クラウドサービスの鍵管理：

暗号化による望ましい結果と制約を理解する

CSAJC cloud security alliance SM Japan Chapter

CSA cloud security alliance®

The permanent and official location for Cloud Security Alliance Cloud Key Management research is:
https://cloudsecurityalliance.org/research/working-groups/cloud-key-management/

Acknowledgments

Lead Authors:

Doug Egan Ashish
Kurmi Paul Rich
Michael Roza
Mike Schrock

Contributors:

Marina Bregkou Subhojit
Goswami Anup Marwaha
Christiane Peters

Peer Reviewer:

Vrettos Moulos
Ruben Trapero

CSA Global Staff:

Marina Bregkou Stephen
Lumpe (Cover)
AnnMarie Ulskey (Layout)

日本語版提供に際しての告知及び注意事項

本書「クラウドサービスの鍵管理」は、Cloud Security Alliance (CSA)が公開している「Key Management in Cloud Services」の日本語訳です。本書は、CSA ジャパンが、CSA の許可を得て翻訳し、公開するものです。原文と日本語版の内容に相違があった場合には、原文が優先されます。翻訳に際しては、原文の意味および意図するところを、極力正確に日本語で表すことを心がけていますが、翻訳の正確性および原文への忠実性について、CSA ジャパンは何らの保証をするものではありません。
この翻訳版は予告なく変更される場合があります。以下の変更履歴（日付、バージョン、変更内容）をご確認ください。

変更履歴

日付	バージョン	変更内容
2021 年 03 月 01 日	日本語版 1.0	初版発行

CSAジャパン成果物の提供に際しての制限事項

日本クラウドセキュリティアライアンス（CSA ジャパン）は、本書の提供に際し、以下のことをお断りし、またお願いします。以下の内容に同意いただけない場合、本書の閲覧および利用をお断りします。

１．　責任の限定

CSA ジャパンおよび本書の執筆・作成・講義その他による提供に関わった主体は、本書に関して、以下のことに対する責任を負いません。また、以下のことに起因するいかなる直接・間接の損害に対しても、一切の対応、是正、支払、賠償の責めを負いません。

(1) 本書の内容の真正性、正確性、無誤謬性
(2) 本書の内容が第三者の権利に抵触しもしくは権利を侵害していないこと
(3) 本書の内容に基づいて行われた判断や行為がもたらす結果
(4) 本書で引用、参照、紹介された第三者の文献等の適切性、真正性、正確性、無誤謬性および他者権利の侵害の可能性

２．　二次譲渡の制限

本書は、利用者がもっぱら自らの用のために利用するものとし、第三者へのいかなる方法による提供も、行わないものとします。他者との共有が可能な場所に本書やそのコピーを置くこと、利用者以外のものに送付・送信・提供を行うことは禁止されます。また本書を、営利・非営利を問わず、事業活動の材料または資料として、そのまま直接利用することはお断りします。

ただし、以下の場合は本項の例外とします。

(1) 本書の一部を、著作物の利用における「引用」の形で引用すること。この場合、出典を明記してください。

(2) 本書を、企業、団体その他の組織が利用する場合は、その利用に必要な範囲内で、自組織内に限定して利用すること。
(3) CSA ジャパンの書面による許可を得て、事業活動に使用すること。この許可は、文書単位で得るものとします。
(4) 転載、再掲、複製の作成と配布等について、CSA ジャパンの書面による許可・承認を得た場合。この許可・承認は、原則として文書単位で得るものとします。

３．　本書の適切な管理

(1) 本書を入手した者は、それを適切に管理し、第三者による不正アクセス、不正利用から保護するために必要かつ適切な措置を講じるものとします。
(2) 本書を入手し利用する企業、団体その他の組織は、本書の管理責任者を定め、この確認事項を順守させるものとします。また、当該責任者は、本書の電子ファイルを適切に管理し、その複製の散逸を防ぎ、指定された利用条件を遵守する（組織内の利用者に順守させることを含む）ようにしなければなりません。
(3) 本書をダウンロードした者は、CSA ジャパンからの文書（電子メールを含む）による要求があった場合には、そのダウンロードしまたは複製した本書のファイルのすべてを消去し、削除し、再生や復元ができない状態にするものとします。この要求は理由によりまたは理由なく行われることがあり、この要求を受けた者は、それを拒否できないものとします。
(4) 本書を印刷した者は、CSA ジャパンからの文書（電子メールを含む）による要求があった場合には、その印刷物のすべてについて、シュレッダーその他の方法により、再利用不可能な形で処分するものとします。

４．　原典がある場合の制限事項等

本書が Cloud Security Alliance, Inc. の著作物等の翻訳である場合には、原典に明記された制限事項、免責事項は、英語その他の言語で表記されている場合も含め、すべてここに記載の制限事項に優先して適用されます。

５．　その他

その他、本書の利用等について本書の他の場所に記載された条件、制限事項および免責事項は、すべてここに記載の制限事項と並行して順守されるべきものとします。本書およびこの制限事項に記載のないことで、本書の利用に関して疑義が生じた場合は、CSA ジャパンと利用者は誠意をもって話し合いの上、解決を図るものとします。

その他本件に関するお問合せは、info@cloudsecurityalliance.jp までお願いします。

日本語版作成に際しての謝辞

「クラウドサービスの鍵管理」の日本語訳は、CSA ジャパン会員の有志により行われました。
作業は全て、個人の無償の貢献としての私的労力提供により行われました。なお、企業会員からの参加者の貢献には、会員企業としての貢献も与っていることを付記いたします。
以下に、翻訳に参加された方々の氏名を記します。（氏名あいうえお順・敬称略）

上田　将司
太田　吏城
菊池　弘治
塩田　英二
松浦　一郎
成田　和弘
舟木　康浩
満田　淳
諸角　昌宏

目次

1. はじめに

この文書の目的は、鍵管理システムがクラウドプラットフォーム固有のものか、外部管理か、自己運用か、あるいは別のクラウドサービスかどうかにかかわらず、クラウドサービスで鍵管理システム（KMS）を使用するためのガイダンスを提供することにあります。推奨事項は、どの形式の鍵管理システムが、異なるそれぞれのユースケースに適しているかを判断するのに役立ちます。

ハードウェアセキュリティモジュールや、暗号化ツールを含む鍵管理システムは、一般的に、セキュリティ上の利点を提供することに加えて、コンプライアンスとデータ制御の要件を満たすために使用されています。参照標準の例としては、鍵管理システムの設計ガイドと規制に広く使用されているものには、NIST（アメリカ国立標準技術研究所）によるFIPS（Federal Information Processing Standard）、コモンクライテリアと、PCI DSS（Payment Card Industry Data Security Standard）が含まれます。この文書では、これらの標準を参照します。

KMSは、目的を達成するための手段であり、それ自体が目的でないことを常に認識しておくことが重要です。KMSによって実現される機能は、ビジネスニーズに不可欠なツールです。KMSベースの機能を構築する前に、ビジネス要件を明確にすることが重要です。一部のビジネス要件には、KMSと暗号化では対処できない上に、暗号化の機能には根強い誤解が続いています。さらに、鍵管理と暗号化の規制要件は、一般的に不明確、未定義、または十分に理解されていません。この、明確さと理解の欠如には、規制当局による暗号化と暗号化システムの機能への理解不足、セキュリティおよびコンプライアンススタッフの思い込み、暗号化機能の誤った伝達、マーケットや報道機関による規制や立法の意図の誤った解釈等、多くの原因があります。暗号化は、比較的専門的な分野であり、一般に専門的な教育と実践的な経験が必要ですが、そのトピックに関心を持つ規制当局および報道機関のスタッフには、そのどちらもが不足していることがよくあります。多くの場合、誤った結論や誤った推奨事項につながるような技術的な要素の理解の誤りというよりは、暗号化が何を達成できるかについて誤解しているものです。この文書では、SaaS、PaaS、IaaSでの鍵管理システムの使用についてわかりやすく記述します。

CSA EKMガイダンスを開発するための基礎となる文書は、NISTIR7956（クラウドサービスにおける暗号鍵管理の問題と課題；2013年9月公開）です。NISTIRは、エンタープライズIT環境と比較したクラウド環境での暗号化鍵管理の複雑さを取り上げています。しかし、NISTIR 7956は、[前の段落で強調した]最初にビジネス要件を理解し、暗号化とKMSが適切なテクノロジーであるかどうかを判断する必要性を問題として取り上げていません。暗号化は、技術的にはデータの送信と保存における秘密/プライバシーのために使用されます。しかし、秘密/プライバシーに使用されるテクノロジーは暗号化だけではありません。暗号化の使用は無駄に費用がかかり、誤った安心感を与える可能性がある場合が多くあります。誤った秘密保護の感覚を伴う暗号化の無意味な使用例として、ライバルビジネスを買収する意図を明かした従業員宛に送信する電子メールを、対称鍵暗号を利用して保護することで、報道機関やライバル企業への漏洩を防ぐことを期待すること、が挙げられます。暗号化は、組織が望むプライバシーを強制することも、組織が、誰が情報を漏らしたかを判断することにも役立ちません。

NISTIR7956では、「所有の違い」はさらに複雑であると主張していますが、その理由については、述べていません。実際、NISTIRは、暗号化鍵の所有が、結果に重大な影響を与える理由を説明していません。鍵の所有はそれ自体が目的ではありません。それは、目的を達成するための手段です。鍵の所有の望ましい目標は、コンプライアンス制御のアサーション、プライバシーの主張、またはその両方である可能性があります。しかしながら、鍵の所有がプライバシーの結果に直接つながることはありません。さらに、コンプライアンス管理の主張は、プライバシー目標を達成する場合と達成しない場合があります[1]。したがって、その目的を達成するための手段の有効性に関して評価を行うことができるように、〔鍵の所有の〕目的が明確であることが不可欠です。

暗号化鍵の所有については、ほとんどの場合、誤った点に関心が示されています。これを説明するために、リースしている物理的な保管庫を考えてみましょう。法的文書、宝石、またはワインや美術品等の収集品など、ある種の貴重な品物の保管に使用されます。この場合、借手はリースを受けた保管庫の鍵を所有し、借手がその保管庫にアクセスする唯一の権利を付与する契約〔を仮定します〕に署名したとします。また、保管庫が従来の南京錠を使用してロックされていると仮定しましょう。借手が南京錠の鍵を貸手に提供することを選択することは珍しくありません。そうすることで、借手がいなくても荷物を受け取ってリース保管庫に入れることができます。このシナリオでは、南京錠の鍵は、貸し手によって“所有”されているといえます。しかし、保管庫に格納されている資産へのアクセスを決定するのは鍵の所有ではなく、保管庫を開く鍵の所持にあります。従って、重要なのは所有ではなく、所持です。法執行機関が鍵の解放を強制する捜査令状を貸し手に示した場合、貸し手は保管庫を開いて捜索を許可することができます。同様に、借手が爆発性化学物質を保管していると貸手が疑う場合、貸手は、そのような捜索が契約に含まれていなくても、借手の許可なしに内容物を検査するために使用できる鍵を持っています。これが鍵の性質です。重要なのは所持であり、所有ではありません。

NIST7956に関しては、クラウドサービスに関して同じ”所有と所持”のロジックが適用されます。クラウドサービスプロバイダと鍵を共有する組織は、クラウドサービスプロバイダが組織のデータにアクセスできるという可能性を受け入れていることになります。このSaaSのケースでは、組織はクラウドサービスプロバイダに対し、利用者のデータにアクセスするであろうことを普通に想定することになります。これは、SaaSが、利用者のデータにアクセスしないとほとんど価値がないためです。たとえば、PaaSまたはSaaSとして提供されるAIまたは機械学習アルゴリズムの場合、利用者データにアクセスしないと、AIまたは機械学習アルゴリズムが処理できるものは何もありません。初期のSaaSの機能は検索でした。YahooまたはLycos(米国を中心に展開されるポータルサイト）にインデックスを付けて、人々がWWWでそのコンテンツを検索して見つけることが出来るようにする機能です。Webサイトがそのコンテンツを暗号化し、検索プロバイダがコンテンツを復号するための鍵を持っていなかった場合、こうしたインデックスは作成できませんでした。

さらに、所有と所持でさえ、データアクセスの考えられるすべてのシナリオを網羅しているわけではありません。例えば、法執行機関が鍵の所持者に鍵の提示を命令する場合を考えてみましょう。法律は、所持者が鍵の所有者でない場合でも、所持者が鍵の所有者に通知することを禁止する一方で、法的要求に応じて鍵を使用するように所持者に指示する場合があります。この状況では、法執行機関が直接鍵を所持することはありませんが、別の主体が鍵を強制的に使用することでコンテンツにアクセスできます。

[1] [クラウドサービスからの]データの保存と取得のみが必要な場合は、プライバシーの目標を達成するために、暗号化鍵の単独の所有、所持、および使用ができます。つまり、鍵の所有者は、鍵を所持して使用できる唯一の主体でもあります。クラウドサービスプロバイダが鍵を所持することはないため、暗号化された形式のデータの保存と取得のみを提供できます。これにより、データがクラウドサービスプロバイダで利用できなくなり、サービスプロバイダへの法的要求による第三者のアクセスがブロックされます。ただし、データ処理が必要なSaaSの場合は、データを暗号化せずにレンダリングする必要があります。SaaSプロバイダが暗号化されていないデータにアクセスできる場合、鍵の所有はデータのプライバシーとは無関係です。

Webセキュリティとプライバシーの標準は、暗号化鍵の所有権ではなく、鍵の所持と使用に焦点を当てるべきです。所有は、データへのアクセスに関して技術的には無関係ですが、暗黙的または明示的に法的権利または義務を提供しているかもしれません。クラウドサービスでデータのプライバシーとセキュリティを提供するために使用される暗号化に関しては、暗号化鍵の所有、所持、および使用を調査する必要があります。

暗号化の使用と、鍵の所有権、使用、および所持権のダイナミクスに関する上記の説明は、このガイドの読者が暗号化を使用するときに、KMSの設計と実装の選択について、結論を出したり、仮定を立てたりする前に、最初に望ましい結果に焦点を当てることによって、クラウド環境で鍵管理システム（KMS）を活用するアプローチを開始する必要があることを示しています。すべての鍵管理システム（KMS）はツールであり、目的を達成するための手段であるため、データを保護するための暗号化の制限だけでなく、「望ましい結果」を理解することが重要です。

1.1 鍵管理の歴史

鍵管理には、長い歴史があります。いくつかの古代の暗号鍵の生成の証拠は、紀元前４～５世紀にさかのぼることが出来ます。しかし、この文書では、近代のクラウドコンピューティングについてのみ説明することを目的とします。”クラウドコンピューティング”への最初の言及は、1990年代半ばになされ、一般化されたクラウドサービスの出現は、Amazon Elastic Computingのサービスが開始された2006年と考えられます。2010年に、Key Management Interoperability Protocol（KMIP)が公開され、企業にとってますますマルチインスタンスで、異種の、相互接続されたテクノロジーの世界になるための橋頭堡が確立されました。多くの技術標準には、PKIと主要な鍵管理基盤がありましたが、これらの技術はまだクラウドコンピューティングの進化に適応していませんでした。AmazonがAmazon Key Management Servicesの立ち上げを発表したのは、AmazonEC2サービスの立ち上げから８年後のことでした[2]。その立ち上げ前は、主要なクラウドサービスプロバイダの利用者は事実上、鍵管理のためのマネージドサービスオプションを持っておらず、旧来の（オンプレミス展開用に構築された）鍵管理製品を使用するか、クラウドサービスごとに利用可能な暗号化オプションを選択していました。たとえば、Amazonは最初に、S3とRedShift[3]、RDS for Oracle、RDS for Microsoft SQL Server用の特注の暗号化ソリューションを提供していました。これらのサービスごとの暗号化オプションは、実際の鍵管理サービスを提供せず、クラウド鍵のみを使用していました。従来の鍵管理アプリケーションは、オンプレミスでの使用のみを念頭に置いて設計されていたため、通常、クラウドアプリケーションおよびサービスと統合するために追加のソフトウェア開発が必要でした。

2013年のAmazon KMSの発売以降、利用者はクラウドネイティブソリューションを選択し、オンプレミスとクラウドの両方で、Amazonサービスと利用者自身のアプリケーションの両方の鍵の生成と使用を制御できるようになりました。その後、オンプレミスで生成および/またはホストされた鍵を統合するオプションが利用可能になりました。同じ頃、MicrosoftはAzure Key Vault[4]と呼ばれる独自の基本的な鍵管理（クラウド）サービスをリリースし、Amazonのオプション/システムと同じ機能の多くを提供しました。間もなく、これらのクラウドサービスは暗号化鍵を超えて機能を拡張し、パスワード、接続文字列、その他の秘密情報を追加できるようになりました。彼らはまた、クラウドであるなしに関わらずあらゆるアプリケーションに統合するためのソフトウェア開発キット（SDK)を提供開始しました。この進化により、クラウド鍵管理サービスは、認証、バージョン管理、より豊富な委任などの鍵管理機能を備えたシークレット管理サービスとして再定義されました。2017年にGoogleは、AmazonおよびMicrosoftと同様の機能を備えた独自の鍵管理サービス[5]を出荷しました。

[2] https://aws.amazon.com/blogs/aws/new-key-management-service/
[3] https://aws.amazon.com/redshift/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

過去数年間で、主要なクラウドプロバイダは鍵管理サービスを強化し、新しい鍵長、管理機能、専用のテナント機能、およびコロケーションオプションを追加しました。これらの変更は、金融サービス、防衛、製薬、政府機関などの、より洗練された利用者の要件を満たしたいという願望を反映しています。クラウドプロバイダはまた、クラウドプロバイダのプラットフォームでホストされているサイトのTransport Layer Security（TLS）を有効にするための暗号化鍵のプロビジョニングを商品化し、DigiCert、Global Sign、Entrustなどの老舗の既存プロバイダから市場シェアを獲得しています。これは、特により多くのデバイスがクラウドサービスと統合され、IDと認証に鍵を使用するようになるにつれて、主要なクラウドプロバイダが鍵管理において、より大きな役割を果たすことの前兆と見ることもできます。

1.2 ゴール

この文書では、KMSをクラウドサービスと組み合わせて使用して、セキュリティとコンプライアンスの要件を満たすことを支援（ユーザ/開発者、設計者、アーキテクト、管理者、オペレーター、開発者、監査人）するための推奨事項を提供します。主要な管理機能を利用者に提供するクラウドサービスプロバイダ向けに、追加の推奨事項が提供されます。

1.3 対象読者

対象読者には、クラウドプロバイダ、クラウド利用者、CISO、規制当局、開発者、アーキテクト、鍵セキュリティやセキュア開発にかかわるセキュリティおよびコンプライアンス担当者、クラウドサービスにおいて、鍵管理サービスを運用、利用する方が含まれます。

2. KMS 概念アーキテクチャ

2.1 定義

暗号モジュールは、「暗号アルゴリズムを含む暗号ロジックまたはプロセスを実装するハードウェア、ソフトウェア、ファームウェア、またはそれらの組み合わせの集まりで、モジュールの物理的な境界を確立する、明示的に定義された隣接するモジュールの「暗号境界」内に含まれるもの」［RFC4949］、［NIST FIPS 140-2］です。ハードウェアベースの暗号モジュールの例としては、トラステッドプラットフォームモジュール（TPM）、スマートカード、ハードウェアセキュリティモジュール（HSM）があり、ソフトウェアベースの暗号化モジュールには、あらゆるタイプの暗号化ライブラリ、暗号化ソフトウェアなどが含まれます。

FIPS（Federal Information Processing Standard）140-2、ISO/IEC 15408 Common Criteria for Information Technology Security Evaluation（コモンクライテリアまたはCCと呼ばれる）は、暗号モジュールの保護レベルを認証するための国際規格です。

様々な業界（特に金融部門）の規制当局は、重要なデータやプロセスを保護するために FIPS 140-2 認証を受けた暗号モジュールのみを使用することを求めています。保護レベルは暗号モジュールの使用状況に依存します。

[4] https://azure.microsoft.com/en-us/services/key-vault/
[5] https://cloud.google.com/security-key-management

特に、規制当局は、ビジネス上重要な暗号鍵を保護するために FIPS140-2 レベル 2 以上の認証を受けた暗号モジュールの使用を要求しています。これらのモジュールは、暗号鍵を管理、生成、安全に保存するために使用される、いわゆるハードウェアセキュリティモジュール（HSM）の形で提供されることがよくあります。

暗号モジュールは、暗号アルゴリズムを自動化しますが、鍵の配布、鍵の登録と登録解除、鍵のバックアップと復元、鍵の更新、鍵の失効、鍵管理の維持などの鍵のライフサイクル管理を提供するとは限りません。監査人は、使用されている暗号アルゴリズムや鍵の保管だけを監査するのではなく、暗号鍵がどのように使用されているかを監査することに注意が重要です。

鍵管理システムは、暗号モジュールを中心に構築され、暗号機能と鍵のライフサイクル管理を活用してアプリケーションやサービスを接続するシステムです。また、鍵管理システムの機能を拡張して、シークレットや証明書を管理することも可能です。NIST [NIST SP [800-57] Part1]では、KMSを暗号鍵とそのメタデータ（生成、配布、保存、バックアップ、アーカイブ、リカバリ、使用、失効、破棄など）を管理するためのシステムと定義しています。自動化された鍵管理システムを使用して、鍵管理プロセスを監視、自動化、保護することができます。

2.1.1 BYOK の意味

頭字語 “BYOK” (Bring Your Own Key) や、“HYOK” (Hold Your Own Key)、“CYOK” (Control Your Own Key) という関連語が、2013年頃からテクノロジー企業のマーケティング用語として使われるようになりました。これらの頭字語の意味は企業間だけでなく、多くのクラウドサービスを提供している１つの企業内でも異なりました（あるサービスでは BYOKという用語を使用して、利用者が所有する KMSから鍵を持ち込むことを意味する場合もあれば、同じ企業の別のサービスでは、クラウド事業者が所有するKMSから鍵を利用者が所有し管理することを許可することを意味する場合もありました）。このような標準化の欠如は、当然のことながら多くの混乱をもたらし、その混乱は現在も続いています。多くのテクノロジー企業がこれらの頭字語を使って製品やソリューションを市場に投入しているため、これらの用語の共通の意味がさらに損なわれています。IETF のような標準化団体が参照規格を公開する技術的な理由がないため、これが変わることはないでしょう。NIST SP 800-57 などの NISTドキュメントには"BYOK" や "Bring Your Own Key” についての言及はなく、同ドキュメントでは、秘密鍵の ”所有者” の定義は、「秘密鍵」を使用することを許可された任意のエンティティであり、対称鍵の場合は「鍵を使用することを許可された任意のエンティティ」となっています。秘密鍵や対称鍵はクラウドサービスプロバイダによって使用されることになるため、SaaS の市場に出回っている BYOK スキームのほとんど(すべてではないにしても)では、この所有者の定義を完全に否定しています。組織が所持を決定するには、「どのエンティティがこの鍵を使用できるか」を確認する必要があります。所持は、鍵が別の当事者によって使用されるように構成されている場合、鍵の作成元に限定されません。

理解しておくべき重要なことは、BYOKは特定の実装ではないということです。この用語/頭字語は、技術的な設計や結果を明確に示すものではなく、またその結果として生じる法的リスクを示すものでもありません。これは単なるマーケティング用語です。したがって技術専門家は、この用語を使用すべきではなく、専門家コミュニティや報道機関の間でこの用語の使用を積極的に思いとどまらせるべきです。

この文書は技術情報と推奨事項を伝えるものであり、BYOK や HYOK などの用語は技術用語ではなく、それらはマーケティング用語です。これらの用語が意図しているのは、何らかの望ましい結果（通常は、データへのアクセスに対する実際の制御や、データへのアクセスに対する制御の外観）を表現することであるため、技術専門家が行わなければならないのは、特定した KMS モデルやパターンを使用することによってどのような結果が得られるか、あるいは得られないかを、実際の技術用語で説明することです。

2.2 一般的なクラウド KMS パターン

クラウドコンピューティングと関係するほとんどの最新のコンピューティング技術と同様に、クラウドコンピューティングと鍵管理のソリューションには、多くの可能性のあるアーキテクチャの組み合わせがあります。組織がアーキテクチャの選択にどのようにアプローチするかに影響を与える主な疑問は、以下のようなものがあります：

1. 組織で既に使用しているKMSシステムは何か？
2. クラウドサービスを利用する上で、必要な機能、運用、ビジネスの成果は何か？

最初の質問への答えは、２番目の質問への答えよりも切実ではありません。実際、一部のクラウドサービスは既存のKMSとの統合を許可またはサポートしていないため、最初の質問の答えは無関係である可能性があります。最初の質問は、組織が２つ目の質問への回答を明確に文書化した後で対応した方が良いでしょう。２番目の質問では、コスト、複雑性、安定性、サービスレベル合意書、プライバシー、透明性、法規制への準拠、導入のスピードなどを考慮して、希望する成果の長いリストが出てくることがあります。これらの結果は、相互運用性、構成可能性、標準への適合性、組織で利用可能な KMS 関連のスキルなどの考慮事項によって、さらに推進される可能性があります。様々なアーキテクチャパターンは、これらの考慮事項に関連した長所と短所を示しています。

後述するアーキテクチャパターンでは、どのクラウドもパブリックまたはプライベートのいずれかであることに注意することが重要です。ここでは、最も一般的に見られるパターンを示していますが、複数のパターンを混在させたり、パターン間の統合を活用したりする実装（例：マルチクラウドKMS）のために、アーキテクチャを一般化することができます。

過去10年間で、クラウドサービスのアーキテクチャと、多くのクラウドサービスをサポート、統合できる鍵管理システムが大きく進化しました。単一の組織が所有するエンタープライズデータセンターハードウェアとして従来考えられていたKMSの機能は、クラウドの規模とパフォーマンスで解き放たれました。この進化により、多くの組織がクラウドKMSオプションに引き付けられます。これは、従来のエンタープライズKMS実装に伴う歴史的な課題（コスト、複雑さ、脆弱性、および活性化への意欲がほとんどない「高齢者」の状態に到達する傾向）によるものです。

業界は急速な変化の時期にあるため、組織はクラウドKMSの新しいパターンに変更、移行、または採用する機能を検討することを推奨します。クラウドKMSパターンのベンダー固有の実装の評価では、別のパターンに変更する機能、またはパターンを統合する機能を決定する必要があります。この柔軟性により、ビジネスの俊敏性が実現し、合併、買収、売却による軋轢が軽減されます。

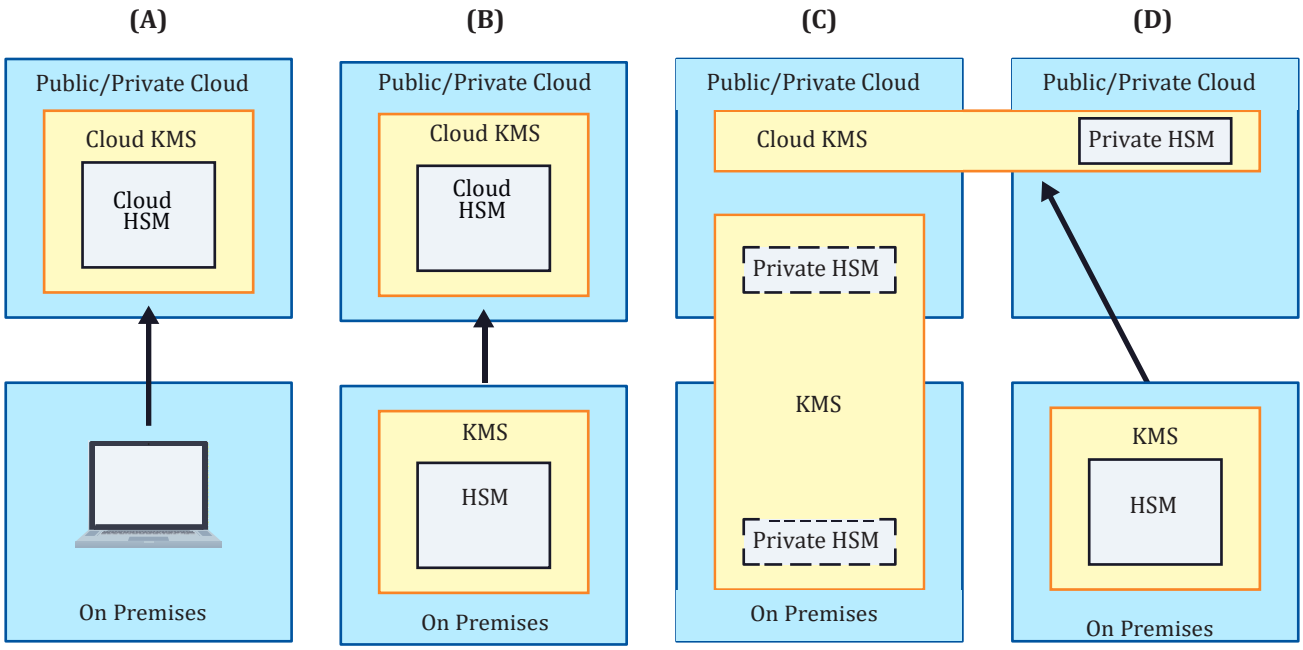

図1：クラウドサービスと鍵管理システムのパターン

図1は、次のクラウドKMSパターンを示しています：

1. （A）は同一クラウド内のKMS（HSMを含む）を活用したクラウドサービスです。
2. （B）は（A）のパターンを拡張し、外部のKMSから鍵の実体をインポートできるようにしたものです。
3. （C）は所有する組織の管理下にありますが、物理的にはクラウドプロバイダのデータセンター内でホストされている専用（プライベート）のHSMを備えたクラウドKMSです。
4. （D）はオンプレミスでもクラウドでもホストすることができ、HSMや暗号カードなどのオンプレミスの暗号モジュールにリンクされているマルチクラウドKMSの統合/管理に使用されるオンプレミスKMSを示しています。

上記の図は、４つのアーキテクチャパターンを表すことだけを目的としたものではありません。これは、基本的なクラウド鍵管理コンポーネントを単一のクラウドに限定したり、複数のクラウドサービスやプロバイダに拡張したり、組織の従来のデータセンター環境内に常駐するシステムを含めたりできるソリューションを構成できることを示しています。次のセクションでは、過去10年間に出現した４つの主要なクラウド鍵管理パターンを検証します。

クラウドKMSシステムの既存の実装のリストは、Appendix B にあります。

2.2.1 クラウドネイティブ鍵管理システム

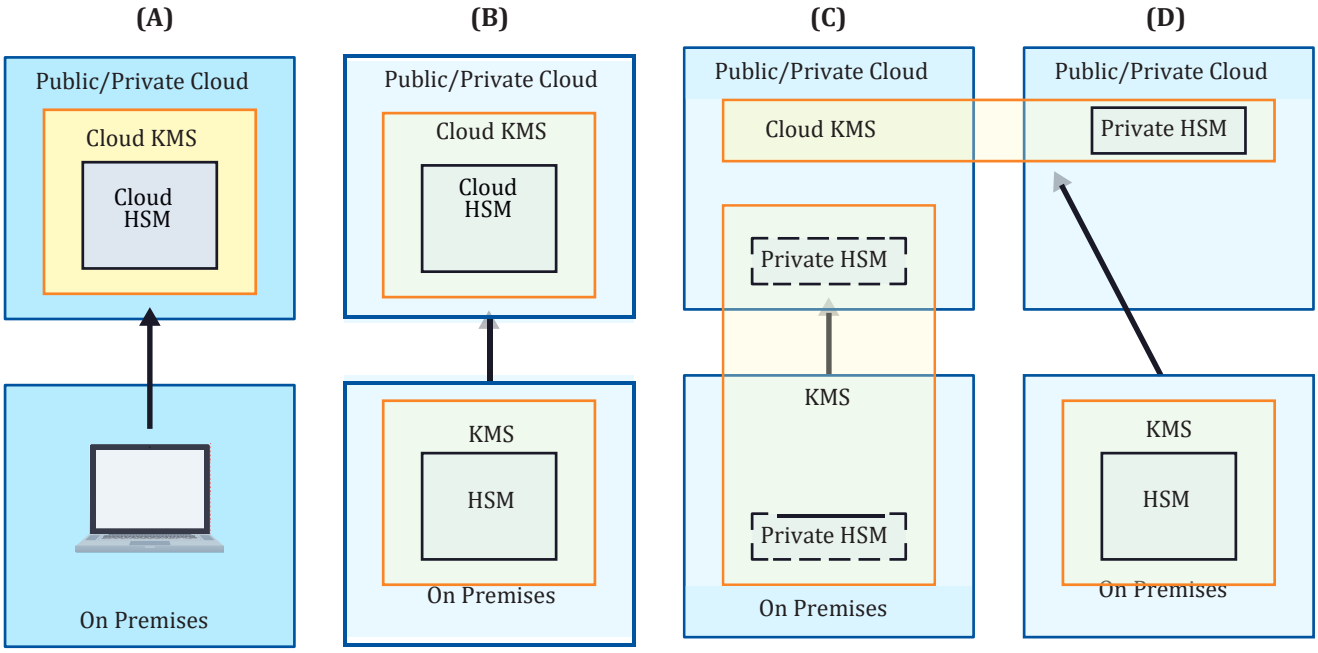

図2: クラウドネイティブパターン

定義

このパターンの特徴は、利用者が使用するクラウドサービスを提供するプロバイダと同一のプロバイダがKMSを構築し所有していること、およびKMSのすべてのコンポーネントがクラウド上にあることです。図1はこれをパターン(A)として図示しています。KMSは、使用されるサービスと一体のものである場合や、同一のクラウドサービスプロバイダ内で別途に提供されるKMSサービスの場合があります。このモデルでは、クラウドプロバイダが選択し運用する暗号アルゴリズムとモジュールを使用します。

属性

- すべてのハードウェアとソフトウェアの構成要素はプロバイダの管理下にあり、KMSはその公開文書を除き、利用者にとっては典型的な「ブラックボックス」です
- 一般的に、クラウドサービスと当該KMS機能の間の職務分掌は限定的なもの、あるいは、全く分掌がありません
- SLAを実施するためには最も悪影響が少ないモデルです
- クラウド環境は、利用者の平文データへのアクセスを持つ、または持つ可能性があります
- このパターンは、プロバイダの外部のKMS機能への拡張性は低い可能性がありますが、同一のプロバイダ内での他のサービスへの容易な拡張性を提供します
- 価格は、利用者が購入するクラウドサービスに組み込まれていることが一般的で、そのため、価格は隠されていたり、識別が困難であったり、ゼロであることが前提とされている場合があります
- コストは、KMSの機能を管理・運用するために要する技術的な専門知識 によって変動します
- 実装に要する時間は最小です
- 一般的には高性能です
- 一般的には高いスケーラビリティを持ちます

- レイテンシーによる影響は少ないです
- 他のクラウドKMSパターンへの進化をサポートできます
- FIPSに関するサポートは当該クラウドKMSによる制限を受けます
- 一般的に、利用者のキーセレモニー要件をサポートすることはできません

課題

このモデルは、異なるクラウドプロバイダ間のデザインの違いを含んでいるため、利用するクラウドプロバイダ毎の独自の技術的知識やスキルが必要となります。さらに、多くの場合、管理モデルはアプリケーション/サービス管理者とKMS管理者を区別していないため、職務分掌が不十分になります。しかし、利用者が利用可能な実装と機能セットのシンプルさは、多くの技術領域の知識を開発する必要性と管理上の負担を軽減することができるかもしれません。

一般的に、特定のアルゴリズムやプロトコルの詳細は利用者の目に触れることはありませんが、クラウドプロバイダの文書や監査資料には、広範な技術的パラメータが記載されている場合があります。利用者は通常、KMSの構成や動作をほとんどもしくは全くコントロールできないため、このモデルは、キーロール（またはキーローテーション）期間や失効・回復等のコンプライアンス要件の全てに関し満たすことはできない可能性があります。最後に、このパターンでは、CSP（クラウドサービスプロバイダ）が鍵を持つため、クラウドプロバイダから利用者データのプライバシーを完全に確保することはできません。

利用に関する推奨事項

- 利用者が他のモデルを使用する明確な理由がない場合
- 利用者のコストが、保管やトランザクション等の、KMSの専門知識のための人員配置以外の要因によって変動する場合
- 組織がKMSの専門知識や運用サポート能力を欠いている場合
- 実装のスピードが重要な要素である場合

2.2.2 外部鍵作成

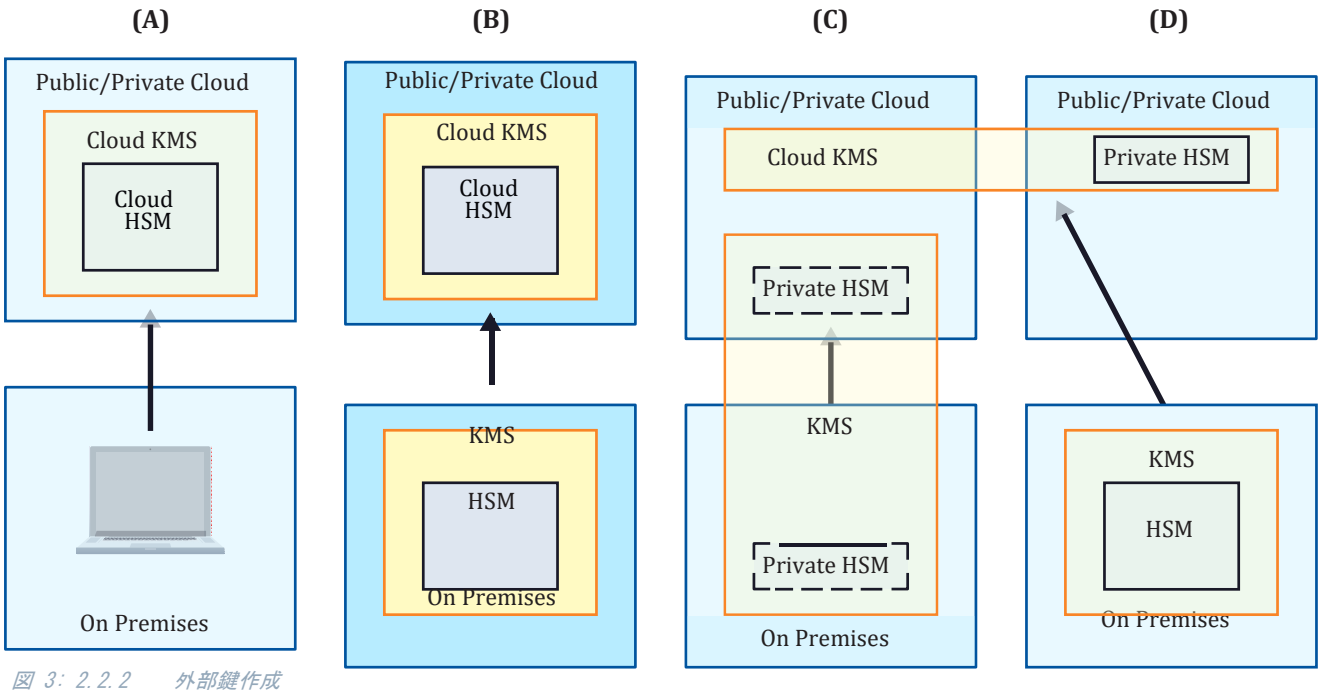

図 3: 2.2.2　外部鍵作成

定義

外部鍵作成パターンは、クラウドネイティブモデルをベースに構築され、外部KMSを用いる鍵生成セレモニー[6]を可能にします。このアーキテクチャは一般的に、何らかの方法による鍵の「所有権」を対象としたコンプライアンス要件に対応するために選ばれます。それらは多くの場合、曖昧で、ビジネスや技術的な成果ではなく、実装の詳細を対象とした用語を使用しています。このパターンでは、外部KMSとクラウドKMSとの統合が無いため、外部KMSは鍵生成機能としてのみ稼働します。最近のユースケースでは、クラウドサービスプロバイダが外部のキーマネージャへ連絡することを余儀なくされる場合がありますが、これはインシデント処理や全体的なパフォーマンスへのレイテンシーの影響が懸念されるため、CSPは一般的に敬遠しています。

BYOKやHYOKという不運な頭字語は、2.1.1節で示すように、当該の用語はいずれも標準的な技術的定義が存在しないため、それらの用語に対してパターンを評価するための信頼のおける方法は無いものの、どちらも当該のモデルが使用されているという期待を反映することができます。
このパターンの主な特徴は、1つ以上の主要な鍵生成プロセスで使用されるソフトウェアとハードウェアを利用者がコントロールしていることです。通常、このパターンはルート鍵のみの外部生成をサポートしており、その他の鍵はすべてクラウドKMS内で生成されます。

[6] See https://en.wikipedia.org/wiki/Key_ceremony

属性

- クラウドのハードウェアおよびソフトウェアコンポーネントはプロバイダの管理下にあり、利用者はクラウドの外部にあるKMSを管理し、また一般的にはルート鍵の生成とクラウドKMSへのインポートに使用します
- ルート鍵生成のための明確な職務分離を提供します
- クラウドのSLAに影響を与える可能性はほとんどありませんが、外部キーのインポートプロセスがリカバリ業務に影響を与える可能性があります
- クラウド環境は、利用者の平文データへのアクセスを持つ、もしくは持つ可能性があります
- このパターンは、プロバイダの外部へのKMS機能の拡張性は低い場合がありますが、同一プロバイダ内の他のサービスへの容易な拡張姓を提供します
- 価格設定は、通常、外部キーインポート機能のライセンシングによって変動します
- コストは一般的に、鍵のインポート機能のライセンス価格、インポートされる鍵の数、および鍵をインポートする外部KMSの管理策によって変動します
- 実装時間は、通常は、鍵の生成とインポートに使用される外部KMSの取得と設定によって変動します
- パフォーマンスは通常、ネイティブKMSパターンと同じです
- スケーラビリティは通常、ネイティブKMSのパターンと同じです
- レイテンシー感度は通常、ネイティブKMSパターンと同じです
- 一般的に、このパターンは、ネイティブKMSパターンに比べ、他のパターンへの進化にさらなるサポートを提供します
- 外部KMSによって生成された鍵に関するFIPS準拠機能を強化できますが、通常は鍵暗号化鍵に限定されます
- 通常 、ルートキーに関する利用者によるキー生成セレモニー要件を満たすことができます

課題

クラウドネイティブパターンの課題がすべて当てはまります。さらに、クラウドプロバイダは、外部キー生成のためのハードウェア製造企業を限定し、サポートしている場合があります。したがって、互換性のあるKMSハードウェア/ソフトウェアを取得する必要がある場合があり、そのためのスキルの習得が必要となることから、実装に要する時間が長くなる可能性があります。

外部KMSを使用して生成された鍵は、バックアップ・コピーとしては有用でない可能性があることに留意すべきです。プロバイダは、利用者がインポートした鍵に重要なメタデータを追加する場合があり、オリジナルの鍵が復元操作に使用できなくなる場合があります[7]。

利用に関する推奨事項

- クラウドネイティブのパターンでは 対応できない特定のキー セレモニーを利用者が要望している場合
- 利用者が、サポートされている最新モデルを介して外部鍵管理の保護を追加したい場合

[7] SalesforceとMicrosoftの両方とも、この特性を実装しているか機能を持っています。https://developer.salesforce.com/docs/atlas.en-us.securityImplGuide.meta/securityImplGuide/security_pe_byok_setup.htm and https://docs.microsoft.com/en-us/microsoft-365/compliance/cus- tomer-key-overview?view=o365-worldwide 参照

2.2.3 外部鍵管理システムを使用したクラウドサービス

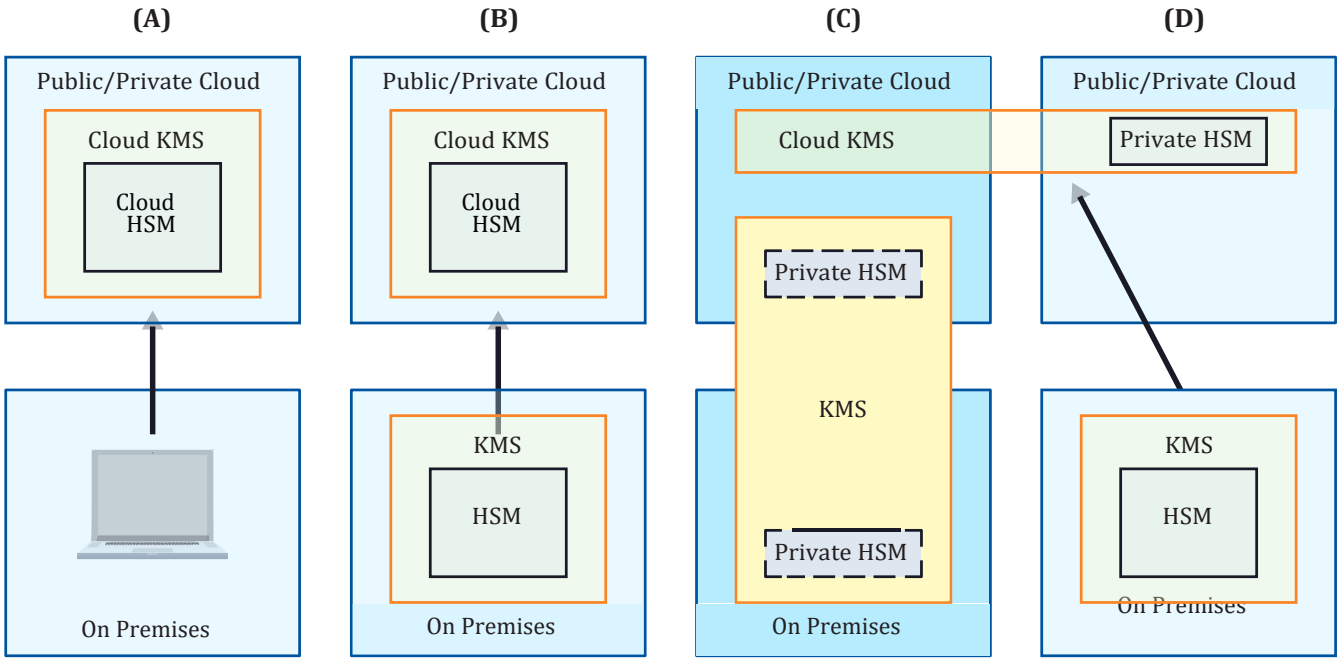

図 4: 2.2.3 外部鍵管理システムを使用したクラウドサービス

定義

外部鍵管理システムは、KMSが完全に外部へホストされたクラウドサービスを使用します。それは利用者の施設内か、利用者が選択したサードパーティによってホストされているか、またはそれらを組み合わせるか、のいずれかに該当します。ハードウェアは利用者もしくはクラウドプロバイダの所有物かもしれませんが、利用者のために用意されるものです。クラウドプロバイダは、専用のクラウドHSMサービスの提供をサポートする場合と、クラウドプロバイダが所有する施設で利用者のハードウェアをホストするコロケーションモデルをサポートする場合、があります。利用者はKMSのあらゆる側面を管理し、KMSがサービスインシデントの原因となった場合を考慮して、サービスレベルアグリーメントに同意するのが一般的です。クラウドプロバイダからデータの機密性を確保するため、CSPによる鍵のラッピングやアンラッピングが実行できない仕組みを用いることがあります。

属性

- 高度な利用者のコントロールと構成管理。アルゴリズム、プロトコル、そして鍵の長さなどを利用者が把握し、構成変更が可能です
- KMSと同様に、KMSとクラウドサービスのアクティビティを完全に分離することが可能です
- KMSのシングルポイントでの統合管理をサポートする可能性があります
- 鍵情報がクラウドプロバイダと共有されないように、また利用者のプレーンテキストデータがプロバイダへ公開されないようにすることが可能です
- KMS機能の大半がクラウドサービスで実装されているため、このパターンは高いポータビリティを提供します

- 専用ハードウェアおよび単一テナントのKMSモデルを選択することでコストが増加します。スタッフの配置も要因となりえます。通常、CSP暗号化がコスト要因になることはほぼありません
- コストは外部KMSとすべてのコンポーネントの管理によって決定する場合が大半です
- このパターンでは実装に掛かる時間が長くなる場合があります。このパターンをすでに組み込んでいる組織は、新しいクラウドサービスを実装するまでの時間を軽減できる可能性があります
- パフォーマンスが制限の要因となる場合があります。パブリッククラウドまたはプライベートクラウドのKMSを使用することで、パフォーマンスの制限が緩和される可能性があります
- 拡張性が制限の要因となる場合があります。パブリッククラウドまたはプライベートクラウドを外部KMSとして使用することで、拡張性の制限を緩和することができます
- このパターンを利用すると、クラウドサービスで遅延が発生する場合があります
- 一般的に、このパターンは追加および削除に関する制限はありません
- すべての鍵にユーザによるFIPSコンプライアンスを提供することができます
- ユーザが定義する鍵生成の要件を満たすことができます

課題

このモデルは他のパターンほど一般的でないため、選択したサービスモデルによっては利用できない場合があります。[8] このモデルを用いるとサービスレベルアグリーメントに触れる場合があり、サービス障害発生の際にユーザが補償を受けることができない場合があります。このモデルを用いる期待は、ユーザがプレーンテキストを利用しても機密性を確保できることであるため、一般的にこのモデルはSaaSや、ユーザ情報をプレーンテキストで処理するようなクラウドプロバイダのシステムとは互換性がありません。ユーザはKMSを管理するための適切なスキルを有していることが求められます。

使用に関する推奨事項

- クラウドサービスプロバイダにネイティブなKMSが用意されていない場合
- ユーザがオンプレミスとクラウドサービスの両方で使用される単一のKMSを望む場合
- クラウドプロバイダからプレーンテキストの秘匿性を保つ要件に対応が可能
- FIPS認定レベルを満たす機能の提供が可能

[8] 2020年6月、WGはGoogle、Salesforce、Oracle、Dropboxを比較し、これらがこのモデルをサポートしていないことを見つけました。 MicrosoftとAmazonだけが“bring-your-own- HSM”または“dedicated HSM”を持っているようで、すべてのサービスで利用できるとは限りません。

2.2.4 マルチクラウド鍵管理システム(MCKMS)

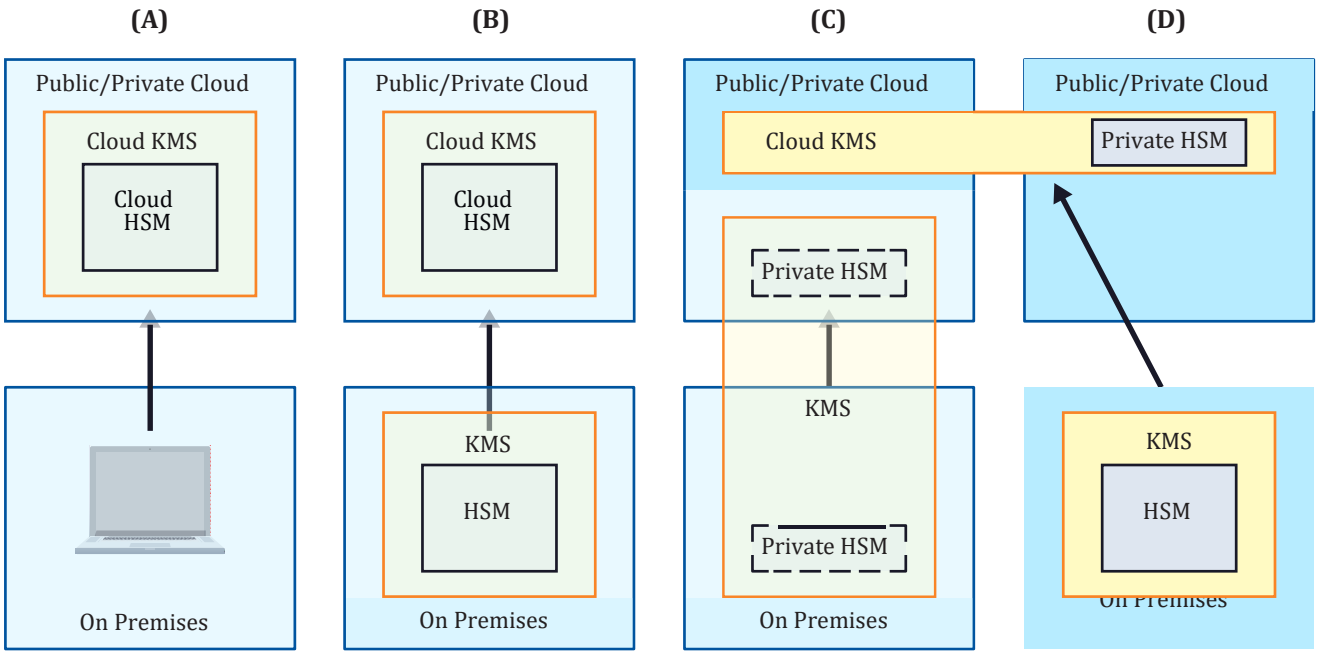

図4: 外部鍵管理システムを使用したクラウドサービス

定義

マルチクラウドKMSパターンはKMSの実装とクラウドサービスを密接に組み合わせる機能を示しています。外部KMSをサポートするクラウドサービスはすでに存在しており、KMSを複数のクラウドへ拡張したり、複数のKMSを選択して拡張したりすることができます。

属性

- このパターンの属性(コスト、複雑さ、コントロール、実装にかかる時間、規模、パフォーマンス、相互運用性)は他のクラウドKMSパターンで実装されている機能です
- クラウドKMSを別のクラウドKMSのバックアップとしても活用できるため、クラウドスケールでのフォールトトレランスが利用可能です
- これまでに培った専門知識とリソースへ投資することで、クラウドサービスの採用を加速させるという副次的な効果が期待できます

課題

一般的にこのパターンは鍵管理における戦略的アプローチのために確保されています。それには幅広い専門的知識が必要とされ、エンジニアと実装に多くの時間が費やされます。また、資産(ライセンス)または運用コスト、あるいはその両方で高いコストが必要とされます。しかしながら、複数のパブリッククラウドプロバイダを活用することで、高度なポータビリティ、フォールトトレランス、拡張性が利用可能となります。

使用に関する推奨事項

- 回復性と拡張性が高く評価されている場合
- 動的な編成が発生する組織 –買収や売却が頻繁に発生する
- 複雑なコンプライアンスの要件
- 成熟したテクノロジーアーキテクチャを備えたすべての組織が対象

3. 暗号鍵管理とコントロール

暗号鍵管理はデータ暗号と復号が必要な運用コントロールを指します。本章はコントロールされた環境、すなわちコントロールとコントロール自体の運用を容易にする設定が対象範囲となります。

3.1 鍵管理コントロール環境

鍵管理コントロール環境とは、企業における鍵管理コントロールの重要性への経営陣の態勢を指します。管理環境は、経営陣によりサポートおよび承認されたすべてのポリシーと手順、およびポリシーから逸脱されたアクションで構成されます。以後の章においては適切なコントロール環境を確保するために必要ないくつかの主なポリシーを確認しています。

ガバナンスとポリシー管理

暗号鍵管理ポリシーの定義と実施はすべての鍵管理ライフサイクルのすべてのステージに影響があります。各暗号化鍵または鍵のグループは、どのデバイス、デバイスのグループ、またはアプリケーションのタイプがそれを要求できるか、およびそのデバイスまたはアプリケーションが実行できる操作（暗号化、復号、署名など）を定義する個別の鍵使用ポリシーによって管理される必要があります。さらに暗号鍵管理ポリシーは鍵管理プロセスにおいて鍵を紛失した場合に復元するため、および、鍵が要求された後にリリースするための、より高いレベルの認可に対する追加要件を指示する場合があります.[9]。

コントロール：

- 暗号と鍵管理ポリシーと手順は作成、承認、実装、維持される必要があります。
- 暗号と鍵管理ポリシーと手順は最低年単位で必要に応じてレビュー、変更されて承認される必要があります。

[9] https://www.thalesesecurity.com/faq/key-secrets-management/what-encryption-key-manage- ment-lifecycle

リスク管理

リスク管理はガバナンスにおいて必須要素です、そしてリスク管理全体において暗号と鍵は重要な役割を果たします。暗号鍵は、認可されていないアクセスを防ぐことや解読できない形式にデータを変換することに利用されます。鍵とともに利用されるアルゴリズムがデータをスクランブルもしくはアンスクランブルするために暗号鍵を構成します。さらに暗号と鍵管理のリスクは低減されなければなりません。

コントロール：

- 暗号と鍵管理の役割と責任は定義されて、権限分掌を実施して実装される必要があります。
- 暗号と鍵管理リスクプログラムはリスクアセスメント、リスク対応、リスク環境とモニタリングとフィードバックが規定され含まれる必要があります。

変更管理

変更管理のゴールは暗号環境の変更によりセキュリティの安全性が低くなる、要件を満たさない結果となることを防ぐ事です。さらに全ての暗号システムへの変更は記録され、変更によってセキュリティが低下または不十分になった場合は、以前の状態に戻せなければなりません。変更管理プロセスでは常に、変更プロセスがスムーズに進められ、変更の実装後も効率的かつ効果的に職務を遂行できるように、ユーザはその変更について事前に通知、相談、教育を受ける必要があります。

コントロール：

- 暗号化および鍵管理に関連するシステム、ポリシー、および手順の変更は、変更の効果、コスト、および利点の決定を含む手順に従って行う必要があります。
- 暗号鍵管理関連システム、ポリシー、手順の変更は全ての関連するステークホルダーと連絡を取る必要があります。
- 変更は元の状態に戻す可能性を考慮して全て記録されます。

鍵管理システム

暗号鍵管理システム（CKMS）は、ソフトウェアモジュールアプリケーションとハードウェアセキュリティモジュール（HSM）の個々のコンポーネントで構成されており、暗号鍵とメタデータの生成、確立、配布、保存、説明、一時停止、取り消し、または破棄に使用され、そして一連の主要な管理機能またはサービスを実行するために使用されます。

鍵管理サービスは鍵の生成、破棄、失効、配布また復元を含みます。いくつかの暗号鍵管理システムサービス（例、CA（認証局サービス））はサービスレベルアグリーメントまたは契約に基づいて第三者により提供される場合があります.[11]。

[10] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf Page 2, Sec 1.1, Para 2, last sentence

[11] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt2r1.pdf Page 2, Sec 1.1, Para 3

コントロール：

- 暗号プロトコルを使用して機密データを保管時、使用時または通信時に保護する必要があります。
- 暗号アルゴリズムは暗号対象のデータの機密性に基づき保護するために利用されます。
- 暗号鍵管理ストレージ、インベントリシステムは、鍵または/もしくは電子証明書が自動的に記録される必要があります。
- 鍵セキュリティ機能はパーティションやドメイン等によって、セキュリティ以外の機能から分離する必要があります。

ログの集中管理

セキュリティログは攻撃を特定するためのセキュリティイベントを記録します。
しかしながらログデータはしばしば大量であり、自動化を支援するシステムなしではレビューすることは不可能です。さらにクラウド環境においてアプリケーションはコンテナで稼働し、ログファイル内に記載する代わりに標準出力に作成されます。このようなスタンダードアウトプットはログ集中管理システムに転送され正しく解析される必要があります。
さらに、運用に関する規制では、フォレンジックで使用するためのログデータの収集、保存、保護がタイムリーに対応されていることが求められています。

コントロール：

- 重要な処理と管理活動を証明する変更不可能なログ情報のセキュアな監査証跡を維持する必要があります。

メトリック、監視とレポート

メトリックは、パフォーマンス評価を行うための基準を提供する指標です。これらは暗号化処理時間といった技術的指標、もしくはリスク管理ポリシーのレビュー頻度といった手動によるものであっても構いません。モニタリングとは、パフォーマンス低下の修正をしたり、良いパフォーマンスを再現するためのアクションが行えるような、十分な頻度と透明性を持っているものを指します。レポートとは、パフォーマンス低下の修正をしたり、良いパフォーマンスを再現するためのアクションが行えるように、適切な人に適切なタイミングで作成されるものを指します。

コントロール：

- モニタリングとレポーティングシステムは暗号操作またはポリシー、プロセス、手順と管理に透明性を提供すべきです。

監査

暗号化の監査の目的は、ガバナンス、リスク、コンプライアンスが、鍵管理ライフサイクル/フェーズど同様に鍵管理コントロール環境にも適切に組み込まれていることを確実にすることです。計画、パフォーマンス、報告、フォローアップで構成される暗号化監査プロセスは、ユニークではありません。ユニークなのは、テストされるエリアと、暗号化が会社の業務のコンテキスト内でどのように適合するかです。監査中に実行されるテストには、これらの両方のセクションにリストされているコントロールのテストが含まれますが、これらに限定されません。

コントロール：

- 暗号と鍵管理システム、ポリシーとプロセスは少なくとも年一回個別に監査される必要があります。

- テストするコントロールは、コントロールの設計と運用の有効性を判断するために、リスクベースの方法を使用して選択する必要があります。

3.2 鍵管理コントロール

鍵管理のライフサイクル/フェーズは、鍵の作成、使用、保管および破棄を管理する操作です。主な操作は、生成、アクティブ化、配布、非アクティブ化、危殆化、一時停止、有効期限、アーカイブ、リカバリ、ローテーション、失効、置換および破棄です。各操作の説明の後に、コントロールの例が示されています。

生成

鍵は生成されましたが、使用が許可されていません。この状態では、鍵は所有証明または鍵確認の実行にのみ使用できます。対称鍵または公開鍵/秘密鍵のペアは、必要な場合にのみ生成する必要があります。

制御

- すべての鍵は、乱数生成器によるセキュアな暗号モジュール内で生成されるべきです。

アクティブ化

この状態の鍵は、暗号論的に情報を保護するため(例えば、平文を暗号化するか、デジタル署名を生成する)、および/または以前に保護された情報を暗号論的に処理するため(例えば、暗号文を解読するか、デジタル署名を検証する)に使用できます。

対称鍵または秘密鍵は、使用される必要がある場合にのみアクティベートされる必要があります。公開鍵は、利用可能になったとき、または関連するメタデータに示された日付(例えば、X.509公開鍵証明書のnotBefore日付)にアクティベートされます。

コントロール:

- 鍵は事前にアクティベートされた状態(データの暗号化に使用する準備ができていない状態)で作成する必要があります。事前にアクティベートされた鍵は、有効期間/暗号化期間の開始日を入力することでアクティベートする必要があります。アクティベートを必要とせずに作成された鍵は、すぐにアクティベートすることができ、有効期限または破棄されるまで有効なままにすることができます。アクティベートと有効期限は、より適切に管理するために推奨されます。

配布

鍵の生成後、鍵、および生成された場合には乱数生成器(RBG)などのキーマテリアルは、2つ以上のエンティティ間で共有される必要があります。この鍵およびキーマテリアルの共有または転送は、手動または自動の方法で行うことができます。

コントロール:

- 鍵はセキュアなチャネルを使って、手動または自動で鍵トランスポートを使って転送されるべきです。

非アクティブ化

非アクティブ化状態の鍵は、暗号保護の適用には使用できませんが、暗号で保護された情報の処理には使用できます。

対称鍵または秘密鍵は、データへの暗号保護の適用に不要になった時点で非アクティブ化する必要があります。これらの鍵の非アクティブ化に続いて、破棄またはアーカイブが行われる場合があります。公開鍵は非アクティブ化されません。公開鍵は失効する場合(X.509公開鍵証明書のnotAfterフィールドの日付など)、または一時停止する場合(X.509標準の証明書失効リスト(CRL)など)、または失効する場合(X.509標準のCRLなど)があります。

コントロール:

- 鍵は、鍵がデータを暗号化できなくなる有効期限日に非アクティブ化する必要があります。有効期限日を指定せずに鍵を作成した場合、鍵は無期限に有効なままとなり、不適切に使用される危険性が高くなります。

危殆化

一般に、鍵は、許可されていないエンティティに配布されるか、またはそのエンティティによって決定されると危殆化されます。危殆化された鍵は、情報に暗号保護を適用するためには使用しないでください。ただし、場合によっては、鍵ペアの危殆化された秘密鍵に対応する危殆化された鍵または公開鍵を使用して、暗号で保護された情報を処理することがあります。

コントロール:

- 危殆化された鍵の情報に適用される暗号保護は停止しなければならず、危殆化された鍵は取り消されなければならない。

失効

鍵が失効された場合(言い換えると、危殆化以外の理由で失効された場合)、その鍵は処理のために使用され続けることができます。

公開鍵の失効情報は、X.509公開鍵証明書の場合、CRLまたはOnline Certificate Status Protocol(OCSP;オンライン証明書ステータスプロトコル)応答を介してリライングパーティにセキュアに伝達されます。シークレットキー(秘密鍵を含む)は、CRLまたはOCSP応答で漏洩または識別されたときに失効させる必要があります。その後、その鍵を危殆化鍵リストに追加する必要があります。

コントロール:

- 鍵権限は、許可された暗号サービス管理者によってのみ失効されます。

一時停止

一時停止された鍵は暗号による保護(例えば、平文の暗号化やデジタル署名の生成)のために使われるべきではありません。しかしながら、一時停止された鍵は一時停止される前に保護されていた情報(例えば、暗号文の復号やデジタル署名の検証)を処理するために使われる可能性があります。

一時停止された鍵または鍵ペアは、後でアクティブ状態に復元されることもあれば、非アクティブ化または破棄、あるいは危殆化された状態に移行することもあります。

鍵は、鍵の未知の状態や鍵の所有者が一時的に不在であることなど、様々な理由で使用が一時停止されることがあります。公開鍵の場合、関連する当事者に、対の秘密鍵の一時停止が通知されます。この場合にはCRLおよびOCSP応答における「保留中」が失効理由コードとして通知されるでしょう。

コントロール:

- 鍵移行、およびあらゆる状態から一時停止への移行および停止からの移行の理由は、監視され、レビューされ、承認されるべきです。

ローテーション

鍵のローテーションとは、暗号化鍵を廃棄し、その後新しい暗号化鍵で置き換えることを指します。鍵のローテーションにより、任意の1つの鍵で暗号化される情報の量が削減されます。これにより、鍵が危殆化された場合に公開されるデータの量が削減されます。ローテーションと暗号化期間は、適切なローテーションが、とりわけ鍵によって暗号化されると予想される情報の量を考慮する暗号化期間に依存するという点で、密接に関連する2つの概念です。

コントロール:

- 鍵は、新しい鍵を生成し、そのバージョンを新しいデータの暗号化に使用されるプライマリ鍵としてマークすることによってローテーションさせる必要があります。非プライマリ鍵は、以前に暗号化されたデータの復号に使用できます。

置換

鍵の危殆化または鍵の暗号期間の終了により、鍵の置換が必要になる場合があります。置換は、鍵の再生成処理または鍵の更新処理によって実行できます。

鍵の再生成は、古い鍵の値から完全に独立した方法で生成された新しい鍵で鍵を置換することです（つまり、古い鍵を認識しても置換された鍵の値は認識されず、その逆も同様です）。鍵の更新処理では、新しい鍵は通常、置換される古い鍵と何らかの関係があります。

コントロール：

- 置換鍵は鍵更新プロセスではなく、鍵の再生成によって生成する必要があります。

有効期限

1つまたは複数の鍵の使用が許可されている期間が終了すると、暗号期間が失効したと見なされます。

コントロール：

- 鍵および証明書に割り当てられた有効性/暗号期間は、データの機密性および保護データの量に比例する必要があります。

アーカイブ

鍵アーカイブとは、必要に応じて後で回復できるように、証明書の秘密鍵を保存することです。

鍵は、通常の使用では不要になったときに保存されますが、鍵の暗号期間の後に必要になる場合があります。秘密鍵または秘密鍵の例として、アーカイブされたデータの復号があります。公開鍵の例として、アーカイブされた署名付き文書の検証があります。

コントロール：

- アーカイブされた鍵は、厳密なアクセス制御を必要とするセキュアなリポジトリで管理する必要があります。

リカバリ

鍵リカバリとは、暗号鍵またはアーカイブされた鍵を元の状態に戻すことです。割り当てられた使用属性を保持するために、関連付けられた鍵属性を元の状態に厳密に複製することが重要です。鍵情報は、鍵の有効な暗号期間中のバックアップから、または情報がアーカイブされている場合はアーカイブからリカバリできます。

すべての鍵リカバリは、次の項目に基づいて適切に承認される必要があります。

- 鍵リカバリの運用シナリオ： 法執行鍵リカバリ、エンタープライズ鍵リカバリ、個別鍵リカバリなど
- 鍵リカバリポリシー：
  - 鍵リカバリ情報が生成されたとき
  - 鍵リカバリ情報が受信されたとき
  - 鍵リカバリ情報がどのように処理/検証されるか
  - 鍵リカバリ/エスクローエージェントの使用

コントロール:

- リカバリのためにどの鍵情報を保存する必要があるかを判断するには、運用の継続性に対するリスクと、鍵情報のコントロールが失われた場合に鍵情報がさらされるリスクとを比較して評価する必要があります。

破棄

鍵は破棄された状態で、使用できません。秘密鍵または対称鍵のすべてのコピーは、それらが不要になったと慎重に判断された時点で破棄する必要があります(アーカイブや再構築のため)。鍵は、物理的または電子的な手段で復元できないように、鍵生成情報のすべてのトレースを削除する方法で破棄する必要があります。公開鍵は、必要に応じて保持または破棄できます。

コントロール:

- 鍵が削除対象としてマークされると、この状態を元に戻す機能はありません。この不可逆的な状態に入る前に、ユーザに十分な警告を与える必要があります。
- ソフトウェアベースの鍵は、すべての保管場所および使用場所から確実に破棄されたことを証明することは困難です。暗号鍵のコピーが作成されている場合は、そのコピーが最終的に破棄されるように注意する必要があります。
- ユースケースで署名付き破棄証明が必要な場合は、FIPS140-2ハードウェアを使用し、鍵のライフサイクル全体を通して監査ログを有効にすることをお勧めします。
- 鍵とそれに関連付けられた属性値は破棄されますが、鍵メタデータ(鍵名、タイプ、ハッシュ値、暗号期間、鍵状態遷移履歴、使用方法など)は組織のデータ保持ポリシーに従って保持されます。
- それらの存在の記録は、将来の監査目的のために削除されない場合があります。鍵エスクローはそのような監査可能なユースケースの1つであり、破棄の証明がCSPとエンドユーザとの信頼関係にとって最も重要になる場合があります。この場合、監査ログは、セキュリティで保護されたハードウェアデバイスを使用して、タイムスタンプ付きの署名で強化します。
- 鍵が破棄された後に、破棄された鍵の危殆化が決定される可能性があります。このような危殆化状態は、鍵状態遷移履歴テーブルに記録することをお勧めします。

4. プログラマティックライブラリインターフェイス

REST (REpresentational State Transfer)とSOAP (Simple Object Access Protocol)は、業界で最も一般的に使用されているAPIアーキテクチャです。よく比較されますが、本質的に異なる目的に使用されます。SOAPは、Microsoftによって開発された標準ベースのWebサービスアクセスプロトコルです。REST APIはURI (Uniform Resource Identifier)、HTTPプロトコル、データフォーマットのためのJSONを利用します。SOAPは長い間、Webサービスインターフェースの標準プロトコルでしたが、現在の業界はRESTアーキテクチャに支配されています。

RESTとSOAPの両方に対する推奨事項を以下に示します。

4.1 REST 推奨事項

RESTは、APIを構築するためのリソースベースのソフトウェアアーキテクチャスタイルです。RESTfulなWebサービスは、HTTPを除き、規定の標準に従っていません。したがって、開発を容易にし、かつクライアントの導入を増やすために業界のベストプラクティスに従って、RESTful APIを構築することが重要です。REST固有のセキュリティのベストプラクティスをいくつか紹介します。

シンプルにする/メカニズムの経済性

デザインはできるだけシンプルであるべきです。すべてのコンポーネントインタフェースとその相互作用は、理解するのに十分なほど単純でなければなりません。ソリューションが 「不必要に」 複雑になるたびに、穴が残る可能性が高くなります。

認証

常にHTTPSを使用する必要があります。常にTLSを使用することで、認証の資格情報は、HTTP Basic認証のユーザ名フィールドで配信されるランダムに生成されたアクセストークンで単純化されます。使用することは比較的簡単で、多くのセキュリティ機能が追加コストなしで含まれています。
パフォーマンスを改善するためにHTTP2を使用することは、検討する必要があります。1つの接続で複数のリクエストを送信できるため、後のリクエストでTCPおよびTLSハンドシェイクのオーバーヘッドを完全に回避できます。

ユーザ名/パスワードの代わりに、生成されたAPIキーを使用することが望ましいです。APIセキュリティキーは、所有者のみが読み取り可能なファイルに安全に保存する必要があります。これはSDKで使用できる鍵を維持しながら、鍵の公開を制限します。

認可

HTTPメソッドは保護される必要があります。Incoming HTTPメソッドがセッショントークン/APIキーに有効であり、リソース・コレクション、アクション、およびレコードに関連づけられることを保証される必要があります。ホワイトリストで許可されるメソッド:許可されるバーブだけが動作し、他のすべてのバーブが適切な応答コード(例:403 Forbidden)を返すように、許可されるバーブを適切に制限する必要があります。特権を持ったアクションおよび機密リソース・コレクションは保護する必要があります。セッショントークンまたはAPIキーは、権限のあるコレクションまたはアクションが不正使用から適切に保護されるように、Cookieまたは本体パラメータとして一緒に送信する必要があります。

入力バリデーション

強力なバリデーションチェックが使用されるべきであり、バリデーションチェックが失敗した場合、リクエストは即座に拒否します。APIレスポンスでは、ユーザエクスペリエンスを改善するために、関連するエラーメッセージと正しい入力フォーマットの例を送信する必要があります。

入力バリデーションチェックの失敗はログに記録する必要があります。特に、クライアント側のコードがWebサービスを呼び出す可能性が高い場合に記録します。1秒間に何百回も入力バリデーションに失敗している人は、良からぬことを行っていると推測されます。悪用を防ぐには、APIを1時間または1日あたりのリクエスト数を制限するように考慮することが重要です。

URLバリデーション:　一般的な入力改ざん攻撃の一般的な名称には、強制ブラウジング、コマンド挿入、クロスサイトスクリプティング、バッファオーバーフロー、フォーマット文字列攻撃、SQLインジェクション、クッキーポイズニング、隠しフィールド操作などがあります。

Incoming Content-Typeをバリデーションする必要があります。Content-Typeヘッダーとコンテンツが同じタイプであることを常にチェックする必要があります。Content-Typeヘッダーがない、あるいは予期しないContent-Typeヘッダーの場合、サーバは406 Not Acceptable応答でコンテンツを拒否します。

Acceptヘッダーに許可されるタイプのいずれかが明確に含まれていない場合は、レスポンスタイプを検証し、要求(理想的には406 Not Acceptable応答を含めて)を拒否する必要があります。

XML入力バリデーション:　XML外部エンティティの攻撃、XML署名のラッピングなどから保護することが重要です（http://ws-attacks.org）。

URLに含まれる情報の暴露

ユーザ名、パスワード、セッショントークン、APIキーは、URLには含めないでください。これらはWebサーバログに記録されるため、容易に悪用されます(例:https://api.domain.com/user-management/users/{id}/someAction?apiKey=abcd 123456789)。

OAuth 2.0の使用

OAuth 2.0認可フレームワークを使用すると、リソース所有者とHTTPサービスとの間で承認のやり取りを調整すること、またはサードパーティ製アプリケーションにHTTPサービスへの制限付きアクセスを許可することによって、リソース所有者の代わりにサードパーティアプリケーションがそれ自身のためにアクセスを得るためにHTTPサービスへの制限付きアクセスを取得できるようになる、あるいはそれ自身のためにアクセスを得ることになります。資格情報として、短命の転送にXMLが大きく依存しているため、アクセストークンを使用することによって、資格情報が損なわれる危険性は軽減されます。

クリアなHTTPステータスコードの使用

HTTPはステータスコードを定義します。REST APIの設計では、成功のために200だけを使用したり、エラーのために400だけを使用したりしてはいけません。

許可された最新のXMLライブラリの排他的使用

ここ数年、セキュリティ研究者は、XML eXternal Entity Injection (XXE) 攻撃などのXML標準およびライブラリのセキュリティ脆弱性を数多く発見しています。SOAPは転送においてXMLに大きく依存しているため、SOAPメッセージの処理には、許可された最新のXMLライブラリのみを使用する必要があります。

4.2 SOAP推奨事項

SOAP (Simple Object Access Protocol)は、クライアントとサーバが情報を交換するためのXMLベースのプロトコルです。SOAPは、OASISおよびW3C推奨をサポートします。セキュアなトランスポートなど、前述の一般的なHTTP関連のセキュリティ機能の多くは、ここでも適用できます。SOAP固有のセキュリティに関するベストプラクティスを次に示します。

暗号化

暗号化は、メッセージの機密性を保証します。メッセージの内容は、意図した送信者と受信者だけが読むことができることを保証します。攻撃者は、たとえ暗号化されたメッセージを傍受したとしても、実際のメッセージの内容を読むことはできません。理想的にはすべてのヘッダーと本体を含むリクエスト/レスポンス全体を暗号化する必要があります。「WS-Security」 標準では、リクエスト/レスポンスの特定の部分の暗号化を許可するために「XML暗号化」標準を使用します。「XML暗号化」標準は、暗号化に複数の異なるアルゴリズムを定義しています。アルゴリズムの選択は、弱い暗号実装がアルゴリズムによって提供される数学的保証を非常に損なうので、セキュアな構成の重要な部分です。

暗号化/デジタル署名バリデーション

サーバが検証用の公開鍵を抽出するためにメッセージに依存する必要がないように、公開証明書の交換にはオフラインチャネルを使用することをお勧めします。信頼できる公開証明書はすべて、マシンの証明書マネージャにプリインストールされている必要があります。暗号化バリデーション操作には、これらの信頼できる公開証明書だけを使用してください。

SOAPAction

SOAPActionパラメータは、HTTPリクエストヘッダを利用して、リクエストのSOAP本文の関数呼び出しを表します。ファイアウォールは、特定のソースからの要求を受け入れるかどうかを決定するために、このパラメータを使用することがよくあります。この設計により、ファイアウォールはXML文書全体を解析することなく、決定を下すことができます。Webサービスの中には、このパラメータを無視して、リクエスト本文に含まれている関数を実行するだけのものもあります。この不一致により、攻撃者はHTTPリクエストヘッダーを変更して、すべてのファイアウォールの制限をバイパスし、Webサービスに許可されていない関数を実行させることができます。リクエスト本文の実際の関数呼び出しをチェックせずに、SOAPActionパラメータが参照する関数を実行する実装は他にもあります。これにより、攻撃者は、リクエスト本文が暗号化/デジタル署名で保護されていても、実行する機能を変更できます。SOAPActionパラメータが使用されていない場合は、この機能を無効にすることを強くお勧めします。

WS-Addressing

この標準では、SOAPメッセージのヘッダーにアドレス指定情報を追加できます。攻撃者はこれらの要素を悪用して、直接アクセスできないWebサービスやその他のサーバに到達する可能性があります。この機能は、どうしても必要な場合にのみ使用することをお勧めします。それ以外の場合は、この機能を明示的に無効にする必要があります。

5. APIの実践的な考慮事項

このセクションでは、重要なAPIエンティティを識別し、最も重要な3つのAPIの関係について説明します。そして、セキュアな暗号化を実現するために、これらの関係とテクノロジーがどのように管理されるかについて説明します。

5.1 APIのリレーションシップのプラクティス

5.1.1 インターフェースのトライアングル

次の図に示すように、APIの観点からは、相互作用する必要のある3つのエンティティがあります。

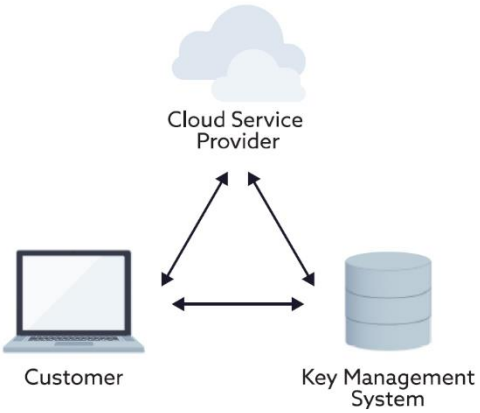

図6: インタフェース・トライアングル

5.1.2 リレーションシップ

利用者/クラウドプロバイダ

利用者とクラウドプロバイダのリレーションシップでは、利

用者はAPIを使用してクラウドプロバイダからのクラウドサービスを利用します。

要件：通常、豊富で幅広いAPIが、多くの機能をカバーします。APIのクライアントは通常、グラフィカルユーザインターフェイス（GUI）、コマンドラインインターフェイス（CLI）、またはスクリプトです。APIは通常、待ち時間やパフォーマンスよりも使いやすさと統合を優先しています。これらのAPIはクラウドプロバイダ固有のものですが、通常は利用者に同じプロバイダから提供される、他の非KMSクラウドリソースを管理する方法と同じ種類です。これらのAPIは通常、インターネット経由でアクセスできます。

利用者/ KMS

このリレーションシップでは、利用者はKMS管理APIを使用して鍵を管理します。鍵の管理には、アクセスコントロールポリシーの設定、セキュリティおよび暗号化ポリシーの適用、鍵のライフサイクルとローテーションの管理などが含まれます。

要件：　ここの要件は、利用者/クラウドプロバイダ・インターフェイスの要件と同じです。セキュリティ保護を強化するために、多くのKMSプロバイダは、これらのAPIの使用を、分離されたプライベートネットワーク内の少数の信頼できる機器にのみ制限しています。一部のKMSプロバイダは、これらのAPIをローカルでのみ公開することにより、物理アクセスを実施します。

クラウドプロバイダ/ KMS

クラウドプロバイダは、KMSへの限定されたインターフェースを使用して、必要に応じて利用者鍵を取り出し、または使用し、クラウド内の利用者のデータ資産を暗号化/復号します。

要件：　クラウドプロバイダがSLAを維持できるようにするには、これらのAPIの待ち時間を短くする必要があります。これらのインターフェースは、待ち時間と処理のオーバーヘッドを最小限に抑えるためにバイナリインタフェースにすることができます。

5.2 実践的なAPIマネジメント

このセクションでは、いくつかの既存の鍵管理APIの特性の概要を示し、Key Management Interoperability Protocol（KMIP）、Network-Attached Encryption - XML（NAE-XML）、およびKeySecureRESTに焦点を当てます。業界にはいくつかの標準化されたプロトコルがありますが、実践の上では、上記のインターフェースのいずれによっても、異なるクラウドサービスとKMSプロバイダ間で相互運用できることはまれです。

セキュリティ

説明
セキュリティは、APIの保護方法に関係しています。

KMIP
Mutual Transport Layer Security（TLS）認証（mTLS）

NAE-XML
Transport Layer Security（TLS）。mTLSが一般的ですが、クライアント証明書はオプションです。

KeySecure REST
TLS. mTLSが一般的ですが、クライアント証明書はオプションです。

認証

説明
このカテゴリでは、エンティティがAPIを認証する方法を取り上げ、サポートされているエンティティと資格情報のタイプについて説明します。認証が接続レベルであるか、要求ごとであるか、またはその両方であるかを調査して明確にします。

KMIP
KMIPの仕様は、接続レベルのセキュリティのためにクライアント証明書を要求します。そのクライアント証明書は有効である必要があります。接続が確立されると、各リクエストには追加のログイン資格情報を含むヘッダーが含まれるでしょう。したがって、KMSはTLS証明書から動作主体を認証することを選択することも、リクエストヘッダーに追加の資格情報を要求することもできます。KMIPリクエストヘッダーは、ユーザ名/パスワードの資格情報と、クライアントのアイデンティティの「証明」の両方をサポートします。この「アテステーション」メソッドは非常に柔軟でオープンです。チャレンジアンドレスポンス方式をサポートしたり、クライアントがJson Web Token（JWT）、アクセストークン、SAMLアサーションなどの任意のタイプのアサーションを送信したりするために使用できます。

NAE-XML
NAE-XMLは、独自仕様のXMLログインAPIです。クライアントはポートに接続し、ログインコマンドを発行する前にいくつかのコマンド（バージョン情報など）を発行できます。他のコマンドは、ログインコマンドが完了するまで発行できません。資格情報は通常TLS証明書ですが、ユーザ名/パスワードもサポートされています。認証コンテキストは、個々の要求ではなく、TCP接続に関連付けられています。X分間アクティビティがない場合、接続を閉じるためのタイムアウトがあります。クライアントエンティティは、KeySecure RESTと同様に、人間または機器のいずれかですが、NAE-XMLはそれらを区別しません。

KeySecure REST
OAUTH2.0でAPIを保護します。
資格情報は通常TLS証明書ですが、ユーザ名/パスワードもサポートされています。クライアントは、最初に1つまたは複数のトークンエンドポイントからアクセストークンを取得してから、後続のすべてのリクエストでベアラ方式を使用し、Authorizationヘッダーにおいてこのアクセストークンを用いる必要があります。認証は、ベアラトークンを検査することにより、要求ごとにKMSによって実行されます。トークンは、有効期限が5分のJWTです。クライアントには、有効期限の長い更新トークンも発行され、これを使用することで追加のアクセストークンを取得できます。クライアントエンティティは、ユーザまたは機器のいずれかです。ユーザと機器のプロビジョニング方法には違いがあり、通常、ユーザはユーザ名/パスワードの資格情報を持っており、機器は証明書を持っています。ただし、アクセストークンエンドポイントと実際のKMSリクエストは同じです。更新トークンは通常、X分間操作がないと期限切れになるように構成されています。

認可

説明
このカテゴリには、操作の認可方法が含まれます。

KMIP
対象外

NAE-XML
鍵には、鍵に完全にアクセスできる所有者がいます。鍵の所有者は、他のユーザのグループに鍵を使用するための制限付きアクセスを許可できます。

KeySecure REST
これは非常に柔軟なABACモデルです。箱から出してすぐに、鍵の所有者とグループのアクセス許可を使用して、NAE-XMLと同じように機能します。

ローテーション

説明
ローテーションは、鍵のローテーションまたはバージョン管理の方法、およびクライアントが現在のバージョンを知る方法に関係します。

KMIP
重要な「バージョン」についてもともとの概念を持っていません。各鍵は単独のオブジェクトです。ただし、鍵は他の鍵への「リンク」を持つことができ、各リンクにはセマンティック的な目的があります。KMIPで鍵をローテーションまたは「置換」するには、新しい鍵を生成し、前の鍵の「名前」を新しい鍵に移動し、2つの鍵間にリンクを作成します。1つのリンクタイプ「replacedObject」は新しい鍵から古いものを指し、リンクタイプ「replacementObject」は古いものから新しいものへのリンクを指します。鍵の最新バージョンのみを必要とする保護または署名操作を実行するクライアントは、その「名前」一つで鍵を参照します。これらの名前は古い鍵から新しい鍵に移動され、名前は常に最新バージョンの鍵を参照します。（復号および検証操作は、変わることのないIDによって鍵を参照します）。

NAE-XML
NAE-XMLは、バージョン管理された鍵とバージョン管理されていない鍵の両方をサポートします。バージョン管理されていない鍵のローテーションはサポートしていません。バージョン管理された鍵の場合には、鍵の複数のバージョンを含めることができます。ある意味で、「鍵」は実際には一連の鍵のバージョンを指します。鍵には名前があるため、名前で鍵を参照することで、常に最新バージョンの鍵を指すことができます。クライアントは、名前とバージョンで鍵を参照することもできます（復号/検証操作用）。

KeySecure REST
KeySecureRESTはNAE-XMLに似ています。鍵には、単一の「名前」とバージョンシーケンス番号があります。鍵をローテーションさせると、同じ「名前」と新しいバージョンシーケンス番号で新しい鍵が作成されます。鍵には、0～n個の「エイリアス」（名前と同様の意味を持つ）があります。エイリアスは古いバージョンから新しいバージョンに移動されます（KMIPと同様）。KMIPと同じ意味を持つ「リンク」もあり、「replacementObject」および「replacedObject」リンクは自動作成されます。他のすべての点では、鍵のバージョンは独自のプロパティと属性を持つ独立したオブジェクトです。鍵のローテーションでは、鍵の最新バージョンのクローンが作成され、クローンとして作成された新しいバージョンは（KMIPのように）独立したオブジェクトになります。最新バージョンの鍵が必要なクライアントは、「名前」またはそのエイリアスの1つで鍵を参照します。

識別子

説明
このカテゴリは、鍵の識別方法、鍵の識別子の数、鍵バージョンの識別方法、および各識別子のセマンティックに関係しています。

KMIP
KMIPは一意の識別子であり、完全なopaque文字列であり、グローバルに（テナントをまたいでも、再利用されることもなく、KMS全体で完全に）一意です。これは通常、KMS名、クライアントによって選択された0-nのopaque文字列識別子によって生成されます。ローテーションすると新しい鍵に移動します。もう1つのオプションはshortUniqueIdentifierです。KMIP 2.0の新機能で、これは短い識別子です。それ以外の点では、uniqueIdentifierと意味的に同じです。uniqueIdentifierのハッシュであることが推奨されています。

NAE-XML
鍵名、これは、クライアントによって選択された、またはサーバ上で生成されたイミュータブルな文字列識別子であり、鍵のすべてのバージョンを参照します。NAE-XMLは、単一のテナント/アカウント/ドメイン内でのみ一意です。鍵が削除された場合、名前は再利用される可能性があります。

KeySecure REST
ID、MUID、UUID、およびUniqueIdentifier：これらはすべてopaque型の文字列識別子であり、KMSによって生成されます。それらはすべて同じセマンティックを持っていますが、さまざまなクライアントのニーズに合わせるためにわずかに異なる形式を持っています。名前；名前は一意の文字列であり、NAE-XML名と同じセマンティックです。
エイリアス：　エイリアスは0～n個の一意の文字列識別子であり、クライアントによって割り当てられ、再利用できます。単一のドメイン内でのみ一意であり、ローテーション時に最新の鍵にコピーされます。これらは、意味的にはKMIPの「名前」属性に似ています。

属性/エクスポート可能性

説明
このカテゴリは、鍵がエクスポート可能であるか、エクスポート可能、かつ、エクスポートされたことがあるか、という概念がAPIにあるかどうかに関係します。

KMIP
KMIPでは、鍵のエクスポート可能性の特性を説明するためのいくつかの属性を持っています。

- その属性の1つはFreshです。これは、オブジェクトがエクスポートされたことがない（まだクライアントに提供されていない）場合に当てはまります。
- （仕様では「Get」操作で示され、非常によく似た「Export」操作については言及されていません。Exportがこのフラグをtrueからfalseに反転するトリガーも必要かどうかは明確ではありません。）
- もう1つの属性はExtractableです。これは、オブジェクトをエクスポートできるときにtrueです。
- 'Never Extractable'属性は、オブジェクトが抽出可能になったことがないときにtrueになります。
- 次の属性は「Sensitive」で、これは、鍵がラップされている場合にのみエクスポートできるときにtrueです。ラップされていないエクスポートは許可されません。
- 最後の属性は「AlwaysSensitive」です。これは、オブジェクトをAlwaysSensitiveとしたときにtrueになります。

NAE-XML
対象外

KeySecure REST
REST APIのセマンティックはKMIPと似ていますが、属性名が（「unexportable」、「neverExported」、「neverExportable」と）異なります。SensitiveおよびAlwaysSensitiveと同等の属性がロードマップにあります。

エクスポート

説明
このカテゴリには、鍵のエクスポート方法、鍵がラップされているかどうか、およびラップがオプションであるかどうかが含まれます。

KMIP
KMIPは、エクスポートの2つの操作「Get」と「Export」をサポートしています。それぞれをどのようなときに使用するのが適切かについては、ドキュメントでは明確ではありません。「get」はキーマテリアルのみを返し（属性は返しません）、マテリアルを使用したいクライアントをターゲットにしているようです。「Export」には、マテリアルとすべての属性が含まれ、あるKMSから別のKMSへの鍵の移行を対象としているようです。どちらのコマンドも、オプションでラッピングパラメータを渡すことができます。ラッピングパラメータには、ラッピングの形式、ラッピングおよびHMACに使用する鍵のUniqueIdentifiersを定義します（KMSにはすでに別の鍵が存在している必要があります）。

KeySecure REST
REST APIは、鍵とすべての属性をエクスポートするPOST /keys/:id/export endpointをサポートします。エクスポートパラメータには、対称鍵（KMSにすでに存在する）の「名前」か、RSA公開鍵を明示的に含むラッピング命令が含まれる場合があります。

ライフサイクル

説明
このカテゴリは、鍵の状態のライフサイクルに関係しています。

KMIP
KMIP鍵のライフサイクルについては、https://docs.oasis-open.org/kmip/kmip-spec/v2.0/cs01/kmip-spec-v2.0-cs01.html#_Toc6497510に説明があります。

ライフサイクルの状態は、アクティブ前（Pre-Active）、アクティブ（Active）、非アクティブ（Deactivated）、侵害された（Compromised）、破壊された（Destroyed）、危殆化され破壊された（DestroyedCompromised）です。

ライフサイクルは一方向にのみ流れます。以前の状態に戻ることはできません。一時保留の概念はありません。KMIPにも「アーカイブ」の概念がありますが、これは単純なバイナリ状態であり、ライフサイクルに直交しています。

KMIPには、プロセス開始日と保護停止日もあります。これらの日付は、鍵を最初に復号/検証に使用できる時期（プロセス開始）と、鍵を最後に暗号化/署名に使用できる時期（保護停止日）を制約します。これらの日付には、ライフサイクルの状態に関連する値に制約があります（たとえば、保護停止日は非アクティブ化した日より前であっても、非アクティブ化した日と同じでもかまいません）。

鍵の状態をクライアントが直接変更することはできません。ライフサイクル内のある状態から別の状態への遷移は、別の属性で日付を渡した結果（たとえば、アクティブ化した日）、または状態の変化を引き起こす何らかの操作（たとえば、取り消し）の結果として発生します。

NAE-XML
NAE-XMLは、アクティブ（Active）、制限付き（Restricted）、廃止（Retired）、および除去（Wiped）の4つのライフサイクル状態をサポートします。アクティブとは、すべての操作が可能であることを意味します。制限付きとは、復号または検証操作のみが許可されることを意味します。廃止とは、操作が許可されていないことを意味します。除去とは、鍵マテリアルが破壊されることを意味します（廃止の不可逆バージョンのようなものです）。アクティブ、制限付き、および廃止は、他の状態に自由に変更できます。これらの状態は、バージョン管理された鍵の個々のバージョンにのみ適用されます。（バージョン管理されていない鍵にはライフサイクルがないようです）。

クライアントが状態属性を直接設定すると、アクティブ、制限付き、および廃止の間の状態変化が発生します。除去は、鍵を破壊したときに副次的に遷移します。

KeySecure REST
これはKMIPと同じです。

属性

説明
このカテゴリは、APIが鍵の属性を処理する方法に焦点を当てています。

KMIP
KMIPは、多数の標準属性のセットをサポートしています。鍵はこれらすべての属性を定義する必要はありませんが、仕様によって定義され、厳密に制約されている属性は定義する必要があります。KMIPクライアントおよびサーバは、オプションで任意の数の追加のカスタム属性を定義できます。属性は、TTLVのシステム型（KMIPが使用するネイティブバイナリエンコード形式）で定義する必要があります。

NAE-XML
鍵には、常に存在する組み込み属性の固定セットがあります。ほとんどは読み取り専用です（サーバによってのみ管理されます）。クライアントは、有限の数のカスタム属性を追加できます。

KeySecure REST
これはKMIPに似ています。鍵には、クライアントが自由に設定できるJSONドキュメントの「メタ」属性があります。クライアントは、任意の属性やネストされたJSONドキュメントを「メタ」に追加できます。

6. まとめ

クラウドサービスの採用は、技術的なアジリティ、弾力性のあるスケール、市場投入までのスピード、設備投資の削減など、組織に多くの利点をもたらします。クラウドサービスはまた、データのプライバシーとセキュリティの分野で特に課題があります。クラウドサービスを採用する場合、組織はデータプライバシーの期待と義務を非常に明確に理解することが重要です[12]。技術用語（実装の詳細とサイバーセキュリティの概念）を考える前に、組織はまず「何を」という質問に明確に答えることができなければなりません。私たちは、どのような結果を達成することを期待していますか？」 この質問は、次のような幅広い回答につながる可能性があります:

- クラウドサービスを使用することによるデータプライバシーのリスクは、完全に文書化され、明確に理解され、組織の法務およびリスク管理チームによって受け入れられる必要があります。
- データは、クラウドサービスの採用前と同じくらいセキュアにします。
- プライバシーの義務はクラウドベンダーの認証によってサポートされており、ベンダーの認証に変更があった場合は、サービスを正常に終了するための適切な通知が提供されます。
- データは、クラウドサービスの人員に（読み取り可能/理解可能な形式で）公開されるリスクはありません。
- クラウドベンダーがデータを読み取り可能/理解可能な形式で第三者に公開することはできません。

上記の例はいずれも、技術、管理策、暗号化などに言及していないことに注意してください。上記の各ステートメントは、期待された結果を満たすためのさまざまな技術的手段を可能にし、実装には暗号化、鍵、および鍵管理システムを含みます。データのプライバシーとセキュリティに関する組織の義務と目標を理解することは、暗号化の使用を含む、技術的なソリューションと実装を選択する前段階となります。暗号化を実装することで、多くの人的エネルギーと時間が浪費され、その結果、期待されるデータのプライバシーやセキュリティが提供されず、困惑、データの損失、監査の失敗、および仕事の不安定さにつながります。明確なビジネスとデータのプライバシーとセキュリティの期待を確立することで、これらの望まれない結果を回避できます。

[12] これらの結果は、クラウドサービスを使用する際に、他の考慮事項（弾力性のあるスケール、高性能、地理的フットプリント）とバランスを取る必要もあります。

一度暗号化が確立されると、暗号化鍵の生成、配布、処理、および破棄の変遷過程を理解することが重要です。暗号化は非常に多くのテクノロジーアーキテクチャの一部であるため、鍵と鍵管理システムの使用は、多くの組織内で非常に複雑なエコシステムになる傾向があります。
次に、組織は、組織のデータ保護、コンプライアンス、およびプライバシーの期待に応えることを目的とした、特定のビジネスおよび技術的なユースケース要件に適した鍵管理システムの形式を決定するための支援を必要としています。

組織がクラウドサービスで暗号化を使用することを選択したら、データのプライバシーとセキュリティについて期待される結果を再検討する必要があります。暗号化鍵の使用の変遷過程は、達成されるデータプライバシーの結果において基本的な役割を果たします。これは、セクション2.1.1に記載されています。いわゆるBYOK（Bring Your Own Key）や同様のモデルをクラウドサービスで使用したBYOKの意味は、通常期待される結果が得られないことを示しています。これらのいわゆるBYOKモデルを使用しようとしているほとんどの組織は、クラウドサービスプロバイダが利用者のデータを裁判所や法執行機関などの第三者に引き渡すことを強制できないことを期待しています。ただし、いわゆるBYOKモデルのほとんどのベンダーにおける実装では、クラウドサービスがデータ暗号化鍵を使用しているため、必要に応じてエクスポート用に暗号化されていないデータを生成できるので、実際にはその結果を防げません。BYOK（およびその変化形）という用語は、鍵の使用ではなく鍵の生成に誤って焦点をあてています。データの復号は、キーの生成方法と全く関係しません。クラウドサービスがデータを復号する鍵を使用できる限り、テクノロジーは復号鍵（またはその親または親の親など）がどこでどのように生成されたかをまったく気にしません。したがって、データのプライバシーとセキュリティに対する期待を評価するには、鍵をどのように使用できるか、つまり、どのエンティティがどのような状況で、どのような目的で、どのような制約の下で鍵を使用できるかを調べる必要があります。これらの情報はすべて、使用中の暗号化ソリューションをサポートする鍵管理システムの運用モデルとパラメータを決定するために必要です。

暗号化鍵とそれをサポートする鍵管理システムのもう1つの側面は、脅威とその相対的な重要性の分析です。すべてのサイバーセキュリティの専門家が知っているように、セキュリティはタダではなく、リソース支出と引き起こされる問題の影響との最適なバランスを達成するには、脅威に優先順位を付ける必要があります。たとえば、ある暗号化鍵はデータの破棄にのみ役立ちます。この場合、一般的なディスク/ボリューム暗号化技術において、鍵はデータの可用性を制御しますが、セキュリティを制御しません。鍵自体はディスクの内容を復号するために必要ですが、データの機密性を間接的に提供しているだけです（通常、オペレーティングシステムで使用する暗号化鍵を「ロック解除」するためにPINまたはヒューマンファクターアーティファクトが使用されます）。このタイプの暗号化システムでは、鍵の盗難は事実上問題ではないため、優先すべき脅威ではありません。ただし、鍵を破棄すると、ディスクの内容全体が完全に失われる可能性があります。したがって、対処する必要がある最も優先度の高い（そしておそらく唯一の）脅威は、適切な認可とサポート手順がなければ、鍵（およびエスクローされたコピー）の破棄が非常に困難であることを保証することです。このタイプのダイナミクスは、サポートする鍵管理システム、したがってKMSの選択に明らかな影響を及ぼします。

CSAは、組織が、計画中、進行中、または確立されたクラウドの採用に関係なく、ビジネスおよびプライバシー/セキュリティの成果を達成するための潜在的な使用について評価できる観点から、既存の鍵管理システムの機能を文書化することを推奨します。これにより、クラウドサービスでKMSを使用する可能性を評価するための基盤が確立されます。同じ評価をクラウドKMSサービスで実行できます。

要約すると、ハードウェアセキュリティモジュール（HSM）やその他の暗号化ツールを含む鍵管理システムは、コンプライアンス、データ制御要件、およびビジネスニーズへのサービスにおけるセキュリティ上の利点を満たすために使用されます。この文書では、クラウドサービスでの4つのKMSパターンの使用について説明しました。それらは、クラウドネイティブ鍵管理システム（Cloud Native Key Management System）、外部鍵作成（External Key Origination）、外部鍵管理システムを使用したクラウドサービス、およびマルチクラウド鍵管理システムです。それらが定義され、それらの属性と課題が列挙され、それらの使用に関する推奨事項が示されました。次に、鍵管理制御について2つのセクションで説明しました。コンプライアンス、データ制御、およびセキュリティを確保するために必要な制御について、制御環境および制御そのものについて説明しました。この後、文書は、最も使用されている2つのAPIアーキテクチャの使用に関する推奨事項を確認および作成し、実用的なAPIの考慮事項で締めくくりました。

KMSのパターン、管理、APIアーキテクチャ、および実践的な考慮事項は、組織がデータのプライバシーとセキュリティの期待に応える主要な管理目標を達成するためのガイドとして使用できます。

References

[NISTIR 7956]	NISTIR 7956. Cryptographic Key Management Issues and Challenges in Cloud Services, September 2013. https://csrc.nist.gov/publications/detail/nistir/7956/final
[NIST SP 800-57]	NIST SP 800-57 Part 1 Rev. 4. Recommendation for Key Management, Part 1: General, January 2016. https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final
[NIST SP 800 57] Part 2	NIST SP 800-57 Part 2 Rev. 1. Recommendation for Key Management: Part 2: Best Practices for Key Management Organizations, May 2019. https://csrc.nist.gov/publications/detail/sp/800-57-part-2/rev-1/final
[NIST SP 800-57] Part 3	NIST SP 800-57 Part 3 Rev. 1. Recommendation for Key Management, Part 3: Application-Specific Key Management Guidance, January 2015. https://csrc.nist.gov/publications/detail/sp/800-57-part-3/rev-1/final
[NIST SP 800-57] Part 1	NIST SP 800-57 Part 1 Rev. 5. Recommendation for Key Management: Part 1 – General. May 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5
[NIST SP 800-52]	NIST SP 800-52 Rev. 2, Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, August 2019. https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/final
[NIST SP 800-133]	NIST SP 800-133 Rev. 1. Recommendation for Cryptographic Key Generation, July 2019. https://csrc.nist.gov/publications/detail/sp/800-133/rev-1/final
[OpenAPI Initiative]	OpenAPI Initiative (OAI) v3.0. https://github.com/OAI/OpenAPI-Specification
[OWASP]	OWASP Key Management Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html
[CSA's Security Guidance]	Security Guidance for Critical Areas of Focus in Cloud Computing v4.0. https://cloudsecurityalliance.org/artifacts/security-guidance-v4/
[OASIS Key Mgmt Interoperability Control]	OASIS Key Management Interoperability Protocol (KMIP). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip

Appendix A: Acronyms

Selected acronyms and abbreviations used in this paper are defined below.

AI	Artificial Intelligence
API	Application Programming Interface
BYOK	Bring Your Own Key
CC	Common Criteria
CISO	Chief Information Security Officer
CLI	Command Line Interface
CMK	Customer Master Keys
CRL	Certificate Revocation List
CKMS	Crypto Key Management System
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
CYOK	Control Your Own Key
DEK	Data Encryption Keys
EKM	External Key Manager
FIPS	Federal Information Processing Standard
GRC	Governance Risk and Compliance
GUI	Graphical User Interface
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HYOK	Hold Your Own Key
IAM/IdAM	Identity and Access Management
IT	Information Technology
JSON	JavaScript Object Notation
JWT	JSON Web Tokens
KMS	Key Management System

KMIP	Key Management Interoperability Protocol
KYOK	Keep Your Own Key
ML	Machine Learning
MCKMS	Multi-Cloud Key Management System
MTLS	Mutual Transport Layer Security (TLS) Authentication
NAE-XML	Network-Attached Encryption (NAE) -eXtensible Markup Language (XML) (NAE-XML)
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency Report
OAUTH	Open Authorization
OCSP	Online Certificate Status Protocol
OS	Operating System
PaaS	Platform as a Service
PCI DSS	Payment Card Industry Data Security Standard
PKI	Public Key Infrastructure
RBAC	Role-Based Access Control
RDS	Relational Database Service
REST	Representational State Transfer
SAML	Security Assertion Markup Language
SaaS	Software as a Service
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TPM	Trusted Platform Module
TLS	Transport Layer Security
mTLS	Mutual Transport Layer Security
TTLV	Tag, Type, Length, Value
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

UOID	Unique Object Identifier
WS-Security	Web Services Security
XML	Extensible Markup Language
XXE	XML External Entity

Appendix B: Big 5 CSP KMS Comparison

Presented below is an overview of four key public cloud service providers' KMS Offerings.

Alibaba Cloud	
Alibaba has one managed service that is relevant to this document.	
Alibaba Cloud KMS	Alibaba Cloud Key Management Service (KMS) provides secure and compliant key management and cryptography services to help customers/users/organizations encrypt and protect sensitive data assets. KMS is integrated with a wide range of Alibaba Cloud services to allow customers/users/organizations to encrypt data across the cloud and to control its distributed environment. KMS provides key usage logs via ActionTrail, supports custom key rotation, and provides HSMs that have passed FIPS 140-2 Level 3 or other relevant validation, to help customers/users/organizations meet their regulatory and compliance needs.
AWS	
AWS offers two managed services that are relevant to this document.	
KMS	AWS Key Management Service (AWS KMS) is a managed service that makes it easy for customers to create and control the encryption keys used to encrypt the data. It uses hardware security modules that have been validated under FIPS 140-2 or are in the process of being validated. AWS KMS allows users to manage customer master keys (CMK) and use them for various cryptographic operations. A CMK is a logical representation of a master key. CMKs are used for managing data encryption keys (DEK) that are used for encrypting data, including large amounts of data and other data encryption keys.
CloudHSM	AWS CloudHSM is a cloud-based hardware security module (HSM) that manages encryption keys on the AWS Cloud using FIPS 140-2 Level 3 validated HSMs. This service essentially provides a CloudHSM Cluster. Clusters can contain multiple HSM instances spread across multiple Availability Zones in a cloud region. HSM instances in a cluster are automatically synchronized and load-balanced. Customers receive dedicated, single-tenant access to each HSM instance in their cluster. Each HSM instance appears as a network resource in their virtual cloud network. AWS provides HSM cluster management APIs.
Azure	
Azure has two managed services that are relevant to this document.	

Azure Dedicated HSM	Azure Dedicated HSM is an Azure service that provides cryptographic key storage in Azure using dedicated HSMs. It is the ideal solution for customers who require FIPS 140-2 Level 3-validated Thales Luna devices and complete and exclusive control of the HSM appliance. It is also suited for applications which need Common Criteria EAL 4+, NITES, or Brazil ITE and needs cryptography other than RSA and ECC. HSM devices are deployed globally across several Azure regions. They can be easily provisioned as a pair of devices and configured for high availability. HSM devices can also be provisioned across regions to assure against regional-level failover. Microsoft delivers the Dedicated HSM service by using the SafeNet Luna Network HSM 7 (Model A790) appliance from Gemalto.
Azure Key Vault	Azure Key Vault helps customers with secrets management, key management and certificate management. Secrets and keys are safeguarded by Azure, using industry-standard algorithms, key lengths, and hardware security modules (HSMs). The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. Authentication is done via Azure Active Directory. Authorization may be done via role-based access control (RBAC) or key vault access policy. RBAC is used when users deal with the management of the vaults, and key vault access policy is used when they are attempting to access data stored in a vault. This service has native integration with a large number of other Azure services.
Google Cloud	
Google has two managed services that are relevant to this document.	
GCP Cloud KMS	GCP Cloud KMS is a scalable cloud-hosted key management service that lets customers manage cryptographic keys for their cloud services the same way they do on premises. It supports AES256, RSA 2048, RSA 3072, RSA 4096, EC P256, and EC P384 cryptographic keys. Cloud KMS is integrated with Cloud IAM and Cloud Audit Logging so that customers can manage permissions on individual keys and monitor how these are used. Cloud Key Management Service stores cryptographic keys in a hierarchical structure designed for useful and elegant access control management. Cloud KMS has a built-in 24-hour delay for key material destruction in order to prevent accidental or malicious data loss. Cloud KMS can also generate HSM-backed keys with GCP Cloud HSM.

GCP Cloud EKM	GCP Cloud External Key Manager (EKM) is in early access and allows to encrypt data with encryption keys that are stored and managed in a third-party key management system. Cloud EKM lets customers/users encrypt data in BigQuery and Compute Engine with encryption keys that are stored and managed in a third-party key management system that is deployed outside Google's infrastructure. External Key Manager allows customers/users/organizations to maintain separation between data at rest and encryption keys while still leveraging the power of cloud for compute and analytics.
IBM	
IBM has two managed services that are relevant to this document.	
IBM Cloud HSM	IBM Cloud Hardware Security Module (HSM) Luna 7.0 from Thales protects the cryptographic infrastructure of some of the most security- conscious organizations in the world by securely managing, processing and storing cryptographic keys inside a tamper-resistant, tamper- evident device. With IBM Cloud HSM 7.0, customers/users/organizations can solve complex security, compliance, data sovereignty and control challenges associated with migrating and running workloads on the cloud.
IBM Key Protect	IBM Key Protect is a cloud-based security service that provides life cycle management for encryption keys that are used in IBM Cloud services and non-IBM cloud applications. Key Protect provides roots of trust, backed by a FIPS-140-2 Level 3 hardware security module (HSM). It supports importing the customer's root of trust encryption keys into the service. It can generate, store, and manage customer keys with a secure, application-friendly, cloud-based key management solution for encryption keys. When keys are deleted, they can never be recovered, and any data that is encrypted under those keys cannot be recovered. All programmatic interfaces are secured by TLS and mutual authentication.

Appendix C: Key State Lifecycle Layered View

Minimal standards:

This document proposes that the CSPs follow each of the states in the key state lifecycle. Recognizing that not all application developers will find it useful to move through the key states. This document proposes a two-tiered key state transition. Implementing five Tier 1 state models will be mandatory for all keys, while Tier 2 sub-states/ reasons will be optional.

Tier 1	Tier 2
1. Generated	
2. Suspended: includes sub-states	
	a. Distribution
	b. Replacement
	c. Rotation
3. Activated	
4. Deactivated: includes reasons such as	
	a. Compromised
	b. Revoked
	c. Expired
	d. Archived
5. Destroyed	

The advantage of this view is that a developer who wants to do a quick key state check can simply work with these five values (and state transitions) and still meet the key lifecycle management criteria for logging/ auditing.

Also, a particular CSP may want to limit their actions based on these Level 1 states, e.g. if the key is in a deactivated state to only allow decrypt/ verify/ MAC verify/ unwrap operations. However, if an auditor wants to explore the reasons for the deactivated state, there is an option for them to delve into Level 2 details about why a key is currently deactivated.