf{att}(x) \qquad d_x = g_2^{r_x}$$

Encrypt$_{\mathbf{KEM}}$(PK, γ;u)→(Key,CT)。暗号化アルゴリズムは、公開パラメータ PK、属性 γ のセット、および擬似乱数生成器 G(実装の詳細についてはセクション 2.4.2 を参照)へのオプションの入力シード $u \in \{0,1\}^k$ を入力とする。アルゴリズムは、まずランダムな $s \in$ Zp を選択し(シード u が指定されている場合は、ランダム性のソースとして G を使用する)、次に以下を返す:

$$\text{キー} \leftarrow {}_{H(1)}(e(g_{(1)},g_{(2)}))^{ys}), CT = (\gamma, E'' = g_2^s, \{E_i = T(i)^s\}_{i \in \gamma})$$

Decrypt$_{\mathbf{KEM}}$(CT,SK) = Key.復号アルゴリズムは、暗号文 CT と、アクセス構造 T を埋め込んだユーザの秘密鍵 SK を入力とする。アルゴリズムはまず、アクセス構造 T が暗号文の属性 γ を満たすかどうかを判定する(例えば、T (γ) = 1 の場合)。もしそうであれば、T を満たすのに必要な属性の最小セット S に対してラグランジュ係数 ω を回復する(*例えば*、recoverCoefficients(T , γ)→(ω,S))。したがって、そのような各属性 $i \in$ S について、対応する係数 ω_i と、T における対応する SK 成分(*例えば*、$D_{\rho((i)())}$ と $d_{\rho(i)}$ によって定義される[6]、アルゴリズムはまず計算する:

[4] H_1 は、SHA-256 を使ってインスタンス化された鍵付きハッシュ関数である。

[5] ランダムオラクル T は、SHA-256 を使用してインスタンス化された鍵付きハッシュ関数である。15]のセクション 2.2.2 を参照のこと。

[6] ρ を葉ノードを属性に写す射影関数とする。

$$\prod_{i \in S} \left(\frac{e(D_{\rho(i)}, E'')}{e(E_i, d_{\rho(i)})} \right)^{\omega_i} =$$

$$\prod_{i \in S} \frac{e(g_1, g_2)^{\lambda_x \omega_i s} \cdot e(T(i), g_2)^{\omega_i r_x s}}{e(T(i), g_2)^{\omega_i r_x s}} = \prod_{i \in S} e(g_1, g_2)^{\lambda_x \omega_i s} = e(g_1, g_2)^{ys}$$

アルゴリズムは、Key= $_{H(I)}(e(g_1,g_2)^{ys})$ を計算することができる。

実施内容

- *最適化。k*= | S|とする。ラグランジュ係数をペアリングの内側に持ってくることで、ペアリングの数を 2k から *k* + 1 に減らすことができる。ペアリング演算は指数演算よりも計算量が多いため、この最適化により復号化全体の速度が向上する。OpenABE での復号の結果：
$$\frac{e(\prod_{i \in S} {D_{\rho(i)}}^{\omega_i}, E'')}{\prod_{i \in S} e({E_i}^{\omega_i}, d_{\rho(i)})}$$
- *CCA セキュリティー。*上記のスキームは CPA セキュアなだけなので、実際の実装ではセクション 2.3.8 による CCA セキュア変換を使用することを思い出してほしい。

2.3.6 暗号文ポリシー属性ベースの暗号化

このセクションでは、Waters Large Universe システム[20, Appendix A]の Ciphertext-Policy ABE KEM スキームの変形について説明する。構成は 4 つのアルゴリズムからなる。セットアップアルゴリズ ムは、単一の権威に対して公開パラメータとマスターシークレットを生成する。オーソリティは keygen を実行し、特定のユーザに属性のセットを付与する秘密鍵を生成することができる。暗号化アルゴリズムはアクセス構造を入力とし、共通鍵と暗号文を出力する。復号アルゴリズムは、認可されたユーザが暗号文を復号するために使用する。

セットアップ(τ ,*n*) → PK,MSK.セットアップアルゴリズムは、セキュリティパラメータ τ を入力とし、BSetup(1^{τ}) → ($q,g_1,g_2,$G1,G2,GT,e) を実行し、ランダムな指数 $\alpha,a\in$ Zp を選択する。さらに, 耐衝突ハッシュ関数として, H_1 : $GT \rightarrow \{0,1\}^n$ と H_2 : $\{0,1\}^* \rightarrow$ G1 (どちらもランダムオラクルとしてモデル化する)を用いる.[7] 当局は, 公開パラメータ *PK* を公開し, マスターシークレット *MSK* を以下のように設定する.

$$\text{PK} = \{g_1, g_2, g_1^a, e(g_1, g_2)^{\alpha}\} \quad \text{MSK} = \{\alpha, g_2^a\}$$

鍵生成(PK,MSK, γ) → SK。鍵生成アルゴリズムは、マスターシークレットと属性 γ のセットを入力とする。アルゴリズムはまず、ランダムな $t\in$ Zp を選択する。秘密鍵 SK は以下のように生成される：

$$K = g_2^{\alpha} \cdot g_2^{a \cdot t} \qquad L = g_2^t \qquad \forall x \in S \; K_x = H_2(x)^t$$

Encrypt$_{\text{KEM}}$(PK,T;*u*)→(Key,CT)。暗号化アルゴリズムは、公開パラメータ PK、アクセス木構造 T、および擬似乱数生成器 G(実装の詳細についてはセクション 2.4.2 を参照)へのオプションの入力種 $u\in\{0,1\}^k$を入力とする。アルゴリズムは、最初にランダムな指数 $s\in$ Zp を選択する（シード $u\in\{0,1\}^k$が指定された場合、ランダム性のソースとして G を使用する）。*G* の出力長は T のサイズによって異なることに注意。）次に、アクセス構造 T に従って、

[7] H_1 と H_2 は、SHA-256 を使ってインスタンス化された鍵付きハッシュ関数である。

s のランダム化共有を計算する（*例えば*、computeSecretShares(*s*,T) → *λ*）。ここで、{*λ*1,...,*λ*ℓ}を *s* の共有とする。ℓ は T のリーフノードの数であり、ρ はリーフノードを属性にマップする射影関数である。さらに、アルゴリズムはランダムな $r_1,...,r_\ell \in$ Zp を選び、以下を返す：

キー= $_{H(1)}(e(g_1,g_2)^{\alpha s})$、
CT = $\{C' = g_1^s, (C_1 = g_1{}^{a\lambda} H_2(\rho(1))^{-r_1}, D_1 = g_2^{r_1}), \ldots, (C_\ell = g_1{}^{a\lambda_\ell} H_2(\rho(\ell))^{-r_n}, D_\ell = g_2^{r_\ell})\}$

Decrypt$_{KEM}$(CT,SK) = Key.復号アルゴリズムは、アクセス構造 T の暗号文 CT と属性 *γ* の集合の秘密鍵を入力とする。アルゴリズムはまず、属性 *γ* が暗号文のアクセス構造 T を満たすかどうかを判定する（*例えば*、*T*(*γ*)=1）。そうであれば、T を満たすのに必要な属性の最小セット *S* についてラグランジュ係数 *ω* を回復する（*例えば*、recoverCoefficient(T ,*γ*)→(*ω*,*S*)）。このような各属性 *i*∈ S と対応する係数 ω_i に対して、復号アルゴリズムはまず計算する：

$$\frac{e(C', K)}{\prod_{i \in S}(e(C_i, L) \cdot e(K_{\rho(i)}, D_i))^{\omega_i}} =$$

$$\frac{e(g_1, g_2)^{\alpha s} \cdot e(g_1, g_2)^{ast}}{\prod_{i \in S}(e(g_1, g_2)^{ta\lambda_i \omega_i})} = e(g_1, g_2)^{\alpha s}$$

アルゴリズムは、Key= $_{H(1)}(e(g_1,g_{(2)}))^{\alpha s}$) を計算することができる。

実施内容

- *最適化。*k=｜S|とする。KP-ABE KEM 実装と同様に、ここでもラグランジュ係数をペアリングの内側に持ってくることで、ペアリングの数を 2k + 1 から *k* + 2 に減らすことができる。したがって、この最適化により、復号化時間が顕著に高速化される。OpenABE での復号の結果：

$$\frac{e(C', K)}{e(\prod_{i \in S} {C_i}^{\omega_i}, L) \cdot \prod_{i \in I}(e({K_{\rho(i)}}^{\omega_i}, D_i))}$$

- *CCA セキュリティー。*上記のスキームは CPA セキュアなだけなので、実際の実装ではセクション 2.3.8 による CCA セキュア変換を使用することを思い出してほしい。

2.3.7 CPA KEM から標準暗号への変換

ここで、任意の CPA 安全 ABE スキームを CCA 安全 ABE スキームに変換する一般的な方法について説明する。(OpenABE のコンテキストでは、CPA スキームコンテキストは、CPA KEM スキームバリアント（*すなわち*、上記の ABE スキームのいずれか）と擬似ランダム生成器（PRG）（セクション 2.4.2 で定義）を組み合わせたものであり、CCA セキュア ABE スキームを構築するために使用するプリミティブを形成する。CPA^0 または CPA KEM という添え字を使用して、そのアルゴリズムが生成された CPA スキームコンテキストであるか、既存の CPA KEM スキームであるかをそれぞれ示す。CPA スキームコンテキストでは、基礎となるスキームのセットアップアルゴリズムとキー生成アルゴリズムは変更されません。この構成では、暗号化と復号を以下のように変更する：

Encrypt$_{CPA^0}$(PK,*M*,A;*u*) → CT^0。暗号化アルゴリズムは、公開パラメータ PK、メッセージ *M*∈{0,1}ℓ、アクセス構造または属性セット（A と表記）、およびランダムなシード *u* を入力とする。

1. 実行 **Encrypt**$_{\text{CPAKEM}}$(PK,A;u) → (K,CT$_0$)
2. 実行 **G**(K, `) → y
3. CT$_{(1)\,y}$= ⊕ M とする。
4. 出力暗号文 CT = (CT$_{(0}$,CT$_{(1}$))

復号 $_{\text{CPA}^0}$(SK,CT) = M'

1. 実行 **Decrypt**$_{\text{CPAKEM}}$(SK,$CT_{(0)}$) = K'
2. 計算 `$'$= **length**($CT_{(1}$))
3. 出力 M' = CT$_1$⊕ G(K'^0, `$'^0$))

2.3.8 選択された暗号文の安全な変換

上で定義したプリミティブを基に、どのような CPA スキームコンテキストからどのように選択暗号文の安全な ABE システムを構築するかを示すことができる。CCA KEM アルゴリズムはブラックボックス方式で CPA スキームコンテキストを使用する。さらに、ランダムオラクルとしてモデル化した耐衝突ハッシュ関数 H: {0,1}* → {0,1}^{k}を使用する[8]。ここでは、アルゴリズムが生成された CCA KEM スキームであるか、既存の CPA スキームコンテキストであるかを示すために、それぞれ CCA KEM または CPA^{0}という添え字を使用する。CCA KEM 変換では、基礎となるスキームのセットアップおよびキー生成アルゴリズムは変更されません。

最後に、基礎となる IND-CPA セキュア ABE システムは、暗号文を作成するために使用されるアクセス構造を、公開鍵のみを使用してその暗号文から効率的に抽出できるという特別な特性を有すると仮定する。特に、決定論的アルゴリズム **ExtractAccessStructure**(PK,C) →の存在を仮定する。
という正しさを持っている。

$$\textbf{Extract AccessStructure}(\text{PK},C=\ \textbf{Encrypt}_{\text{CPA}^0}(\text{PK},\text{A},M;u)) = \text{A}$$

我々が使用する全ての IND-CPA ABE スキームは、(些細なことではあるが)この性質を持つ。

この変換は、暗号化と復号化を次のように変更する:

Encrypt$_{\text{CCA-KEM}}$(PK,A) → (K,C)。暗号化アルゴリズムは、公開パラメータ PK と、アクセス構造または属性のセットを入力とする。

1. ランダムな K∈{0,1}^{n}を選ぶ。
2. ランダムに r∈{0,1}^{n}を選び、u= H(r K||| A)とする。

[8] H は SHA-256 を使ってインスタンス化された鍵付きハッシュ関数である。

3. **Encrypt**$_{\text{CPA}^0}$(PK,A,M=(K,r);u)→C を実行する。
4. 鍵 K と暗号文 C を出力する。

Decrypt$_{\text{CCAKEM}}$(PK,SK,C) = K^0 ∪ ⊥.

1. 実行 **Decrypt**$_{\text{CPA}^0}$(SK,C) = M^0 = ($K^{(0}$,$r^{(0}$))
2. A^0= **ExtractAccessStructure**(PK,C).
3. u^0= H(r^0||$^{K(0)}$||$^{A(0)}$)とする。
4. **Encrypt**$_{\text{CPA}^0}$(PK,A^0,M^0 = (K^0,r^0);u^0) → $C^{(0)を}$実行する。
5. C^0=$^?$ C をチェックし、等しければ $K^{(0)を}$出力する。そうでなければ⊥を出力する。

暗号化に追加される時間はほとんど無視できる。復号化に追加されるオーバーヘッドは、有効性チェックの一環として暗号文を再暗号化しなければならないことである。

2.3.9 ハイブリッド暗号化（CCA KEM を使用）

システム内のデータを暗号化するために、OpenABE は、元のデータを暗号化する目的で CCA KEM スキームを基にした CCA スキームコンテキストを定義する。具体的には、CPA スキームコンテキストは、CCA KEM と認証された暗号化スキームを組み合わせます（詳細 はセクション 2.3.12 を参照）。CCA スキームコンテキストは、基礎となるスキームのセットアップと keygen アルゴリズムを変更せずに残す。この構成では、暗号化と復号を以下のように変更する：

暗号化 $_{\text{CCA}}$(PK,M,A) → CT

1. **Encrypt**$_{\text{CCAKEM}}$(PK,A)→(K,$C_{0)}$を実行する。
2. AAD = **ExtractHeader**(C_0) 3.**AuthEncrypt**(K,M;AAD) → C_1 を実

   行。

4.出力 CT = (C_0,C_1).

復号 $_{\text{CCA}}$(PK,SK,CT) = M^0 ∪ ⊥

1. **Decrypt**$_{\text{CCA- KEM}}$(PK,SK,C_0) = $K^{(0)を}$実行する。
2. AAD' = **ExtractHeader**($C_{(0)}$)
3. K^0 6= ⊥の場合、M^0= **AuthDecrypt**(K^0,$C_{(1}$;AAD^0) を実行する。
4. エラーがなければ $M^{(0)を}$出力する。そうでなければ⊥を返す。

ExtractHeader メソッドは、ABE 暗号文から以下のメタデータを抽出します：OpenABE ライブラリのバージョン、楕円曲線識別子、ABE アルゴリズム識別子、一意の暗号文識別子（または UUID）。

2.3.10 失効

ABE における失効には、時刻を属性として使用することで暗号的に強制できる柔軟なセマンティクスがある。シンプルにすることを推奨する。例えば、KP-ABE または CP-ABE で実装可能な 2 つのオプションを考えてみる。

1. また、暗号文は、失効が有効かどうかを示すフラグを持つ。
2. 暗号文 t はその作成時刻 t を持つ。秘密鍵は時刻 $X_0, X_{(1)}$ のウィンドウを持ち、t が X_0 と $X_{(1)}$ の範囲内でなければ復号できない。

これらのアプローチは、失効リストの管理など、任意の秘密鍵を失効させる 標準的な方法を追加的に適用することで、さらに強化することができる。

2.3.11 公開鍵暗号化

楕円曲線。楕円曲線の定義を以下に示す。この定義は、素数体上の曲線に関する NIST FIPS 186-4 文書[14]を若干引用したものである。各素数 p に対して、素数次数 n の擬似乱数曲線は以下のようにリストされる：

$$E: y^2 \equiv x^{(3)} - 3x + b \pmod{p}$$

したがって、これらの曲線では、係数は常に $h = 1$ である。楕円曲線生成器 ECSetup は、セキュリティ・パラメータ 1^τ を入力すると、上記の擬似乱数曲線のパラメータ($p,n,SEED,c,a,b,g,$G)を出力するアルゴリズムである、ドメインパラメーターシード)、SHA-1 ベースのアルゴリズムの出力 c、係数 $a \equiv -3$、係数 b（$b^2c \equiv -27 \pmod{p}$ を満たす）、および基点 g の x および y 座標。

建設

このセクションでは、OpenABE に実装されている公開鍵暗号化鍵のカプセル化について説明します。これは、NIST SP800-56A Revision 1 [2, Section 6.2.2.2]に記載されている One-Pass Diffie-Hellman 鍵合意プロトコルから派生したものです。

構成は 4 つのアルゴリズムからなる。セットアップ・アルゴリズムはセキュリティ・パラメータを入力とし、楕円曲線グループを選択する。鍵生成アルゴリズムはユーザ識別子を入力とし、公開鍵と秘密鍵のペアを出力する。暗号化アルゴリズムは送信者の識別子、受信者の識別子、受信者の公開鍵を入力とし、共通鍵と暗号文を出力する。復号アルゴリズムは、認可された受信者が暗号文を復号するために使用する。

セットアップ(τ)。セットアップアルゴリズムは、セキュリティパラメータ τ を入力とし、ECSetup(1^τ)→ECSetup($1^{\tau)}$を実行する。

($p,n,SEED,c,a,b,g,$G)とする。KDF を鍵導出関数(詳細はセクション 2.4.1 参照)とし、` を導出される鍵の長さとする。

Keygen(ID) → (PK,SK)。鍵生成アルゴリズムは、グループパラメータと鍵識別子 ID∈{0,1}^{k} を入力とする。アルゴリズムはまずランダムに $a \in Z_p$ を選び、公開鍵 PK = g^a と秘密鍵 SK = a を出力する。

Encrypt$_{KEM}$(ID_S,ID_R,PK)→(K,C)。暗号化アルゴリズムは、送信者の識別子と受信者の公開鍵を入力とする。アルゴリズムはまずランダムな $b \in Z_p$ を選択し、共有鍵
$Z = g^{(ab)}$として、鍵 K と暗号文 C を返す。

$$K = \text{KDF}(Z, `, \text{AlgID} || \text{ID}_S || \text{ID}_R), \quad C = \{\text{ID}_S, g^b\}.$$

Decrypt$_{KEM}$(ID_R,SK,C) = K. 復号アルゴリズムは、受信者の秘密鍵と暗号文を入力とし、受信者の秘密鍵 を用いて $Z = g^{(ab)}$を復元する。そして、アルゴリズムは派生鍵を出力する。K = KDF(Z, `,AlgID|| ID_S|| ID_R)。

PKE スキーム・コンテキスト

上記の PKE KEM プリミティブと認証された暗号化プリミティブを組み合わせて、CCA セキュア公開鍵暗号方式を構築する。その構成は以下の通りである：

暗号化 $_{CCA}$(ID_S,ID_R,PK,M) → CT

1. **Encrypt$_{KEM}$**(ID_S,ID_R,PK)→(K,C_0)を実行する。
2. **AuthEncrypt**(K,M) → C_1を実行する。
3. 出力 CT = (C_0,C_1)).

復号 $_{CCA}$(ID_R,SK,CT) = $M^0 \cup \perp$

1. **Decrypt$_{KEM}$**(ID_R,SK,C_0)) = K^0を実行する。
2. M^0= **AuthDecrypt**(K^0,C_1)を実行する。
3. **AuthDecrypt** 中にエラーが発生した場合、M^0 または⊥を出力する。

*公開鍵および秘密鍵の検証。*実装には、[2, Section 5.6.2.5]の ECC 完全公開鍵/秘密鍵検証ルーチンに従った鍵の検証も含まれる。この検証では、公開鍵が無限遠の点ではないこと、公開鍵の各座標が一意に正しい 要素表現を持つこと、公開鍵が正しい楕円曲線上にあること、そして正しい次数/EC サブグループを持つことを保証する。秘密鍵については、正しい順序を持つことだけを主張すればよい。これらのアサートがすべて成立すれば、鍵はロードされ、暗号化/復号に使用 できるようになる。そうでない場合、鍵検証エラーが設定される。

2.3.12 認証された暗号化

OpenABE には、大きなデータオブジェクトの機密性と完全性を提供するための認証済み共通鍵暗号化（AE）プリミティブが含まれています。我々は、OpenSSL ライブラリ [19] が提供する AES-Galois Counter Mode (GCM) 暗号を 256 ビットの鍵で使用している。AES-GCM を選んだ理由は、効率的であるだけでなく安全だからである。

加えて、パイプライン・サポートにより、ハードウェア実装は低コストで高速を達成でき（例えば、Intel の AES-NI）、レイテンシも低い。このようにセキュリティと高スループット/効率のバランスが取れているため、AES-GCM 暗号は OpenABE に適した AE ビルディングブロックだと考えています。さらに、OpenABE の暗号文には、スキーム識別子、基礎となる楕円曲線識別子、OpenABE ライブラリのバージョン番号、一意の暗号文識別子を含むメタデータをエンコードするヘッダが含まれます。OpenABE 暗号文のこれらの部分を保護するために、各暗号化スキームのコンテキストは、タグ検証の一部として AES-GCM 関連データを利用します。これにより、暗号文の内容とともに OpenABE 暗号文ヘッダの改ざんが検出されます。以下は、認証された暗号化プリミティブへのインターフェースです：

AuthEncrypt(*K,M;AAD*)→*C*。暗号化アルゴリズムは、入力として共通鍵、メッセージ $M \in \{0,1\}^*$、オプションとして認証された関連データ $AAD \in \{0,1\}^k$ を取る。アルゴリズムは暗号文 *C* を出力する。

AuthDecrypt(*K,C;AAD*) = M∪⊥.復号アルゴリズムは、対称鍵、暗号文 *C*、およびオプションとして認証された関連データ *AAD* を入力とする。このアルゴリズムは、タグ検証が成功した場合、メッセージ *M* を出力する。そうでない場合は、⊥を返す。

2.3.13 デジタル署名

OpenABE には、OpenSSL ライブラリ [19]が提供するデジタル署名プリミティブが含まれています。OpenSSL の実装は、ANSI X9.62、米国連邦情報処理標準（FIPS）186-2 デジタル署名標準（DSS）[12]から派生したものです。この実装では、SHA-256 を使用して ECDSA プリミティブのメッセージダイジェストを計算する。ここでは、ECDSA の簡略化されたインターフェイスについて説明し、スキームの詳細については FIPS 186-2 DSS 文書 [12] を参照されたい。

Keygen(τ) → (PK,SK)。鍵生成アルゴリズムは、セキュリティパラメータ τ を入力とし、ECSetup(1^τ)を実行して楕円曲線パラメータを選択し、公開鍵と秘密鍵を出力する。

Sign(SK,*M*) → σ。署名アルゴリズムは、秘密鍵 SK とメッセージ $M \in \{0,1\}^*$ を入力とし、署名 σ を出力する。

Verify(PK,*M*, σ) = {*true,false*}。検証アルゴリズムは、公開鍵 PK、メッセージ *M*、署名 σ を入力とする。アルゴリズムは、署名が M と PK に関して有効であれば*真*を出力する。そうでない場合は*偽*を出力する。

2.4 暗号ツール

2.4.1 鍵導出関数(KDF)

OpenABE では、2 種類の鍵導出関数を実装している。最初のタイプは、NIST SP 800-56A [2, セクション 5.8.1]で説明されている連結鍵導出関数（Concatenation Key Derivation Function: KDF）です。この構造では、基礎となるハッシュ関数として SHA-256 を使用する。この特定の KDF は、セクション 2.3.11 で説明する公開鍵暗号化スキームのコンテキストで使用される。連結 KDF アルゴリズムは以下のように記述される：

kdf(*z, `,m*) → k∪⊥.KDF アルゴリズムは、共有秘密鍵 *Z*(バイト列表現)、生成される秘密鍵材料の ビット長を示す鍵長 ` 、アルゴリズム ID、送信者 ID、 受信者 ID を連結したメタデータ *M*∈{0,1}^{n}を入力とする。アルゴリズムは指定された長さの派生鍵を出力する。

1. *reps* = *d `/hashlene*.
2. *reps* > (2^{32} − 1)の場合、⊥を出力して停止する。
3. 32 ビット、ビッグエンディアンのビット列*カウンタを* $0000000 1_{16}$として初期化する。
4. *カウンタ*|| *Z*|| *M が最大ハッシュ inputlen* ビットより長ければ、⊥を出力して停止する。
5. *i*=1 から *reps まで*(1 刻み)、以下のようにする:
   (a) *Hash*$_i$= H(*counter Z*||| M)を計算する。
   (b) 結果を符号なし 32 ビット整数として扱う。
6. (`/*hashlen*)が整数である場合、*Hhash を* $Hash_{reps}$に設定する。そうでない場合、*Hhash を* $Hash_{reps}$の( ` mod hashlen)左端のビットに設定する。
7. 出力 *K*= $_{Hash(1)}$|| $Hash_{(2)}$|| ...|| $Hash_{(reps)-1}$|| Hhash.

2 つ目のタイプの KDF は、パスワードベースの鍵導出を可能にするものである。この KDF は OpenSSL ライブラリの PBKDF2 実装を利用し、基礎となる擬似乱数関数を HMACSHA256 に設定する。この代替 KDF は、パスワード、鍵の長さ、鍵のソルト([9]、 反復回数を入力として受け取る。アルゴリズムは、指定された長さの派生鍵を出力する。ブルートフォース攻撃に対抗するため、最低でも 10,000 の反復回数で PBKDF2 を実行する。

2.4.2 疑似ランダム生成器(PRG)

OpenABE には、任意の長さの疑似ランダムビット列を生成する PRG が含まれています。この実装は、NIST が承認したブロック暗号、カウンタモード(CTR)の AES-256 を使用してインスタンス化されており、その詳細は NIST SP 800-90A, Rev 1 [3]に記載されています。特に、セクション 10.2「ブロック暗号に基づく DRBG メカニズム」を実装する。この構築は、以下に要約する 4 つの手順で構成される:

1. _提供されたデータを使用して PRG の内部状態を更新する。**update**(*提供データ,K,V*) → (*K,V*):
   (a) *temp Null*=
   (b) while (**len**(*temp*) < *seedlen*) do:
      i. *V* = (*V* + 1) mod 2^{outlen}
      ii. *出力ブロック*= **AES ECB**(*K,V*)

[9]ソルトは、追加のエントロピーとして使用されるランダムデータである。

iii. *temp*= *temp*|| *出力ブロック*

(c) *temp*= **leftMostBits**(*temp,seedlen*)

(d) *temp*= *temp*⊕ *提供データ*

(e) *K*= **leftMostBits**(*temp,keylen*) (f) *V*= **rightMostBits**(*temp,outlen*) (g) return *K* and *V*.

2. **CTR DRBG_ Instantiate** 関数[3、セクション 10.2.1.3.2]。擬似乱数ビットを生成する前に呼び出される。入力パラメータをチェックし、シード材料を初期化し、PRG の初期状態を導出する。**Block_** Cipher df を[3, Section 10.3.2]で規定される導出関数とする。

   CTR DRBG Instantiate(*entropy input,nonce,person string*) → (*K,V,reseed counter*):

   (a) *シード_材料*= *エントロピー入力*|| *ノンス*|| *人物文字列*

   (b) $seed\_material =$

   (c) $K = 0^{keylen}$

   (d) $V = 0^{outlen}$ **ブロック暗号 df**(*シード材料,seedlen*)

   (e) (*K,V*) = **CTR DRBG 更新**(*シード材料,K,V*)

   (f) *リシード・カウンター* = 1

   (g) *K,V* を返し、*カウンター*を初期状態として*再シードする*。

3. **CTR_ DRBG_** Reseed 機能 [3、セクション 10.2.1.4.2]。擬似ランダムビットの生成に追加のエントロピー入力を挿入する。これは、PRG への要求が 100,000 回を超えたときにトリガされる（NIST が推奨するデフォルト値であるが、必要に応じて調整可能）。

   CTR DRBG Reseed(*seed material,K,V*) → (*K,V,reseed counter*):

   (a) *seedmaterial*= *entropy input*|| *追加入力*。*シード材料*の長さがちょうど *seedlen* ビットであることを確認する。

   (b) *seedmaterial*= **ブロック暗号 df**(*seed material,seedlen*)

   (c) (*K,V*) = **CTR DRBG Update**(*seed material,K,V*) (d) *再シードカウンタ* = 1.

   (e) *K,V* を返し、*カウンタ*を新しい作業状態として*再シードする*。

4. **CTR_ DRBG_ 生成**関数 [3、セクション 10.2.1.5.2]は、インスタンス化または再シード後に擬似ランダムビットを生成する。入力パラメータの妥当性をチェックし、十分なエントロピーを得るために reseed 関数を呼び出します（必要と判断された場合）。次に、generate アルゴリズムを使用して、要求された疑似ランダム・ビットを生成する。最後に、PRG の動作状態を更新し、要求された疑似ランダム・ビットを呼び出し元のアプリケーションに返します。

   CTR DRBG Generate(*K,V,reseed counter,* `) → (*prand bits*,K^0,V^0,*reseed counter*^{0}):

(a) (*reseed カウンタ*>*reseed 間隔*)の場合、*エラーステータス*= **RESEED** を設定し、reseed が必要であることを示す。

(b) if (*add input* 6= *Null*), then

- *addl input*= **ブロック暗号 df**(*addl input,seedlen*)
- (*K,V*) = **CTR DRBG Update**(*addl input,K,V*)

(c) else *addl input* = $0^{seedlen}$

(d) *temp Null*=

(e) *アウトレン* = `

(f) while (**len**(*temp*) < *outlen*) do:

i. *V* = (*V* + 1) mod 2^{outlen}

ii. *出力ブロック*= **AES ECB**(*K,V*)

iii. *temp*= *temp*|| *出力ブロック*

(g) *prand bits*= **leftMostBits**(*temp*)

(h) (K^0,$V^{(0)}$) = **update**(*addl input,K,V*)

(i) *リシード・カウンター*= *リシード・カウンター* + 1

(j) *prand ビット*、K^0、$V^{(0)}$を返し、新しい動作状態として*カウンタを再シードする*。

以上の手順が定義されたので、次にスキームで使用される高レベルの PRG 関数を以下のように定義する:

G(*K,* `)→*R*。高レベルでは、このアルゴリズムは、入力として鍵(または初期シード)*K*∈{0,1}$^n$、出力長 ` を取り、アルゴリズムはビットの擬似ランダムシーケンス *R*∈{0,1}` を返す。さらに、CTR DRBG 仕様で要求されるノンスを生成するために、耐衝突ハッシュ関数 H_l : {0,1}^{n} → {0,1}^{m}を使用する。アルゴリズムは以下のとおりである:

1. *エントロピー入力*= K と*ノンス*= H_l(*K*)。
2. *person string*= Null。パーソナライズ文字列は DRBG のオプションであるため、実装では除外しています。これは将来変更される可能性があります。
3. もし *PRG init* でなければ
   (a) $(Key, V, reseed\_counter) \leftarrow$
   (b) $PRG\_init = true$ **CTR DRBG Instantiate**(*エントロピー入力,nonce,人物文字列*)
4. (*R,Key*^{0},*V*^{0},*reseed counter*^{0}) ← **CTR DRBG Generate**(*Key,V,reseed counter,* `)
5. *エラーステータス*= *RESEED* の場合

   (α) **CTR DRBG Reseed**(*seed 素材,Key*^{0},*V*^{0})) 6. 擬

似乱数ビット *R* を返す。

2.4.3 乱数生成(RNG)

OpenABE には、OpenSSL ライブラリ[19]が提供する暗号的に強力な擬似乱数生成器をラップしたデフォルトのグローバル乱数生成器（RNG）が含まれています。OpenABE の初期化では、まず OpenSSL RNG（デフォルトでは /dev/urandom からエントロピーを取得）をシードします。ここで、PRG（セクション 2.4.2 参照）プリミティブが RNG インタフェースを拡張し、OpenABE の任意の場所で RNG を PRG と交換できることに注意してください。さらに、ペアリングおよび楕円曲線モジュール（セクション 2.5 参照）がグループ要素の選択などのためにどのようにランダム性を取得するかを制御するためのコールバックメカニズムを提供します。事実上、このアプローチにより、どのような楕円曲線/ペアリング数学ライブラリでも、そのライブラリの乱数ソースに依存したり、そのライブラリを信頼したりすることなく、サポートできる可能性があります。

2.4.4 キーストア

OpenABE Keystore は、ライブラリでサポートされるすべてのコア暗号アルゴリズムの公開鍵および秘密鍵を管理します。鍵の保管、メモリへの鍵のインポート/エクスポート、PBKDF2（セクション 2.4.1）を介した指定したパスフレーズによる鍵の暗号化に対応する。鍵ストアの実装には、与えられた暗号文にマッチする鍵の検索機能など、ABE 鍵の管理を容易にするための機能が含まれている。ABE 鍵はさらに、効率性、鍵の有効期限など、多くのメトリクスに基づいて検索できる。さらに、鍵ストアは鍵マテリアルのカプセル化を提供し、単に文字列識別子による鍵の参照を可能にする。このアプローチにより、OpenABE を既存のアプリケーションにシームレスかつ直感的に統合することができます。

2.5 ゼウトロ数学ライブラリ

Zeutro 数学ライブラリ（ZML）は、基本楕円曲線とバイリニア演算をサポートするその他の曲線をサポートする汎用数学 API を提供します。前述したように、この API は一元化されているため、上位レベルのスキーム実装に影響を与えることなく、サブコンポーネントの更新や置き換えが容易です。さらに、この設計手法により、さまざまな外部数学ライブラリの特定の長所を利用した異なるバージョンの OpenABE を展開することができます。

以下のセクションでは、ZML を構成するコンポーネントとそのコア機能について説明します。

2.5.1 ペアリング・モジュール

ZML ペアリング・モジュールは、属性ベースの暗号化スキームをサポートするために、ペアリングに適した普通の楕円曲線を提供します。このモジュールは RELIC ライブラリ[1]の上に構築されており、すべての双線形演算（ペアリング演算を含む-セクション 2.3.1 参照）を提供する。我々は、埋め込み次数 $k = 12$（または一般的に BN-254 と呼ばれる）の最先端の Barreto-Naehrig（BN）曲線[5]を使用してスキームをインスタンス化する。[10]この特殊な非対称曲線は、非常に効率的なペアリング実装と AES-128 と同等のセキュリティレベルを実現することが知られています。その結果、ABE スキーム実装の全体的な性能は、以前の取り組みよりも向上しています。BN 曲線のその他の利点としては、グループ要素の表現を圧縮できることが挙げられます。これは ABE の暗号文をよりコンパクトに

[10]より高いセキュリティ・レベル（AES-256 とほぼ同等）を提供する BN-638 を使うこともできる。

することに直結し、伝送コストを大幅に削減する。欠点としては、BN 曲線は *Type-III* ペアリングであるため、グループ G_1 に対してのみ効率的なハッシュが可能であることが挙げられる。

あるいは、ペアリング・モジュールは、ペアリングの実装効率は劣るが、より高いセキュリティ・レベルを提供する類似の非対称曲線でインスタンス化することもできる。これには、埋め込み次数 $k=24$ の Barreto-Lynn-Scott（BLS）曲線[4]や、埋め込み次数 $k=18$ の Kachisa-Schaefer-Scott（KSS）曲線[13]がある。一般に、ペアリング曲線の選択は、必要とされるペアリング計算の数や必要とされるペアリング以外の演算（*例えば*、指数、乗算、ハッシュなど）を含む様々な要因に依存する。新しいペアリングに適した曲線が発見された場合、セキュリティを保ちながら効率を最大化できるのであれば、これらの曲線を統合することができる。

2.5.2 楕円曲線（EC）モジュール

ZML は、標準化された NIST 推奨の楕円曲線のみを公開し、公開鍵暗号化と電子署名アルゴリズムをサポートします。このモジュールは OpenSSL の EC ライブラリ[19]をベースに構築されている。オプションとして、EC モジュールは RELIC ライブラリ[1]の基本楕円曲線コンポーネントでインスタンス化することもできます。

いくつかの特殊な NIST 曲線[14]から選ぶことができるが、ここでは EC モジュールを以下のサイズの素数場に限定する：P-192、P-224、P-256、P-384、P-521 である。デフォルトでは、スキームは曲線 P-256（AES-128 ビット・セキュリティに相当）で初期化される。これらの特殊な曲線を使用することで、指数化（またはスカラー倍）や乗算（またはポイント加算）を含む楕円曲線演算の効率を最適化することができます。

第 3 章

安全性の証明

3.1 セクション 2.3.8 の CCA 変換の安全性の証明

3.1.1 KEM からメッセージ暗号への変換の証明

まず、KEM スキームから安全な CPA スキームへの変換の安全性を証明する。最初のゲームは ABE スキームに対する IND-CPA セキュリティである。我々は、確率的多項式時間(PPT)攻撃者が、連続するゲーム間で無視できる優位性の差を持つこと、また、最後のゲームにおける攻撃者の優位性が 0 であることを示す。

まずは試合の流れを説明する。

試合（公認会計士）

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。
2. 攻撃者は 2 つのメッセージ $M_0, M_{(1)}$を挑戦者に提出する。
3. 次に挑戦者はコイン $\beta \in \{0,1\}$を裏返す。
4. 挑戦者は $\mathbf{Encrypt}_{\mathbf{CPAKEM}}(\mathrm{PK}, \mathbb{A}; u) \to (K', \mathrm{CT}_0^*$ を実行し、$K=$ $^{K(0)}$を設定する。）
5. 挑戦者は $\mathbf{G}(K, `) \to y$ を計算する。
6. 最後に $\mathrm{CT}_1^* = y \oplus M_\beta$
7. チャレンジ暗号文 $\mathrm{CT}^* = (\mathrm{CT}_0^*, \mathrm{CT}_1^*)$が攻撃者に送られる。
8. 攻撃者は $\beta^0 \in \{0,1\}$を提出し、$\beta=$ $\beta^{(0)の}$場合に勝利する。

ゲーム 」

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。
2. 攻撃者は 2 つのメッセージ $M_0, M_{(1)}$を挑戦者に提出する。
3. 次に挑戦者はコイン $\beta \in \{0,1\}$を裏返す。
4. 挑戦者は $\mathbf{Encrypt}_{\mathbf{CPAKEM}}(\mathrm{PK}, \mathbb{A}; u) \to (K', \mathrm{CT}_0^*)$ 。
   KEM キースペース。

5. 挑戦者は **G**(*K*, ')→*y* を計算する。
6. 最後に $\mathrm{CT}_1^* = y \oplus M_\beta$ を計算する。
7. チャレンジ暗号文 $\mathrm{CT}^* = (\mathrm{CT}_0^*, \mathrm{CT}_1^*)$が攻撃者に送られる。
8. 攻撃者は β'∈{0,1}を提出し、$\beta = \beta'$場合に勝利する。

ゲーム $_2$

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。
2. 攻撃者は 2 つのメッセージ M_0, M_1を挑戦者に提出する。
3. 次に挑戦者はコイン β∈{0,1}を裏返す。
4. 挑戦者は **Encrypt**$_{\mathbf{CPAKEM}}(\mathrm{PK}, \mathbb{A}; u) \to (K', \mathrm{CT}_0^*$ を実行し、KEM 鍵空間から *K* をランダムに選択する。
5. 挑戦者は y∈{0,1}' を一様にランダムに選ぶ。
6. 最後に $\mathrm{CT}_1^* = y \oplus M_\beta$ を計算する。
7. チャレンジ暗号文 $\mathrm{CT}^* = (\mathrm{CT}_0^*, \mathrm{CT}_1^*)$が攻撃者に送られる。
8. 攻撃者は β'∈{0,1}を提出し、$\beta = \beta'$場合に勝利する。

Lemma 3.1.1 *もし我々の基本的な ABE KEM が IND-CPA セキュアであれば、ゲーム $_1$ とゲーム $_{\mathrm{CPA}}$ における PPT 攻撃者の利得の差は無視できる。*

ABE-KEM IND-CPA セキュリティゲームを行うアルゴリズム B を考える。まず ABE 挑戦者から公開鍵 PK を受け取り、ステップ 1 のためにこれを A に渡す。次にステップ 2-3 を自ら実行する。次に KEM チャレンジ暗号文 CT^*_0と鍵 *K* を受け取る。
K はランダムに選択されるか、**Encrypt**$_{\mathbf{CPAKEM}}(\mathrm{PK}, \mathbb{A}; u) \to (K', \mathrm{CT}_0^*)$. $\mathcal{B}$ 、PRG チャレンジャーから与えられたチャレンジ値 *K* を使用してステップ 5~8 を実行する。攻撃者が勝った場合(すなわち、$\beta = \beta'$)、B は、*K* が暗号化の呼 び出しから生成されたことを示すために 1 を出力する。

K が暗号化から生成された場合、B はゲーム$_{(\mathrm{CPA})}$をシミュレートし、ランダムに選択された場合、ゲーム $_1$ をシミュレートすることが観察される。したがって、ゲーム $_1$ とゲーム$_{(\mathrm{CPA})}$における攻撃者の優位性の差は、IND-CPA セキュリティゲームにおける B の優位性となる。

Lemma 3.1.2 *もし我々の基礎となる擬似ランダム生成器 G が安全であれば、ゲーム $_2$ とゲーム$_{(1)}$における PPT 攻撃者の優位性の差は無視できる。*

PRG セキュリティゲームを行うアルゴリズム B を考える。これはチャレンジ *y*∈{0,1}' を一様にランダムに、あるいは **G**(*K*, ')→*y* として生成する。縮小アルゴリズム は、*K* を選択しないことを除いて、ゲーム $_1$ のステップ 1-4 を実

行する。攻撃者が勝った場合（つまり$\beta = \beta^0$）、B は 1 を出力して y が擬似的に選ばれたことを示す。そうでない場合は 0 を出力してランダムに選ばれたことを示す。

y が擬似ランダムに選ばれた場合、B はゲーム $_1$ をシミュレートし、ランダムに選ばれた場合、ゲーム$_{(2)}$をシミュレートすることが観察される。したがって、ゲーム $_2$ における攻撃者のゲーム $_1$ との優位性の差は、PRG セキュリティゲームにおける B の優位性となる。

定理 3.1.3 *ゲーム$_{(2)}$における(必ずしも PPT ではない)攻撃者のアドバンテージは 0 である。*

これは、y が一様にランダムに選ばれるので、$y \oplus$ $_{M(\beta)}$は一様にランダムな文字列として分布し、β *に関する*いかなる情報も持たないという事実から導かれる。

定理 3.1.4 *基礎となる PRG が安全であり、基礎となる CPA KEM が安全であると仮定すると、ABE CPA KEM から ABE メッセージ暗号化スキームへの変換は、IND-CPA 安全スキームを生成する。*

定理は、上記の 3 つのレンマから直ちに導かれる。

3.1.2 CCA 変形の証明

次に、IND-CPA セキュア ABE スキームから CCA セキュア CPA スキームへの変換の安全性を証明する。これは、ABE スキームに対する IND-CCA 安全性を持つ一連のハイブリッドゲームを提供することによって行う。我々は、確率的多項式時間(PPT)攻撃者が、連続するゲーム間で無視できる優位性の差を持つこと、また、最後のゲームにおける攻撃者の優位性が 0 であることを示す。我々の安全性の証明は、ハッシュ関数 H をランダムオラクルとしてモデル化する。

まずは試合の流れを説明する。

試合$_{(CCA)}$

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。
2. 挑戦者はすべてのクエリーを記録し（r K||| A）、入力が初めてクエリーされた場合は、ランダムな出力 u で応答する。そうでなければ、前の応答を返す。このオラクル操作はゲームを通して実行される。
3. 挑戦者は提出された任意の暗号文 CT に対して、任意の多項式回数の復号を実行する。
4. 挑戦者は攻撃者からチャレンジアクセス構造体 A* を受け取る。
5. 次に挑戦者はコインβ ($\{0,1\}$)を投げる。次に **Encrypt$_{\text{CCA- (KEM)を}}$**計算する。$(\mathrm{PK}, \mathbb{A}^*) \to (K', \mathrm{CT}^*)$. β = 0 であれば K^* = $^{K(0)}$を設定する。そうでなければ K をランダムに選択する。
6. CT* ,K^* の値が攻撃者に送られる。
7. 挑戦者は、任意の暗号文 CT 6= CT* に対して、任意の多項式回数の復号を実行する。
8. 攻撃者は$\beta^0 \in \{0,1\}$を提出し、$\beta = \beta^{(0)の}$場合に勝利する。

ゲーム $_1$

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。
2. 挑戦者は、すべてのクエリ(r K||| A)を記録し、入力が初めてクエリされた場合、ランダムな出力 u で応答する。さらに $\mathbf{Encrypt}_{\mathbf{CPA}^0}$(PK,A,$M$=($K$,$r$);$u$)→CT を計算し、($r$ K||| A)のテーブルに CT,u の両方を入力する(u のみが応答として返される)。
3. 暗号文 CT が与えられたとき、挑戦者は CT がランダムオラクルテーブルの暗号文エントリとして現れるかどうかをチェックする。そうでなければ、⊥を出力して拒否する。
4. 挑戦者は攻撃者からチャレンジアクセス構造体 A* を受け取る。
5. 次に挑戦者はコイン β ←{0,1}を投げる。次に $\mathbf{Encrypt}_{\mathbf{CCAKEM}}(\mathrm{PK}, \mathbb{A}) \to (K', \mathrm{CT}^*)$. を計算する。もし β = 0 であれば K^* = $^{K(0)}$をセットする。そうでなければ K^* をランダムに選択する。
6. CT* ,K^* の値が攻撃者に送られる。
7. 暗号文 CT 6= CT* が与えられたとき、挑戦者は CT がランダムオラクルテーブルにあるかどうかをチェックする。そうでなければ、⊥を出力して拒否する。
8. 攻撃者は $\beta^0 \in \{0,1\}$を提出し、β= $\beta^{(0)の}$場合に勝利する。

ゲーム $_2$

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。
2. 挑戦者は、すべてのクエリ(r K||| A)を記録し、入力が初めてクエリされた場合、ランダムな出力 u で応答する。さらに、$\mathbf{Encrypt}_{\mathbf{CPA}^0}$(PK,A,$M$=($K$,$r$);$u$)→$C$ を計算し、(r K||| A)のテーブルに C,u の両方を入力する(u のみが応答として返される)。
3. 暗号文 CT が与えられたとき、挑戦者は CT がランダムオラクルテーブルに現れるかどうかをチェックする。そうでなければ、⊥を出力して拒否する。
4. 挑戦者は攻撃者からチャレンジアクセス構造体 A* を受け取る。
5. 次に挑戦者はコイン β ←{0,1}を投げる。次に、挑戦者は、ランダムな $K^0 \in \{0,1\}^n$、ランダムな $r \in \{0,1\}^n$、およびランダムな u を選択することで、チャレンジ暗号文を計算する(u はランダムオラクルを呼び出して選択されるわけではないことに注意)。次に、$\mathbf{Encrypt}_{\mathbf{CPA}^0}$(PK,A* ,$M$ = (K^0,r);u) → CT* と設定する。β = 0 であれば K^* = $^{K(0)を}$設定する。そうでなければ K^* をランダムに選択する。
6. CT* ,K^* の値が攻撃者に送られる。

7. 暗号文 CT 6= CT^* が与えられたとき、挑戦者は CT がランダムオラクルテーブルにあるかどうかをチェックする。そうでなければ、⊥を出力して拒否する。

8. 攻撃者は$\beta^0 \in \{0,1\}$を提出し、$\beta = \beta^{(0)の}$場合に勝利する。

ゲーム $_3$

1. 挑戦者はセットアップを実行し、ABE スキームの公開鍵 PK を攻撃者に渡す。

2. 挑戦者は、すべてのクエリ(r K||| A)を記録し、入力が初めてクエリされた場合、ランダムな出力 u で応答する。さらに、$\mathbf{Encrypt}_{CPA^0}$(PK,A,M=(K,r);u)→C を計算し、(r K||| A)のテーブルに C,u の両方を入力する(u のみが応答として返される)。

3. 暗号文 CT が与えられたとき、挑戦者は CT がランダムオラクルテーブルのある C として現れるかどうかをチェックする。そうでなければ、⊥を出力して拒否する。

4. 挑戦者は攻撃者からチャレンジアクセス構造体 A^* を受け取る。

5. 次に挑戦者はコインβ ($\{0,1\}$)を投げる。次に、ランダムな$K', \tilde{K} \in \{0,1\}^n$、ランダムな $r \in \{0,1\}^n$、ランダムな u を選択することで、チャレンジ暗号文を計算する(u はランダムオラクルを呼び出して選択するのではないことに注意)。次に、$\mathbf{Encrypt}_{CPA^0}$(PK,A^* ,M = (K,$\tilde{r}$);u) → CT^* と設定する。β = 0 の場合、K^* = $K^{(0)を}$設定する;

   そうでなければ、ランダムに K を選ぶ。

6. CT^* ,K^* の値が攻撃者に送られる。

7. 暗号文 CT 6= CT^* が与えられたとき、挑戦者は CT がランダムオラクルテーブルのある C として現れるかどうかをチェックする。そうでなければ、⊥を出力して拒否する。

8. 攻撃者は$\beta^0 \in \{0,1\}$を提出し、$\beta = \beta^{(0)の}$場合に勝利する。

定理 3.1.5 *もし基礎となるスキームが IND-CPA セキュアであるならば、任意の PPT 攻撃者のゲーム CCA とゲーム $_1$ の差分アドバンテージは無視できる。*

この 2 つのゲームにおける唯一の違いは、復号クエリの処理方法である。2 つのケースを考える。最初のケースでは、クエリーCT が与えられ、CT はランダムオラクルテーブルの暗号文エントリである。しかしこの場合、CT はまさにランダムネス K,r で構造体 A にアクセスするために暗号化したときに生成される暗号文である。これは鍵 K の正しい暗号化なので、復号すると K が生成されるはずである。

番目の場合、CT はリストにない。この場合、元のゲームは解読されるかもしれないが、ゲーム$_{(1)は}$常に拒否される。元のゲームが復号に成功する可能性は無視できると主張する。ここで、$\mathbf{Decrypt}_{CPA^0}$(SK,CT) = M^0 = (K^0,r^0)

かつ $u^0 = H(r_K||||A)$と仮定する。CCA 復号アルゴリズムは、$\mathbf{Encrypt}_{CPA^0}(PK,A,M^0 = (K^0,r^{(0)}));u^0) \rightarrow CT$ の場合を除き、拒否する。しかし、ランダムオラクルは$(r_K||||A)$に問い合わせたものではないので、この事象が発生する確率は、与えられたメッセージの暗号化によって出力された暗号文を（暗号化に使用されたランダム性を知ることなく）事前に推測する確率によって制限される。基礎となるスキームが IND-CPA セキュアである場合、これは無視できる確率で起こるはずである。

異なる復号をする暗号文が生成される確率はごくわずかなので、優位性の差は無視できる。

Lemma 3.1.6 *基本スキームが IND-CPA セキュアである場合、PPT 攻撃者のゲーム 1 とゲーム 2 におけるアドバンテージの差は無視できる。*

(A^*, K^0, r) をチャレンジ暗号文の生成に使用されるタプルとする。攻撃者がこのタプルに対してランダムオラクルをクエリするイベントを EVENT とする。このイベントが起こるまでは、ゲーム $_1$ とゲーム $_2$ における攻撃者の見解は情報理論的に同じであることがわかる。したがって、EVENT が無視できる確率で起こるのであれば、アドバンテージの差は無視できるほど小さいと主張できる。

ここで、EVENT は本当に無視できる確率で発生することを論証する。仮に、無視できない確率で EVENT を発生させる攻撃者 A がいたとする。我々は、以下のような削減アルゴリズム B を作成する。ゲーム $_1$ と同様にセキュリティゲームを実行するが、IND-CPA 挑戦者から ABE 公開鍵 PK を受け取る。ゲーム $_1$ では、秘密鍵なしで復号鍵に答えることができる。A^* を受け取り、$(A^*, M_0 = (K^0, r))$を提出する。

$(\mathbb{A}^*, M_1 = (K', \tilde{r}))$を、ランダムに選ばれた $r, r\sim$の IND-CPA 挑戦者に返し、暗号文 CT^* これはゲームのシミュレーションに使用される。シミュレーションは、$(A^*, (K^0, r))$または$(A^*, (K', \tilde{r})$に対する問い合わせがあるまで実行される。）前者であれば 0 を推測し、そうでなければ 1 を推測する。どちらも問い合わせがない場合は、ランダムに推測する。

もしチャレンジ暗号文が(K^0, r)の暗号化であれば、$(A^*, (K^0, r))$に対するオラクルクエリは確率的に発生し、$(A^*, (K', \tilde{r})$に対するクエリは、～r がセキュリティパラメータ長であるため、無視できる確率で発生することがわかる。同様に、チャレンジ暗号文が$(K', \tilde{r})$の暗号化であった場合、$(A^*, (K', \tilde{r})$に対するオラクルクエリは確率的に発生し、$(A^*, (K', r)$に対するクエリは無視できる確率で発生する。このことから、B は IND-CPA を.に無視できるほど近い優位性で破ることになる。

定理 3.1.7 *もし基礎となるスキームが IND-CPA セキュアであれば、PPT 攻撃者のゲーム 2 とゲーム 3 におけるアドバンテージの差は無視できる。*

(1)復号化には秘密鍵は使用されず、(2)挑戦暗号文の IND-CPA 部分を生成する際のランダム性は、真にランダムに選択されるため、安全性の証明は IND-CPA ゲームに即座にマッピングされる。削減アルゴリズム B は、ステップ 1～4 を自ら実行し、K^0| r とランダムな $K^{\sim}$| r を、（攻撃者から与えられた）アクセス構造 A^* とともに IND-CPA 挑戦者にメッセージとして提出し、IND-CPA 暗号文 C を返す。$\tilde{K}$ が暗号化されている場合、攻撃者の見方はゲーム $_3$ と同じである。K^0 が IND-CPA 挑戦者によって暗号化された場合、ゲーム $_2$ における攻撃者の利点と同じである。

定理 3.1.8 *基本スキームが IND-CPA セキュアである場合、PPT 攻撃者のゲーム 3 におけるアドバンテージは 0 である。*

K~が(K'の代わりに)暗号化されている場合、攻撃者が *K*= $^{K(0)}$とランダムに選ばれた *K* を区別する利点は無視できる。

定理 3.1.9 *基礎となる ABE スキームが IND-CPA セキュアであると仮定すると、変換後のスキームはランダムオラクルモデルにおいて CCA セキュアである。*

定理は上記のレンマから直ちに導かれる。

参考文献

[1] D. F. Aranha and C. P. L. Gouvˆea. RELIC is an Efficient Library for Cryptography. http://code.google.com/p/relic-toolkit/.

[2] Elaine Barker, Don Johnson, and Miles Smid. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. NIST Special Publication 800-56 Rev 1 of March 2007.

[3] Elaine Barker and John Kelsey. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST Special Publication 800-90A Rev 1 of January 2012.

[4] PauloS.L.M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer Berlin Heidelberg, 2003.

[5] PauloS.L.M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer Berlin Heidelberg, 2006.

[6] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[7] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs #1. In *CRYPTO*, pages 1–12, 1998.

[9] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Communications of the ACM*, 55(11):56–64, 2012.

[10] J. Daemen and V. Rijmen. FIPS 197: Announcing the Advanced Encryption Standard (AES). NIST Special Publication of November 2001.

[11] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for finegrained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[12] Timothy A. Hall and Sharon S. Keller. The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS). NIST Special Publication of March 2014.

[13] EzekielJ. Kachisa, EdwardF. Schaefer, and Michael Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In StevenD. Galbraith and KennethG. Paterson, editors, *Pairing-Based Cryptography Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer Berlin Heidelberg, 2008.

[14] Cameron F. Kerry and Patrick D. Gallagher. FIPS 186-4 Digital Signature Standard (DSS). Federal Information Processing Standards Publication (Issued July 2013).

[15] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 99–112, New York, NY, USA, 2006. ACM.

[16] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. *Journal of Computer Security*, 18(5):799–837, 2010.

[17] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[18] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[19] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS, September 2014. www.openssl.org.

[20] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer Berlin Heidelberg, 2011.