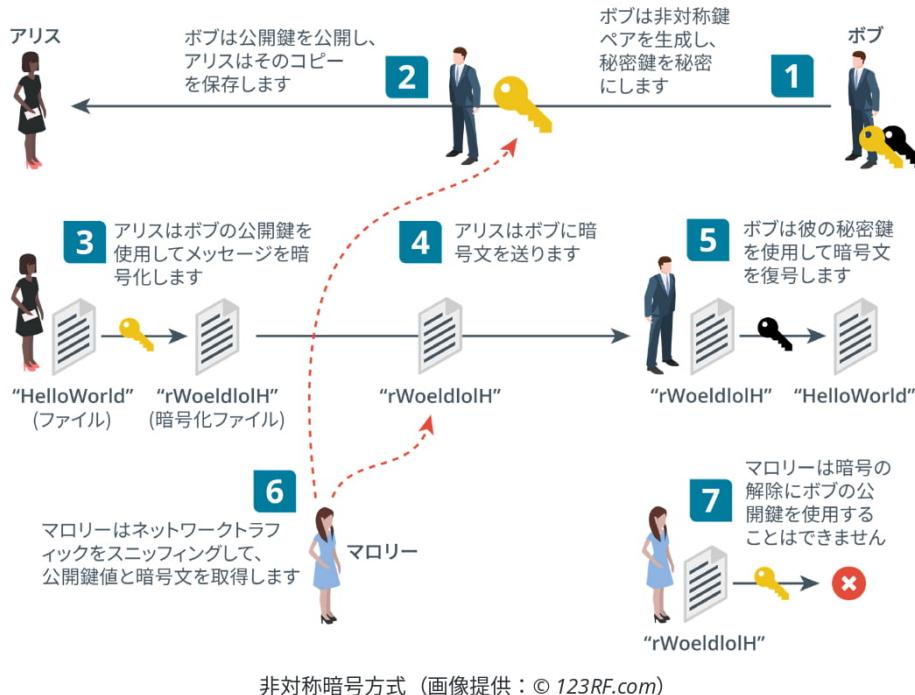


4. アリスはボブに暗号文を送ります。
5. ボブはこのメッセージを受け取って、自分の秘密鍵を使って復号することができます。
6. マロリーがスヌーピングしていた場合、メッセージと公開鍵の両方を傍受することができます。
7. しかし、マロリーは、メッセージの復号に公開鍵を使用することができないため、システムは安全に保たれます。



非対称暗号方式 (画像提供: © 123RF.com)

アイデンティティを証明する場合には、非対称暗号方式を使用することができます。他の誰かが秘密鍵の保持者に成りますことはできません。非対称暗号方式には、対称暗号方式と比べてコンピューターのオーバーヘッドをかなり要するという欠点があります。メッセージは、鍵の長さを超えることはできません。大容量データをディスク上で暗号化したりネットワーク転送する場合、非対称暗号方式は効果的ではありません。

したがって、非対称暗号方式は、認証および否認防止、鍵の合意および交換に最も多く使用されます。鍵の合意/交換とは、一括暗号化に使用する秘密の共通鍵を誰にも発見されることなく決定することを言います。

公開鍵（パブリックキー）暗号化アルゴリズム

非対称暗号方式はしばしば公開鍵暗号方式と呼ばれます。公開鍵暗号方式を使用する商品の多くは、**RSAアルゴリズム**に基づきます。1977年、Ron Rivest、Adi Shamir、および Leonard Adlemanらが、RSA暗号を公開しました(rsa.com)。RSAアルゴリズムは、鍵のペアを導出し、暗号化および復号の演算を実行する数学的特性を提供します。このタイプのアルゴリズムは、公開鍵を使うと容易に実行できますが、秘密鍵を知らないと反転することが難しいため、**トラップドア関数**と呼ばれます。

楕円曲線暗号方式(ECC)は、公開鍵暗号方式で使用可能な別のタイプのトラップドア関数です。RSAアルゴリズムよりもECCアルゴリズムが実質的に優れている点は、鍵の長さにかかわらず、暗号や計算を破る「ショートカット」を誰も知らないからです。したがって、鍵の長さが256ビットのECCは、鍵の長さが2048ビットのRSAにほぼ匹敵します。



RSAキーのペアのセキュリティは、非常に大きい整数の素因数発見の困難さに依存しています（冪剰余）。ECCは、離散対数の問題に依存しています。Cloudflare（クラウドフレア）は、相違点に関する優れた概要を生成しました(blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography)。

レビュー アク ティビティ：

暗号文

次の質問にお答えください。

1. 単純な暗号システムで秘密にしなければならないのは、暗号、暗号文、鍵のうちどれでしょうか？
2. 暗号ハッシュ化が一方で、ダイジェストを元に戻すことができない場合、なぜハッシュ化が有用なセキュリティ技術となるのですか？
3. 公開鍵暗号方式が保証するのはどのセキュリティ特性ですか？
4. 公開鍵/秘密鍵のペアの特性を挙げてください。

トピック5B

暗号動作モードを要約する



対象試験範囲

2.8 暗号化コンセプトの基本を要約することができる

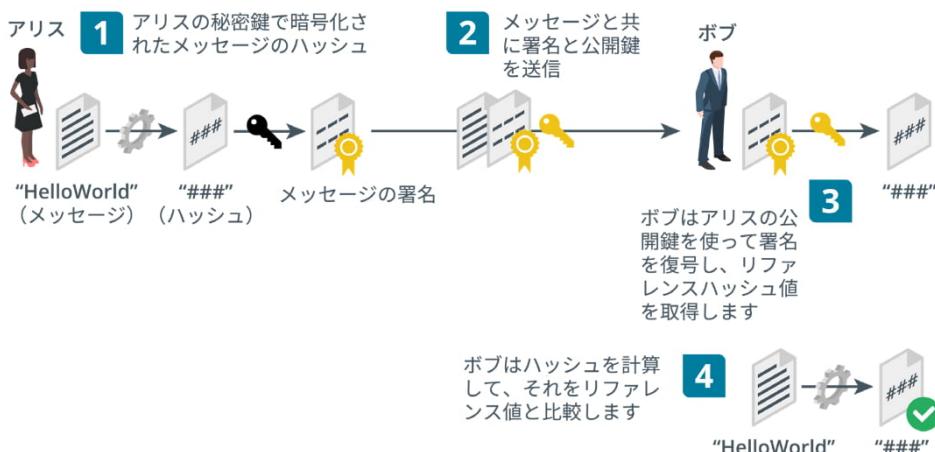
動作モードとは、機密性または完全性などのセキュリティの目標を達成するために、製品内で暗号アルゴリズムを使用することです。動作モードを概説できるようになると、デジタル署名や転送暗号化などセキュリティ管理の実施やサポートにおいて役立ちます。

デジタル署名

公開鍵暗号方式では、送信者は他の誰にもできない方法でメッセージを暗号化する秘密鍵を管理しているため、送信者を認証することができます。しかし、公開鍵暗号方式は、非常に短いメッセージにしか使用できません。ハッシュ化は、入力から一意のチェックサムを計算することで完全性を証明します。これら2つの暗号機能を組み合わせると、送信者の認証とメッセージの完全性の証明を行うことができます。この使用法を**デジタル署名**と呼びます。RSA暗号を使用してデジタル署名を作成する場合、次のプロセスを使用します。

1. 送信者（アリス）が、SHA256のように予め合意したハッシュアルゴリズムを使ってメッセージのダイジェストを作成し、自分の秘密鍵を使ってダイジェストを暗号化します。
2. アリスは、オリジナルメッセージにデジタル署名を添付し、署名とメッセージをボブに送信します。
3. ボブは、アリスの公開鍵を使って署名を復号し、元のハッシュを生成します。
4. 次に、ボブは（アリスと同じアルゴリズムを使って）受け取った文書のチェックサムを計算し、アリスのハッシュと比較します。

2つのハッシュが一致した場合、データが送信中に改竄されることではなく、アリスのアイデンティティは保証されます。データが変更されるか、悪意のあるユーザー（マロリー）がメッセージを変更したり異なる秘密鍵を使用した場合、ダイジェストが一致することはありません。



デジタル署名を使ったメッセージ認証と完全性。(画像提供: © 123RF.com)



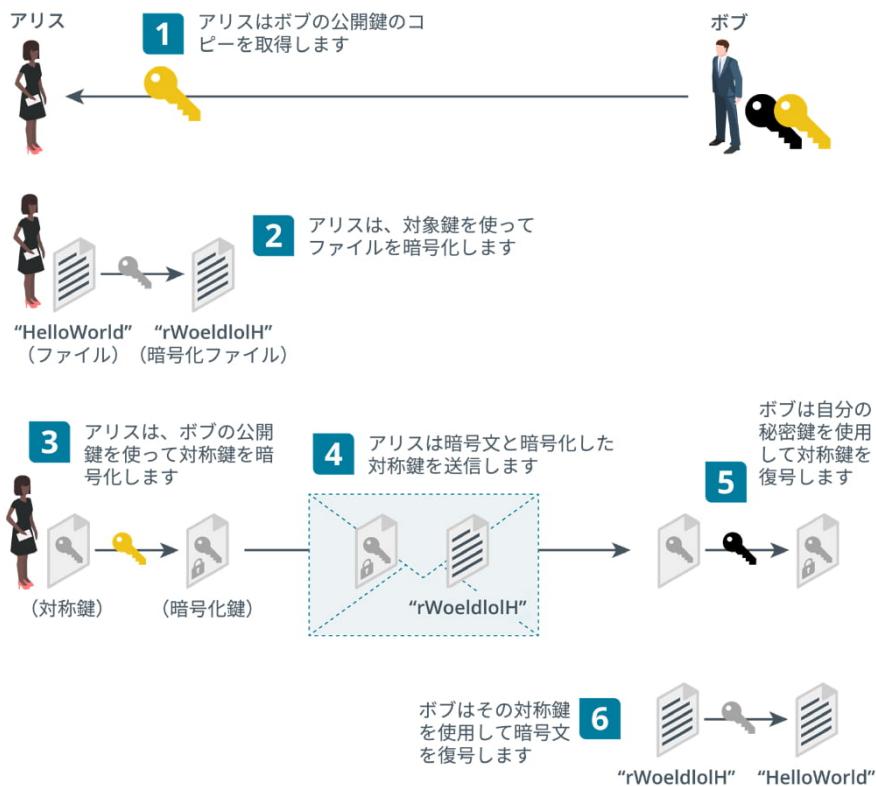
デジタル署名は、秘密鍵を使用して暗号化されるハッシュ値であることを忘れてはいけません。暗号化しない場合、第三者が容易にファイルやハッシュを傍受し、ファイルを修正し、新しいハッシュを計算し、修正済みのファイルとハッシュを受信者に送ります。受信者は、公開鍵が本当にAliceによって発行されたものであることを確認する手段を持っていなくてはならないことを認識していかなければなりません。また、デジタル署名はメッセージの機密性を提供しないことをご留意ください。

DSA(Digital Signature Algorithm)は、同様の目標を達成するためのわずかに異なる形式です。DSAは、RSA暗号ではなく楕円曲線暗号方式(ECC)を使用します。

デジタルエンベロープと鍵交換

対称暗号方式（共通鍵暗号方式）は、大量のデータを暗号化および復号する唯一の実用的な手段ですが（一括暗号方式）、安全に共通鍵を配布するのは困難です。公開鍵暗号方式を使用すると、鍵の配布は簡単ですが、その効果が発揮されるのは、データが少量の場合だけです。そのため、デジタルエンベロープやハイブリッド暗号方式と呼ばれる鍵交換システムで、同一製品内で両者が使用されます。デジタルエンベロープでは、送信者と受信者が、共通暗号方式で使用する鍵を、公開鍵暗号方式を使って安全に交換することができます。

1. アリスはボブの公開鍵のコピーを取得します。
2. アリスは、AESのような共通鍵暗号方式を使ってメッセージを暗号化します。このコンテキストにおいて、共通鍵は、セッション鍵とも呼ばれます。
3. アリスは、ボブの公開鍵（パブリックキー）を使ってセッション鍵を暗号化します。
4. アリスは、暗号化されたセッション鍵をデジタルエンベロープ内の暗号文メッセージに添付してボブに送ります。
5. ボブは、自分の秘密鍵を使ってセッション鍵を復号します。
6. ボブは、セッション鍵を使って暗号文メッセージを復号します。



このプロセスで留意すべきは、受信者の公開鍵が共通鍵の暗号化で使用され、受信者の秘密鍵（プライベートキー）が共通鍵の復号で使用される点です。デジタルエンベロープ全体の有効性は、メッセージ認証符号を使用すると証明できます。



これらの実装において、秘密鍵の安全を確保し、権限のあるユーザーのみが使用できるようにすることが重要です。

デジタル証明書

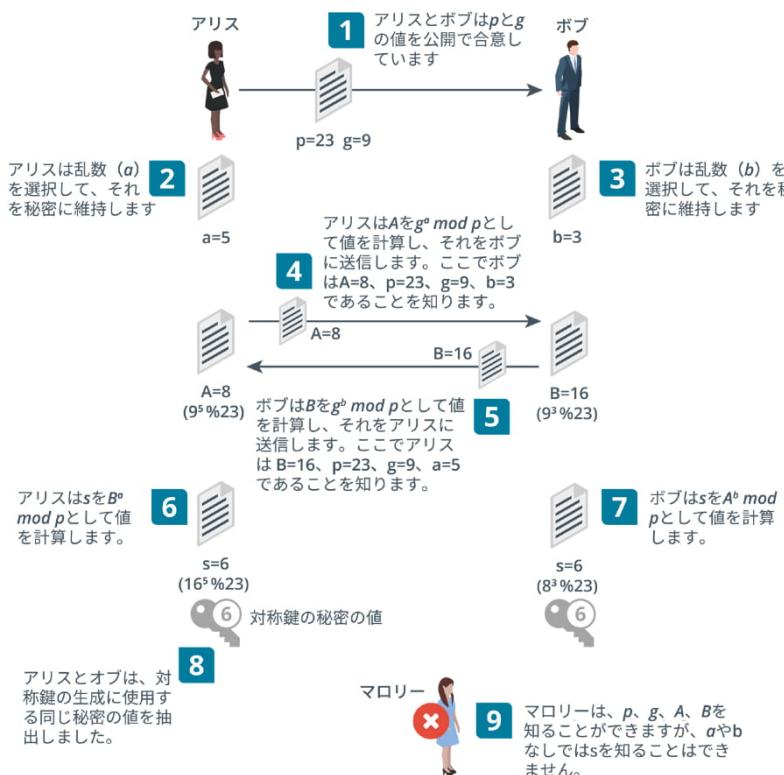
公開鍵/秘密鍵のペアを使用する場合、対象者は、自分の公開鍵を他人が自由に使用できるようにします。これによって、メッセージの受信者はデジタル署名を読むことができます。同様に、デジタルエンベロープでメッセージを暗号化する場合は、受信者の公開鍵を使用します。これは、意図した受信者以外はメッセージを読むことができなくなります。

次に、公開鍵を発行する人物またはサーバーのアイデンティティを信頼するにはどうしたらよいかという疑問が生じます。解決策の一つは、認証局(CA)と呼ばれる第三者が、証明書を発行することによって公開鍵の所有者を認証することです。この証明書はCAによって署名されています。受信者もCAを信頼すれば、証明書の中に含まれる公開鍵を信頼することができます。証明書の発行および確認プロセスは、公開鍵インフラストラクチャ (PKI: Public Key Infrastructure) と呼ばれます。

Perfect Forward Secrecy

デジタルエンベロープを使用する場合、選択した共通鍵暗号方式で使用する一括暗号化のための共通鍵の交換または合意をしなくてはなりません。初期の電子エンベロープの実装では、サーバーとクライアントは、やりとりのぞき見から保護するために、サーバーのRSAキーペアを使用して、共通鍵を交換していました。この鍵交換モデルで、セッションのデータが記録され、後にサーバーの秘密鍵が侵害された場合、セッション鍵の復号や機密セッションデータの復元にこの秘密鍵が使われる可能性があります。

PFS (Perfect Forward Secrecy)を使用することで、RSAキー交換によって生じるリスクが緩和されます。PFSは**DH (Diffie-Hellman key agreement : ディフィー・ヘルマン鍵共有)**を使用して、サーバーの秘密鍵を使用せずに一時的（エフェメラル）なセッション鍵を作成します。DHを使用すると、アリスとボブは、トランプドA関数に関連付けられたいつかの値に合意するだけで、共有の秘密を導出することができます。合意プロセスにおいて、2人はいくつかの値を共有しますが、他の値は秘密にしておきます。マロリーは、公に交換される値から秘密を知ることはできません([en.wikipedia.org/wiki/Diffie%20%93Hellman_key_exchange](https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange))。サーバーから送られた値の認証は、デジタル署名を使って証明されます。



ディフィー・ヘルマンを使用して秘密の値を導出すると、公開されたチャネルを介して安全に共有の共通鍵を生成できます。(画像提供：© 123RF.com)

一時的（エフェメラル）なセッション鍵を使用すると、将来的にサーバーが侵害されても、記録されたデータが攻撃を受けることはありません。また、脅威アクターがあるセッションの鍵を取得しても、その他のセッションの秘密は守られます。これによって暗号解読量が格段に増えるため、脅威アクターは「会話」全体を復元しなくてはならなくなります。

DHE (またはEDH) (Diffie-Hellman Ephemeral mode : ディフィー・ヘルマン鍵共有エフェメラルモード) またはECDHE (Elliptic Curve Diffie-Hellman Ephemeral mode : 楕円曲線ディフィー・ヘルマン鍵共有エフェメラルモード) アルゴリズムのいずれかを使用すると、PFSを実装することができます。PFSを使用する場合、サーバーとクライアントは、相互にサポートしている暗号スイートを使用することを交渉しなくてはなりません。



2014年、OpenSSLのいくつかのバージョンによって、サーバーメモリコンテンツから情報(64キロバイト)が露出してしまう可能性があるという、ハートブリードバグが発見されました(heartbleed.com)。これには、秘密通鍵が含まれるため、サーバーと通信することによって侵害される可能性があります。このバグは約2年にわたって存在していました。ここでは、PFSの価値を説明していますが、皮肉なことに、多くのサーバーが、PFSをサポートするためにバグのあるOpenSSLバージョンにアップデートしている可能性があります。

暗号スイートと動作モード

TLS (Transport Layer Security)のようなプロトコルでは、サーバーのアイデンティティの認証やサーバーとクライアント間の通信の暗号化の両方の要件を満たす必要があります。これらは、サーバーとクライアントで個別の暗号化製品や暗号アルゴリズムの実装によって行われることになります。サポートされている暗号アルゴリズムの組み合わせは、**暗号イースト**と呼ばれます。TLSハンドシェイクの一環として、サーバーとクライアントはお互いに、互換性のある暗号スイートをネゴシエートします。

これまで暗号スイートの2つの部分について確認しました：

- 署名アルゴリズムは、サーバーの公開鍵のアイデンティティについて主張し、認証を容易にするために使用されます。
- 鍵交換/合意アルゴリズムは、同じ一括暗号化のための対称鍵を導出するためにクライアントとサーバーが使用します。

暗号スイートの最後の部分は、一括暗号化のための暗号アルゴリズムを決定します。AESが対称暗号方式として選択される場合、ネットワークデータのストリームをサポートする**動作モード**で使われる必要があります。

CBC (Cipher Block Chaining)モード

CBC (Cipher Block Chaining)モードは、鍵が任意の平文から一意の暗号文を確実に生成できるような初期化ベクトル(IV)を最初の平文ブロックに適用します。次に、最初の暗号文ブロックの出力は、**XOR**演算を使用して、次の平文ブロックと連結されます。このプロセスは、同じ暗号文を生成する平文ブロックがないことを（再度）確認するために、ブロックの「連鎖」が続くかぎり繰り返されます。CBCは、暗号するデータがブロックサイズのちょうど2倍になるように、パディングを使用する必要があります。



XORとは、入力が1および0の時のみ1を出力する論理演算です。

カウンターモード

カウンターモードを使用すると、AESアルゴリズムはストリーム暗号として機能します。カウンターモードは、キーストリームを生成するための鍵にIVとインクリメントカウンタ値を適用します。次に、キーストリームは、平文内のデータにXORされます。各ブロックは、個別かつ平行に処理することができるため、パフォーマンスが向上します。また、カウンターモードではパディングを使用する必要がありません。最後のブロックで使用されていないスペースは単に破棄されます。

認証された動作モード

対称アルゴリズムは、メッセージに完全性または認証を提供しません。基本のCBCとカウンターモードは認証されません。中間者は、共通鍵が無いと直接復号することはできませんが、暗号文は、選択暗号文攻撃と呼ばれる暗号方式をやぶるために挿入または変更される任意のデータに対して脆弱です。

認証済み暗号化方式

MAC (message authentication code : メッセージ認証符号) は、メッセージ出力と共有の共通鍵の組み合わせをハッシュ化することによって認証および完全性のメカニズムを提供します。受信者は、自分の共通鍵のコピーを使って同じプロセスを実行すれば、データを照合できます。このタイプの認証済み暗号化方式は、「HMAC-SHAを使用するAES CBC」のような個別の機能として、暗号スイートで指定されます。残念ながら、AES CBCにこのタイプの認証済みモードを実装すると、パディングオラクル攻撃と呼ばれる暗号化攻撃タイプに対して脆弱です (docs.microsoft.com/en-us/dotnet/standard/security/vulnerabilities-cbc-mode)。

Authenticated Encryption with Additional Data (AEAD)

パディングのメカニズムが原因でCBCの脆弱性が生じるため、ストリーム暗号またはカウンターモードが非常に好まれます。こういった場合は、AEAD (Authenticated Encryption with Additional Data)モードを使用します。AEAD方式では、関連データを使用すると、受信者は、ペイロードが異なる通信ストリームから再生されていないことを確認するためのメッセージヘッダを使用することができます。

AEADモードは、AES-GCMやAES-CCMのように、ハイフンでつながれた1つの機能名で識別されます。ChaCha20-Poly1305ストリーム暗号は、AESの代替として開発されました。

レビュー アク ティビティ： 暗号動作モード

次の質問にお答えください。

1. 電子的にメッセージに署名するプロセスは何と呼ばれますか？
2. デジタルエンベロープでは、どの鍵でセッション鍵を暗号化しますか？
3. 次の記述は正しいですか、誤りですか？PFSDA (Perfect forward secrecy)は、サーバーの秘密鍵の脆弱性が原因で、以前サーバーに送られたトラフィックのコピーが解読のリスクにさらされないようにします。
4. DH (Diffie-Hellman key agreement : ディフィー・ヘルマン鍵共有) がPFS (Perfect forward secrecy)をサポートする理由は何ですか？
5. 最高のセキュリティを提供するのはどのタイプの一括暗号化モードの動作ですか？

トピック5C

暗号のユースケースと脆弱性を要約する



対象試験範囲

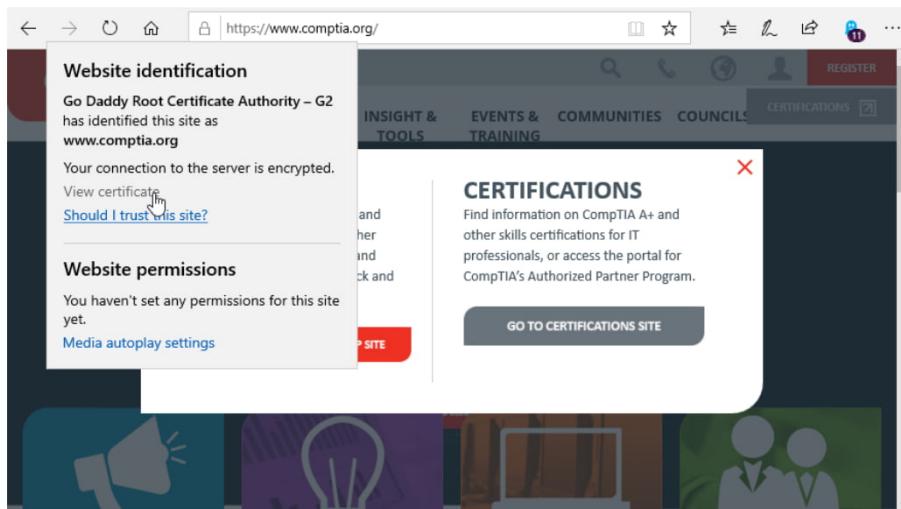
- 1.2 与えられたシナリオに基づいて、可能性あるインジケーターを分析して攻撃のタイプを特定することができます
- 2.8 暗号化コンセプトの基本を要約することができます

対称暗号アルゴリズム、非対称暗号アルゴリズム、ハッシュ関数などは複数存在します。これらの暗号アルゴリズムの特徴により、デバイスごとの制約への適合性を検討することができます。たとえば、バッテリー駆動のデバイスにおける制約などです。いくつかの暗号アルゴリズム自体や、製品に実装された暗号アルゴリズムには、使用が不安定になる脆弱性が存在します。目的にかなった管理を展開するには、こういったユースケースや脆弱性を説明できることが重要です。

認証および否認防止をサポートする暗号方式

1つのハッシュ関数、共通鍵暗号方式、または公開鍵暗号方式は、**暗号プリミティブ**と呼ばれます。完全な暗号システムまたは製品では、暗号スイート内など、複数の暗号プリミティブが使用される可能性があります。対称/非対称/ハッシュのそれぞれタイプの特性や、各タイプの特定の暗号アルゴリズムの特性が、さまざまな状況や目的での使用を制限します。

特定の方法でメッセージを暗号化できる場合、メッセージの受信者は、やり取りしている相手を知っています（すなわち、送信者が認証されているため）。これは、暗号化によって識別、認証、およびアクセス制御システムの基本を形成できることを意味します。



暗号化によって、主体は自身を識別および認証できるようになります。主体とは、人またはWebサーバーのようなコンピューターである可能性があります。

否認防止は、識別および認証と関連づけられます。これは、送信者はメッセージの送信を否定できないというコンセプトです。送信者だけが知っている方法でメッセージが暗号化されている場合、その暗号は送信者が構成したことになります。

認証および否認防止は、メッセージを暗号化できない受信者に依存します。そうでなければ受信者は送信者になりますことができてしまいます。このことは、認証および否認防止をサポートするには、受信者は、暗号プロセスを利用して認証および整合性データを復号できるが、暗号化できないことを意味します。このユースケースは、非対称暗号方式および公開鍵/秘密鍵のペアによってサポートされます。

鍵のペアを使用する場合、ユーザーまたはサーバーが関連する鍵を生成します。秘密鍵は、安全に保管され、他の人が使用できないようにアカウントパスワードによって保護されます。重要なのは、決まったユーザーまたはサーバーのみが秘密鍵を使用できるようにすることです。公開鍵は通常、クライアントまたは通信相手にデジタル証明書の形式で与えられます。

ユーザーやサーバーが認証を必要とする場合、秘密鍵を使って合意したハッシュデータを暗号化し、デジタル署名としてクライアントに送信します。クライアントは公開鍵を用いて署名を復号し、同じハッシュ値を抽出できます。

機密性をサポートする暗号方式

この暗号技術方式を使用すると、セキュアな媒体を介してメッセージを保存または転送する必要がなくなります。暗号文が盗まれたり傍受されても、脅威アクターは盗んだものを理解したり変更することができないため問題ありません。暗号のこの使用方法で、機密性の目的を満たすことができます。このユースケースの場合、データ量が大きいとアルゴリズムを効果的に暗号化できなければ、単純に非対称暗号方式と秘密鍵/公開鍵のペアを使用することはできません。例えば、RSA非対称暗号方式は、鍵の長さ（バイト）は、最大メッセージサイズから11バイト引いたものです。鍵の長さが2048ビットの場合、最大メッセージサイズは245バイトです：(2048/8)-11。ネットワークトラフィックのディスクまたはストリームのコンテンツを暗号化するためにこのタイプのアルゴリズムを使用すると、計算処理上のオーバーヘッドが極めて大きくなります。

したがって、一括データ暗号化では、AESのような対称暗号方式を使用します。対称暗号方式は、データファイルとネットワークトラフィックのストリームを迅速に暗号化および復号することができます。共通鍵暗号方式は、データファイルとネットワークトラフィックのストリームを迅速に暗号化および復号することができます。問題なのは、対称鍵（共通鍵）を安全に配布するのが困難であることです。鍵の値を知る人が多ければ多いほど、機密性が下がります。脅威アクターが鍵を取得するリスクも加速度的に高くなります。幸い、対称鍵（共通鍵）の長さは、たったの128ビットまたは256ビットであるため、公開鍵を使用すれば、容易に暗号化できます。そのため、ほとんどの暗号化システムでは、対称暗号方式と非対称暗号方式の両方を使用します。

機密性をサポートする暗号は、保存中データ（ファイル暗号化）および転送中データ（転送暗号化）に使用されます。：

- ファイル暗号化—ユーザーは、非対称暗号アルゴリズムの鍵のペアを割り当てられます。秘密鍵は、セキュアストレージ(TPM, Trusted Platform Module)に書き込まれ、ユーザーが自分のアカウントの認証を行ったときだけ使用できます。公開鍵は、ランダムに生成されたAES暗号キーを暗号化するために使用します。ユーザーがファイルの暗号化または復号を試みると、暗号化演算または復号演算に使用できるように秘密鍵を使ってAES暗号キーが復号されます。
- 転送暗号化—デジタルエンベロープまたはPFS (Perfect Forward Secrecy)を使用します。HTTPSの場合、Webサーバーに鍵のペアが割り当てられ、秘密鍵を安全に保存します。公開鍵は、デジタル証明書によってクライアントに配布されます。クライアントとサーバーは、セッション鍵として使用するために、鍵のペアを使って一つまたはそれ以上のAES暗号キーを交換または合意します。

完全性と復元力をサポートする暗号方式

完全性はハッシュ化アルゴリズムによって証明され、それによって、2人の関係者が同一チェックサムを抽出することができ、メッセージまたはデータが改竄されていないことがわかります。基本的なハッシュ関数を共有シークレットとともに使用して、**MAC (message authentication code : メッセージ認証符号)** こともできます。これにより中間者によるチェックサムの改ざんを防ぐことができます。

個々のメッセージレベルの完全性を提供するだけでなく、暗号は、高度な耐性を持った制御システムの設計にも使用できます。制御システムは、センサー、ワークステーション、サーバーなど複数の部品と、複雑な動作ロジックを持つものです。システムのほんの一部が危険にさらされたことで、システム全体が侵害されるのを防ぐことができるようなシステムには耐性があります。暗号は、制御システムで配信されるメッセージの認証および完全性をか確実にすることによって、この目標をサポートできます。

コンピューターコードにとっても完全性と耐性は課題です。脅威アクターが管理者特権を保有している場合、正当なコードをマルウェアとして機能するように変更することができます。開発者は、難読化を使用して改竄をより困難にすることができます。**難読化**は、メッセージを理解しにくくする技術です。難読化されたソースコードは、コンピューターのコンパイルやコードの実行に影響しない方法で書き換えられますが、コードを読む人にとって、コードの動作の理解が難しくなっています。

暗号方式は、メッセージを難読化するのに非常に効果的な方法ですが、ソースコードであまりにも効果的なので、コンピューターでさえコードを理解（実行）できない可能性があります。どこかのポイントで、コードを実行するために復号しなければなりません。復号に使用する鍵は、通常ソースコードにバンドルされている必要がありますが、これは強い暗号方式ではなく隠ぺいによるセキュリティに依存していることを意味します。ホワイトボックス暗号方式として知られる、コードの機能を保持したまま埋め込み鍵を保護しようとする試みは、すべて破られてきました。この問題を克服するために現在市場で入手可能な解決策はありませんが、研究のテーマとして注目されています。

暗号の性能限界

暗号アルゴリズム間の違いにより、リソースに制約のある環境では有効なものとそうでないものがある。主な性能要素は次のとおりです：

- 速度 — 対称暗号方式とハッシュ関数の場合に、速度は一秒間に処理できるデータ量のことです。公開鍵暗号方式は、一秒間の操作量によって測定されます。大量のデータが処理されたときに最も影響を受けるのはスピードです。
- 時間/レイテンシー — ユースケースによっては、結果ができるまでにかかる時間がデータレートより重視されます。例えば、セキュアなプロトコルがハンドシェイクフェーズで暗号アルゴリズムに依存している場合、ハンドシェイクが終了するまでデータを転送することはできません。数ミリ秒で測定できるこのレイテンシーは、重要な性能になります。
- サイズ — 暗号アルゴリズムのセキュリティは、鍵の長さとの関連性が強く、鍵が長ければ長いほど安全性が高まります。アルゴリズムの比較に鍵の長さは使用できないのでご注意ください。例えば、256ビットのECCキーは、2048ビットのRSAキーより強力です。鍵の長さが長くなればなるほど各演算の計算処理上のオーバーヘッドが高くなり、その結果、スピードが遅くなり、レイテンシーが高くなります。
- 計算処理上のオーバーヘッド — 鍵の長さの選択に加え、暗号アルゴリズムはそれぞれ一意の性能特性を有しています。他の暗号アルゴリズムより多くのCPUやメモリーリソースを必要とするものもあり、リソース制約のある環境での使用にあまり適さないものもあります。

特定のユースケースのために製品または個々の暗号アルゴリズムを選択する場合、入手可能な最高のセキュリティに対する要求と実装可能なりソースの間で妥協を余儀なくされます。

- 低電力デバイス — 長い鍵を使用して設定された技術または暗号アルゴリズムには、より多くの処理サイクルおよびメモリースペースが必要となる場合があります。この場合、装置の速度が遅くなり、より多くの電力を消費します。したがって、アルゴリズムや鍵強度によっては、特にバッテリー電源で動作する小型デバイスや組み込みシステムに使用には不向きな場合もあります。読み取り装置からのみ電力供給を受け、記憶容量がかなり制限されている非接触型スマートカードは、サポートされる最大鍵の長さに影響を及ぼします。
- 低レイテンシーの使用 — これは、プロトコルハンドシェイクの設定時間に影響を及ぼす可能性があります。ハンドシェイクが長いとユーザーにとって遅延の原因となり、アプリケーションによってはタイムアウトが発生する可能性があります。同様に、暗号技術が音声または映像などリアルタイムに敏感なチャネルで展開される場合、送信機と受信機の両方の処理オーバーヘッドは、信号の品質に影響を与えないように十分低くなければなりません。

暗号方式のセキュリティの限界

リソース制約によって、セキュリティと性能の間で妥協を迫られる可能性がありますが、あまりにも妥協しすぎるとよくありません。

エントロピーと弱い鍵

エントロピーは無秩序さの尺度です。平文は、メッセージを人間の言葉、プログラミング言語、またはデータ構造で表すため、通常低いエントロピーを示します。平文は、人間、コンピュータープロセッサ、またはデータベースが理解できるように保たれている必要があります。無秩序な暗号文を作成することが強力な暗号化アルゴリズムの要件の一つです。別の言い方をすれば、暗号文は高いレベルのエントロピーを示す必要があります。もし平文の秩序の要素が残っている場合、暗号文を解読しやすくなる原因となり、アルゴリズムが弱いことを示します。

重要なのは、AESのようなアルゴリズムが強いと考えられているからといって、この暗号アルゴリズムを実装すればプログラミングライブラリも強くなるということを認識しておくことです。この実装が脆弱性を持っている可能性もあります。このタイプのプログラミングコードの状態を監視し、迅速にアップデートすることが極めて重要です。脆弱性が見つかったら、弱いバージョンで発行した鍵を交換し、データを再暗号化する必要があります。

弱い鍵は、あるべきものより低いエントロピーの暗号文を生成します。キースペースが弱い鍵を含んでいる場合、その暗号アルゴリズムを使用する技術は、これらの鍵の使用を防ぐ必要があります。DESと**RC4**は、弱い鍵を持っていることで知られるアルゴリズムの例です。ソフトウェアに暗号アルゴリズムを実装する段階で、弱い鍵を使用してしまう可能性もあります。2008年に発見されたDebian LinuxのOpenSSL server softwareの疑似乱数生成器で発生したバグがその一例です(wiki.debian.org/SSLkeys)。弱い乱数生成器は、多くの公開鍵が共通の因子を持つことにつながります。暗号解読者は、こういった要因があるかどうかを調査し、鍵全体を非常に簡単に導出することができます。そのため、暗号実装における**真性乱数生成器(TRNG)**または**疑似乱数生成器(PRNG)**モジュールは、その強度を左右する重要な要素です。

```

administrator@LAMP16:~$ gpg --gen-key
gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)

Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
  0 = key does not expire
  <n> = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0) 20
Key expires at Fri 30 Aug 2019 06:27:41 AM PDT
Is this correct? (y/N) y

You need a user ID to identify your key: the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
  "Heinrich Heine (Der Dichter) <heinrich@duesseldorf.de>"

Real name: gtslearning
Email address: support@gtslearning
Comment:
You selected this USER-ID:
  "gtslearning <support@gtslearning>"

Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? o
You need a Passphrase to protect your secret key.

gpg: gpg-agent is not available in this session
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation: this gives the random number
generator a better chance to gain enough entropy.

Not enough random bytes available. Please do some other work to give
the OS a chance to collect more entropy! (Need 237 more bytes)

```

疑似乱数生成器(PRNG)はGPGを使った鍵生成中に動作します。
この方法では、ユーザーがマウスやキーボードを使用するとエントロピーが増加します。



真性乱数生成器および疑似乱数生成器の詳細については、random.orgをご参照ください。

予測可能性と再利用

予測可能性とは、暗号文のエントロピーをより低くして暗号解読者の解読を容易にする暗号操作または特定の鍵の値の弱さです。同じセッション内で同じ鍵を再利用すると、このタイプの弱さが生じます。RC4ストリーム暗号といいくつかのCBCモードは、予測可能性を示すため、他の暗号モードほど安全ではありません。多くの場合、こういった暗号アルゴリズムと組み合わせて、追加のランダム値または疑似ランダム値を使用する必要があります。

- ノンス** — ノンスの主な特徴は、同一の範囲内（つまり同一鍵の値）では再利用できないことです（「一度だけ使われる数」）。この数はランダム値、疑似ランダム値、あるいはカウンター値である場合があります。
- 初期化ベクトル(IV)** — IVの主な特徴は、ランダム（または疑似ランダム）であることです。IVについては、（ノンスと同様に）再利用できないという要件がある場合もありますが、これは主要な特徴ではありません。
- ソルト** — これもまた、乱数、疑似乱数、または文字列です。用語ソルトは特に、ハッシュ化パスワード値と組み合わせて使用されます。

寿命および暗号化攻撃

弱い暗号スイートおよび実装の使用は、組織にとって重大な脆弱性を示す可能性があります。これは、保存中または処理中のデータが安全でない可能性を意味します。このデータを使用すると、悪意のある脅威アクターが、なりすましにより偽装されたデジタル証明書を作成し、甚大な風評被害を引き起こすこともできます。

ある種の暗号アルゴリズムの弱点により、安全でないものがあり、短中期的に安全でなくなる可能性があると考えられるものがあります。ある意味、寿命は、人々が暗号アルゴリズムに対して持つ信頼度の高さを測るもので、暗号解読は、脆弱性を検出するために、暗号システム上で行われます。しかし、特定の暗号アルゴリズムや、研究条件下での暗号アルゴリズムの実装で発見された弱さがアルゴリズムの非推奨につながったとしても、必ずしもシステムがすぐに実質的に脆弱になるわけではありません。

RC4およびDES/3DESは、既に非推奨になっています。RSAは、その有用性が終わりに近づいていると見られていますが、ECCおよびその他のアルゴリズムは、優れたセキュリティおよび性能特性を提供しています(thesistore.com/blog/is-it-still-safe-to-use-rsa-encryption)。MD5とSHA-1には既知の弱点がありますが、互換性が最優先の懸念事項である場合、必ずしも安全でないとは言えません。

別の意味では、寿命は、データをどれくらい長く安全に保つかという考慮事項でもあります。暗号文は、ある時点で暴露されると仮定すると、暗号文は、暗号解読にどれくらいの期間耐える必要があるでしょう？例えば、NSAが操作するラップトップが盗まれたとします。盗んだ人は、現在の計算リソースで暗号を破れると思っていませんが、暗号化メカニズムはどれくらいの期間データを保護する必要があるでしょうか？暗号解読の進化によって、5年、10年、または20年以内にリスクに晒された場合、安全なアルゴリズムの選択は可能でしょうか？セキュリティ、コスト、および相互運用性の間で常に妥協を迫られます。悪意のある数学的攻撃は実行が難しいため、最新の実証済みの技術や標準に対して成功する可能性はほとんどありません。価値の無くなったアルゴリズムを使用していても、慌てる必要はありませんが、影響を受けたシステムを注意深く監視し、より優れた技術をできるだけ迅速に実用化するための計画が必要です。

中間者攻撃とダウングレード攻撃

2人の関係者間のやり取りをキャプチャすることに依存する攻撃もあります。この場合、暗号システムがやぶられることはありましたが、使用される方法で脆弱性が悪用されます。中間者攻撃(MITM)の対象は普通、公開鍵暗号方式です。

1. マロリーは、アリスとボブのチャネルを盗み聞きし、アリスがボブの公開鍵を要求するのを待ちます。
2. マロリーは、通信を傍受し、ボブの公開鍵を手元に残して自分の公開鍵をアリスに送ります。
3. アリスはマロリーの鍵を使ってメッセージを暗号化し、それをボブに送ります。
4. ボブはこのメッセージを受け取り、自分の秘密鍵を使って復号します。
5. 次にマロリーは、ボブの公開鍵を使ってメッセージを暗号化（あるいは変更）しますが、アリスとボブは自分たちの通信が侵害された事実に気づくことはありません。

この攻撃は、鍵と証明書を関連付けるなど、公開鍵のセキュアな認証を使用することで防ぐことができます。アリスがマロリーの公開鍵を確実に拒絶することが必要です。

ダウングレード攻撃は、サーバーがより弱い暗号アルゴリズムと短い鍵の長さを持つ、より低い仕様プロトコルを使用するよう要求することにより、中間者攻撃を容易に実行できるようにするために使用します。例えば、クライアントは、サーバーが好みがちなTLS 1.3ではなくSSLを使用するよう要求します。こうすることで、マロリーは簡単に、アリスが信頼する認証機関の署名を偽造して、アリスがマロリーの公開鍵を信頼するよう仕向けることができます。

キーストレッ칭とソルト化

暗号システムにおいて、パスワードのようなユーザが生成したデータを利用する場合、エントロピーが問題となります。ユーザーは、覚えやすいという理由で、低いエントロピーのパスワードを選択しがちです。この場合、いくつかの技術で補完を試みます。

キーストレッ칭

キーストレッ칭は、ユーザーのパスワードから生成した鍵を取得し、より長くよりランダムな鍵に繰り返し変換します。最初の鍵に対して数千回のハッシュ化が行われる場合もあります。脅威アクターにとってこの手順を再現するのは難しくないため、実際これによって鍵がより強力になるわけではありませんが、脅威アクターは可能性のある鍵の値ごとにすべての処理を再度行わなくてはならないため、攻撃速度が低下します。キーストレッチは、特定のソフトウェアライブラリを用いて、パスワード作成時にハッシュ化して保存することで行うことができます。**PBKDF2 (Password-Based Key Derivation Function 2)**は特に、WPA (Wi-Fi Protected Access) の一部として、上記の目的のために広範囲で使用されています。

ソルト化

ハッシュとして保存されたパスワードは、ブルートフォース攻撃および辞書攻撃に対して脆弱です。ハッシュ関数は一方向であるため、パスワードハッシュは復号できません。しかし、脅威アクターは、ネットワークトラフィックまたはパスワードファイルから取得したパスワードハッシュに一致するものを見つけるためにハッシュを生成することができます。ブルートフォース攻撃は単に、文字列、数字、記号の考えられるあらゆる組み合わせを実行します。辞書攻撃は、一般的な単語およびフレーズから成るハッシュを作成します。

どちらの攻撃も、ハッシュの作成時にソルト値を追加すると速度を落とすことができ、計算式は以下のとおりです。

$$(\text{salt} + \text{password}) * \text{SHA} = \text{hash}$$

ハッシュを確認するシステムがソルト値を知っている必要があるため、ソルトを秘密にしておくことはできません。これは単純に、脅威アクターは事前に計算されたハッシュの表を使用できないことを意味します。ハッシュ値は、パスワードごとに、特定のソルト値で再コンパイルする必要があります。

衝突と誕生日攻撃

誕生日攻撃は、ハッシュ関数の衝突を悪用することを目的とするブルートフォース攻撃の一種です。衝突とは、1つの関数が2つの異なる平文に対して同じハッシュ値を作成することです。このタイプの攻撃は、デジタル署名を偽造目的で使用することができます。この攻撃は次のように動作します：

1. 脅威アクターは、同じハッシュ値を生成する、悪意のある文書と無害の文書を作成します。脅威アクターは、ターゲットから署名入手するために、無害の文書を送信します。
2. 次に、脅威アクターは、無害の文書から悪意のある文書に署名を移し、ターゲットの署名を偽造します。

無害の文書と同じハッシュを出力する悪意のある文書を作成することが、この攻撃の手口です。誕生日のパラドックスとは、実行に必要な計算時間が予想より短いことを意味します。誕生日のパラドックスは、グループ中の2人が同じ誕生日である確率が50%の場合、そのグループには何人の人がいるかを質問します。答えは23ですが、パラドックスに気づかない人は、しばしば180 (365/2)くらいと答えます。要するに、自分と同じ誕生日の人がいる確率は小さくても、任意の2人の誕生日が同じである確率は、グループの人数が増えれば増えるほど高くなるということです。 $1 - (365 * (365-1) * (365 - 2) ... * (365 - (N-1)) / 365^N$

このパラドックスを悪用する場合、脅威アクターは小さな変更（パンクチュエーション、余分なスペースなど）を加えた悪意のある文書と無害の文書を複数作成します。ハッシュの長さと、疑われない程度の変更の限界にもよりますが、脅威アクターが十分なバリエーションを生成できれば、ハッシュ出力の一致確率は50%より高くなる可能性があります。

つまり、誕生日攻撃から保護するには、暗号化アルゴリズムが衝突回避を実証する必要があるということです（すなわち、別個の入力が同じ出力を生成する機会を減らすこと）。誕生日パラドックスを悪用する場合、脅威アクターはたいてい文書/メッセージの両方を操作できる必要があります。これは、選ばれたプレフィックス衝突攻撃(Chosen-prefix collision attack)と呼ばれます。<https://sha-mbles.github.io>。誕生日のパラドックス方式は、MD5関数の衝突を悪用して、信頼できるルートチェーンの認証局が署名したかのように見せかける偽のデジタル証明書の作成に成功した方法です。<http://trailofbits.files.wordpress.com/2012/06/flame-md5.pdf>。

レビュー アク ティビティ：

暗号化のユースケースと脆弱性

次の質問にお答えください。

1. 次の記述は正しいですか、誤りですか？暗号技術とは、物事を秘密にしておくことであるため、否認防止システムの基準として使用することはできません。
2. 暗号技術は、高回復力をどのようにサポートできますか？
3. どのタイプのシステムにとって、高いレイテンシーを示す暗号スイートが問題になりますか？
4. エントロピーと暗号化機能にはどんな関連性がありますか？
5. あなたの会社は、暗号化したパスワードを保存するデータベースを必要とするソフトウェアを作成しています。ブルートフォース攻撃に対してより抵抗力のあるパスワードデータベースを作るにはどんなセキュリティ制御が必要ですか？

トピック5D

その他の暗号化技術を要約する



対象試験範囲

2.8暗号化コンセプトの基本を要約することができる

暗号プロセスの開発および使用環境は常に進化しています。セキュリティの専門家であるならば、これらのトレンドを常に把握しておくことが重要です。そうすることで、より優れたセキュリティ制御を実装する新たな機会や、技術の進歩による既存の制御に対する脅威を認識できるようになります。

量子およびポスト量子

量子は、特定のタスクで従来のコンピュータの性能に十分に対抗できるよう量子メカニズムの特性を使用するコンピューターを指します。

計算

量子コンピューターは、*qubit*（量子ビット）と呼ばれる単位で処理を実行します。*qubit*は、0、1、または0か1の可能性のある重ね合わせ(*superposition*)と呼ばれる不確定状態に設定できます。この可能性は均衡状態か、いずれかに片寄っています。量子計算の能力は、*qubit*がもつれ状態であることで発揮されます。*qubit*の値が読めると、1または0で設定され、その他のもつれた*qubit*すべてが同時に設定されます。このアーキテクチャの強みは、一度の操作で*qubit*として表される大量の状態変数を利用できることで、クラシカルコンピューターのCPUの場合は、メモリのビットごとに読み取り、実行、書き込みサイクルを実行しなくてはなりません。これが特定のタスクの解決に非常に適しています。そのうちの2つが、RSA暗号をサポートする因数分解問題およびECC暗号をサポートする離散型アルゴリズム問題です。

通信

量子計算によって、現在の暗号文の強度がリスクにさらされる可能性がある一方で、よりセキュアな暗号システムをサポートする可能性もあります。もつれ、重ね合わせ、折りたたみの特性は、セキュアな鍵合意を可能にする、改ざんの有無が一目瞭然になる通信システムの設計に適しています。

ポスト量子

ポスト量子とは、有用な処理を行うことができる量子コンピュータが実現したときに期待されるコンピューティングの状態のことです。現在、*qubit*およびもつれの物理的特性が、量子コンピューターのスケールアップを非常に難しくしています。本稿執筆時点では、最も強力な量子コンピュータは約50*qubit*を搭載しています。量子コンピューターで有益なアプリケーションを実行する場合は、約100万*qubit*を必要とします。