ZHQM ZMGM ZMFM – G JULIUS CAESAR  
KXJEY UREBE ZWEHE WRYTU HEYFS KREHE GOYFI WTTTU OLKSY CAJPO BOTEI ZONTX BYBWT GONEY CUZWR GDSON SXBOU YWRHE BAAHY USEDQ – JOHN F KENNEDY  
  
暗号化は、セキュリティエンジニアリングと数学が出会う場所です。  
これは、分散システムを保護するための重要なテクノロジーですが、正しく行うことは驚くほど困難です。  
残念ながら、利用可能な暗号化ツールは必ずしも非常に使いやすいとは限りません。  
ある医学の友人がかつて私に言ったのは、彼女は若い頃、経済的な理由から医学の学位を短縮し、できるだけ早く専門家を作ることに集中していた国で海外で働いていたということです。  
外科医はファイルに尿検査がなかったという理由で劇場から患者を送り返しました。  
医者が生理学だけでなく手術を理解する必要があるのと同じように、セキュリティエンジニアは少なくとも暗号の基本（およびその他多く）に精通している必要があります  
大まかに言って、暗号に近づくことができる3つのレベルがあります。  
この章では、あなたは暗号の訓練を受けていないと仮定し、基本的な直感を説明することに着手しました。  
暗号化のノウハウが必要な理由の1つは、多くの一般的な構造が混乱し、多くのツールが安全でないデフォルトを提供することです。  
 エンジニアに電子コードブックモードの使用を促します。この章の終わりまでに、それが何であるか、なぜそれが悪いのか、そして代わりに何をすべきかを理解する必要があります。  
暗号化とは、暗号を設計する科学と芸術を指します。それらを解読する科学と芸術に対する解読。暗号学は、しばしば単に暗号化に短縮されますが、両方の研究です。  
その後、状況はやや複雑になります。  
ブロック暗号は、暗号化と復号化の両方に1つのキーを持つ場合があり、その場合、それらは共有キー（秘密キーまたは対称）と呼ばれます  
デジタル署名方式は、特別なタイプの非対称暗号プリミティブです。  
次に、暗号学者が使用するセキュリティモデルを導入することにより、完全な機密性、具体的なセキュリティ、区別不可能性、ランダムオラクルモデルなどの定義を微調整します。  
途中で、人々が弱い暗号を破った方法と、強い暗号を使用した弱い構造の例を示します。  
5.2歴史的背景  
アウグストゥスシーザーが王位を昇格したとき、彼は帝国の暗号システムを変更し、「C」が「A」に、「D」が「B」などに書かれるようになりました。  
驚くべきことに、同様のコードがシチリアのマフィアのカポディトゥッティカピであるベルナルドプロヴェンツァーノによって使用され、「a」は「4」、「b」は「5」などと書かれていました。  
アラブ人は、この考えをモノアルファベットの置換に一般化しました。キーワードは、暗号アルファベットを置換するために使用されます。  
トリックは、いくつかの文字、および文字の組み合わせが他のものよりもはるかに一般的であるということです。英語で最も一般的な文字は、e、t、a、i、o、n、s、h、r、d、l、uの順です。  
文字とダイグラム（文字のペア）の使用  
強力な暗号を作成するには、基本的に2つの方法があります。ストリーム暗号とブロック暗号です。  
  
この初期のストリーム暗号は一般的に、チャールズ9世に仕えた外交官であるフランス人ブレーズドヴィジェンテルに帰属します。  
数学者はこれをC = P + K mod 26として記述します。たとえば、Pを追加すると（15）  
 35になり、26を引くと9になります。  
 J、またはより単純にU + P = Jです。  
印刷されたテーブルから真鍮の暗号化ホイールに至るまで、これをすばやく行うための技術が開発されました。  
 女性化したGiacomo CasanovaからコンピューティングのパイオニアであるCharles Babbageまで、多くの人々がポリアルファベット暗号を解く方法を考え出したようです。  
彼は、十分に長い暗号文が与えられた場合、繰り返しパターンがキーワード長の倍数で現れることに気づきました。  
3は6と9の両方を分割するため、3文字のキーワードを推測する可能性があります。  
  
このタイプのストリーム暗号を攻撃から保護する1つの方法は、キーシーケンスをプレーンテキストと同じ長さにし、繰り返さないようにすることです。  
したがって、対戦相手が実行できる計算の量に関係なく、暗号文が与えられている場合、対戦相手は賢いわけではなく、その長さのすべての可能な平文が等しく可能性があります。  
ここに例があります。したがって、図5.3に示すように、ワンタイムパッドの最初の10文字はwclnb tdefjでした。プレーンヘリヒトラーキーwclnbtdefj暗号DGTYIBWPJA図5.3 –スパイのメッセージしかし、彼が主要な素材で絹を焼くと、スパイは主張できます。彼は実際にはアンダーグラウンドレジスタンスのメンバーであり、メッセージは実際に「ハングヒトラー」と言った。  
したがって、このスパイを問題にしたい場合は、暗号文をDCYTI BWPJAに変更できます（図5.4）  
実際、戦争中、クロードシャノンは、可能な平文として可能な限り多くのキーがあり、すべてのキーが等しく可能である場合にのみ、暗号が完全な秘密性を持っていることを証明しました。したがって、ワンタイムパッドは、完全な秘密を提供する唯一のシステムです。  
ワンタイムテープは、第二次世界大戦後期からの双方によるトップレベルの通信、その後NATOの同盟国間の戦略的通信、および1963年からのUS-USSRホットラインに使用されました。  
ただし、このような暗号化は、トラフィックと同じくらい多くのキーマテリアルを消費するため、ほとんどのアプリケーションにとって高額です。  
次に、キーストリーム（一度に1つのシンボル）とデータを組み合わせてデータを暗号化します。  
初期の例は、ローターマシン、非常に長い一連の疑似ランダム状態2を生成し、それらをプレーンテキストと組み合わせて暗号文を取得する機械的ストリーム暗号デバイスでした。  
銀行は一般的に興味がありませんでした。理由は以下で説明しますが、第二次世界大戦の戦闘員がローターマシンを使用して無線トラフィックを暗号化し、連合国がドイツのトラフィックを解読しようとする取り組みには、戦後のコンピュータ業界のキックスタートに貢献した、ColossusのAlan Turing氏など。  
ただし、ブロック暗号はより柔軟で、現在設計されているシステムではより一般的であるため、次にそれらを見てみましょう。  
5.2.3初期のブロック暗号– Playfair  
ホイートストン暗号と呼ばれていないのは、政治家であるプレイフェア男爵にそれを示したからです。プレイフェアはアルバート王子とパーマーストン子爵（後の首相）にそれを示した  
この暗号は5 x 5のグリッドを使用します。ここにアルファベットを配置し、1で並べ替えます。マシンに関する情報は、暗号博物館（https：// www）で見ることができます。  
米国で使用されているHagelinマシンの場合は2文字、ドイツ語のエニグマとイギリスのTypexの場合は順列、キーワード「J」は省略（図5.6を参照）  
プレイフェアが彼のナプキンに書いた例は、「ロードグランビルの手紙」であり、これは「ロードグラムバイヴィルクスルスルテテル」になります。  
たとえば、「am」は「LE」に暗号化します。それ以外の場合、2つの文字はテーブルの長方形の2つの角にあり、この長方形の他の2つの角にある文字に置き換えます。  
これで、次のように標本テキストを暗号化できます。第一次世界大戦ではフィールド暗号として、第二次世界大戦ではアメリカ人とドイツ人によって軍。  
 単一の文字ではなく、分布がより広く、攻撃にはより多くの暗号文が必要です。  
Playfairの暗号文はランダムに見えます。ただし、プレーンテキストのペアの1文字を変更すると、多くの場合、暗号文の1文字のみが変更されるという特性があります。  
結果の1つは、十分な暗号文または考えられるいくつかの単語を指定すると、テーブル（または同等のもの）  
実際、この章の冒頭にある引用は、将来のジャックケネディ大統領がモーター魚雷艇が沈没した後、10人の他の生存者とともに小さな島に閉じ込められた若いジャックケネディ大統領によって送信された、プレイフェアで暗号化されたメッセージです。日本の駆逐艦との衝突。  
より強力な暗号の場合、暗号の入力の小さな変更による影響が、出力全体に拡散するようにします。  
これらのアイデアは、次のセクションでさらに詳しく説明します。  
たとえば、Data Encryption Standard（DES）  
他のほとんどのアプリケーションで置き換えられたのブロック長は、この2倍です。たとえば、銀行口座番号が常にトランザクションの同じ場所に表示される場合、それを含むトランザクションが同じキーで暗号化されるたびに同じ暗号文が生成される可能性があります。  
詐欺師が銀行の電話会社で働いていて、彼が「IBMに10,000,000ドル支払う」と言っているのを知っていた暗号化されたトランザクションを監視したとします。  
したがって、暗号ブロックがメッセージと同じ大きさでない限り、暗号文には複数のブロックが含まれるため、ブロックを結合する何らかの方法が必要になります。  
5.2.4ハッシュ関数  
これは、メッセージの整合性と信頼性を保護するために進化しました。平文で予測可能な変更を引き起こすような方法で誰かが暗号文を操作できるようにしたくない場合です。  
「有線」とは、「To Lombard Bank、London」などの支払い指示です。  
1234567890 HSBC Bank Cambridge no。の口座を持っている456 Chesterton RoadのJohn Smithへの1000ポンドの合計。  
米国カリフォルニア州サンタバーバラの最初のカウボーイバンクから。  
19世紀には、銀行、電信会社、海運会社がトランザクションを保護するだけでなく、トランザクションを短縮できるコードブックを開発しました。これは当時の国際電信のコストを考えると重要でした。  
ですから、「私たちの口座からあなたと一緒に支払わないでください」 「AFVCT」になる可能性があります。  
銀行は、ストリーム暗号もコードブックもメッセージの信頼性を保護しないことを認識しました。  
銀行の目的にとって重要な革新は、コードブックを使用することでしたが、コードグループをテストキーと呼ばれる数にまとめて追加することにより、コーディングを一方向にすることでした。  
 これは簡単な例です。  
、54（300,000）  
 と71（6,000）  
これにより、207のテストキーが得られます。  
優れたシステムでは、コードグループは2桁ではなく4桁であり、攻撃者がテーブルを再構築するのをより困難にするために、テストキーは圧縮されました。追加すると、キー '7549'は '23'になります。 1桁目と2桁目、および3桁目と4桁目で、桁上げは無視されます。  
実際、一方向関数は少なくとも17世紀から存在していました。  
（目標は、アイディアの優先順位を設定すると同時に、アイディアに対してさらに作業を行う時間を与えることでした。）  
設計の詳細に応じて数十から数百のテスト済みメッセージが与えられると、患者アナリストはトランザクションを偽造するのに十分なテーブルを再構築できます。  
しかし、銀行はそれを回避しました。テストキーは、19世紀後半から1980年代まで細かく機能しました。  
銀行業務と簿記の章で、テストキーに代わるシステムについて説明します。  
彼らは、核兵器の指揮統制で使用される認証コードに現代の重要な暗号を用いた重要な現代の子孫を持っています。  
任意の長さのメッセージMがあり、128ビット長の認証コードAを計算したいとします。また、必要なプロパティは、同じキーの下の認証コードが同じである別のメッセージM 0を見つけることができないはずです。 A、確率が2ffi128である幸運な推測を除いて、彼らが鍵を知っている場合を除きます。  
 ここで、鍵は2つの128ビット番号k1およびk2で構成されています。  
 M 0をAに認証します。  
256ビットのキーと128ビットのタグしかないため、これは無制限の計算能力を持つ敵対者にも当てはまります。そのような敵対者は、メッセージとタグの各ペアに対して2128の可能なキーを簡単に見つけることができますが、それらの間で選択する方法はありません。 。  
  
最後に、暗号化と復号化に異なるキーが使用されるという点で、一部の最新の暗号化システムは非対称です。  
詳細については後で説明します。  
あなたは私にそれを宛てて、それをポストボックスに落とすことによって私にプライベートメッセージを送ることができます。  
もちろん、多くのことがうまくいかない可能性があります。あなたは私に間違ったアドレスを取得する可能性があります（エラーによるものか、欺瞞の結果によるものか）  
同様のことが公開鍵暗号化でもうまくいかない可能性があります。システムに偽の公開鍵が挿入されたり、コンピューターがハッキングされたり、人々が強制されたりする可能性があります。  
暗号化のもう1つの非対称アプリケーションは、デジタル署名です。  
繰り返しになりますが、原稿の署名や印鑑の形でコンピューター以前の類似物があります。繰り返しになりますが、古いやり方と新しいやり方の両方で、うまくいかないことがある非常によく似たものがあります。5.3セキュリティモデル  
セキュリティモデルは、暗号が「良い」という考えを形式化しようとしています。  
同様に、キーを1回だけ使用する認証スキームは、それに対する最良の偽造攻撃がランダムな推測になるように設計できます。その成功確率は、十分に長いタグを選択することで、必要なだけ低くすることができます。  
執筆の時点では、68ビットの暗号パズルを解いて新しいブロックを掘り起こすには、存在する最も強力な敵、デンマークの州と同じくらいの電力を消費するビットコインマイナーのコミュニティが約10分かかります。  
したがって、1000年後でも、偶然に正しいキーが見つかる確率は2ffi35または10億分の1です。  
時間tで働いている敵が最大で✏の確率で暗号を破ることに成功した場合は-secure  
これにより、関心のある暗号の特定の特性について推論することができます。  
これは、暗号学者と暗号解読者にゲームをプレイしてもらうことで形式化されます。そこでは、分析者が無視できない確率で識別できないはずのものの効率的な弁別子を見つけて勝ちます。  
通常、セキュリティの証明は、ランダム化された（つまり、  
 無視できない確率で情報を学習するnの時間多項式で実行されるアルゴリズム。これにより、すでに信頼されている基礎となる暗号プリミティブの効率的な識別が可能になります。  
これにより、KatzやLindell [1022]などの標準テキストで見つけられるかなりの数学的詳細がスキップされます。  
暗号プリミティブをそのタイプのランダム関数と区別する効率的な方法がなく、特に、適用されるすべての統計的およびその他のランダム性テストに合格する場合は、暗号プリミティブを疑似ランダムと呼びます。  
図5.9：–ランダムなオラクルランダムなオラクルを視覚化するために、物理的なランダム性のソースといくつかの保存手段を備えたブラックボックスに座っているエルフを想像します（図5.9を参照）。  
エルフは特定のタイプの入力を受け入れ、スクロールを調べて、このクエリが以前に応答されたことがあるかどうかを確認します。  
さらに、有限の帯域幅を想定します。エルフは毎秒非常に多くのクエリにのみ応答します。  
  
ランダムオラクルの最初のタイプはランダム関数です。  
同じ入力で同じ出力が得られますが、出力のセットはランダムに表示されます。  
ランダム関数は、暗号化ハッシュ関数のモデルです。  
たとえば、警察があなたのラップトップを押収した場合、標準のフォレンジックツールはすべてのファイルのチェックサムを計算して、どのファイルがすでに知られているか（システムファイルなど）を特定します  
。  
また、特定の日付までに特定の電子ドキュメントを所持していたという証拠が必要な場合は、それをオンラインのタイムスタンプサービスに送信するか、ビットコインブロックチェーンにマイニングすることができます。  
これは、上記のセクション5.2.4で説明したフックのアナグラムの現代版です。  
5.3.1.1プロパティ  
入力xの知識があれば、ハッシュ値h（x）を簡単に計算できます  
 そのような入力がまだ知られていない場合は、xを見つけます。  
 出力はランダムであるため、攻撃者がランダム関数を反転させるためにできる最善の方法は、幸運になるまでより多くの入力を送り続けることです。 nビット出力の場合、これは平均で約2nffi1の推測になります。  
したがって、偶然に目的のターゲット出力を持つ入力を相手が推測できないほど多くの可能な出力がある場合、疑似ランダム関数も一方向関数になります。  
たとえば、SHA256が疑似ランダム関数であると主張する場合、特定の256ビット値にハッシュする入力を見つけるための実用的な方法は、すでに知っていて、その値を計算するために使用しない限り、存在しないということです。 。  
したがって、秘密鍵kと連結してh（x、k）を計算することにより、値xの一方向暗号化を取得できます。  
ただし、ハッシュ関数が十分にランダムではない場合、この方法で一方向の暗号化にハッシュ関数を使用すると問題が生じます。  
 出力が十分に長い疑似ランダム関数の3番目の特性は、衝突、つまりh（M1）の異なるメッセージM1 = M2を見つけるのが難しいことです。。  
 次に、衝突を見つける最良の方法は、大量のメッセージMiとそれに対応するハッシュh（Mi）を収集することです。  
ハッシュ関数の出力がnビットの数値で、2nの可能なハッシュ値がある場合、敵が一致を見つけることができるようになる前に計算する必要があるハッシュの数は、これの平方根になります。 2n / 2ハッシュ。  
  
誕生日の定理は次の問題からその名前を得ます。  
ほとんどの生徒は直感的にそれはありそうにないと考え、数学の教師は生徒に次々に自分の誕生日を述べるように頼みます。  
これはほとんどの人を驚かせるように、「誕生日のパラドックス」としても知られています。  
湖にN匹の魚がいて、m匹を捕まえ、鳴らしてから投げ戻したとします。すでに鳴っている魚を最初に釣ったとき、mはNの平方根の「約」になるはずです。  
この定理には、セキュリティエンジニア向けの多くのアプリケーションがあります。  
これは、可能なペアがほぼ2億個になるためです。  
ただし、主張されたIDを検証するためにそれを使用しても問題ない可能性があります（他の多くのことがうまくいかない可能性があります。議論については、パート2の生体認証に関する章を参照してください）。  
衝突検索攻撃が問題にならないアプリケーションがいくつかあります。たとえば、攻撃者が発行されたチャレンジへの回答を見つけなければならないチャレンジレスポンスプロトコルや、チャレンジの繰り返しを防ぐことができる場合などです。  
 たとえば、一般的な機器の応答長は48〜80ビットです。  
しかし、衝突が許容できない他のアプリケーションがあります。  
、後で説明するいくつかの理由から。  
 = h（M2）  
 。  
。  
/ NmはN 'm2 / 2log（1 / ffi）によって漸近的に解決されます  
それらを衝突のないようにするのに十分な大きさ。  
ただし、衝突検索はハッシュ関数の強度に上限を与えるだけであり、これらの特定の関数はどちらもがっかりすることがわかりました。これについては、セクション5.6.2で後述します。  
ただし、外部から与えられた既存のハッシュの2番目のプリイメージを誰も見つけられないことを確認する必要があるだけの場合は、おそらくそれより少なくすることができます。  
5.3.2ランダムジェネレーター–ストリーム暗号  
これもランダム関数ですが、入力が短く出力が長いという点でハッシュ関数の逆です。  
これは、次のようにバックアップデータの機密性を保護するために使用できます。キーストリームジェネレーターに移動し、キーを入力し、ランダムビットの長いファイルを取得し、プレーンテキストデータと排他的論理和して、暗号文を取得します。クラウドのバックアップサービスに送信します。  
 エルフが新しいキーを提示されるたびに必要な長さのランダムなテープを生成し、それを私たちに渡して、同じ入力キーが再び与えられた場合に備えて、参照用にスクロールにコピーを保持していると考えることができます。  
キーストリームジェネレーターにアクセスできる他の人は、キーを知らない限り、同じキーストリームを生成できません。  
そのために、ファイルのハッシュを作成し、安全な場所に保管する場合があります。  
ワンタイムパッドシステムは、時間ではなく空間全体の通信を保護するために使用されるという点を除いて、理論モデルにぴったりです。2つの通信する当事者が事前にキーストリームのコピーを共有しています。  
キーストリームジェネレーターの実際の問題は、複数のバックアップテープを暗号化する場合でも、通信チャネルで送信される複数のメッセージを暗号化する場合でも、同じキーストリームが複数回使用されるのを防ぐことです。  
ただし、M1 + K = C1およびM2 + K = C2の場合、対戦相手は2つの暗号文を組み合わせて2つのメッセージの組み合わせを取得できます。C1ffi C2 = M1 ffi M2、およびメッセージMiに十分な冗長性がある場合、回復した。  
1人の元NSAの主任科学者の言葉では、それは「2回限りのテープ」になりました。  
 そのため、毎回異なる場所でキーストリームを開始します。  
ここでは、古いキーストリームを再利用させようとする可能性のある敵対者がいる場合でも、両方の当事者が適切に機能するキーで同期することを確認する必要があります。  
5.3.3ランダム順列–ブロック暗号  
ここでは、関数は反転可能であり、入力平文と出力暗号文は固定サイズです。シンボルの数と基になるアルファベットに関係なく、暗号化は固定長のブロックに作用します。  
 次のようにブロック暗号化を視覚化できます。  
これには、左側に平文の列があり、右側に暗号文の列があります。  
そうでない場合、サイコロを振って適切なサイズのランダムな暗号文を生成します（これはまだスクロールの右側の列に表示されていません）。  
レコードが見つかった場合は、右側の列から対応する暗号文が得られます。  
そうでない場合、彼はランダムに新しいメッセージを生成し、それを書き留めて私たちに渡します。  
キーごとに、他のすべてから独立した単一の順列があります。  
直感的な考え方は、暗号化マシンは平文と鍵を与えられた暗号文を出力し、暗号文と鍵を与えられた平文を出力するべきであるが、平文と暗号文だけを与えられれば何も出力すべきではないということです。  
プロトコルの章で暗号化用に確立された表記を使用してブロック暗号を記述します。C = {M} Kランダム置換モデルでは、ブロック暗号に対するさまざまなタイプの攻撃を定義することもできます。  
選択された平文攻撃では、対戦相手は特定の数の平文クエリを入力し、対応する暗号文を取得することができます。  
選択された平文/暗号文攻撃では、彼はどちらのタイプのクエリを作成することも許可されています。  
いずれの場合も、攻撃者の目的は、まだ行っていないクエリに対する答えを推測することです（偽造攻撃）。  
。  
誰かが暗号プリミティブに脆弱性を発見した場合、それはアプリケーションに関連する場合とそうでない場合があります。  
そのため、どのような攻撃が検出され、どのようなパラメータがあるかを正確に見つけるために、注意深く調べる必要があります。  
 次の主要な攻撃（線形暗号解読）中にキーを回復するために247の選択された平文が必要  
これらの攻撃は科学的に非常に重要でしたが、実際のエンジニアリング効果はゼロでした。実用的なシステムではそれほど知られていないテキスト（選択されたテキストはもちろんのこと）がないためです。  
このような非実用的な攻撃は、実用的な悪用を提供するのではなく、暗号のセキュリティ認証に影響を与えるため、しばしば認証と呼ばれます。  
他のいくつかのケースでは、証明として開始された攻撃は、後のアイデアによってエクスプロイトに開発されました。  
たとえば、ブロードキャストエンターテインメントシステムを使用すると、ハッカーはデコーダを購入し、多くの映画を見て、暗号化された放送信号と比較することができます。そのため、既知の平文攻撃が主な脅威になる可能性があります。  
歴史的な例は第二次世界大戦で、アメリカのアナリストは、彼らがミッドウェイを意味すると疑った島「AF」に対する日本の意図を知った。  
ミッドウェイが日本の目的であることを知って、チェスターニミッツ提督は彼らを待っていて、4隻の日本の空母を沈め、戦争の流れを変えました[1001]。  
選択された平文/暗号文攻撃は、脅威が昼食時の攻撃である場合に心配になる可能性があります。許可されたユーザーが不在の間に暗号デバイスに一時的にアクセスし、許可されたすべての操作を選択したデータでしばらくの間試行する人。  
。  
 無視できる確率以外は。  
5.3.4公開鍵暗号化とトラップドア一方向  
私たちの類推を続けるために、ユーザーは彼女とエルフだけが知っている巻物に秘密の名前を付け、エルフのパブリック一方向関数を使用してこの秘密の名前のハッシュを計算し、ハッシュを公開して、エルフにこのハッシュを引用する人のために暗号化操作を実行します。  
必要なのは、彼らがオラクルにアクセスできることだけです。  
これは誰でも実行できる計算ですが、秘密鍵などのトラップドアを知っている人だけが元に戻すことができます。  
それにもかかわらず、それを正式に述べましょう：公開キー暗号化プリミティブは、ランダムな入力Rが与えられると2つのキーKR（公開暗号化キー）を返す関数で構成されます  
 以下の特性を備えています1。  
; 2。  
。  
復号化キーKRffi1を使用して暗号文Cに適用される復号化関数があり、元のメッセージM = {C} KRffi1を生成します。金属ではなく数学を使用してその機能を実装します。  
これよりもさらに要求の厳しいモデルがあります。たとえば、ターゲットの暗号文を除いて、対戦相手が選択した暗号文を復号化できる場合のセキュリティを分析します。  
  
ここで定義する最終的な暗号プリミティブは、デジタル署名です。  
したがって、紙の世界で通常の署名が行うのと同じ機能を電子の世界で実行できます。  
署名方式も決定論的またはランダム化できます。最初に、メッセージの署名を計算すると常に同じ結果が得られ、2つ目では異なる結果が得られます。  
 また、署名方式はメッセージの回復をサポートする場合としない場合があります。  
正式には、公開キー暗号化スキームのような署名スキームには、ランダムな入力Rが与えられると2つのキーffiR（秘密署名キー）を返すキーペア生成関数があります。  
 以下の特性を備えています1。  
メッセージMと秘密署名鍵ffiRを指定すると、署名SigffiR {M}を生成するデジタル署名関数があります。および3。  
メッセージの回復が不要な場合は、単純なデジタル署名アルゴリズムをランダム関数としてモデル化して、入力メッセージを固定長の一方向ハッシュ値に減らし、その後に特別な種類のブロック暗号を適用することができます。 elfは、1つのプリンシパルに対してのみ、署名と呼ばれる一方向の操作を実行します。  
この単純なスキームの場合、署名検証とは、エルフ（または署名検証アルゴリズム）を意味します  
しかし、メッセージ回復のあるスキームでは、誰でも署名を入力して、それに対応するメッセージを取り戻すことができます。  
。  
メッセージの回復を使用するアプリケーションは、機械で印刷された切手または印です。切手は、郵便メーターによって作成されたデジタル署名付きの2次元バーコードで構成され、値、日付、送信者の情報などが含まれます。および受信者の郵便番号。  
一般的なケースでは、メッセージの回復は必要ありません。署名するメッセージの長さは任意であるため、最初にハッシュ関数を通過させてから、ハッシュ値に署名します。  
  
これで定義が整理されたので、内部を調べて、それらを実際に実装する方法を確認します。  
実際、研究レベルでも、ほとんどの暗号化は数学と同じくらいコンピューターサイエンスです。暗号に対する最新の攻撃は、ビットの推測、パターンの検索、可能な結果の並べ替えなどからまとめられ、何よりも工夫と永続性が必要です。特に眉毛。  
5.4.1 SPネットワーク  
たとえば、入力テキストのブロックにいくつかのキーマテリアルを追加してから、入力のサブセットを表示し、この方法で何度も続けることができます。  
ブロック暗号は拡散と混乱を必要とします。  
図5.10は、16ビット数のビットとして想像できる16の入力と、4ビットの反転可能な置換ボックス（またはSボックス）の2つのレイヤーを持つSPネットワークを示しています。  
この配置のポイントは、デジタルロジックに任意の16ビットから16ビットの関数を実装する場合、220ビットのメモリが必要です。つまり、単一の出力ビットごとに216ビットのルックアップテーブルが1つ必要です。  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
 ？  
パラメータを適切に選択することで、この単純な構造を反復することによって生成される関数が、ランダムな16ビットから16ビットの関数と、キーの値を知らない対戦者との区別がつかなくなることを望むかもしれません。  
このような設計を安全に行うには、3つのことを行う必要があります.1。  
十分なラウンドが必要です、そして3。  
  
まず、16ビットのブロックで動作するブロック暗号は、対戦相手が平文と暗号文のブロックの辞書を、それらが観察されたときに構築できるため、かなり制限されます。  
そのため、実用的なブロック暗号は通常、平文と64ビット、128ビット、またはそれ以上の暗号文を扱います。  
 またはそれらの32（128ビットブロックサイズの場合）  
  
第二に、十分なラウンドが必要です。  
たとえば、右端の12ビットを一定に保ち、左端の4ビットを微調整して、左上のSボックスの値を推定することができます。しかし、それはまだ基本的な学生の練習です。）  
簡単な例では、Sボックスの1つのラウンドからの各出力ビットが次のラウンドの1つの入力ビットにのみ接続されるため、拡散は非常に遅くなります。  
暗号化だけでなく復号化にもブロック暗号を使用する場合は、この線形変換を可逆にする必要があります。  
  
Sボックスの設計は、セキュリティに必要なラウンドの数にも影響し、悪い選択を研究することで、ブロック暗号のより深い理論への入り口が得られます。  
 出力へ（5,7,0,2,4,3,1,6,8,10,15,12,9,11,14,13）  
次に、入力の最も重要なビットは、出力の最も重要なビットとして変更されずに渡されます。  
私たちの暗号が疑似ランダムであるとは断言できませんでした。  
5.4.1.4線形暗号解析  
Sボックスには、入力のビット1が出力のビット2とビット4に等しいという特性があったことがわかるかもしれません。より一般的には、一定の確率で成立するSボックスへの線形近似があります。  
確率p = 0.5 + 1 / Mで暗号全体を保持する線形関係を見つけることができる場合、確率理論のサンプリング定理によれば、約M 2の既知のテキストが得られたら、キービットの回復を開始することが期待できます。  
次に、暗号は線形暗号解読に対して安全であると見なします。  
5.4.1.5差分暗号解析  
8ビットSボックスの一般的な観察は、「入力ビット2、3、および7を同時にフリップすると、11/16の確率で、フリップする出力ビットは0と1だけになる」というものです。  
分析手順は、考えられるすべての入力差分パターンを調べ、それらの値ffii、ffioを探し、ffiiの入力の変化が特に高い（または低い）ffioの出力の変化を生成するようにします。  
線形暗号解読の場合と同様に、次に、暗号に入力できる入力差が既知の出力差を生成し、有用な確率で複数のラウンドにわたって生成されるように、結合方法を検索します。  
線形暗号解読と同様に、攻撃に必要なテキストの数が、そのキーのさまざまなテキストの可能な総数よりも多い場合は、暗号が安全であると考えるのが一般的です。  
多くのキーの下に多くのテキストがある場合、最終的に現在のキーを解決します。）  
たとえば、高い確率の違いを探す代わりに、起こり得ない（またはまれにしか起こらない）違いを探すことができます。  
これは不可能な暗号解読という魅力的な名前を持っていますが、多くのシステムに対して非常に完全に可能です[242] 4。  
たとえば、ラウンドを注意深く設計することで、ラウンドごとの情報漏えいを減らし、必要なラウンド数を減らすことができます。  
単純なラウンドも分析が容易な場合があります。  
しかし、暗号が私たちが知っているすべての攻撃に抵抗し、ある程度の安全マージンがあることを示すことはできるかもしれませんが、これは新しいタイプの攻撃に抵抗するかどうかについてはほとんど言いません。  
   
Advanced Encryption Standard（AES）  
128ビットブロックで動作し、長さが128、192、または256ビットのキーを使用できます。  
AESは、バイト入力に作用してバイト出力を提供する単一のSボックスを使用します。  
256バイト。実際には、式S（x）によって定義されます。  
 +フィールドGF（28）を超える  
この構造は、厳密な微分および線形境界を与えます。  
最初のステップはshu✏eです。4バイトの一番上の行は変更されずに残り、2番目の行は左に1桁、3番目の行は2桁、4番目の行は3桁シフトされます。  
これを図5.11に示します。これは、例として、最初の列の3番目のバイトの値の変更がどのように伝搬されるかを示しています。  
。  
。  
つまり、ラウンドごとに16バイトの鍵素材が必要になります。これらは、反復関係を使用して、ユーザーが指定したキーマテリアルから派生します。  
これらは、AESの競争の時点で実際に期待したとおり、およびこの章の前の版で説明したように、実用的なセキュリティを提供するには十分です。  
関連するキーがある場合、より速いショートカット攻撃が知られています。  
AESを信頼すべきですか？  
（Camelliaは、自身のAES候補者が最初のラウンドでノックアウトされたチームによって設計されました。）さて、私はAESファイナリストSerpent [94]の設計者の1人で、コンテストで2位になりました。勝者Rijndaelが86票、Serpent 59票、Twofish 31票、RC6 23票、MARS 13票を獲得しました。  
しかし、単純な事実は、Serpentの方が安全ですが、Rijndaelの方が速いということです。業界および暗号の研究者が最後のAES会議で投票し、NISTがこれを標準として承認しました。  
また、AESは蛇よりも安全ではありませんが、実用的なセキュリティはすべて実装に関するものであり、現在、AESの実装には膨大な経験があります。  
前者の主なリスクは、対戦相手がキャッシュミスを観察し、それらを使用してキーを計算することです。  
両方について、パート2の排出ガスのセキュリティに関する章で詳しく説明します。対策には、AESを実行するための多くのCPUでの特別な操作が含まれます。これは、アルゴリズムが現在標準となっているため、正確に利用できます。  
（後で、複数のアルゴリズムを使用して何かがひどく壊れたいくつかの例を確認します。）  
 当時の米国の業界にとって十分な暗号を提供するような方法で、外国政府はそれが安全でないと信じさせたので、代わりに独自の弱い設計を使用しました。  
AESはこのプレイブックに従っているようです。数学的に十分強力なアルゴリズムを選択し、その安全な実装にスキルと注意が必要なアルゴリズムを選択することにより、米国政府は、ロシア、中国、日本などの企業が、スキルと労力が少ないため安全性の低いシステムを使用することになると考えました実装に投資されています。  
ああ、そしてNSAは2005年以降、最大SECRETまでの情報を保護するための128ビットキーと、TOP SECRETのための192ビットまたは256ビットキーを備えたAESを承認しています。  
AESの明確な仕様は連邦情報処理標準197であり、その発明者はその設計を詳細に説明した本を書いている[507]。  
5.4.3 Feistel暗号  
その後FeistelはIBMに移り、Data Encryption Standard（DES）を作成する研究グループを設立しました  
Feistel暗号は、図5.12に示すラダー構造を持っています。  
左半分の丸め関数f1が計算され、排他的論理和（キャリーなしのバイナリ加算）を使用して右半分と結合されます。  
（排他的論理和にはffiという表記を使用します。）  
最後に（ラウンド数が偶数の場合）  
Feistel暗号で表示される表記は（f、g、h、...）です。  
連続したラウンド関数です。  
。  
 =（f2k、f2kffi1、...、f2、f1）  
したがって、ラウンド関数fiは可逆である必要はなく、Feistel構造により、一方向関数をブロック暗号に変換できます。  
  
Feistel暗号に関する主要な理論的結果は、1988年にMike LubyとCharlie Rackoffによって証明されました。  
 は、選択された平文攻撃の下でのランダムな順列と区別がつかず、この結果はすぐに拡張され、（f1、f2、f3、f4）  
（私はいくつかの専門性を省略しています。）  
したがって、信頼のあるハッシュ関数がある場合、それからブロック暗号を構築するのは簡単です。Feistelネットワークで4ラウンドのキー付きハッシュを使用します。  
5.4.3.2 DES  
広く導入された「キラーアプリ」はATMネットワークでした。そこから ？  
 XXXXXXXXXXXXXXXXXXXX⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠⇠ffi•f2k ffi ffi？  
。  
？  
 ffi•f2 ffi ffi？  
 左半分右半分図5.12：–前払いメーター、交通機関のチケットなどに広がるFeistel暗号構造。  
そのラウンド関数は32ビットのハーフブロックで動作し、3つの演算で構成されます。•最初に、ブロックが32ビットから48に拡張されます。 •次に、排他的論理和を使用して48ビットのラウンドキーが混合されます。 •結果は8つのS-boxの行を介して渡されます。各S-boxは6ビットの入力を取り、4ビットの出力を提供します。 •最後に、出力のビットは固定パターンに従って並べ替えられます。  
DESの完全な仕様は[1397]に記載されています。  
最も印象的な批判は、キーが短すぎるということでした。  
Whit DiffieとMartin Hellmanは、1977年に、DESキーサーチマシンは100万個のチップで構築でき、それぞれが毎秒100万個のキーをテストすると主張しました。 100万は約220なので、キーを見つけるには平均で215秒、つまり9時間以上かかります。科学者がDESを発明したIBMは、そのようなマシンを構築するために2億ドルを米国政府に請求することになると反論しました。  
 1980年代、さまざまな諜報機関がDESキー検索マシンを構築しているという噂が絶え間なくありましたが、最初の成功した公開キー検索攻撃は1997年に起こりました。  
1998年、Electronic Frontier Foundation（EFF）  
これには、40MHzで動作する1,536個のチップが含まれており、各チップには24個の検索ユニットが含まれ、それぞれがテストの復号化に16サイクルかかりました。  
クラッカーのデザインは公開されており、[619]にあります。  
100,000台のマシンを備えた最新のボットネットは数時間かかります。  
DESに対するもう1つの批判は、IBMが米国政府の要請に応じて設計原則を秘密にしていたため、おそらく簡単にアクセスできる「罠」があったことです。  
その話は、IBMが1972年にこれらの技術を発見したこと、そして米国国家安全保障局（NSA）  
IBMはNSAの要求に応じて設計の詳細を秘密にしました。  
これで、DESをかなり完全に分析できました。  
DESは20ラウンド以上で安全ですが、実用上、そのセキュリティはキーの長さによって制限されます。  
したがって、既知のショートカット攻撃は問題ではありません。  
AESと同様に、タイミング分析と電力分析に基づく攻撃もあります。  
銀行ネットワークは、1999年以来標準であるトリプルDESに大きく移行しています[1397]。  
正式には：3DES（k0、k1、k2; M）  
）  
 3つのキーを等しく設定すると、単一のDES暗号化と同じ結果が得られ、レガシー機器との下位互換性モードが提供されます。  
 ほとんどの新しいシステムはデフォルトの選択としてAESを使用しますが、ATM、POS端末、および銀行ネットワークで使用される多くのプロトコルで使用されるメッセージ形式のため、多くの銀行システムは8バイトブロックのブロック暗号の使用に専念しています。お互いに話してください、そして顧客のPINを生成して保護するためにブロック暗号を使用しているためです（これについては、銀行と簿記の章で説明します）。  
トリプルDESは、予見可能な将来のためのそのような目的のための完全にサービス可能なブロック暗号です。  
 美白です。  
この複合暗号は、DESXとして知られています。  
 = DES（k0; M ffi k1）  
ホワイト化されたブロック暗号は一部のアプリケーションで使用されます。具体的には、以下で説明するXTS操作モードで使用されます。  
  
一般的な失敗は、暗号化ライブラリが開発者に不適切な操作モードの使用を可能にする、または推奨することさえあります。  
 任意の長さのメッセージを処理するように拡張できます。  
それらを理解することは非常に重要であり、特にいくつかの一般的なツールはデフォルトで弱いツールを提供するため、ジョブに適したものを選択できます。  
 次に説明するモード。  
5.5.1ブロック暗号を使用しない方法  
これは、チャレンジ/レスポンスやいくつかの主要な管理タスクなど、単一のブロックを使用するプロトコルに適しています。現金自動支払機システムでPINを暗号化するためにも使用されます。  
たとえば、図5.14は、DESをECBモードで使用して暗号化すると、漫画の画像がどうなるかを示しています。  
前世紀から人気のある企業の電子メールシステムで使用された暗号化は、8文字のパスワードから派生したキーを使用したDES ECBでした。  
これは、フィールドDES暗号化システムに対してこれまでで最も単純な攻撃の1つを与えました：辞書の各パスワードでヌルブロックを暗号化し、回答を並べ替えるだけです。  
さらに、ECBモードを使用して、銀行の支払いメッセージなど、認証を必要とする複数のブロック長のメッセージを暗号化することは、ブロック境界に沿ってカットアンドスプライス攻撃を開始するため、特にばかげています。  
（a）  
 ECB暗号文図5.14：Linuxペンギン、平文でECB暗号化（ウィキペディアから、Larry Ewingによって作成された画像から派生）  
  
暗号化ブロックチェーン（CBC）モードを使用するために使用される複数のブロックを暗号化するほとんどの商用アプリケーション。  
その中で、排他的または暗号文の前のブロックを、暗号化前の平文の現在のブロックに排他的に（または図5.15を参照）  
このモードは、平文のパターンを偽装します。各ブロックの暗号化は、前のすべてのブロックに依存します。  
 ストリーム暗号と同じように、ステレオタイプ化された平文メッセージヘッダーが同一の暗号文に暗号化することにより、情報が漏洩しないようにします。。  
そのため、通常、CBC暗号化は別の認証コードと共に使用する必要があります。  
これは2002年にSerge Vaudenayによって発見されました。その変種は、2016年までにSSL、IPSEC、TLSに対して使用されました[1949]。  
 ？  
 •••EK EK EK？  
 ？  
 ？  
 ---IV P1 P2 P3 C1 C2 C3 ...  
 モード  
ブロック暗号の暗号化のフィードバックモードは、暗号化の問題だけが原因ではなく、流行から脱落しています。  
CBCでは、暗号のブロック全体を各ブロック入力と各ブロック出力の間で計算する必要があります。  
シリコンは安価なので、暗号化チップをパイプライン処理して、新しいブロックを暗号化します（またはキーストリームの新しいブロックを生成します）。  
最も簡単な解決策は、AESをストリーム暗号として使用することです。  
上記のセクション5.2.2で述べたように、追加ストリーム暗号には2つの体系的な脆弱性があります。  
2つ目は、メッセージの整合性を保護できないことです。  
これらのメッセージは高度に構造化されています。たとえば、37〜42バイト目に転送される合計が含まれていることを知っているかもしれません。  
あなたは銀行に入り、共犯者に500ドルを送ります。  
バイト37〜42のMiを知っているので、Kiを回復して、受信銀行に500ドルではなく500,000ドルを支払うように指示する変更されたメッセージを作成できます。これは、詳細な攻撃の例です。これは、ワンタイムパッドから得られる完全な機密性だけでなく、はるかに単純なストリーム暗号の価格でもあります。  
  
2つの古いストリーム暗号操作モード、出力フィードバックモード（OFB）を見つけることができます。  
。  
初期化ベクトルのIVを書くと、K1 = {IV} KおよびKi = {IV} K（iffi1）になります。  
ただし、OFBモードのnビットブロック暗号は、通常、2n / 2ブロックのサイクル長を持ちます。その後、誕生日の定理は、IVにループバックすることがわかります。  
（CBCモードでも、誕生日の定理により、約2n / 2ブロック後に、繰り返しが見られるようになります。）  
カウンタモードは2002年にNIST規格になっただけなので、このOFBがまだ使用されています。  
これは自己同期するように設計されており、バーストエラーが発生して数ビットをドロップしても、システムは1ブロック長後に同期を回復します。  
しかし、これはビットごとに1つのブロック暗号操作を要し、非常に悪いエラー増幅特性を持っています。今日では、人々はトラフィック層で暗号と組み合わせようとするのではなく、同期とエラー訂正のために専用のリンク層プロトコルを使用する傾向があります。  
5.5.5メッセージ認証コード  
これはメッセージ認証コード、つまりMACです。  
（中間結果は、スプライシング攻撃を防ぐために秘密にされます。）  
メッセージの長さが固定されていれば安全です。 Mihir Bellare、Joe Kilian、およびPhilip Rogawayは、これらの状況下でMACに攻撃を行うと、基盤となるブロック暗号に攻撃ができることを証明しました[211]。  
この問題を修正するために、NISTはCMACを標準化しました。CMACでは、最後の暗号化の前に、キーのバリアントがXORされています[1405]。  
 MACが最後の出力ブロックの半分だけで構成され、残りの半分が捨てられているか、他のメカニズムで使用されているレガシーシステムが表示される場合があります。  
  
上記のモードはすべて1970年代と1980年代にDES用に開発されました（ただし、カウンターモードは2002年に米国政府の正式な規格になりました）  
整合性と機密性を保護する必要があるバルク暗号化には効率的ではありません。 CBCモードまたはカウンターモードのいずれかを使用してデータを暗号化し、CBC-MACまたはCMACを使用してデータの整合性を保護する場合、処理するデータのブロックごとにブロック暗号を2回呼び出すため、操作を並列化できません。  
ガロアカウンターモード（GCM）  
テキストのブロックごとにブロック暗号の呼び出しを1つだけ使用し、並列化できるため、低コストで低レイテンシの高速データリンクで高いスループットを得ることができます。  
タグの計算は、セクション5.2.4で説明した種類のユニバーサルハッシュ関数であり、キーが再利用されない限り安全であることが証明されています。  
したがって、出力は平文と同じ長さの暗号文に、IVと通常128ビットのタグが追加されます。  
GCMは、シスコのDavid McGrewとJohn Viegaによって発明されました。彼らの目標は、高性能ネットワークハードウェアでの使用に適した効率的な認証済み暗号化モードを作成することでした[1268]。（以前の複合モードであるCCMがあり、これはBluetooth 4.0以降で使用されます。これは、カウンターモードとCBC-MACを組み合わせているため、計算に約2倍の労力がかかり、並列化したり、段階的に再計算することはできません[1406 ]。）  
5.5.7 XTS  
これは、プレーンテキストの長さを維持する操作モードを好むハードディスク暗号化などのアプリケーションでは非常に不便です。  
 ffi ffi E-Mi•hiffi1 hi図5.16：–フィードフォワードモード（ハッシュ関数）  
これはコードブックモードですが、ディスクセクターから派生した微調整キーによってプレーンテキストが白くなります。  
 ここで、微調整キーKTjは、異なるキーを使用してIVを暗号化し、それを適切な定数で繰り返し乗算して、ブロックごとに異なるホワイトナーを提供することによって導出されます。  
暗号文の操作を検出するために上位層のメカニズムが必要ですが、単純なチェックサムで十分です。  
5.6ハッシュ関数  
ブロック暗号からハッシュ関数を構築することも可能です5。  
。  
stハッシュ値は、ラウンドiの出力と排他的または排他的です。  
。  
誕生日の定理はここで別の形をしています。ハッシュ関数hがnビットブロック暗号を使用して構築されている場合、h（M1）で2つのメッセージM1 = M2を見つけることが可能です。  
 約2n / 2の労力で（多くのメッセージMiよりも少し多くハッシュして一致を探します）  
したがって、64ビットのブロック暗号は適切ではありません。メッセージを偽造すると、232メッセージのオーダーのコストがかかり、非常に簡単です。  
  
1990年代から2000年代にかけて最も一般的に使用されていたハッシュ関数は、512ビットのキーと128ビットから512ビットに増加するブロックサイズを持つブロック暗号の変種として進化しました。  
衝突は2017年に発見され[1828]、攻撃のより強力なバージョンは2020年に発見されました[1146]。 •2002年にSHA-2に置き換わったSHA-2には、256ビットバージョンと512ビットバージョンがあります（SHA256およびSHA512と呼ばれます）。  
これらのハッシュ関数の基礎となるブロック暗号は似ています。それらのラウンド関数は、32ビットプロセッサで使用可能なレジスタ操作の複雑な混合です[1667]。  
MD4は1998年にHans Dobbertinによって破られました[568]。 MD5は、2004年にXiaoyun Wangとその同僚によって破られました[1979年、1981年]。意味のあるテキストを含み、デジタル証明書に使用されるようなメッセージ形式に準拠している文字列の間でも、衝突を簡単に見つけることができます。  
2017年2月、アムステルダムとGoogleの科学者が1つを発表し、要点を証明し、人々にSHA-2 [1828]（および以前のバージョンのTLSからTLS 1.3へ）などのより強力なハッシュ関数に移行するよう説得しました。  
2020年に、Ga¨etanLeurentとThomas Peyrinは、選択されたプレフィックスの衝突を計算する改良された攻撃を開発し、数万ドルのコストで偽造を証明できるようにしました[1146]。  
 代替ハッシュ関数ファミリを見つけるための競争を組織しました[1409]。  
したがって、標準のハッシュ関数としてSHA-2およびSHA-3を選択できるようになりました。  
衝突が特定のアプリケーションを壊すかどうかは、複雑な問題になる可能性があります。  
銀行システムが実際に顧客が作成した「Pay X the sum Y」というメッセージを受け取り、それをハッシュして署名した場合、詐欺師は「Pay X the sum Y」と「Pay X the sum Z」の2つのメッセージを見つける可能性があります。同じ値にハッシュされ、一方が署名され、もう一方と交換されます。  
通常、実際のトランザクションではデジタル署名ではなくMACを使用します。ログはトランザクションのすべての関係者によって保持されるため、衝突するペアのいずれかに潜入することは簡単ではありません。  
  
しかし、衝突発見アルゴリズムが悪者に本物の金銭を盗むことを許す可能性のあるアプリケーションはほとんどないかもしれませんが、脆弱性の存在は依然としてシステムの価値を損なう可能性があります。  
これはおそらく間違いです。  
裁判官は「写真が撮られてから変更されていなかったという合理的な疑いを超えて満足していない」と運転手を無罪とした。彼の奇妙な判決は翌年上訴された[1432]。  
ハッシュ関数には他にも多くの用途があります。  
単純な方法は、メッセージを次のキーでハッシュすることです：MACk（M）  
。  
2つのハッシュ演算は、2つの異なる定数で排他的論理和をとることによって導出された、キーのバリアントを使用して行われます。= h（k ffi B、h（k ffi A、M）  
。  
ハッシュ関数が弱い側にある場合、この構成により、悪用可能な衝突を見つけにくくすることができます[1089]。  
ハッシュ関数のもう1つの使用法は、後で明らかになるコミットメントを作成することです。  
その場合、ドキュメントのハッシュを公開したり、商用のタイムスタンプサービスに送信したり、ビットコインブロックチェーンにマイニングしたりできます。  
繰り返しますが、タイムスタンプを行うときにペアを渡す必要があるため、衝突ペアを生成するアルゴリズムはこれを壊しません。  
入力は、ツリーの葉を形成する値にハッシュされます。各非リーフノードには、子ノードのすべてのハッシュのハッシュが含まれているため、ルートのハッシュは、リーフのすべての値のハッシュです。  
また、ブロックチェーンアプリケーションでも広く使用されています。実際、ブロックチェーンは単なるマークルツリーです。  
たとえば、Lamportデジタル署名はハッシュ関数から作成できます。512個のランダムな256ビット値kiの秘密鍵を作成し、検証鍵Vをそれらのマークルツリーハッシュとして公開します。  
 hのi番目のビットがゼロの場合はk2iを明らかにし、そうでなければk2i + 1を明らかにします。  
マークルは、マスター秘密鍵でカウンターを暗号化することによって一連の秘密鍵を生成し、ツリーを使用して結果の公開鍵をハッシュできることを理解しました。  
ハッシュ関数のセキュリティプロトコルの使用の1つに言及する価値があります。キーの更新と自動キーイングです。  
。  
妥協の連鎖は、ハッシュ関数の一方向性によって破壊されます。  
バリアントは自動プリンティングです。プリンシパルは、最後のキー変更以降にプリンシパルが交換したメッセージでハッシュすることによりキーを更新します。Ki+ 1 = h（Ki、Mi1、Mi2、。  
。）  
攻撃者が自分のシステムの1つを侵害してキーを盗んだ場合、攻撃者が観察または推測できないメッセージを交換するとすぐに、セキュリティが回復します。繰り返しになりますが、妥協の連鎖は壊れています。  
オーストラリアのEFT決済端末のバンキングで最初に使用されました[207、209]。  
次に、これがどのように機能するかを説明します。  
5.7非対称暗号プリミティブ  
ここでは簡単に概要を説明し、アプリケーションについて説明するときにパートIIのメカニズムのいくつかを詳しく見ていきます。  
ほとんどすべての実際のシステムで使用される2つの問題は、因数分解と離散対数です。  
5.7.1因数分解に基づく暗号化  
定義上、1は素数ではありません。つまり、素数は{2、3、5、7、11、...}です。  
素数を見つけてそれらを掛け合わせて合成数を得るのは簡単ですが、合成数をその因子に分解するのははるかに困難です。  
2020年に因数分解される2つの大きなランダムな素数の最大の複合積はRSA-250で、829ビットの数値（10進数で250桁）でした。  
これには、単一の2.2GHzコアで2700年相当の作業が必要でした。以前のレコードである2019年のRSA-240は、900年に相当しました[301]。  
1024ビットの数値については、NSAがすでにそれらを因数分解できると思います。第2版で、「因数分解の履歴の外挿から、最初の因数分解が2018年に公開されることを示唆している」と述べました。ムーアの法則は減速しています。そして私たちは一年遅れています。  
ファクタリングに基づいて公開鍵暗号化とデジタル署名を行うために一般的に使用されるアルゴリズムはRSAで、その発明者であるRon Rivest、Adi Shamir、Len Adlemanにちなんで名付けられました。  
 （証明：セット{1、2、...、p ffi 1}を取り、それらのそれぞれにpを法としてaを掛け、次にキャンセルします（p ffi 1）  
。  
 ⌘1（mod p）  
 n未満の共通の除数がない正の整数の数です（証明は同様です）。  
したがって、nが2つの素数pqの積である場合、ffi（n）  
（q ffi 1）  
RSAでは、暗号化キーは因数分解が困難な係数Nです（ランダムに選択された2つの大きな素数pとq、たとえばそれぞれ1024ビットの場合、N = pqとします）  
秘密鍵は、秘密に保たれている因子pおよびqです。  
 復号化は逆の操作です：M⌘ep C（mod N）  
。  
 =（p ffi 1）  
 eにはffi（N）との共通の要素はありません  
）  
ep C（mod N）  
、および復号化はフェルマーの定理のために機能します：Cd⌘{M e} d⌘M ed⌘M 1 + kffi（N）  
 ⌘M.1⌘M（mod N）  
 ⌘M d（mod N）  
 そして、メッセージMが復元されたことを確認します：M⌘（Sigd（M）e（mod N）  
その理由は、暗号化は代数的プロセスであるため、特定の代数的性質が保持されるためです。  
この特性は、乗法的同型として知られています。準同型は、いくつかの数学的構造を保存する関数です。  
公開鍵暗号化のもう1つの一般的な問題は、平文が「攻撃」や「退却」などの小さなセットから取り出され、暗号化プロセスが確定的である場合です（RSAと同様）。  
RSAでは、代数的攻撃につながる可能性があるため、小さな指数eを使用して同じメッセージを複数の受信者に暗号化することも危険です。  
「攻撃」などの同じショートメッセージを暗号化するたびに、完全に異なる暗号文を取得し、これらを互いに、および「後退」の暗号文と区別できないようにしたいと考えています。  
暗号理論家は、非対称暗号でうまくいかない可能性のあるすべてのものを分析し、それを整理する方法を見つけるために何十年もの間取り組んできました。  
つまり、選択された平文攻撃だけでなく、選択された暗号文攻撃にも対抗する暗号化が必要です。  
通常の実際のソリューションは、最適な非対称暗号化パディング（OAEP）です。  
 C2 = N ffi h（C1）  
その結果、C1、C2の組み合わせがRSAで暗号化されて送信されます。  
 MをC1 ffi h（N）として回復  
これは最終的に安全であることが証明されました。  
これらは、20世紀に発見され、人々がほとんど忘れてしまった多くの攻撃をブロックします。たとえば、同じメッセージを2つの異なるRSAキーで暗号化すると相手が検出できるという事実です。  
サイドチャネル攻撃やデバイスの物理的な調査でさえ、より多くの作業が必要です。  
一般に、秘密鍵を適用する前にメッセージをハッシュするだけで十分な場合がよくあります：Sigd = [h（M）  
; PKCS＃7は、メッセージダイジェストに署名するための簡単なメカニズムを記述しています[1008]。  
実際の実装でうまくいかなかったことの多くは、サイドチャネルとエラー処理に関係しています。  
cd（mod n）として復号化されたときに、特定のcかどうかをターゲットから学習できた場合  
一般的な公開鍵の実装に対しては、通常、解読にかかる正確な時間を測定することにより、さらに多くのサイドチャネル攻撃があります。  
計算のエラーは、係数の1つの係数を法として正しいが他の係数を間違った結果をもたらすこともあり、係数を因数分解できます。エラーは、暗号デバイスに干渉することによって戦術的に挿入することも、64ビット乗算の特定の値を誤って計算するようにチップメーカーが調整するなど、戦略的に挿入することもできます。  
  
RSAはSSLおよびSSHプロトコルに最初に導入された公開鍵暗号化アルゴリズムでしたが、現在最も一般的な公開鍵アルゴリズムは離散対数に基づいています。  
最初に通常のケースを説明します。  
、次に53を52⇥5または4⇥5として計算できます。これは20で、6（モジュロ7）に減少します。  
 52 =⌘4（mod 7）  
 54⌘6 x 5⌘2（mod 7）  
 56⌘3 x 5⌘1（mod 7）  
これは、任意のyが与えられた場合、常に方程式y = 5x（mod 7）を解くことができることを意味します  
このような小さな例は検査によって解決できますが、ランダムな素数pが大きい場合、これを効率的に行う方法がわかりません。  
 f（x + y）という追加のプロパティを持つ一方向関数です  
f（y）  
 = f（x）  
つまり、一方向の準同型です。  
  
古代ローマに戻って、アンソニーがブルータスに秘密を送りたがっていると想像してください。利用可能な唯一の通信チャネルは、信頼できない宅配便（たとえば、シーザーに所属する奴隷）です。  
アンソニーはメッセージを受け取り、ボックスに入れ、南京錠をかけ、宅配便業者にそれをブルータスに届けてもらえる。  
次に彼は南京錠を取り外し、それをブルータスに持ち帰り、ついにそれを開けた。  
アリスはメッセージMを取得し、キーKAで暗号化して{M} KAを取得し、ボブに送信します。  
しかし、可換性のプロパティは、これが{{M} KB} KAであることを意味するため、アリスは自分のキーKAを使用して復号化し、{M} KBを取得できます。  
適切な可換暗号化はどのように実装できますか？  
アリスがランダムなキーxAを選択し、ボブM ffi xAを送信し、ボブがM ffi xBを返し、アリスが最終的にM ffi xA ffi xBを送信すると、攻撃者はこれらの3つのメッセージを単純に排他的ORすることができます。すべてのXに対してX ffi X = 0なので、xAとxBの2つの値は両方とも相殺され、プレーンテキストMが残ります。pを法とする原始根に基づく離散対数問題が難しい場合、暗号化関数として離散指数を使用できます。  
同様に、ボブは乱数xBを選択し、pを法とするgxAxBを形成し、それをアリスに返します。  
（pffixA）  
 ボブに送信します。  
このスキームの安全性は、離散対数問題の難易度に依存します。  
  
Whitfield DiffieとMartin Hellmanによって1976年に公開された最初の公開鍵暗号化方式は、固定されたプリミティブルートgを持ち、gxAxBモジュロpを共有鍵暗号化システムへの鍵として使用します。  
これを見ていきましょう。  
アリスは秘密の乱数xAを選択し、yA = gxAを計算して、会社の電話帳で自分の名前の反対側に公開します。  
ボブと通信するために、アリスは電話帳からyBをフェッチし、gxAxBであるyBxAを形成し、これを使用してボブへのメッセージを暗号化します。  
あるいは、アリスとボブは一時キーを使用して、フォワードセキュリティを提供するメカニズムを取得できます。  
アリスは乱数RAを選択し、gRAを計算してボブに送信します。ボブは同じことを行い、乱数RBを選択してgRBをアリスに送信します。次に、両方がgRARBを形成し、セッションキーとして使用します（図5.19を参照）  
！ B：gRA（mod p）  
 ！ B：{M} gRARB図5.18 – Diffie-Hellman鍵交換プロトコルアリスとボブは、セッション鍵gRARBを使用して会話を暗号化できるようになりました。  
このプロトコルが開始される前に対戦相手が両方のマシンを検査し、格納されているすべての秘密鍵を知っていたとしても、いくつかの基本的な条件が満たされていれば（たとえば、乱数ジェネレーターが予測不可能であり、マルウェアが残されていなかった場合）  
これは、セクション5.6.2で参照したフォワードセキュリティプロパティの強力なバージョンです。  
アリスとボブの両方が使用後に共有シークレットを破棄する場合、それらには後方セキュリティもあります。その後、機器にアクセスした対戦相手は、後方に進んで古いトラフィックを遮断することはできません。  
 作業しているグループを明示する必要がなく、秘密鍵RAと公開鍵gRAを明示的に書き出す必要がない場合。  
パラメータpとgを選択するときは注意が必要です。たとえば、スノーデンの開示から、NSAが一般的に使用される1024ビットの素数の離散対数問題を解くことができると推測できます6。  
しかし、このプロトコルには小さな問題があります。アリスとボブは最終的にセッションキーを取得しますが、どちらもそれを共有する本当の考えを持っています。  
奴隷は箱をアンソニーに戻し、アンソニーは南京錠を取り外し、箱を開けたシーザーに箱を戻します。  
1つの修正は、AnthonyとBrutusが封印をロックに適用することです。  
チャーリーはアリスのボブへのメッセージを傍受し、それに返信します。同時に、彼はボブと鍵交換を開始し、アリスのふりをします。  
 彼はアリスと共有し、別のキーDH（RB、RC）  
彼がネットワークの真ん中に座り、それらの間でメッセージを翻訳し続ける限り、彼らは彼らの通信が危険にさらされていることを検出するのに苦労するかもしれません。  
STU-2電話（現在は使用されていませんが、フォートミードのNSA博物館で見ることができます）では、2人のプリンシパルが生成したキーの8桁のハッシュを読み取り、以前と同じ値であることを確認します分類された問題について話し始めました。  
プロトコルは複数の攻撃を受け、最近ではBluetooth（KNOB）の主要なネゴシエーションが発生しています  
この規格では、1〜16バイトのキーの長さが許可されています。 6可能性のある離散ログアルゴリズムであるNFSは、各素数の大きな計算に続いて、その素数を法とする各離散ログの小さな計算を含みます。  
]、10年前の3倍の効率のNFSバージョンを使用しています。  
キー長のネゴシエーションは平文で実行されるため、攻撃者は長さを下限に強制できます。  
以前のバージョンのBluetoothは、セクション14.3.3.3で説明されているHomePlugプロトコルの「ジャストワーク」モードに似ていますが、主に、害のない環境で適切なデバイスとのペアリングキーを設定できるように設計されていました。敵対的な攻撃における巧妙な攻撃から防御するよりも。非常に多くのことがうまくいきません。非常に弱い鍵を生成または受け入れるため、保護のように見えるだけのプロトコル。解読にかかる時間など、サイドチャネルを介してキーをリークするプログラム。スタックのオーバーフローやその他のハッキングにつながるソフトウェアの脆弱性。  
また、実際の暗号コードを自分で記述しないでください。適切にそれを行うには、計算数理論​​からサイドチャネル分析や形式的方法まで、さまざまなスキルが必要です。  
  
ベースpとジェネレーターgが何らかの適切な方法で選択された公開値であり、メッセージに署名することを望む各ユーザーが公開署名検証キーY = gXを持つ秘密署名キーXを持っていると仮定します。  
メッセージキーkをランダムに選択し、フォームr = gk（mod p）  
ここで、k、r、メッセージM、秘密鍵Xの線形方程式を使用して署名sを作成します。  
/ k;これはffi（p）を法として行われます  
両側が一方向準同型写像f（x）を通過するとき  
機能するデジタル署名方式を取得するには、さらにいくつかの詳細を修正する必要があります。  
また、ハッシュ関数を使用してメッセージMをハッシュし、任意の長さのメッセージに署名できるようにします。これにより、対戦相手がアルゴリズムの代数構造を使用して、署名されたことのないメッセージの署名を偽造できなくなります。  
 これは米国の標準であり、政府のアプリケーションで広く使用されています。  
、pを法とする整数の次数qの要素g、秘密署名鍵x、公開検証鍵y = gx。  
、は（r、s）  
）  
 s⌘（h（M）  
/ k（mod q）  
DSAは、メッセージを復元しないランダム化されたデジタル署名方式の典型的な例です。  
  
離散対数とその類似体は、他の多くの数学的構造に存在します。  
これらの曲線には、それらに対する加算演算を定義でき、結果のMordellグループを暗号化に使用できるという特性があります。  
ただし、楕円曲線暗号システムは、少なくとも2つの理由で興味深いものです。  
楕円曲線暗号は、EMV支払いカードの最新バージョンからビットコインまでのアプリケーションで使用されています。  
RSAとDiffie-Hellmannでは、ユーザーは自分の秘密鍵を選択してから、対応する公開鍵を計算したことを思い出してください。  
7 1990年代には、pは512〜1024ビット、q 160ビットの範囲になる可能性があります。これは、2001年に1023〜1024ビット[1402]に、2009年に1024〜3072ビットに変更され、qの範囲は160〜256ビット[1403]になりました。  
ムーアの法則やアルゴリズムの進歩が心配な場合は、より大きなサイズを選択できます。  
誰でもあなたの身元へのメッセージを暗号化できる鍵。秘密鍵を使用してこれを復号化できます。  
どちらの場合も、秘密鍵は、中央機関が自分自身だけが知っているシステム全体の秘密鍵を使用して計算されます。  
電子メールアドレスや他の識別子から人々の秘密鍵を計算することはきちんとしたハックのように見えるかもしれませんが、政府部門が再編成されたり名前が変更されたりするとコストがかかる可能性があります[115]。  
  
公開鍵の暗号化とデジタル署名ができるようになったので、ユーザーを鍵にバインドするためのメカニズムが必要です。  
ローレンコーンフェルダーによるより一般的な解決策は、認証局（CA）向けです。  
CAはローカルシステム管理者によって実行される場合があります。ただし、最も一般的には、Verisignなどのサードパーティのサービスであり、そのビジネスは、公開鍵に名前が付けられたプリンシパルによって制御されているかどうかについてデューデリジェンスを行った後、公開鍵に署名することです。  
 （5.1）  
このようにして、信頼できる方法ですべてのプリンシパルに通信する必要があるのは、管理者の公開署名検証キーだけです。  
まず、ネーミングは難しいです。これについては、分散システムに関する次の章で説明します。  
政府のシステムでは、ユーザーの名前や役割だけでなく、セキュリティクリアランスレベルを確立することがよくあります。  
商用システムでは、多くの場合、リモートユーザーを役割ベースのアクセス制御にリンクします。  
システムエンジニアリングのレベルでの認証で失敗する可能性のある他の多くの事柄があります。  
悪い証明書の失効は、それが機能する場合、通常不安定です。  
これらの警告を踏まえて、最も一般的に使用されている公開鍵プロトコルであるTLSを見てみましょう。5.7.5 TLS  
これは本質的にTLSプロトコル（当時はSSLと呼ばれていました）です。  
1995年に両方向の暗号化と認証をサポートするためにPaul KocherとTaher ElGamalによって開発され、httpリクエストとレスポンスの両方を盗聴と操作の両方から保護できるようになりました。  
TLS v1のプロトコルの基本バージョンの簡単な説明は次のとおりです：1。  
サーバーは、名前S、トランザクションシリアル番号S＃、ランダムナンスNS、および公開鍵KSを含む証明書CSを含むサーバーhelloメッセージで応答します。  
クライアントは、サーバーの公開鍵KSで暗号化された事前マスター秘密鍵K0を含む鍵交換メッセージを送信します。  
 これまでのすべてのメッセージで計算されます。  
このキーは、クライアントとサーバーによって送信されたナンスでプレマスターシークレットキーをハッシュすることによって計算されます。K1 = h（K0、NC、NS）  
この時点以降、すべてのトラフィックが暗号化されます。これをクライアントサーバー方向では{...} KCS、サーバーからクライアント方向では{...} KSCと記述します。  
4。  
その後、最終的にデータの送信を開始します。  
2011年にイランはCA Diginotarをハッキングし、2019年にカザフスタンは市民に地元の警察の証明書をブラウザに追加するように強制しました。  
もちろんこれは主権の問題を引き起こします。  
} KCS S！ C：{終了、MAC（K1、everythingtodate）  
  
完全なプロトコルはこれよりも複雑であり、いくつかのバージョンを経てきました。  
これにより、中間者が弱いキーの使用を強制できるダウングレード攻撃が発生しました。  
TLSには双方向認証のオプションもあるので、クライアントが証明書も持っている場合、サーバーがこれを確認できます。  
TLSは、Webトラフィックの暗号化に使用されるだけでなく、Windows 2000以降のWindowsの認証オプションとしても利用できます。企業ネットワークでの認証にKerberosの代わりに使用できます。  
  
SSLの初期のバージョンにはいくつかのバグ[1973]がありましたが、SSL 3.0以降には問題がないようです。 SSL 3.0以降のバージョンはTLS 1.0に名前が変更されました。  
しかし、それ以来20年以上の間に、12を超える深刻な攻撃がありました。  
TLS 1.1は、CBC暗号化とパディングエラーの悪用に対する保護を備えて2006年に登場しました。 TLS 1.2は2年後に続き、ハッシュ関数をSHA256にアップグレードし、認証された暗号化をサポートしました。その間、出現したさまざまな攻撃に対処するパッチがいくつかありました。  
これは、古くなった暗号スイートを拒否し、厳密なトランスポートセキュリティ（STS）などの機能を追加するようにブラウザーを変更する大手サービス会社によって対処されています。  
。  
  
コアプロトコルの最新の主要なアップグレードであるTLS 1.3は、2年間の議論の後、2019年1月にIETFによって承認されました。  
これは、コンプライアンスの目的で監視を行うために暗号化されたセッションを定期的に傍受する銀行業界との間で論争を引き起こしています。  
  
他の何十もの公開鍵プロトコルが広く使用されており、その多くは後で詳しく説明します。  
  
コードサイニングは、人々がソフトウェアをディスケットではなくダウンロードするようになった1990年代に導入されました。  
バージョンNがバージョンN + 1への更新を検証できるようにソフトウェアに公開署名検証キーを含めることは、公開キー暗号化の単純なアプリケーションになると思うかもしれませんが、これは事実とはかけ離れています。  
iPhoneなどの一部のプラットフォームでは、署名されたコードのみが実行されます。これは、ソフトウェアの出所を保証するだけでなく、セクション22.3.2で説明するように、プラットフォームの所有者がアプリを収益化できるようにします。ゲーム機も似ています。  
利用できない場合は、難読化されたコードを使用して検証を行い、マルウェア（または顧客）を困難にすることができます。  
署名キーに関しては、開発者はそれをハードウェアセキュリティモジュールに保持することができます。これは高価で、セクション20.5で説明されている微妙な方法で壊れます。商用CAに戻る信頼の連鎖があるかもしれませんが、政府による法的強制を心配する必要があります11 COVID-19のパンデミックは、少し休息を与えました：マイクロソフトは、2020年春にレガシーバージョンのTLSのサポートを削除する予定でしたが、これを遅らせた。つまり、コード署名は見た目ほど簡単ではなく、特にユーザーが敵である場合にそうです。  
5.7.6.2 PGP / GPG  
暗号活動家Phil Zimmermannがオープンソースの暗号化製品Pretty Good Privacy（PGP）を書いた  
紙の本でソースコードを公開することによる輸出規制。これは、投稿、スキャン、コンパイルが可能です。  
たとえば、sysadmins、コンピューター緊急対応チーム（CERT）  
また、メッセージングをサポートするために犯罪組織に販売されているカスタマイズされた電話にも組み込まれています。これについては、後でセクション25.4.1で説明します。  
署名キーを使用してメッセージに署名したり、目的の各受信者の公開キーを使用してメッセージを暗号化したりするためのコマンドラインオプションがあります。  
1990年代のPGPの導入と使用から多くのことがわかりました。  
12のサブジェクトのうち4つだけが他のサブジェクトに暗号化された電子メールを正しく送信でき、すべてのサブジェクトが少なくとも1つの重大なエラーを発生させました。  
5.7.6.3 QUIC  
これは、サーバーによって暗号化されたクライアントの最後のIPアドレスを保持するCookieによって実現されます。  
グーグルはそれが検索待ち時間を8％、YouTubeのバッファ時間を18％削減すると主張しています。  
一般的な原則として、IBMの1950年代のEBCDICコーディング標準と1970年代のSNA、またはMicrosoftがメール標準とセキュリティの両方を「採用および拡張」しようとするかどうかに関係なく、オープンスタンダードを独自のものに置き換えようとする企業の試みには注意が必要です。 1990年代以降のプロトコル、またはFacebookによるアフリカでのインターネットアクセスの促進により、ユーザーは壁に囲まれた庭にとどまった。  
  
研究者は、特別な特性を持つ多数の公開鍵と署名プリミティブを発明しました。  
しきい値暗号は、署名鍵または復号鍵をn個のプリンシパル間で分割できるメカニズムであり、nのうちの任意のkがメッセージに署名（または復号）できるようにします。  
k = nの場合、構成は簡単です。  
。  
+ dn。  
 [554]。  
ブラインド署名は、メッセージが何であるかを知らずにメッセージに署名を付ける方法です。  
、それを計算する署名者に渡します（ReM）  
。  
ここで、なぜ誰かがその内容を知らずにドキュメントに署名したいのかと尋ねるかもしれませんが、いくつかのアプリケーションがあります。  
銀行は、公開鍵（e、n）を使用して正しいことが検証された署名が付けられた、一意のシリアル番号と指定された形式の冗長性を持つ任意の文字列Mを10ドルで受け取ることに同意する場合があります  
ブラインド署名プロトコルにより、顧客は銀行にシリアル番号を知らなくても銀行にコインに署名してもらうことができ、プロトタイプの道路通行料システムで使用されました。  
デジタルキャッシュの主な問題は、同じコインを2回使う人を検出することでした。これは、セクション20.7で説明したように、ブロックチェーンまたは他の元帳メカニズムを使用して最終的に修正されました。  
匿名のデジタル認証情報が認証に使用されるようになりました。PCマザーボードのTPMチップは、ユーザーを特定せずに、マシンで実行されているソフトウェアについて何かを証明する場合があります。  
 つまり、侵害された単一のデバイスがエコシステム全体を破壊します。  
  
動機？  
楕円曲線システムはこの限界を達成するように見えます。 256ビットの楕円スキームは、128ビットキーを使用した128ビットブロック暗号とほぼ同じくらい解読するのが難しい場合があります。 NSAの軍事アルゴリズムスイートBで使用されている唯一の公開鍵暗号化方式は、384ビットの楕円曲線システムです。  
その結果、楕円曲線暗号システムはより高速になります。  
2019年の第3版までに、ビットコインマイナーは10分ごとに68ビットハッシュの衝突を発見しており、RSA-768が因数分解されており、エドスノーデンはNSAが1024ビットプライムモジュラスの離散ログを実行できると語っています。  
Peter Shorは、十分に大きな量子コンピュータを構築できれば、因数分解と離散対数計算の両方が容易になることを示しました[1725]。  
私は懐疑的です（多くの物理学者がそうであるように）  
私は量子暗号の価値についてさらに懐疑的です。一括暗号化にAESを使用するライン暗号化デバイスの鍵を再生成できる場合がありますが、その方法はすでにわかっています。  
理論物理学は、1970年代初頭、ジェラルド・フフトがヤン・ミルズのくりこみ可能性を証明することによって標準モデルを完成させて以来、行き詰まっています。量子情報理論は最新の熱意です。  
しかし、 't Hooftのセルラーオートマトンモデル[916]やGrisha Volovikの超流動モデル[1967]などの別の解釈は、ベルテストが超流動の秩序パラメーターのように、量子真空における長距離秩序の存在を実証するだけであることを示唆しています。 。  
量子暗号システムについて主張された証明にそのような抜け穴があることを指摘すると、支持者の唯一の答えは個人的な虐待であるように見えます。  
これらの曲線には多くの構造があります。彼らは、世界で最も賢い純粋な数学者によって集中的に研究されています。小さな特性の曲線用のより良い離散対数アルゴリズムが2013年に発見されました[168]。そして、NSAは楕円曲線暗号の使用から離れているようです。  
2020年に、NISTは量子暗号化後の標準化プロセスの提出物の第3ラウンドのパブリックレビューを開始しました。  
1つ以上のアルゴリズムが選択され、標準化されるため、それらを使用する暗号スイートは、アップグレードとしてTLSなどのプロトコルにドロップできます。  
楕円対数が容易になれば、これらのリソースが手に入り、プライムフィールドの離散ログやRSAにフォールバックすることもできます。  
したがって、惑星の将来を気にする数学者は、楕円対数問題について考えるよりも悪い結果をもたらすかもしれません。  
5.7.9他に何がうまくいかない  
実際、20世紀に設計されたシステムへの攻撃があり、主に輸出管理規則、無知な設計、またはその両方によって短すぎたキーが含まれていました。  
後の章で、暗号戦争とその輸出管理規則がドアロックへの攻撃をもたらした例を示します（セクション13.2.5）。  
 および著作権の執行（セクション24.2.5）  
最近のほとんどの攻撃はこの実装を悪用しています。  
不十分な乱数ジェネレーターは、他の多くの失敗を引き起こしました：共通の要因を持つRSAキー[1140]、離散ログの予測可能なシード[1676]など。  
過去20年間に大幅な変更を余儀なくされた暗号実装への実際の攻撃の多くは、タイミングや電力分析などのサイドチャネルを悪用しています。私はこれらに19章を捧げます。  
第20章では、Signalメッセージングプロトコル、TOR匿名システム、暗号通貨など、公開鍵メカニズムを複雑な方法で使用して興味深い創発的なプロパティを取得する多くのシステムについて説明します。  
これらには興味深い障害モードもありますが、サイドチャネルに関連するものもありますが、すべてではありません。  
  
多くの暗号は悪用されているため失敗します。そのため、セキュリティエンジニアは、さまざまな種類の暗号が何をするのかを明確に理解する必要があります。  
次のレベルは、AESなどの個々の暗号、または公開鍵暗号システムとデジタル署名メカニズムの基礎となる数論メカニズムの設計です。  
次のレベルは実装の悪さを含み、これははるかに扱いにくく、厄介です。  
実際のシステムの内部でピアリングを行い、対称鍵アプリケーションのブロック暗号を置換と順列の慎重な組み合わせによって構築する方法について説明しました。公開鍵暗号化やデジタル署名などの非対称アプリケーションでは、数論を使用します。  
他の種類の暗号（ストリーム暗号とハッシュ関数）は、適切な操作モードで使用することにより、ブロック暗号から構築できます。  
ポピュラーな暗号ライブラリはプログラマーに危険なデフォルトを公開することにより不適切な操作モードを使用するように促すため、多くのシステムが失敗します。  
サイドチャネル攻撃から貧弱な乱数発生器まで、うまくいかない可能性のある他の多くのものがあります。  
 また、暗号メカニズムがアクセス制御や物理的セキュリティなどの他の手段とうまく統合されている場合。  
教訓は、自分で転がしてはいけないということです。独自のプロトコルや暗号を設計しないでください。絶対に必要でない限り、独自の暗号コードを記述しないでください。  
;あなたは関連する専門家の資料を読み、専門家に話し、有能な人々がそれを破ろうとする必要があります。  
暗号の設計はチェーンソーのジャグリングに少し似ています。致命的なエラーを発生させるのは簡単すぎます。それらの多くは、暗号が数学の特定の分岐（数論、代数幾何学、複雑性理論、組み合わせ論、グラフ理論、および情報理論）と出会うところです。  
ビジネスの経験的な終わりは、暗号化、署名、および複合操作のためのプリミティブの設計に関係しており、利用可能なプラットフォーム上で適切に機能します。  
理論的には何が起こっているのかを理解する最良の方法は、Crypto、Eurocrypt、Asiacryptなどの過去数年間の研究会議の議事録を読むことです。暗号設計に関する作業はFast Software Encryptionに掲載されています。実装への攻撃はしばしばCHESに現れます。一方、暗号がシステムでどのように使用されるかに対する攻撃は、IEEEのセキュリティとプライバシー、CCS、Usenixなどの上位のシステムセキュリティ会議で見つかります。  
Bruce SchneierのApplied Cryptography [1667]は、非数学者が理解できるレベルで多くの根拠をカバーしており、米国の輸出管理法にも関わらず、1990年代に暗号コードを入手しましたが、現在は少し古くなっています。  
カッツとリンデルは、数学のために生徒に読んでもらう本です。  
 しかし、日付も付けられています。たとえば、[1022]のように、GCMについて言及していません。  
差分暗号解読に関する聖書は、発明者のEli BihamとAdi Shamir [245]によるものであり、線形および差分暗号解読に関する優れた短いチュートリアルは、Howard Heys [895]によって書かれました。  
動作モードに関する最初の本は、Carl MeyerとSteve Matyas [1301]によるものです。  
過去20年間のTLSへの実際の攻撃については、Christopher MeyerとJoerg Schwenkによる調査報告[1302]と、この本の後半のサイドチャネルに関する章を参照してください。  
ランダム性とアルゴリズムへのより完全ではありませんがより読みやすい導入は[835]にあります。  
暗号学の歴史は魅力的であり、多くの古い問題が再発し続けています。  
メリーランド州フォートジョージミードにあるNSA博物館も一見の価値がありますが、おそらくイギリスのブレッチリーパークにある博物館が一番でしょう。  
しかし、EllisがGCHQで働いていたため、彼の仕事は分類されたままでした。  
この話は[625]で語られています。  
そして、分類されたコミュニティはデジタル署名を事前に発明しませんでした。彼らはホイット・ディフィーとマーティン・ヘルマンの業績のままです。