共通鍵暗号

渡辺 大†

1. まえがき

暗号技術の起源は紀元前にまで遡り、主に軍事目的で、秘密裏に情報をやりとりする技術として発達してきました。しかし、昨今では、インターネットを中核とするIT技術の普及に伴い、私たちの日常を支える技術として発展を続けています。

本講座でも、これまで暗号技術にまつわるトピックが何度か取り上げられていますが、これから数回に分けて暗号技術とはどのようなものなのか、もう少し掘り下げて見ていきたいと思います。今回はその第一歩として、昔からある共通鍵暗号の現在について紹介します。

共通鍵暗号の用途は、ざっくり言うと「データの保護」です。では、データの保護とはどういうことでしょうか？もちろん、「データがなくならないように保管する」という観点もありますが、セキュリティ的な観点からすれば

- (1) データを共有すべき人以外にはデータが見えないようにしたい (情報の秘匿)
 - (2) データが勝手に書換えられては困る (情報の完全性保証)
- という2点が主な課題になります。これらの課題をどのように解決するか、ここから議論したいと思います。

2. 共通鍵暗号と公開鍵暗号

暗号技術は、共通鍵暗号と公開鍵暗号の2つに分けられ

ます。共通鍵暗号とは、通信する二者間で同じ秘密鍵を共有することで、秘密通信を実現します。一方の公開鍵暗号は、暗号化で使う鍵と復号に使う鍵が異なるのが特徴です。

共通鍵暗号は古来より使われてきた暗号化方式ですが、「どうやって秘密鍵を共有するか？」という課題があります。軍事目的で使われていた頃は、通信相手が特定できていたため、出発前にあらかじめ秘密鍵を渡しておく、という方法でこの問題を解決していました。しかし、インターネットのように不特定多数との通信を前提とする社会では、この方法は使えません。この課題を解決するのが公開鍵暗号です。

公開鍵暗号では、データを暗号化するときを使う鍵 (公開鍵と呼びます) と暗号化されたデータを復号するときを使う鍵 (秘密鍵と呼びます) は、2つで対となる、異なる値です。また、公開鍵から秘密鍵を推定することが難しいという特徴があります。ある人 (アリスさんとします) が対となる公開鍵と秘密鍵を生成したとします。そして、暗号化に使う公開鍵だけ公開します (これが名前の由来です)。この公開により「誰でも暗号化できる」ようになります。ただし、「復号できるのは秘密鍵を持っているアリスだけ」というシステムができます。

このように、公開鍵暗号は共通鍵暗号に比べていろいろな使い方ができるのですが、残念ながら、処理が複雑なために暗号化/復号処理に時間がかかってしまうというダメ

図1 ハイブリッド暗号

† 株式会社日立製作所横浜研究所

"Security Technologies on Image Information (6): Symmetric Key Encryption" by Dai Watanabe (Yokohama Laboratory, Hitachi, Ltd., Yokohama)

リットがあります。そこで、鍵共有には公開鍵暗号を、データが大きいコンテンツの暗号化には共通鍵暗号を、という感じで用途に応じて暗号の種類を使い分けるのが一般的な利用法です。図1に共通鍵暗号と公開鍵暗号をハイブリッドに使用した、メッセージの伝送方法を示します。

3. データの暗号化と情報の秘匿

前者の課題を解決する基本的な手段はデータを暗号化して、秘密鍵を持っている人だけが読めるようにすることです。歴史上でも、紀元前から暗号が使われていたことが知られており、スパルタのスキュタレー暗号や、ローマのシーザー暗号が有名です。ちなみに、スキュタレー暗号は、文章を共有したい人が同じ太さの棒(スキュタレー)を持ち、棒にでたらめに見える文字列が書かれた革ひもを巻きつけると、棒の長さ方向に文章となる方式です。図2にその例を示します。ひもには「いちよろろりたむちはぬ…」という訳のわからない文字列が書かれています。この紐を棒に巻くと「いろはにほへと…」と読めるわけです。少し専門的に言うと、一種の転置処理であり、文字の順番を入れ替えて読むことで元の平文がわかるわけです。棒の太さが鍵になっています。シーザー暗号は、文章を共有したい人がある数値を共有し、アルファベットをその数値分だけずらした(一種の換字処理をした)文章を送受する方式です。送り手がアルファベットを3つずらせば、受け手は反対方向に3つずらせば平文を読むことができます。このずらす幅が鍵となっています。図3にひらがなを利用したシーザー暗号の例を示します。ここでは3文字ずらすことが鍵となっており、「はなのいろはうつりにけりな…」という文章が「へうやにほへおらをとえをう…」という暗号文になっています。

現在では、データの暗号化を行う技術はより高度化しています。暗号化方式を大きく分けるとストリーム暗号とブロック暗号の二つになります。ストリーム暗号はデータを乱数でマスクする方法です。ブロック暗号は、データをブロック単位に区切り、秘密の置換テーブルを使って別の値に置換する方法です。1ブロックの大きさ(ブロック長)は、通常8バイトか16バイトです。ここで「置換テーブルを使って変換する」と書きましたが、ブロック長が8バイトの場合

図2 スキュタレー暗号

図3 シーザー暗号

でも、置換テーブルの大きさは($2^{64} \times 8 \times 2 =$) 256 EB (エクサバイト: 1 EB=10¹⁸ バイト)になってしまいます。現実には変換テーブルをメモリー上に持っておくかわりに、特定のアルゴリズムにしたがって、テーブルの値を毎回計算しています。

はじめての商用暗号(アルゴリズム公開型暗号)であるDES(Data Encryption Standard)¹⁾のインタフェースは、ブロック長64ビット、鍵長56ビットです。開発当時から、DESの鍵長は安全性の観点から短すぎるのではないかと、という懸念がありました。計算機の処理能力が大きくなるにつれ、この懸念が大きな問題になってきます。そこで、RSA社は、1997年からDESの解読コンテストを開催し、鍵の総当りがどれくらい現実的であるかを明らかにしようとなりました。表1に解読コンテストの結果を示します。初回のチャレンジでは、7万台のPCを使って鍵の総当たり探索を行い、96日で解読に成功しました。また、2年後の1999年に開催された第3回のチャレンジでは、10万台のPCに加え、

表1 ブロック暗号の解読コンテスト

対象暗号	鍵長	コンテスト名	解読日	解読時間	計算量
DES	56ビット	DES Challenge	1997/1	96日	PC約7万台
		DES Challenge II-1	1998/1	39日	PC約5万CPU
		DES Chal	1998/7	56時間	専用ハードウェアDES Cracker (制作費25万ドル)
		DES Challenge III	1999/1	22時間15分	DES Cracker+ PC 10万台
RC5-64	64ビット	RC5-64 Secret-Key Challenge	2002/9	約4年	PC約7万台
SHA-256*	—	ビットコインの採掘	2015/3時点	2 ⁵⁸ 回/秒	—

*現在の計算リソースを過去のコンテスト時と現在の計算リソースとを比較するために表に掲載しました。

SHA-256の安全性が損なわれているわけではありません。

専用の解読用ハードウェアを使って、解読時間を22時間まで短縮しています。かなりの計算コストではありますが、DESチャレンジは、DESを安全に使える時代の終わりを告げるものでした。

余談になりますが、最近では、ネットとコンピュータを用いて一儲けを考えている人たちの間で、ある膨大な計算を行い、解答を出すことで、ネット上でやり取りされている仮想通貨であるビットコインを大量に入手できる（“金脈を掘り当てる”と言われていています）、いわゆる“採掘”を目的として暗号処理（ハッシュ関数SHA-256）の計算が大量に行われています。ビットコインに関する最新の統計情報を提供しているBlockchain.info²⁾を見れば、現時点でどれくらいの暗号計算ができるのか、簡単に調べることができます。試しに、採掘状況を調べてみると、なんと1秒にハッシュ関数の計算を2⁵⁸回以上実行していました（2015年3月19日時点）。これは、DESの鍵を1秒に4個解読できてしまう、ということの意味しています。金の採掘を目的とする欲望の前では、56ビットというDESの鍵が提供する安全性がいかに脆弱なものなのか、おわかりいただけるのではないのでしょうか。

DESの延命を図る技術として、Triple-DESという方式が導入されました¹⁾。これは、DESの暗号化処理を3回繰り返す、というもので、2つの鍵を使うものと、3つの鍵を使うものがあります。処理速度を犠牲にして、安全性を改善するというわけです。

しかし、より根本的な解決を目指すべく、DESチャレンジ

ジと時を同じくして、米国標準技術研究所（NIST: National Institute for Standards and Technology）は、DESの後継を担う暗号の選定プロジェクトを開始しました³⁾。次世代標準暗号（Advanced Encryption Standard: AES）の選定は、プロセスの透明性を重視し、オープンなコンペティション方式で行われました。コンペティションには、世界中から21のアルゴリズムが応募され、3年をかけて審査が行われました。審査の内容は、安全性の観点での評価に加えて、ICカードやPC、ハードウェアを含むさまざまなプラットフォームにおける実装性能評価が行われました。安全性評価についても、当初想定されていた以上に厳しい審査が行われ、評価技術の向上にも繋がりました。このように、多大なコストをかけて行われたプロジェクトの結果、当時ベルギーの学生であったRijmen氏とDaemen氏が提案したRijndaelがAESに選ばれました。余談ですが、Rijndaelという名称は、提案者の名前を繋げたものなのだそうです。AESの標準仕様は2001年に文書化され⁴⁾、現在に到るまでインターネットで使われている暗号の主流となっています。

4. ブロック暗号 (AES) のアルゴリズム

それでは、共通鍵暗号の代表格であるAESの暗号化アルゴリズムを簡単に紹介しましょう。AESはブロック長が128ビットで、DESのブロック長の2倍あります。また、鍵長は128ビット、192ビット、256ビットの3種類が用意されています。

次に処理構造を見ていきましょう。図4にブロック暗号の

図4 ブロック暗号の基本構造

基本構造を示します。ブロック暗号の暗号化関数は、一般的にデータランダム化部と鍵スケジュール部に分けられます。データランダム化部は、ラウンド関数と呼ばれる処理の繰り返しになっています。ラウンド関数は、毎回異なるラウンド鍵を使いますが、秘密鍵からラウンド鍵を生成するのが鍵スケジュール部です。一般には、ラウンド処理の回数が多いほど、より安全になると考えられており、AESは鍵長が128ビットの場合は10ラウンド、鍵長が256ビットの場合は14ラウンドの処理を行います。

ブロック暗号の設計でキモとなるのが、ラウンド関数です。AESのラウンド関数は、128ビットの入力に対し、SubBytes, ShiftRows, MixColumns, AddRoundKeyという4つの処理が順番に行われ、128ビットのデータを出力する関数です。それぞれの処理を図5に示します。

いずれもバイト単位の処理になっていることが、AESの特徴の1つになっています。SubBytesは、1バイトデータの置換表(S)を使って、128ビット(16バイト)の値を1バイトずつ置き換えます($x_{i,j} \leftarrow S[x_{i,j}]$)。ShiftRowsは、バイトの位置を入れ替えます($x_{i,j} \leftarrow x_{i,j+i \bmod 4}$)。つまり、SubBytesはシーザー暗号に代表される換字処理、ShiftRowsはスキュタレー暗号に代表される転置処理を行っているわけです。しかし、これだけではバイト単位の出現頻度分布が変わりませんので、次のMixColumnsで、4バイトを一括で線形変換($f_i()$)する多文字変換処理を行います($x_{i,j} \leftarrow f_i(x_{0,j}, x_{1,j}, x_{2,j}, x_{3,j})$)。ShiftRowsとMixColumnsの組合せは、2ラウンドの処理を行うと、1バイトの入力値が16バイト全体に影響するように設計されています。また、これら3つの処理は公知のもので、誰でも逆変換を構成できてしまいます(つまり、解読できます)。そこで、最後にAddRoundKeyでラウンド鍵を排他的論理和し、秘密情報を混ぜ込みます($x_{i,j} \leftarrow x_{i,j} \oplus r_{i,j}$: ただ

し、 $r_{i,j}$ はラウンド鍵、 $\oplus$ は排他的論理和演算を示します)。このラウンド鍵は鍵スケジュール部に入力された秘密鍵を用いて生成されます。

このように、AESを構成する処理をひとつひとつ見ていくと、昔の暗号と大きく変わっているというわけではありません。しかし、置換表の作り方や線形変換の選び方に関する知識は大幅に増え、良い要素を選ぶ理論も確立されています。

5. モード：ブロック暗号の使い方

さて、ブロック暗号を実際に使うことを考えましょう。AESを使うならば、まずデータを16バイトごとに区切ります。データ長は16バイトの倍数とは限りませんから、何らかの数値を追加して、16バイトの倍数にします。

次に、AESを使って順に暗号化すれば良いように思えます。この暗号化方式はECB (Electric Code Book)⁵⁾モードと呼ばれています。図6にECBモードのブロック図を示します。 E_K が共通鍵暗号のブロックで、鍵Kが使用されることを示します。Pnが入力の平文、Cnが出力となる暗号文を示します。

ECBは、任意長のデータを暗号化できますが、同じデータが連続するような場合には、平文の情報が漏えいしてしまうかもしれません。1ブロック=16バイトもあるので、なかなか同じ入力が続くことは考えにくいかもしれませんが、例えば、図7は、冗長性が高い画像ファイルをECBモードで暗号化してみた結果です。暗号化されていても、

図6 ブロック暗号のECBモード

図7 暗号化処理の結果
(Wikipedia「暗号利用モード」より転載)

図5 AESのラウンド関数を構成する4つの処理

図8 ブロック暗号のCBCモード

図9 CBC-MACモード

ペンギンの輪郭が確認できるのではないのでしょうか。

ECBの問題点を解決するために考案されたのがCBC (Cipher Block Chaining)⁵⁾モードと呼ばれる暗号化方式です。図8にCBCモードのブロック図を示します。IVは初期ベクトル (Initial Vector) であり、P_nと同じサイズの乱数です。

CBCモードでは、暗号文ブロック (ほとんど乱数です!) を次の平文ブロックと排他的論理和 (⊕) することで、ブロック暗号への入力に常に変化し続けるようになっており、ECBモードのような問題は起こりません。CBCモードは単純な仕掛けながら安全性が高いということで、DESやAESとセットで広く使われてきた暗号化方式のモードです。しかしながら、本講座の第3回⁶⁾でも触れられていましたが、「パディングオラクル攻撃」という攻撃を使うと、CBCモードで暗号化されたデータを復号できる可能性があることが2002年にわかりました⁷⁾。これは、CBCモードが弱い、というよりはCBCモードの使い方が間違っていることに起因する脆弱性なのですが、昨今では、TLSなど標準暗号プロトコルの安全性を脅かすようになり、CBCモードの継続使用には黄信号が灯っている状態です。

6. メッセージ認証コード

ところで、パディングオラクル攻撃では、暗号文を改ざんしてサーバの応答を観測することで、暗号文を復号していきます。したがって、改ざんを検出するメッセージ認証技術 (正しく) 使うことで、パディングオラクル攻撃を防ぐことができます。メッセージ認証技術の基本は誤り検出に用いる巡回冗長検査 (CRC: Cyclic Redundancy Check) と同じです。まず、事前にデータの特徴値を抽出し、データに付加しておきます。データが改ざんされているかチェックするときは、データの特徴値を改めて計算し、事前に計算しておいた値と一致するかを確認します。もちろん、CRCがデータの破損チェックに使えるのは「データが壊れているのに、『たまたま』CRCは一致するなんてことはないだろう」という前提があります。ところが暗号解読者の視点からすれば、「意図的に」CRCの値が一致するようにデータを改ざんするのは簡単です。そこで、特定の人の計算できないような特徴値の計算方法として、メッセージ

図10 CBC-MACの脆弱性

認証コード (MAC: Message Authentication Code) が必要になります。MACの簡単な構成例として、ブロック暗号を使うCBC-MACがあります。図9にCBC-MACのブロック図を示します。Tagが最終的に出力される認証用のコードとなります。名前からもわかるように、CBC-MACは暗号化に用いるCBCモードをMACに流用しています。

CBC-MACは、固定長のデータを取り扱う分には安全であることが理論的に証明されていますが、いろいろな長さのデータを取り扱う場合には、安全性が損なわれる可能性があります。例えば、2つのデータをつなぎ合わせて新しいデータを偽造する攻撃に弱いことが知られています。図10で攻撃方法を説明してみましょう。具体的には、MACを生成するための鍵を持たない攻撃者が、未知のメッセージとTagのペアを生成できます。まず、何らかの方法で、攻撃者が事前に2つのメッセージM1=(P1, P2)とM2=(P3, P4)に対する認証コードT1とT2を入手したとします。(M1, T1)と(M2, T2)のメッセージとタグのペアを出力しても、これはすでに知っているもので、攻撃とはなりません。次に、M=(P1, P2, T1⊕P3, P4)というタグの値を知らない未知のメッセージを考えてみましょう。3ブロック目の「T1⊕P3」が攻撃のキモになっています。(P1, P2)に対す

図11 CMACモード

る認証コードがT1であることがわかっていますので、3ブロック目の暗号化処理の入力は $T1 \oplus (T1 \oplus P3) = P3$ になります（同じ値T1の排他的論理和 $T1 \oplus T1$ はゼロになります）。したがって、(P1, P2, $T1 \oplus P3$, P4)に対する認証コードはT2になり、攻撃者は未知のメッセージMとタグT2のペア(M, T2)を偽造できたことになります。

このような脆弱性を避けるため、現在では、CBC-MACを改良したCMAC⁸⁾が標準技術になっています。図11にCMACのブロック図を示します。K'は副鍵と呼ばれ暗号化 E_K に使われる鍵とは別の鍵です。

K'が攻撃者に知られない限り、CBC-MACと同じ攻撃はできなくなっています。

この他によく使われるMACとしては、ハッシュ関数を利用するHMAC⁹⁾があります。

7. ストリーム暗号、ハッシュ関数

本稿では、ブロック暗号を中心に解説を進めてきました。この他に、共通鍵暗号に分類される技術として、ストリーム暗号とハッシュ関数があります。ストリーム暗号は、データと（擬似）乱数を排他的論理和する、という単純な処理で暗号化を実現しています。ブロック暗号のように「なんにでも使える」という便利さはありませんが、暗号化処理がビット単位で実行できるので、①半端な長さのデータを暗号化するのに向いている、②エラーが他のビットに伝播しない、という特徴があり、昔から無線通信の暗号化に使われてきました。AESが普及した現在でも、携帯電話の標準3GPP (Third Generation Partnership Project) ではストリーム暗号を多く利用しています。ちなみに、昨年、WEPやSSL/TLSの脆弱性の一因を担っているRC4もストリーム暗号の一種です。

ハッシュ関数は、MACと同じくデータの特徴値を抽出する技術で、方向性を持っていることが特徴です。方向性とは「出力から入力を特定できない」ことを意味しています。通常の暗号技術でも「鍵を知らない限り」暗号文から平文の情報を得ることはできませんが、ハッシュ関数は、鍵なしで方向性を実現しているという点で、暗号技術の中でも異色の存在です。ハッシュ関数は前述のメッセージ認証にも使えます。CMACでは、安全性を確保するために、

最後のブロックでもう1つ別の鍵を使いましたが、ハッシュ関数を使ってメッセージ認証を行うHMACモードでも同じようなテクニックで「2つのデータをつなぎ合わせる攻撃」を防いでいます。この他にも、ハッシュ関数は、公開鍵暗号の基本技術である電子署名や公開鍵証明書でも使われており、複雑な暗号方式の構成では必須の要素技術になっています。

標準のハッシュ関数として、Rivestが開発したMD5¹⁰⁾とNSA (National Security Agency)が開発したSHA-1¹¹⁾が長らく使われてきましたが、2004年頃にこの2つのアルゴリズムに対する攻撃が発表されました^{12) 13)}。特に、MD5に対する攻撃は深刻で、2009年には認証局のルート証明書が偽造されるという事件に発展しました¹⁴⁾。幸いにも、NISTは、これに先立ってDES、SHA-1といった旧来のアルゴリズムをAES、SHA-2¹¹⁾に置き換えていく暗号移行プログラムを推進していました。置き換えるアルゴリズムがないまま立ち往生する最悪の可能性は避けることができたのです。また、NISTは、SHA-2が安全ではなかった場合を想定して、SHA-3を準備することにしました¹⁵⁾。この選考は、AESコンペティションを踏襲したスタイルで2008年から約5年をかけて行われました。筆者もLuffaというアルゴリズムを応募し、1次選考は通ったのですが、残念ながら、最終選考に進むことができませんでした。SHA-3には、AESの開発者でもあるDaemen氏が率いるチームが提案したKeccakが選ばれました。

8. 共通鍵暗号技術に関する標準

最後に、標準化されている共通鍵暗号を少し紹介しておきましょう。表2を参照してください。もっとも包括的に暗号技術をまとめているのはISO (International Organization for Standardization)で、ブロック暗号ではTriple DESやAESの他にもMISTY1やCamelliaが、ストリーム暗号ではSNOW 2.0やMUGIが収録されています^{16) 17)}。その一方で、おそらくもっとも参照されて、実際に利用されているのは米国のNIST標準暗号でしょう。NISTはブロック暗号として3-key Triple DESとAES、ハッシュ関数としてSHA-1とSHA-2だけを標準化しています。そして、擬似乱数生成やTLSなどの暗号プロトコルで利用される鍵の伸長など、共通鍵暗号技術に分類される機能はブロック暗号とハッシュ関数のモードを利用するように規定されています。日本や欧州でも、同様に安全な暗号アルゴリズムを選定するプロジェクト(CRYPTREC¹⁸⁾、NESSIE¹⁹⁾)がAESコンペティション後に始まり、独自のラインナップを描いています。

9. むすび

本稿では、広く使われているAESとその周辺に話題を絞って、共通鍵暗号技術を紹介してきました。暗号というと、とかく「難しい」、「わからない」と敬遠されがちですが、

表2 標準暗号

	64ビットブロック暗号	128ビットブロック暗号	ストリーム暗号	ハッシュ関数	メッセージ認証コード
ISO	TDEA (triple DES) MISTY1 CAST-128 HIGHT	AES Camellia SEED	MUGI SNOW 2.0 Rabbit Decim v2 KCipher-2	RIPEMD-128 RIPEMD-160 SHA-1 SHA-256 SHA-284 SHA-512 WHIRLPOOL	
NIST	3-key triple DES	AES	—	SHA-1 SHA-256 SHA-384 SHA-512	CMAC HMAC
CRYPTREC (推奨暗号)	3-key triple DES	AES Camellia	KCipher-2	SHA-256 SHA-384 SHA-512	CMAC HMAC
CRYPTREC (推奨候補暗号)	CIPHERUNICORN-E Hierocrypt-L1 MISTY1	CIPHERUNICORN-A CLEFIA Hierocrypt-3 SC2000	Enocoro-128v2 MUGI MULTI-S01	—	PC-MAC-AES
NESSIE	MISTY1	AES Camellia SHACAL-2*	—	Whirlpool SHA-256 SHA-384 SHA-512	UMAC TTMAC EMAC HMAC

* SHACAL-2は256ビットブロック暗号に分類される。

スマート化が進む社会では、日常的に暗号を利用したサービスが提供されています。本稿を通して、暗号が少しでも皆さんの身近になれば、これに勝る喜びはありません。

(2015年4月1日受付)

〔文 献〕

- 1) National Institute of Standards and Technology: "Data Encryption Standard (DES) : specifies the use of Triple DES", Federal Information Processing Standards Publication 46-3, Oct. 25, 1999 (Withdrawn: May 19, 2005)
- 2) <https://blockchain.info/>
- 3) National Institute of Standards and Technology: "Advanced Encryption Standard (AES) Development Effort", <http://csrc.nist.gov/archive/aes/>
- 4) National Institute of Standards and Technology: "Advanced Encryption Standard", Federal Information Processing Standards Publication 197 (Nov. 2001)
- 5) National Institute of Standards and Technology: "Recommendation for Block Cipher Modes of Operation - Methods and Techniques", NIST Special Publication 800-38 A (Dec. 2001)
- 6) 神田雅透: "SSL/TLSの仕組みを知っていますか?", 映像学誌, 69, 3, pp.228-233 (2015)
- 7) S. Vaudenay: "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ...", Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, 2332, pp.534-546, Springer (2002)
- 8) National Institute of Standards and Technology: "Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication", NIST Special Publication 800-38B (May 2005)
- 9) National Institute of Standards and Technology: "The Keyed-Hash Message Authentication Code (HMAC) ", Federal Information Processing Standards Publication 198-1 (July 2008)

- 10) R. Rivest: "The MD5 Message-Digest Algorithm", IETF RFC 1321 (1992)
- 11) National Institute of Standards and Technology: "Secure Hash Standard (SHS) ", Federal Information Processing Standards Publication 180-4 (May 2012)
- 12) X. Wang and H. Yu: "How to break MD5 and other hash functions", Advances in Cryptology, Eurocrypt2005, Lecture Notes in Computer Science, 3494, pp.19-36, Springer (2005)
- 13) X. Wang, L. Yin and H. Yu: "Finding collision in the full SHA-1", Advances in Cryptology, CRYPTO2005, Lecture Notes in Computer Science, 3621, pp.17-37, Springer (2005)
- 14) A.K. Lenstra, D. Molnar, D.A. Osvik, B. de Weger, M. Stevens, A. Sotirov and J. Appelbaum: "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate", Advances in Cryptology-CRYPTO 2009, Lecture Notes in Computer Science, 5677, pp.55-69 (2009)
- 15) National Institute of Standards and Technology: "Cryptographic Hash and SHA-3 Standard Development", <http://csrc.nist.gov/sgroups/ST/hash/>
- 16) ISO/IEC 18033-3:2010, Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers
- 17) ISO/IEC 18033-4:2011, Information technology - Security techniques - Encryption algorithms - Part 4: Stream ciphers
- 18) CRYPTREC: Cryptography Research and Evaluation Committee, <http://cryptrec.go.jp/>
- 19) NESSIE: New European Schemes for Signatures, Integrity and Encryption, <https://www.cosic.esat.kuleuven.be/nessie/>

渡辺 大 1996年、東北大学理学研究科数学専攻博士前期課程修了。1999年、(株)日立製作所に入社。システム開発研究所に配属。現在は同横浜研究所に従事。共通鍵暗号の設計および安全性評価業務に従事。