

クラウドを支えるこれからの暗号技術

光成滋生

2015年3月23日

目次

はじめに	viii
第Ⅰ部 暗号の基礎	1
第1章 共通鍵暗号	4
1.1 暗号	4
1.2 古典的な方法	4
1.3 Vernam 暗号と情報理論的安全性	5
1.4 共通鍵暗号	6
1.5 ブロック暗号と計算量的安全性	7
1.6 暗号の危険化	7
1.7 計算量の指標	8
1.8 決定的アルゴリズムの問題点	9
1.9 CBC モードと初期化ベクトル	9
1.10 パディングオラクル攻撃	10
1.11 この章のまとめ	11
第2章 亂数	12
2.1 擬似乱数	12
2.2 暗号論的擬似乱数	13
2.3 擬似乱数とストリーム暗号	13
2.4 CTR モードとストリーム暗号	14
2.5 Intel CPU の <code>rdrand</code> 命令	15
2.6 この章のまとめ	16
第3章 公開鍵暗号	17
3.1 鍵の共有	17
3.2 余りの世界	17

3.3	Diffie-Hellman 鍵共有	19
3.4	鍵共有のための仮定と条件	20
3.5	CDH 仮定	20
3.6	DH 鍵共有の注意点	21
3.7	巾乗の計算	22
3.8	バイナリ法	23
3.9	公開鍵暗号	24
3.10	余りの世界再び	25
3.11	ElGamal 暗号	26
3.12	強秘匿性と DDH 仮定	27
3.13	ElGamal 暗号の弱点	28
3.14	より強力な攻撃	28
3.15	攻撃と安全性のまとめ	30
3.16	RSA 暗号	31
3.17	公開鍵基盤と認証局	32
3.18	PKI への攻撃	34
3.19	この章のまとめ	35
第 4 章 認証		36
4.1	ハッシュ関数	36
4.2	ハッシュ関数の構成	38
4.3	メッセージ認証符号	39
4.4	HMAC	40
4.5	認証付き暗号	42
4.6	デジタル署名	42
4.6.1	RSA-FDH 署名	43
4.6.2	DSA	43
4.7	ブラインド署名	44
4.8	秘匿共通集合計算	46
4.9	部分ブラインド署名	47
4.10	この章のまとめ	48
第 5 章 楕円曲線暗号		49
5.1	楕円曲線暗号	49
5.2	2 次元の世界へ	50
5.3	ドラクエ 3 の世界	51

5.4	楕円離散対数問題	53
5.5	楕円 DH 鍵共有と楕円 ElGamal 暗号	54
5.6	楕円曲線暗号の利点	55
5.7	ECDSA	55
5.8	前方秘匿性	56
5.9	この章のまとめ	58
第 6 章	群	59
6.1	演算の抽象化	59
6.2	群の定義	60
6.3	巾乗	61
6.4	巡回群	62
6.5	群に対する離散対数問題	63
6.6	離散対数問題に対する攻撃	63
6.7	巡回群の位数	65
6.8	この章のまとめ	66
第 II 部	新しい暗号技術	67
第 7 章	ペアリングと ID ベース暗号	70
7.1	ペアリング	70
7.2	ID を使った鍵共有	71
7.3	3 者間 DH 鍵共有	72
7.4	双線形と非対称	73
7.5	対称ペアリングと非対称ペアリング	75
7.6	DLP と ECDLP とペアリング	77
7.7	ID ベース暗号	78
7.8	ID ベース暗号と公開鍵暗号の違い	79
7.9	タイムリリース暗号	80
7.10	ペアリングを使ったデジタル署名	82
7.11	この章のまとめ	82
第 8 章	検索可能暗号	83
8.1	キーワード検索	83
8.2	キーワード検索公開鍵暗号	85
8.3	問題点と改良	85

8.4	この章のまとめ	86
第 9 章	プロキシ暗号	87
9.1	概要	87
9.2	最初の方式	88
9.3	改良方式	89
9.4	その後の進展	90
9.5	この章のまとめ	90
第 10 章	放送型暗号	91
10.1	放送型暗号の目的	91
10.2	ユーザの排除機能	92
10.3	不正者追跡	93
10.4	放送型暗号の始まり	94
10.5	効率のよい放送型暗号	95
10.6	BGW 方式の安全性	96
10.7	より一般の不正者追跡	96
10.8	この章のまとめ	97
第 11 章	属性ベース暗号	98
11.1	属性ベース暗号とは	98
11.2	秘密分散	99
11.3	(k, n) 閾値法	99
11.4	(k, n) 閾値法の復号	100
11.5	Fuzzy ID ベース暗号	102
11.5.1	セットアップ	103
11.5.2	鍵生成	103
11.5.3	暗号化	103
11.5.4	復号	103
11.6	安全性	104
11.7	属性ベース暗号	104
11.8	not をサポート	106
11.9	この章のまとめ	107
第 12 章	関数型暗号	108
12.1	述語暗号	108
12.2	ベクトルの基底変換と内積	111

12.3	DPVS	112
12.4	DPVS を用いた内積暗号	113
12.5	関数型暗号	114
12.6	この章のまとめ	115
第 13 章 準同型暗号		116
13.1	ElGamal 暗号再び	116
13.2	化合物データベースの秘匿検索	118
13.2.1	Tversky 指数	119
13.2.2	暗号化したまま Tversky 指数を計算する	120
13.2.3	サーバ側の安全性向上	121
13.2.4	悪意あるクライアントへの対処	121
13.3	LWE 問題	122
13.4	RLWE 仮定による完全準同型暗号	123
13.5	この章のまとめ	126
第 14 章 ゼロ知識証明		127
14.1	ゼロ知識証明	127
14.2	離散対数問題のゼロ知識証明	128
14.3	秘匿検索におけるゼロ知識証明	129
14.4	電子投票	131
14.5	シャッフル	132
14.6	シャッフルのゼロ知識証明	132
14.7	この章のまとめ	134
第 15 章 ペアリングの安全性仮定		135
15.1	DDH 仮定と DLIN 仮定	135
15.2	パラメータつき問題	136
15.3	ちょっとしたクイズ	137
15.4	ヒントの多い問題	138
15.5	n -DHI 問題と n -SDH 問題	139
15.6	補助入力付き DLP	140
15.7	この章のまとめ	141
第 III 部 数学的なはなし		142
第 16 章 有限体		145

16.1	Euclid の互除法と有限体の逆数	145
16.2	Euclid の互除法の効率	147
16.3	Fermat の小定理と有限体の逆数	148
16.4	巡回群の大きさ	150
16.5	拡大体	150
16.6	拡大体の元の逆元	153
16.7	この章のまとめ	153
第 17 章 楕円曲線		154
17.1	トーラス上の関数	154
17.2	\wp 関数	155
17.3	\wp 関数の関係式	156
17.4	Weierstrass の方程式	158
17.5	判別式	158
17.6	楕円曲線の加法の図形的な定義	159
17.7	楕円曲線の加法公式	161
17.8	トーラスと楕円曲線の同型対応	163
17.9	射影空間	164
17.10	射影座標での加法公式	166
17.11	Edwards 曲線	168
17.12	Edwards 曲線の加法公式	168
17.13	この章のまとめ	169
第 18 章 ペアリング		170
18.1	関数の零点と極	170
18.2	位数の性質	171
18.3	楕円曲線上の関数の零点と位数の例	172
18.4	因子	173
18.5	楕円曲線上の関数の主因子の性質	174
18.6	ペアリング	176
18.7	Miller のアルゴリズム	178
18.8	この章のまとめ	181
付録 A 安全性仮定に関する問題一覧		182
A.1	乗法群表記の問題	182
A.2	加法群表記およびペアリングの問題	183

参考文献	184
索引	192

はじめに

このテキストでは 2000 年以降に登場した、公開鍵暗号に続く新しい暗号技術を紹介します。

最近はネットショッピングをする人が増えています。安心して買い物ができるためには何が必要でしょうか。勝手に口座からお金を引き出されたり、買い物履歴を他人に見られたり、注文した品物が別のものになったりすると困ります。そんなことが起こらないようにするために情報セキュリティという考え方があります。情報セキュリティとは、特定の人だけが情報にアクセスできること、第三者によって勝手に情報を書き換えられたりしないこと、アクセスしたいときにアクセスできることなどの性質を表す言葉です。情報セキュリティの維持には様々な手法や技術が必要です。

たとえば通信相手が本人であることの保証、情報をやりとりする経路の安全性、サーバの安定運用、運営しているサーバの管理者の意識などが必要でしょう。ある買物システムが安全な通信技術を使って作っていたとしても、その管理者に当事者を装って電話を掛け「昨日買った買い物ってどれでしたっけ」と聞いて「〇〇です。」と答えさせてしまうと情報がもれてしまいます。正しい手続きを踏まずにうっかり答えてはいけないという教育が管理者に対して必要です。

それらの技術の中で鍵となる技術が暗号です。暗号が安全でも他が弱いと情報セキュリティは守れません。しかし、暗号技術が無いと信頼の土台ができないという意味で重要な技術です。ここではその暗号技術について、まず公開鍵暗号を中心で解説します。

また最近ではクラウドサービスという言葉も一般的になってきました。クラウドとはサーバ側の資源を利用することで様々な恩恵を受けるサービス形態です。大抵の場合、今までユーザ側（クライアント）で管理していた情報をサーバ側に預けます。つまり、個人的な情報が大量にサーバにあるため今まで以上に情報セキュリティに注意する必要があります。

管理方法に対するユーザの要求も多様化しています。たとえばクラウドに情報を預けるけれども中身は見せたくない、でも情報処理をクラウドにさせたいという要求です。公開鍵暗号だけではこの様な要求に答えられません。それに応じて暗号理論も日々研究され、進化しています。暗号化したまま検索や演算したりする手法が研究開発されています。

公開鍵暗号が発明されたのが 1970 年代。その技術が一般化し、Amazon や楽天などのサービスが爆発的に普及するには 20 年以上かかりました。新しい暗号技術はまだ研究段階で課題も多いです。しかし、改良されることでこれから一般的に使われていくものもあるでしょう。そのようなこれからクラウドを支えると思われる暗号理論や技術について、難しくなりすぎない範囲でできる

だけ曖昧にせずに解説したいと思います。

このテキストの構成

このテキストは大きく三つに分かれています。

第 I 部の「暗号の基礎」では、共通鍵暗号やハッシュ関数、公開鍵暗号などの基礎的な技術について軽く紹介しつつ最近の話題にも多少触れます。また普及が進む楕円曲線暗号について詳しく紹介します。基礎的な技術については『暗号技術入門』(結城浩) [結城 08] が優れた入門書です。楕円曲線暗号については『楕円曲線暗号入門』(伊豆哲也) [伊豆 13] もよい資料です。

第 II 部の「新しい暗号技術」では、このテキストで登場する数学的な道具立てを説明しつつ、ID ベース暗号を中心とした比較的新しい話題を紹介します。属性ベース暗号や準同型暗号やなどの紹介がこのテキストのメインです。

第 III 部の「数学的ななし」では、第 II 部では説明しきれなかった数学的な部分の穴埋めを中心に行います。高校生～大学初年度の数学のレベルを想定します。

なお URL や注釈、式や節の番号などはクリックするとそこに飛べるようになっています。Acrobat Reader や SumatraPDF で読んでいるなら飛んだ後 [Alt]-← で元の場所に戻れます。

謝辞

書きかけの段階から様々な方が内容を確認してくださいました。特に中谷秀洋さん、緑川志穂さんには細かいところまで目を通して有益なコメントをいただき本当にありがとうございました。竹迫良範さんはレビューを募ってくださいました。ここでは一部の方のお名前を列挙いたします。

荒巻賢一、木村仁美、照屋唯紀、西尾泰和、星野喬、堀田昭人、前田浩邦、
その他匿名のレビューの方々（五十音順、敬称略）

この場を借りてお礼を申し上げます。

連絡先

テキストを読んで間違っている、分かりにくい、こうした方がいいなどのご意見がありましたら、twitter (@herumi) やメール (herumi@nifty.com), github (<https://github.com/herumi/ango>)、あるいは掲示板 (<http://hpcgi1.nifty.com/herumi/bbs.cgi>) にてご連絡ください。最新版は <https://github.com/herumi/ango/raw/master/ango.pdf> から取得できます。

第Ⅰ部

暗号の基礎

I 部の流れ

1 章では共通鍵暗号を軽く紹介し、情報理論的安全性と計算量的安全性という考え方を説明します。それからブロック暗号やその使い方について触れます。SSL3.0 に対する攻撃として話題になった POODLE にも簡単に触れます。

2 章では暗号に必要な乱数の紹介をし、3 章で公開鍵暗号の話に入ります。公開鍵暗号（特に ElGamal 暗号）はこのテキストのメインですので詳しく紹介します。

それから公開鍵暗号への攻撃方法や求められる安全性について説明します。RSA 暗号も軽く紹介し、よく説明されている通常の方法は必ずしも安全ではないことを示します。最後に公開鍵暗号を利用するためには必要な公開鍵基盤（PKI）について紹介します。

4 章では認証の話をします。ハッシュ関数に求められる性質を紹介した後、メッセージ認証の話をします。HMAC ではよくある間違いに対する攻撃の紹介をしてからデジタル署名に触れます。それからブラインド署名や部分ブラインド署名というちょっと変わった署名も紹介します。

5 章では I 部の二つ目の山場である楕円曲線暗号の話をします。2 章で紹介した ElGamal 暗号の話が楕円曲線を使って同じように進むことを紹介します。またアメリカの国家安全保障局の盗聴問題で注目度が増えた前方秘匿性（PFS）についても紹介します。

6 章では群の話をします。群とは有限体上の暗号と楕円曲線上の暗号を統一的に考えるための枠組みです。多少抽象的になるので最初はとっつきにくいかもしれません。

第1章

共通鍵暗号

この章では共通鍵暗号について紹介します。共通鍵暗号は2000年以上前からあったとされる歴史の長いものです。

1.1 暗号

暗号（化）とは、私とあなたとのやりとりを他人が盗み見しても、その内容を知られないようにするやり方です。このテキストでは公開されたある手順にしたがって、あるメッセージを元が分からぬものに置き換える方法を対象とします。暗号化する前のメッセージを平文（ひらぶん）、暗号化されたメッセージを暗号文といいます。暗号文を元の平文に戻す操作を復号といいます。復号するときに使う情報を秘密鍵といいます。秘密鍵は他人に知られてはいけません。

暗号技術の中には、そもそもやりとりをしていることを悟られないようにする方法もあります。大事なデータを音声や映像の中に隠蔽（いんぺい）する技術でステガノグラフィと呼ばれます。データの著作権を管理するための電子透かしなどに使われます。

また暗号化の手順を公開せずに秘密にしておくというやり方もあります。ただ手順自体を秘密にする方法は、その方法がいつか漏れてしまう可能性、あるいはその方法が安全なのか第三者の目で検証できないという点であまり勧められません。

情報を安全に扱うためには他人に情報を見えないようにするだけでは不十分です。もらった情報が正しいものか、あるいは他人になりすましていないかを検証できることも必要です。それらの手法を含めたものを総称して暗号技術、暗号プロトコルということがあります。略して単に暗号ということもあります。文脈によって指している対象が異なることがありますので、ご注意ください。

1.2 古典的な方法

古代ローマでも使われていたとされる古典的な暗号として、文字をずらす方法があります。たとえば文章中のアルファベットを一つ前にするという規則です。変換する前の大文字小文字の情報は

残しておきましょう。たとえば H なら G, e なら d です。また一番前の A は Z にするとします。すると Hello IBM は Gdkkn HAL になります。このメッセージを受け取った人は一文字後ろにして Hello IBM に戻します。

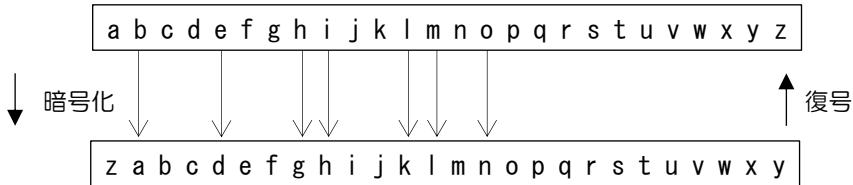


図 1.1 一文字ずらしの暗号

小さいころに秘密のやりとりの遊びでやってみた方もいらっしゃるのではないでしょうか。この場合、ずらす方法が暗号方式、Hello IBM が平文、Gdkkn HAL が暗号文、ずらす文字数が秘密鍵にあたります。

ただこのような文字をずらすだけの方法は容易に破れてしまいます。たとえば英語では e の頻度が高い、q の後ろは殆ど u などの統計的な特徴があります。文字をずらすだけでは文字の頻度の分布は暗号文にも残ります。すると暗号文の中の頻度の高い文字は e だろうという予想を立ててずらされている文字数を推測できます。このように暗号文から元の平文や秘密鍵を見つけることを解読するといいます。現在広く使われている暗号は、たとえ平文のデータに偏りがあったとしてもその暗号文には統計的な偏りが出ないように設計されています。

1.3 Vernam 暗号と情報理論的安全性

Vernam (バーナム) 暗号は絶対に破られないことが保証されている暗号です。ワンタイムパッド (one-time pad) とも呼ばれます。長さ n ビットの平文 m を暗号化するときは長さ n ビットの乱数 r を作ります。乱数については 2 章で紹介します。ここではサイコロを振り、出た目が偶数なら 0, 奇数なら 1 とする操作を n 回繰り返して r を作りましょう。 r を秘密鍵とします。それから m と r のビットごとの排他的論理和 (\oplus) を使って $\text{Enc}(m) := m \oplus r$ とします。排他的論理和は、0 と 1 の演算を

$$\begin{aligned} 0 \oplus 0 &= 0, \\ 0 \oplus 1 &= 1, \\ 1 \oplus 0 &= 1, \\ 1 \oplus 1 &= 0 \end{aligned}$$

という規則で計算します。同じ値同士の排他的論理和は 0 で、異なる値同士なら 1 です。 a, b, c が 0 か 1 のいずれでも $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ が成り立ちます。たとえば平文が 9 (2 進数表記で 0b1001 と書く) で乱数が 13 (0b1101) とします。暗号文は各桁のビットごとに排他的論理和

をとるので

$$9 \oplus 13 = 0b1001 \oplus 0b1101 = 0b0100 = 4$$

です。2進数に変換する方法は3.8節で紹介します。復号は同じ乱数を使って再度排他的論理和をとります。すると

$$(m \oplus r) \oplus r = m \oplus (r \oplus r) = m \oplus 0 = m$$

と元の平文に戻ります。

これがどうして絶対破られないとされているのか、簡単な場合で考えてみましょう。長さ1ビットの平文を暗号化してみます。平文 m は0か1のどちらかです。乱数 r も0か1のどちらかです。すると $\text{Enc}(m)$ も0か1のどちらかです。

$\text{Enc}(m)$ が0だったときに m を推測してみましょう。乱数 r が0なら $m = 0$, $r = 1$ なら $m = 1$ です。 $r = 0$ の確率が $1/2$ なら $m = 0$ の確率も $1/2$ です。これでは $\text{Enc}(m)$ を知らなくても、当てずっぽうで $m = 0$ と推測したときに当たる確率と変わりません。平文の長さが2ビットなら0, 1, 2, 3の4通りで乱数 r も4通りです。どのパターンが出る確率も等しく当てずっぽうで当たる確率は $1/4$ です。ビット長が1増えるごとに平文の種類は2倍になります。したがって長さが n ビットなら平文、乱数の種類は共に 2^n 通りでどのパターンが出る確率も等しく $1/2^n$ です。

どんなビット長でも $\text{Enc}(m)$ を知ったときに m が当たる確率が、 $\text{Enc}(m)$ を知らずに当てずっぽうで当たる確率と同じなので絶対破れない書きました。このような安全性を持つ暗号を情報理論的安全であるといいます。どんなに高性能な計算機を使ってもこの状態は変わりません。

しかし Vernam 暗号が使われることはあまりありません。たとえ安全に m を暗号化できたとしても、それを復号するための秘密鍵 r をどうやって相手に送るかという問題があるからです。元の平文と同じ長さの乱数が必要なのですから、問題が m から r に変わっただけという考え方できます。

1.4 共通鍵暗号

暗号化するときに使う鍵と復号するときに使う鍵が同じである暗号方式を共通鍵暗号といいます。秘密鍵暗号や対称鍵暗号とも呼ばれます。前述の文字をずらす方法や Vernam 暗号は共通鍵暗号の一種です。

Vernam 暗号は秘密鍵の長さと平文の長さは同じでした。それでは使いにくいので、秘密鍵を小さい固定長にした共通鍵暗号が一般的に使われています。

共通鍵暗号は大きく分けてブロック暗号とストリーム暗号の2種類に分類されます。ブロック暗号は、たとえば平文を128ビットのブロックに分けて、そのブロック単位で暗号化します。平文の長さがブロック長の倍数でないときはパディングと呼ばれる長さ調整が必要です。

たとえば平文の長さを50、ブロック長を16とすると、 $50/16 = 3$ 余り2なので最後のデータに14個ダミーのデータを追加して一つのブロックにします。単に追加すると平文の正しい長さが判

らなくなってしまうので、先に追加したダミーの長さを記す、あるいはデータの終端記号を追加してから残りをダミーで埋めることができます。

それに対してストリーム暗号は、ビット（あるいはバイト）単位で平文を処理する暗号方式です。通常パディング操作は不要です。

現在ではブロック暗号が主流です。これはブロック暗号に対する暗号の安全評価の研究の方が進んでいるためです。また2.4節で紹介する方法を使うとブロック暗号を使ってストリーム暗号を構成できます。ブロック暗号にはアメリカ合衆国標準のAES、NTTと三菱電機が共同で開発したCamelliaなどがあります。ソニーが開発したブロック暗号CLEFIAは省電力なハードウェアでも動作する軽量暗号（Lightweight Cryptography）の国際標準規格ISO/IEC 29192の一つに採用されています。

1.5 ブロック暗号と計算量的安全性

ブロック暗号などの秘密鍵の大きさが小さい固定長の暗号は扱いやすいです。しかしその代わりVernam暗号の持っていた情報理論的安全性は失われています。なぜならある暗号の秘密鍵の長さが n ビットだったとします。1.3節で書いたようにこの秘密鍵の種類は 2^n 通りです。したがって平文の長さが n よりずっと長かったとしても最大 2^n 通り試せばその暗号を破れます（今はどの答えが正しいのかの判定方法を考慮しません）。これを総当たり攻撃といいます。

たとえば平文の長さが1024ビットとします。Vernam暗号の場合、平文の候補が 2^{1024} 通りどれも対等にあります。しかし、秘密鍵の長さが64ビットのブロック暗号を使った場合、平文の候補は高々 2^{64} 通りしかありません。Vernam暗号に比べてずっと少ないので。

ある暗号の攻撃に必要な計算量が 2^n のとき、 n ビットセキュリティを持つといいます。ブロック暗号では総当たり攻撃よりも効率のよい攻撃法が見つかっていないものが望ましいです。つまり n ビット鍵長の理想のブロック暗号は n ビットセキュリティを持ちます。

たとえば鍵長が128ビットのときは鍵の種類が $2^{128} = 3.4 \times 10^{38}$ 個あります。一つずつ正しい鍵か確認する場合、1秒間に1京（= 10^{16} ）回確認できたとしても、全部調べるのに1000兆年かかります。鍵長が256ビットのブロック暗号だと 3.6×10^{53} 年です。宇宙の年齢とされる138億（= 1.38×10^{10} ）年よりもずっと大きい数字ですね。最近は128ビットセキュリティ以上の安全性を持つものが望ましいとされています。

このように暗号の攻撃に必要な計算量を見積もり、それよりも少ない計算量では解けないと定義された安全性を計算量的安全性といいます。

1.6 暗号の危険化

計算量的安全性を持つ暗号は、計算機の進歩や解読アルゴリズムの改良に伴い、その安全性は弱くなります。ある暗号が当初想定されていたよりも少ないコストで解読できるようになることを暗

号の危険化（きたいか）といいます。

暗号の危険化は秘匿性（ひとくせい）に関わる重要な問題です。そこでどの暗号が安全でどの暗号が危ないのかといった情報の収拾と評価を行っている専門的な機関があります。日本では CRYPTREC (CRYPTography Research and Evaluation Committees)^{*1} や CELLOS (Cryptographic protocol Evaluation toward Long-Lived Outstanding Security Consortium)^{*2}などの機関です。他にも一般向けに注意喚起するサイトがあります。重要なデータを扱うサーバ管理者は日頃からそれらが提供する情報に触れておくとよいでしょう。

たとえば 3.16 節で紹介する RSA 暗号は 20 年前なら 1024 ビットの鍵サイズで安全でした。しかし近い将来破られると考えられています。すると当時 1024 ビット RSA 暗号で暗号化されていたデータは解読できてしまいます。現在、2048 ビット RSA 暗号や 128 ビット AES が広く使われています。これらの安全性はせいぜい数十年とされています。50 年後も安全かというと恐らくそういうではありません。

インターネット上の買物ならそのときのクレジットカードの番号を秘密に、かつ改竄（かいざん）されずに送受信できればよいので、使用する暗号が当面安全なら特に問題はありません（買物の内容が将来漏洩（ろうえい）する可能性はあります）。クラウドサービスは利用したいけれども全面的に信用できるわけではないから手元でコンテンツを暗号化してアップロードしているという方がいらっしゃるかもしれません。しかし、サービス提供者、あるいは誰かがその暗号化されたデータを盗聴して保持し 20 年後に解読してしまう可能性を知っておく必要があります。

たとえば自分のゲノム情報を 70 年間は絶対に秘密にしたいと思ったとき、どんな暗号を使えばよいのか。それに対する明確な答えは今のところ無いように思います。

1.7 計算量の指標

1.5 節では長さ n ビットの鍵は 2^n 個なので総当たり攻撃に必要な計算量は 2^n と書きました。これは 1 回の演算量がどれくらいかを考慮していません。1 回あたり 10 なのか、あるいは 1000 のなのかによって実際の値は変わってきます（そもそも単位を書いていませんでした）。ただ 100 倍違ったとしても 10^{53} 年でも 10^{55} 年でも「解けない」ことには変わらないですね。また計算量が $1000 \times n^2$ と 2^n とでは、 n が小さいときは前者が大きいですが、 n が大きくなると後者の方がずっと大きくなります。

このように n が大きくなったときに、対象となる計算量がどのように変化するのかについて定数倍を無視して概要を知りたいことがあります。このとき O 記法（オー記法）というものを使います。 f, g を $x \in \mathbb{R}$ に関する実数値関数とします。十分大きい x に対して $|f(x)|$ がせいぜい $|g(x)|$ の定数倍にしかならないとき、 f のオーダーは $O(g(x))$ であるといいます。

^{*1} <http://www.cryptrec.go.jp/>

^{*2} <https://www.cellos-consortium.org/jp/>

たとえば、 $f(x) = 3x^3 + 5x - 5$ なら x が十分大きいときは $3x^3$ の項が効いてきます。

$$\lim_{x \rightarrow \infty} |f(x)/x^3| = 3.$$

ですから $f(x)$ のオーダーは $O(x^3)$ です。先ほどの n ビットセキュリティで必要な計算量は $O(2^n)$ です。

1.8 決定的アルゴリズムの問題点

あるアルゴリズムにおいて入力が決まれば常に同じ結果を出力するとき、そのアルゴリズムを決定的 (deterministic) といいます。これとは対照的に、同じ入力であっても実行するたびに異なる値になる可能性があるとき、そのアルゴリズムを確率的 (probabilistic)，あるいは非決定的 (non-deterministic) といいます。

高校までで習う殆どのアルゴリズムや公式は決定的だと思います。やるたびに結果が変わると困るからです。身近な確率的なアルゴリズムの例としてとして、円周率を求めるモンテカルロ法があります。1辺の長さが 2 の正方形の中に半径の長さが 1 の円を書きます。正方形の中にてたらめに沢山点をうち、円の中に入った個数を数えます。打った点の総数と円の中に入った点の数との比から円周率を求める方法です。沢山回数を重ねても 3.1 付近にすらならない経験をされた方もいらっしゃるかもしれません。

AESなどのブロック暗号は同じ秘密鍵と平文を与えた場合、常に同じ暗号文を出力します。つまりブロック暗号は決定的アルゴリズムです。

暗号が決定的だとちょっと困ったことが起ります。たとえば Aさんは入札に参加していたとします。Aさんは入札額 10 万円をブロック暗号で暗号化して暗号文 c として先方に送っていました。Aさんが無事落札し、入札額 10 万円が公開されます。すると Aさんの通信を盗聴していた Bさんは c が 10 万円を暗号化したものだと分かります。

次に別の入札で Aさんがまた 10 万円で入札しようとした。暗号文は同じ c です。すると Bさんは c を見て Aさんの入札額が分かり、自分は 10 万 1 円で落札してしまったのです。

したがってブロック暗号を決定的なまま使ってはいけません。

1.9 CBC モードと初期化ベクトル

前節ではブロック暗号は決定的アルゴリズムなのでそのまま使ってはいけないと説明しました。同じ平文が常に同じ暗号文になるのが問題なので、そうならないような使い方をすべきです。その方法はいくつかあるのですが、ここでは CBC モード (Cipher Block Chaining) を紹介します。CBC モードはインターネットで安全に通信するためのプロトコルである SSL (Secure Sockets Layer) で使われています。

CBC モードは平文をブロック (たとえば 16 バイト) ごとに暗号化する際に、一つ前のブロックの

暗号文と排他的論理和をとってから暗号化する方法です。先頭ブロックの一つ前は存在しないのでダミーのブロックをランダムに生成して使います。このブロックを初期化ベクトル（Initialization Vector），略して IV といいます。IV は暗号文といっしょに他人に見える形で送ります。

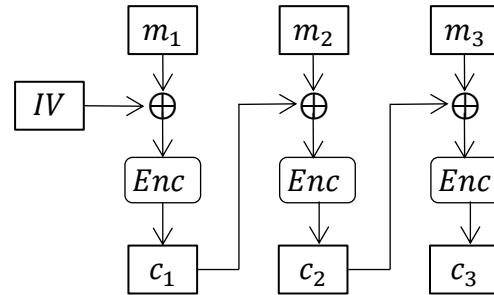


図 1.2 CBC モードの暗号化

IV が異なると同じ平文 m でも暗号文 c が異なるのでより安全になります。

復号は暗号文を復号した後一つ前の暗号文と排他的論理和をとって平文に戻します。先頭のブロックでは一つ前の暗号文の代わりに IV を使います。

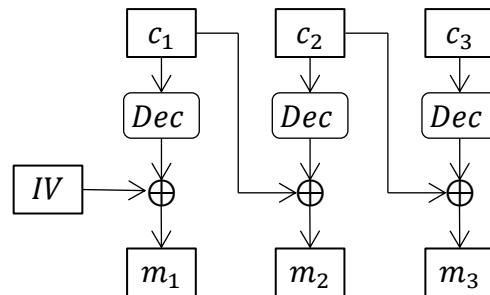


図 1.3 CBC モードの復号

1.10 パディングオラクル攻撃

CBC モードは平文 m_i は一つ前の暗号文 c_{i-1} との排他的論理和をとっています。したがって c_{i-1} のどれか 1 ビットを変更すると、対応する m_i の 1 ビットが変わります。これは通常問題になることはありません。しかし 2002 年にブロック暗号のパディング処理と組み合わせた攻撃の発表がありました [Vau02]。

それは次のような攻撃です。ブロック暗号はデータをブロックサイズに合わせるためにパディング処理が必要でした。攻撃したい暗号文にわざと不正なパディングを追加してサーバに送ります。するとサーバがそれを受け入れるかエラーで弾くかします。そのサーバの返答を使うことで秘密鍵

は分からぬまま元の平文の一部を得るのであります。

この手法はパディングオラクル攻撃 (Padding Oracle Attack) と呼ばれます。オラクル (Oracle) とは神託（しんたく）と呼ばれる、神様が発した言葉のことです。この場合は「攻撃者が作った暗号文が不正か否か」という問い合わせの答えを教えてくれるサーバをオラクルに見立てています。

この攻撃を実際にするにはかなりの量の問い合わせが必要になり、直ちに脅威となるとは考えられていませんでした。しかし 2014 年、SSL3.0 の仕様の不備を使って CBC モードに対して少ない回数でパディングオラクル攻撃ができることが示されました。POODLE (Padding Oracle On Downgraded Legacy Encryption) と呼ばれるこの攻撃により、現在 SSL3.0 は使ってはいけないプロトコルとなっています。CBC モード自体は安全なのですが、プロトコルの作り方や組み合わせ方によっては安全でなくなることがあるという重要な一例です。

1.11 この章のまとめ

共通鍵暗号の一つである Vernam 暗号という乱数を使った暗号を紹介しました。Vernam 暗号は情報理論的に安全な暗号ですが、秘密鍵の大きさが平文と同じ大きさという難点があります。そのため大抵は秘密鍵の大きさが数百ビットですむブロック暗号やストリーム暗号が使われます。ブロック暗号は、どれくらいの回数の計算をすれば解けるかという計算量の見積もりを用いてその安全性が決められています。つまり計算量的安全性を持つ暗号です。

アルゴリズムには入力が同じなら常に同じ値を返す決定的なものと、その都度変わり得る確率的なものがあります。ブロック暗号は決定的なアルゴリズムなので平文と秘密鍵が同じなら常に同じ暗号文になります。この性質は安全性を損なうため、CBC モードなどを使って同じ平文でも異なる暗号文になるようにすべきです。

第 2 章

乱数

この章では安全な暗号を構成するのに不可欠な乱数の紹介をします。1.8 節で書いたように暗号は確率的アルゴリズムであることが望されます。確率的なものを実現するにはよい乱数が必要です。暗号にとってよい乱数とは何か、その満たすべき性質を紹介します。

2.1 擬似乱数

Vernam 暗号では乱数を使いました。乱数とはサイコロやルーレットのように次に何が出るか分からない、再現性の無いでたらめな数字の列のことです。しかし、コンピュータでは本当の乱数を扱うのはなかなかやっかいです。与えられた手順とデータにしたがって常に同じ値を計算することを決定的というのでした（1.8 節参照）。コンピュータは決定的な処理が得意です。ですが、やる度に計算結果が異なる確率的な処理は通常できないのです。そんなプログラムがあったら、それは大抵バグですね。

そのため厳密な乱数を扱うのは諦めて、一見乱数のように見える擬似乱数というものを代わりに扱います。擬似乱数生成器とは、ある決まった数式と初期値を出発点として決定的に計算される数列を出力する装置です。擬似乱数とは擬似乱数生成器により生成された数列のことです。初期値が同じなら常に同じ擬似乱数が出力されます。

大抵のプログラミング言語に用意されている `rand()` という関数は擬似乱数を生成します。

```
(rand() % 6) + 1
```

の値をすることでサイコロの値をシミュレーションされた方も多いでしょう ($a \% b$ は a を b で割った余りを表します)。実行するたびに異なる数列が欲しいときは、初期値にプログラムを起動したときの時刻を設定することが多いです。

`rand()` は線形合同法と呼ばれるアルゴリズムを使って実装されることがあります。線形合同法によって生成される擬似乱数はパラメータによっては偶奇が規則的に現れ、ランダム性が高くありません（そのときは `rand()` をその関数の値のとり得る最大値で割って 0~1 の間の小数にしてか

ら 6 倍して整数に切り捨てるトまだましになります).

様々なシミュレーションでは Mersenne twister (メルセンヌ・ツイスター), 略して MT という擬似乱数生成器が使われるようです. MT は, 高速に計算できる, その擬似乱数列が $2^{19937} - 1$ という長い周期を持つ, 統計的に十分ランダムで均等に分布する, というよい性質を持っているからです. シミュレーションにとって都合のよい擬似乱数が得られるのです. しかし, MT は暗号のシステムでそのまま使ってはいけません.

2.2 暗号論的擬似乱数

暗号の用途で使われる乱数は, シミュレーションなどで使われる擬似乱数とは異なる性質が要求されます. 通常の擬似乱数と区別するために暗号論的擬似乱数と呼ばれることがあります. 暗号を使うということが分かっている場合は単に擬似乱数ということもあります.

暗号論的擬似乱数の直感的な定義は, どれだけ過去の生成列を見ても次の 1 ビットの出力が 0 か 1 のどちらか予測できない ($1/2$ 以上の確率で当てられない) という性質を持つものです. 暗号論的擬似乱数生成器も決定的なアルゴリズムなのですが, 生成列から初期値や次出る値を推測できとはいいけないです.

MT は回路の内部に 624 個の 32 ビットの値を持ちます. その内部変数を逐次更新しながら擬似乱数列として出力します. したがって少なくとも 624×32 ビット程度観察すれば内部状態が推測できます. するとそれから先に出現する値を予測できてしまうので暗号論的擬似乱数生成器ではないのです.

先ほど擬似乱数の初期値に時刻を設定する話をしました. 現在時刻のパターンはそれほど多くはないので, その方法では初期値を推測できてしまいます. ゲームに使うなら十分かもしれません, 暗号を使う擬似乱数生成器の初期値に時刻を使ってはいけません.

コンピュータで暗号論的擬似乱数を扱うときは Linux では /dev/random や /dev/urandom という乱数生成デバイス, Windows では CryptGenRandom() という専用の関数を使うことが推奨されます. これらの関数は, 内部でそのときの HDD, メモリ, マウスやキーボードの情報など常に変動している情報を乱数生成に利用します. そうすることで誰も予測も再現もできない値を生成しています.

2.3 擬似乱数とストリーム暗号

Vernam 暗号は平文と同じ長さの乱数が必要でした. 亂数の代わりに暗号論的擬似乱数生成器を使ってみましょう. 暗号論的擬似乱数生成器に初期値を入れるとそれに応じた決まった乱数列を生成できます. 相手にその初期値を渡します. すると両者で同じ擬似乱数列を共有できます. 亂数全てを共有する必要がある Vernam 暗号に比べて初期値の共有だけですみます. このような考え方で作られた共通鍵暗号をストリーム暗号といいます.

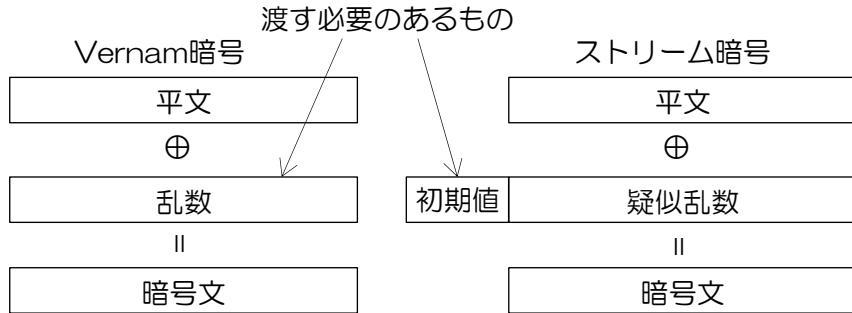


図 2.1 Vernam 暗号とストリーム暗号

ストリーム暗号は初期値 s を秘密鍵として決定的な擬似乱数を生成し、それと平文の排他的論理和をとって暗号文とします。

$$\begin{aligned} r &= \text{rand}(s), \\ \text{Enc}(m) &= m \oplus r. \end{aligned}$$

復号は同じ s を使って同じ値を生成する擬似乱数を用いて暗号文との排他的論理和をとります。

$$\begin{aligned} r &= \text{rand}(s), \\ \text{Dec}(\text{Enc}(m)) &= (m \oplus r) \oplus r = m. \end{aligned}$$

ストリーム暗号は初期値の長さが平文の長さよりも短いので、ブロック暗号と同じく情報理論的安全性を持っていません。総当たり攻撃以外の効率のよい方法が無いことが求められる計算量的安全性を持つ暗号です。計算量的安全性については 1.5 節を参照ください。

ストリーム暗号には RC4 や日立の Enocoro, 九州大学と KDDI による KCipher-2 などがあります。RC4 は長らく使われていましたが 2012 年頃から攻撃方法が見つかり 2013 年は CRYPTREC の運用監視リストに入りました。

2.4 CTR モードとストリーム暗号

擬似乱数をうまく作ればストリーム暗号を構成できます。擬似乱数はどんな方法で作ってもかまいません。安全なブロック暗号を使ってストリーム暗号を作る方法があります。初期値 s と 0 から一つずつ増えていく値 c (カウンタと呼ばれる) を連結したもの $(s||c)$ をブロック暗号で暗号化して作ります。この方法はカウンタモード (counter mode), 略して CTR モードと呼ばれます。CTR モードで擬似乱数を作りて平文と排他的論理和をとるので、暗号化と復号は同じ操作になります。図 2.2 において c_i と m_i を入れ換えると復号を表す図になります。

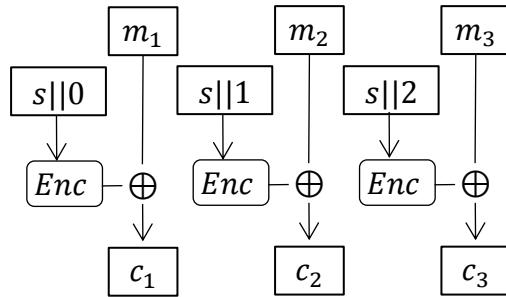


図 2.2 CTR モードの暗号化と復号

1.9 節で紹介した CBC モードは一つ前に作った暗号文が次の暗号文の作成に必要でした。つまり i 番目のブロックの暗号化は $i - 1$ 番目のブロックの暗号化が終わるまで開始できません。しかし、CTR モードは $s||c$ の暗号化は一つ前の処理と独立に行えます。つまり複数の暗号化を並列に実行できるので CBC モードより高速に暗号化できます (CBC モードの復号は並列実行可能です)。

もちろんブロック暗号からストリーム暗号が作られるからといってストリーム暗号が不要となるわけではありません。ブロック暗号を使わなくても擬似乱数を生成する方法はいろいろあるからです。KCipher-2 は AES よりも高速に暗号化できるように設計され、2013 年に電子政府推奨リストに採用されました。

2.5 Intel CPU の `rdrand` 命令

暗号論的擬似乱数は、暗号にとって重要なのに作るのが難しいという悩みがありました。擬似乱数生成器に渡す初期値をどのように決めるかという悩みもあります。そのため Intel の Ivy Bridge 以降の CPU には `rdrand` という命令が追加されています。`rdrand` は CPU 内部の不確かな電子状態を使うことで決定的でない乱数列を生成します。難しいアルゴリズムを知らなくてもこの命令を呼べば暗号に使える乱数が得られるのでとても便利です。実際、`gcc` というコンパイラで提供される `std::random_device` という乱数生成ライブラリの実装では `rdrand` が出力する値をそのまま使ってています (`gcc 4.8` で確認)。

しかし前述の擬似乱数生成デバイス `/dev/urandom` の実装はこの命令だけに依存するような作り方をしていません。なぜでしょうか。仮に `rdrand` の実装が、あるストリーム暗号で生成される擬似乱数を出力していたとしましょう。初期値を秘密にしておけば、通常これは暗号論的擬似乱数と区別ができません。ところが初期値を知っている人 (この場合は Intel) にとっては、この乱数は決定的で予測可能です。これではこの乱数を使って作られた秘密鍵が初期値を知っている人にのみ知られてしまう可能性があります。このような状況をバックドアがあるといいます。

2013 年アメリカの国家安全保障局 (NSA) が様々な盗聴を行っていたという告発がありました。そのためセキュリティに関するオープンでないハードウェア技術は信用するべきではないとい

う考え方方が強くなりました。Intel の `rdrand` の実装にバックドアがある可能性はとても低いと思われます。しかし、`/dev/urandom` の実装者は万一そんなことがあっても安全性を保てるように、`rdrand` だけには依存しない設計をしているのです。

2.6 この章のまとめ

暗号に使われる乱数は、シミュレーションなどで使われる乱数と違い過去のデータから次の値を推測できてはいけません。たとえば MT (Mersenne twister) をそのまま使ってはいけません。もし次の値を推測できる乱数を使ってしまうと、たとえ暗号自体が安全だったとしてもシステム全体としては安全にならないことがあります。暗号プロトコルの中で乱数を使うときは十分な注意が必要です。

ストリーム暗号は擬似乱数を使います。擬似乱数を作る方法はいろいろありますが、ここではブロック暗号を使って作る方法を紹介しました。

第3章

公開鍵暗号

この章では、公開鍵暗号について紹介します。公開鍵暗号は1970年代に初めて構成された比較的新しい種類の暗号です。

公開鍵暗号は、共通鍵暗号の秘密鍵をどうやって安全に受け渡しできるかという問題から生まれました。現在では様々な公開鍵暗号が提案されています。その中でElGamal暗号という公開鍵暗号を紹介します。ElGamal暗号は、このテキストで紹介したい新しい暗号技術において様々な場面で形を変えて登場する重要な暗号です。ElGamal暗号は有限体という枠組みの中で構成されるのでその用語の説明から始めます。

最後に公開鍵暗号を使って運用するために必要な取り決めについて軽く紹介します。

3.1 鍵の共有

安全な共通鍵暗号があったとしましょう。ネットショッピングをする場合、通信はどのような経路を通って行われているか分かりません。もしかしたら誰かが盗聴しているかもしれません。商品名や買い物に使うクレジット番号などを第三者には知られたくないで暗号化して送りたいです。それで共通鍵暗号の一つAESを使ってその番号を暗号化しました。さて、AESの秘密鍵をどうやって先方に知らせましょう。その秘密鍵を更に別の共通鍵暗号で暗号化して送る？その暗号化を使った鍵はどうしましょう。

共通鍵暗号を使うだけではきりがありません。別の手段が必要だと分かります。盗聴者がいたとしても二人の間で盗聴者には分からないように秘密鍵を共有する仕組みです。次節以降でその方法について説明しましょう。

3.2 余りの世界

二人の間で安全に秘密の数字（秘密鍵）を共有するには、余りの世界の計算が必要になります。たとえば日付を無視した時刻の流れがそれになります。1日を24時間制で表し、今19時としま

す。1時間後は20時です。5時間後の24時になると0時に戻ります。50時間後なら2日と2時間後なので21時になります。つまり時刻は24で割った余りを考えています。

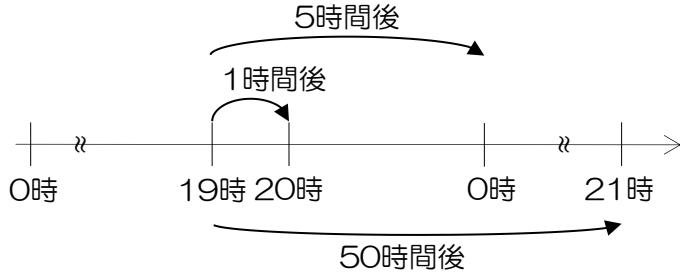


図 3.1 時刻と余り

一般に整数 a を整数 $p (> 0)$ で割った余りを $a \bmod p$ と書くことにします。より正確に書くと $a = qp + r$ ($0 \leq r < p$) となるような整数 q, r を選んだとき r を余りと定義します。 a が負のときでも余りは常に0以上 p 未満になるように選びます。また a と b の差が p で割れるとき $a \equiv b \pmod{p}$ と書きます。例をあげましょう。

$$\begin{aligned} (19+1) \bmod 24 &= 20, \\ (19+6) \bmod 24 &= 1, \\ 50 &\equiv 2 \pmod{24}, \\ (-35) \bmod 24 &= 13. \end{aligned}$$

最後の等式については $-35 = (-2) \times 24 + 13$ だからです。

この余りの世界で足し算と引き算を考えてみましょう。余りを足したもの（を p で割った余り）は足したもの余りです。数式で書くと

$$((a \bmod p) + (b \bmod p)) \bmod p = (a + b) \bmod p.$$

たとえば $15 + 20 = 35$ ですが、 $15 \bmod 12 = 3, 20 \bmod 12 = 8, 35 \bmod 12 = 11$ で $(3 + 8) \bmod 12 = 11$ が成り立っています。

引き算や掛け算でも同じことが成り立ちます。

$$\begin{aligned} ((a \bmod p) + (b \bmod p)) \bmod p &= (a + b) \bmod p, \\ ((a \bmod p) - (b \bmod p)) \bmod p &= (a - b) \bmod p, \\ ((a \bmod p) \times (b \bmod p)) \bmod p &= (a \times b) \bmod p. \end{aligned}$$

掛け算を繰り返すことで巾乗（べきじょう）の計算もできます。このとき上記3番目の性質によつて、掛け算を全てすませてから余りをとるのではなく、掛けている途中の値に対して余りを取っても構いません。たとえば $2^{30} \bmod 13 = 1073741824 \bmod 13 = 12$ ですが、 1073741824 を13で割るのはちょっと面倒です。でも $2^1, 2^2$ の余りを順に計算しながらやると13で割る数はせいぜい2桁なので簡単です。

a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$2^a \bmod 13$	1	2	4	8	3	6	12	11	9	5	10	7	1	2	4

2を12乗したら1になりました。そこからは2, 4, 8, …と続き、12進むごとに元に戻ることを繰り返します。ということは24乗しても1です。だから

$$2^{30} = 2^{24+6} = (2^{12})^2 \times 2^6 \equiv 2^6 \equiv 12 \pmod{13}$$

と計算できます。これからじっくりと説明していきますが、巾乗することは比較的容易なのに対してその逆、何乗したらある値になるかを求めるのは難しいことが知られています。その一方向性が暗号の肝となります。

3.3 Diffie-Hellman 鍵共有

1976年、余りの世界での巾乗計算を使って秘密鍵を共有する方法が考えされました。その方法を提案者の名をとって Diffie-Hellman（ディフィー・ヘルマン）鍵共有（DH 鍵共有）といいます。

次の方法を使ってAさんとBさんの間で秘密の数字を共有します。

1. 整数 g, p を二人の間で決めて平文で共有する。 g, p の条件は後述します。
2. Aは秘密鍵（秘密の値） a を決め公開鍵 $K_A := g^a \bmod p$ を求めてBに渡す。ここで「 $:=$ 」は左辺を右辺で定義するという記号です。
3. Bは秘密鍵 b を決め公開鍵 $K_B := g^b \bmod p$ を求めてAに渡す（以後 $\bmod p$ を省略します）。
4. Aは a とBからもらった K_B を使って $K_B^a = (g^b)^a = g^{ab}$ を求める。
5. Bは b とAからもらった K_A を使って $K_A^b = (g^a)^b = g^{ab}$ を求める。

$K_B^a = (g^b)^a = g^{ab} = (g^a)^b = K_A^b$ なのでAとBは同じ値を共有できました。

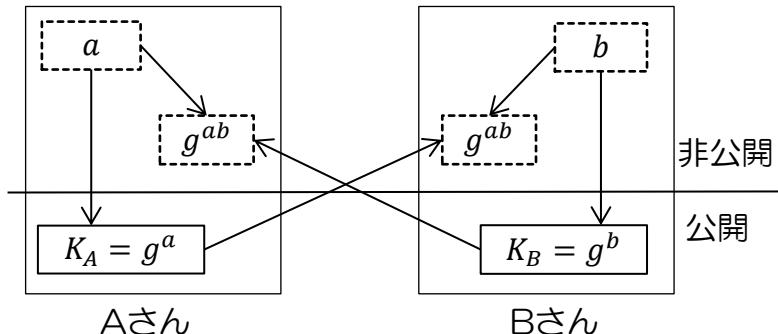


図 3.2 DH 鍵共有

3.4 鍵共有のための仮定と条件

AさんとBさんは同じ値 g^{ab} を共有できましたが、これは安全な方法なのでしょうか。AさんとBさんのやりとりを盗聴していた第三者（攻撃者）がいたとします。その攻撃者は g, p, K_A, K_B の値を知ります。公開鍵 K_A を知っているのだから a を求められそうです。実際 g^1, g^2, \dots, g^{p-1} と順番に計算していくばどれかは K_A に等しくなります。つまり、しらみ潰しに確認すれば求めることは可能です。この問題を定式化すると次のようになります。

「 $g, p, g^a \bmod p$ が与えられたときに a を求めよ。」

これを離散対数問題（Discrete Logarithm Problem），略して DLP といいます。一般に実数 $x, y, a > 0$ について $y = a^x$ を満たすとき $x = \log_a y$ と書き、 x を「 a を底（てい）とする y の対数」とよぶのでした。DLP の場合、 $h = g^a$ とすると a は g を底とする h の対数とみなせます。そして a, h, g は整数なので離散的な値です。それで離散対数問題と呼ばれているのです。

今のところ g, p が次の条件を満たしているときの DLP は現在のコンピュータは求められないと考えられています。DLP を解くのに必要な計算量が大きすぎて現実的な時間で解けないです。

1. p は 1024 ビット以上の素数で、 $p - 1$ の約数の中に p に近い大きさの素数 q がある。
2. g は $i = 1, \dots, q - 1$ に対して $g^i \not\equiv 1 \bmod p$ となる値（このような g を生成元といいます）。

2より大きい素数 p は奇数なので、条件 1 で一番よいのは $q := (p - 1)/2$ が素数となるときです。条件 2 の生成元 g は 2 や 5 が使われることが多いです [TK03]。このような (p, q, g) の組をこのテキストでは DH パラメータと呼ぶことにします（一般的な用語ではありません）。この条件については 6.7 節で再度触れます。

もしかしたら明日、誰かが DLP の効率のよい方法を発見し、その求め方を公開するかもしれません。あるいは既に解読方法を発見したけれども世の中の通信を盗み見するためにその方法を隠している人がいるかもしれません。そんな不安はありますが、現在は DLP はとても難しい問題と考えられています。

すると攻撃者は公開鍵 K_A や K_B からは秘密鍵 a と b を求められません。つまり g^{ab} は分からぬのです。よって g^{ab} は A と B の二人だけしか知らない数字となり、安全に秘密の数字を共有できたことになります。

3.5 CDH 仮定

前節では DLP が難しいので DH 鍵共有は安全だと説明しました。でもちょっと待ってください。 K_A, K_B から a, b を求められなくとも何らかの方法で直接 g^{ab} を計算できたりしないのでしょうか。この問題を定式化すると次のようになります。

「 $g, p, g^a \bmod p, g^b \bmod p$ が与えられたときに $g^{ab} \bmod p$ を求めよ。」

この問題を DH 問題 (DHP : DH Problem) といいます。DLP とはちょっと違います。もし DLP が解けるなら DH 問題は解けます。 g, g^a, g^b が与えられたときに g と g^a から DLP により a を求めて g^b を a 乗すれば g^{ab} を求められるからです。しかし逆に DH 問題が解けるからといって DLP が解けるか否かはすぐには分かりません。DH 問題も十分難しいと考えられていますが、今のところ二つの問題の同値性は示されていないようです。DH 問題が難しいという仮定を CDH 仮定 (Computational DH) といいます。単に DH 仮定、あるいは DHA (A は Assumption) ともいいます。したがってこの用語を使うと DH 鍵共有は CDH 仮定の元で安全といえます。

3.6 DH 鍵共有の注意点

DH 鍵共有は CDH 仮定の元で安全に秘密の数値を共有できることが分かりました。ここで安全というのは、二人がやりとりする経路を攻撃者が盗聴していたとしても g^{ab} は二人だけしか知らないようにできるという意味です。しかし、攻撃者が盗聴だけでなく改竄（かいざん）までしていた場合の安全性を保証しているわけではありません。



図 3.3 正常時の DH 鍵共有

正常時は A の公開鍵 K_A が B に、B の公開鍵 K_B が A に正しく送信されています。このときは安全に共有できます。

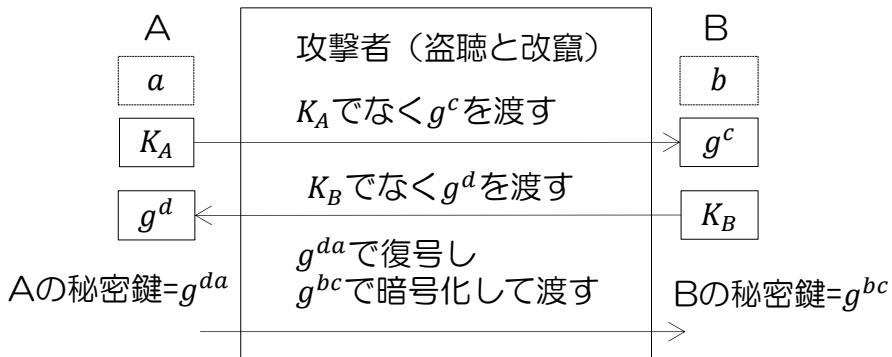


図 3.4 改竄時の DH 鍵共有

しかし、途中経路に通信内容を改竄できる攻撃者がいたとします。攻撃者は適当な数 c, d を決めて A から来た K_A を受け取り g^c を B に渡します。そして B からきた K_B を受け取り g^d を A に

渡します。

A は K_B (と思っているが実は g^d) を a 乗し g^{ad} を共有された秘密の値と思い込みます。同様に B は K_A の代わりに g^c を b 乗し g^{bc} が共有された値と思い込んでいます。A は g^{ad} を鍵として共通鍵暗号を使い送信します。攻撃者は K_A を d 乗し g^{ad} を持っているので復号できて中身を見ることができます。次に K_B の c 乗である g^{bc} を使って中身を暗号化して B に送信します。B は g^{bc} を使って復号できるので盗聴されているとは想像できません。

このように通信の途中に入つて改竄する攻撃を中間者攻撃 (MITM : Man in the Middle) といいます。中間者攻撃されると安全ではないため、 K_A, K_B を改竄されずに正しく送信できたかを確認する機構が必要になります。これは DH 鍵共有に限らない難しい問題で、後で考察します。

3.7 巾乗の計算

DH 鍵共有では p が 1024 ビット以上、つまり 10 進数で 300 桁以上という条件があると書きました。具体的にはたとえば

179769313486231590772930519078902473361797697894230657273430081157732675
805500963132708477322407536021120113879871393357658789768814416622492847
430639474124377767893424865485276302219601246094119453082952085005768838
150682342462881473913110540827237163350510684586298239947245938479716304
835356329624224137859

という数字です。かなり大きいですね。 a も同じような大きさです。素朴な疑問ですが、 $2^a \bmod p$ って計算できるのでしょうか。たとえば 1 回の掛け算を 1 ナノ秒 ($= 10^{-9}$ 秒) でできたとします。これは最近のパソコンの数十倍速い速度です。しかしそれでも $2^2, 2^3, \dots$ をえんえんと 10^{300} 回計算していると $10^{-9} \times 10^{300}$ 秒 $= 3 \times 10^{283}$ 年かかってしまいます。これではまったく使い物になりません。DH 鍵共有は絵に描いた餅なのでしょうか。大丈夫です。そんなことはなく、ずっと少ない掛け算の回数で巾乗を計算する方法があります。

具体例で考えてみましょう。 $2^{100} \bmod 10^8$ を計算してみます。素朴な方法では 99 回の掛け算が必要です。でも $100 = 25 \times 2 \times 2$ と分解すれば掛け算の回数を減らせることがあります。 $2^{100} = (((2^{25})^2)^2)$ なので 2^{25} を求めましょう。 $2^{25} = 2 \times 2^{24} = 2 \times (2^3)^8$ と変形できます。

$$\begin{aligned} 2^3 &= 2 \times 2 \times 2 \equiv 8 \pmod{10^8}, \\ 2^6 &= (2^3)^2 \equiv 64 \pmod{10^8}, \\ 2^{12} &= (2^6)^2 \equiv 4096 \pmod{10^8}, \\ 2^{24} &= (2^{12})^2 \equiv 1777216 \pmod{10^8}, \\ 2^{25} &= 2^{24} \times 2 \equiv 3554432 \pmod{10^8}, \\ 2^{50} &= (2^{25})^2 \equiv 6842624 \pmod{10^8}, \\ 2^{100} &= (2^{50})^2 \equiv 3205376 \pmod{10^8} \end{aligned}$$

というわけで 8 回の掛け算で求められました。これなら手計算でもなんとかできます。このように 2 乗を組み合わせると掛け算の回数を減らせます。

3.8 バイナリ法

バイナリ法とは $g^a \bmod p$ を高速に求める計算アルゴリズムの一つです。先程は $a = 100$ を適当に分割して計算を工夫しました。もう少し一般的に適用できる方法でやってみます。まず 100 を相異なる 2 の巾の和で書きます。

$$100 = 64 + 32 + 4.$$

この分割の方法は 100 を 2 進数に変換すると簡単にできます。2 進数に変換するには 100 を 0 になるまで繰り返し 2 で割り、その時の余りを逆順に並べるとよいです。

$$\begin{aligned} 100/2 &= 50 \text{ 余り } 0, \\ 50/2 &= 25 \text{ 余り } 0, \\ 25/2 &= 12 \text{ 余り } 1, \\ 12/2 &= 6 \text{ 余り } 0, \\ 6/2 &= 3 \text{ 余り } 0, \\ 3/2 &= 1 \text{ 余り } 1, \\ 1/2 &= 0 \text{ 余り } 1. \end{aligned}$$

下から余りを並べると 1, 1, 0, 0, 1, 0, 0 です。 $100 = 0b1100100$ とかけます。先頭の 0b は 2 進数であることを示す記号です。1 がある部分の位置の 2 巾の和になっています。

i	6	5	4	3	2	1	0
2^i	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
100	1	1	0	0	1	0	0

$$100 = 2^6 + 2^5 + 2^2 = 64 + 32 + 4.$$

次に $g = 2$ の 2 巾の値を順次求めます。一つ前の値の 2 乗を繰り返して作ります。

a	1	2	4	8	16	32	64
$2^a \bmod 10^8$	2	4	16	256	65536	94967296	9551616

すると

$$\begin{aligned} 2^{100} &= 2^{64+32+4} = 2^{64} \times 2^{32} \times 2^4 \\ &\equiv 9551616 \times 94967296 \times 16 \\ &\equiv 3205376 \pmod{10^8} \end{aligned}$$

と計算できます。演算回数は 2 の巾を求めるのに 6 回、 2^{100} を求めるのに 3 回の計 9 回です。先ほどのやり方より少し回数は多いですが 99 回よりもずっと少なく求められました。

このやり方は容易に一般化できます。 a を 2 進数展開したときに長さ n だったとします。まず g の 2 乗を繰り返して $g, g^2, g^4, g^8, \dots, g^{2^n} \bmod p$ を作ります。そして a を 2 進数に変換したときの 1 となった場所の $g^{2^i} \bmod p$ を掛けていけばよいのです。

掛け算回数は 2 巾の表を作るのに $n - 1$ 回、2 進数展開したときの 1 となっている場所は最大 $n - 1$ 個なので合計最大 $2(n - 1)$ 回です。 a が 10 進数で 300 衡程度なら、 $10^{300} \approx 2^{1024}$ なので $n = 1024$ 。つまり高々 2000 回程度の掛け算で $g^a \bmod p$ を求められます。

この方法を使えば巾乗を十分高速に求められます。 p の 2 進数展開したときの長さは $\log_2(p)$ なのでこのアルゴリズムの計算量は $O(\log(p))$ です。というわけで DH 鍵共有は現実的な時間で計算できるアルゴリズムだと分かりました。

3.9 公開鍵暗号

共通鍵暗号は秘密鍵をどうやって渡すかが悩ましいでした。そこで暗号化する鍵と復号する鍵を分けたらうまくいくのではないかと考えられた（かどうかは知りませんが）方式が公開鍵暗号です。公開鍵暗号は一般的に次のような枠組みとして定義されます。

1. 全員で使うためのパラメータの初期設定をする。
2. ユーザそれぞれが誰にも教えない秘密鍵 K' と、みなに教える公開鍵 K とを作る。
3. A さんに平文 m を送りたいときは A さんの公開鍵 K_A を使って暗号化する。
 $c := \text{Enc}(K_A, m).$
4. A さんは自分の秘密鍵 K'_A を使って復号する。
 $m := \text{Dec}(K'_A, c).$

公開鍵 K_A を使って暗号化された暗号文は秘密鍵 K'_A を知っている人（A さん）しか復号できません。しかし K_A を知っている人は誰でも K_A を使って暗号化できます。

ここで Enc は確率的アルゴリズムでなければなりません。確率的アルゴリズムとは同じ秘密鍵 K_A と同じ平文 m を使っても $\text{Enc}(K_A, m)$ は毎回異なる値になるものでした（1.8 参照）。

どうして Enc は確率的アルゴリズムでないといけないのでしょうか。なぜならたとえば A さんが yes か no のどちらかを暗号化して暗号文 c を作ったことがわかっているとします。盗聴者は公開鍵 K_A を使って $\text{Enc}(K_A, \text{yes})$ と $\text{Enc}(K_A, \text{no})$ を作れます。すると Enc がいつも同じ値を出力する決定的アルゴリズムだったら、作った暗号文のどちらが c に等しいかを確認することで平文を当てることができます。

このように暗号を解読しようとする人（これを攻撃者といいます）が自分の好きな平文に対してその暗号文を入手できる状況での攻撃を選択平文攻撃（CPA : Chosen Plaintext Attack）といいます。共通鍵暗号を使う場合に CPA が可能な状況はそれほど多くはありません。しかし、公開鍵暗号では常に CPA を想定しなければなりません。そのため公開鍵暗号は少なくとも確率的アルゴリズムである必要があるのです。なお、Dec は決定的アルゴリズムでかまいません。

さて、これから DH 鍵共有方式を変形して公開鍵暗号の一つ ElGamal 暗号を作ります。その前に準備として余りの世界をもう少し見てみましょう。

3.10 余りの世界再び

余りの世界では足し算、引き算、掛け算、巾乗算ができました。ある集合の要素に対して演算の結果がまたその集合に入っているときその演算は閉じているといいます。つまり、余りの世界で足し算、引き算、掛け算は閉じています。整数の世界でも足し算、引き算、掛け算は閉じています。

ところが整数の世界では割り算をすると結果が整数にならないときがあります。たとえば 5 割る 3 の値 $5/3$ は整数ではありません。つまり割り算は整数の世界で閉じていません。それに対して有理数の世界では四則演算は閉じています。分数（整数を含む）同士のそれらの演算の結果はまた分数になるからです。

余りの世界はどうでしょうか。整数と同じ割り算ではやはり閉じていません。しかし整数の世界と異なり、余りの世界にはうまい割り算を導入できます。それを見ていきましょう。

一般に 0 以外の整数 x に対して $1/x$ を x の逆数といいました。 $1/x$ は x を掛けて 1 になる値です。

$$(1/x) \cdot x = 1.$$

たとえば $p = 13$ として

$$2 \times 7 = 14 \equiv 1 \pmod{13}$$

という式を眺めます。2 と 7 を掛けて 1 になっています。それで余りの世界では 2 の逆数 $1/2$ は 7 であると考えることにします。

$$1/2 \equiv 7 \pmod{13}.$$

7 の逆数が 2 であると考えてもよいです。

$$1/7 \equiv 2 \pmod{13}.$$

すると $5/2$ を $5 \times (1/2) \equiv 5 \times 7 = 35 \equiv 9 \pmod{13}$ と考えると割り算ができることがあります。実際 $9 \times 2 = 18 \equiv 5 \pmod{13}$ なので $9 \equiv 5/2 \pmod{13}$ として問題無さそうです。同様に他の数の逆数も見てみましょう。

13 で割った余りの世界の逆数

x	1	2	3	4	5	6	7	8	9	10	11	12
$1/x$	1	7	9	10	8	11	2	5	3	4	6	12

0 以外の x にそれぞれについて、掛けると 1 になる逆数 $1/x$ が見つかりました。効率のよい逆数の求め方については 16.1 節で扱います。このように 0 以外の余りの逆数を余りの世界で作れました。したがって余りの世界で任意の $x, y (y \neq 0)$ に対して x/y を定義できます。つまり、この方式

の割り算を使うなら余りの世界は割り算で閉じています。この余りの世界を \mathbb{F}_p と書きます。 \mathbb{F}_p は 0 から $p - 1$ までの値からなる数の集合で、その中で足し算、引き算、掛け算、上記方式の割り算（0 を除く）ができます。一般に四則演算ができる集合を体（たい）といいます。 \mathbb{F}_p は有限個の集合の体なので有限体といいます。 p を標数といいます。無限個の集合の体としては先ほどの有理数全体の他に実数全体、複素数全体の集合などがあります。

さて、 $p = 13$ の余りの世界は体となりましたが、いつでもそうなるとは限りません。たとえば $p = 12$ の余りの世界を考えます。 $2 \times 6 = 12 \equiv 0 \pmod{12}$ となります。 0 でない 2 個の値を掛けたら 0 になってしまいました。つまり 2 や 6 の逆数は考えられないのです。もしあったらこの式の両辺に $1/2$ をかけると $6 \equiv 0 \cdot (1/2) = 0 \pmod{12}$ となり、6 も 0 ということになってしまいます。他には 3 の逆数を見つけようと、1 から 11 まで順に掛けると、

$$3, 6, 9, 0, 3, 6, 9, 0, 3, 6, 9$$

となり掛けで 1 になる数がありません。つまり 3 の逆数が存在しないのです。 12 で割った余りの世界では割り算について閉じてないので有限体でもありません。

一般に p が素数でないと 1 でない 2 個の数 a, b を使って $a \times b = p \equiv 0 \pmod{p}$ とかけるので、 a や b は逆数を持ちません。つまり合成数の余りの世界は有限体になりません。逆に p が素数のときは必ず有限体になることを示すことができます。証明は後半の 16.1 節で紹介します。

3.11 ElGamal 暗号

有限体 \mathbb{F}_p の説明をしたので ElGamal（エルガマル）暗号という公開鍵暗号を作れます。これは 1984 年に提案されました。次のような方式です。

g と p を固定し、みなに公開します。 x をランダムに選び $y := g^x \pmod{p}$ とします（以後 \pmod{p} は省略します）。 x が秘密鍵で y が公開鍵です。

公開鍵 y を公開している人に対して平文 m を暗号化して送るには次のようにします。整数 r をランダムに選び暗号文

$$\text{Enc}(m) := (g^r, my^r)$$

を計算して $\text{Enc}(m)$ を送ります。

暗号文 $(c_1, c_2) := \text{Enc}(m)$ を受け取った人は秘密鍵 x を用いて次のように復号します。

$$\text{Dec}(c_1, c_2) := c_2 / c_1^x.$$

ここで割り算は前節の有限体の中での割り算を表しています。 m を暗号化して復号すると

$$c_1 = g^r, \quad c_2 = my^r$$

により

$$\text{Dec}(\text{Enc}(m)) = \text{Dec}(c_1, c_2) = c_2 / c_1^x = my^r / (g^r)^x = mg^{rx} / g^{rx} = m$$

となって乱数 r が消えて元の m に戻ります。

ElGamal 暗号は CDH 仮定の元で秘密鍵を知らない人は復号できないことが知られています。なお、暗号化するときに乱数 r を使っているので同じ m を暗号化しても $\text{Enc}(m)$ は毎回異なる暗号文を出力します。だから ElGamal 暗号は確率的アルゴリズムです。

3.12 強秘匿性と DDH 仮定

公開鍵暗号の安全性についてもう少し考えてみましょう。3.9 節でも触れたように A さんと B さんがやりとりするとき、A さんが質問をして B さんが yes か no で答える状況はよくあります。それを盗聴して暗号を解読しようとする攻撃者は、外部の状況から B さんが yes か no のどちらかを答えたと推測していたとします。攻撃者の立場からすると次の問題を考えることになります。

「2 個の異なる平文 m, m' およびこのどちらかを暗号した c が与えられたとする。このときどちらの平文を暗号化したものか当てよ。」

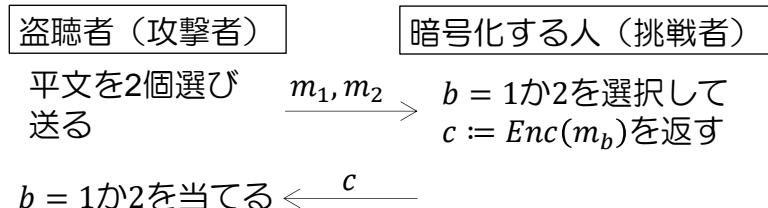


図 3.5 当てっこ問題

当てっこ問題の出題者、つまり暗号文を作る人を挑戦者といいます。暗号が安全であるためには攻撃者が負けること、つまりこの当てっこ問題ができないことが望されます。できないというのは攻撃者がどんなにがんばったとしても、何も考えずに答えて当たる確率 $1/2$ と殆ど同じにしかできないという意味です。このとき暗号文の情報は何も漏れていないと考えられます。この性質を強秘匿性（きょうひとくせい）といいます。

ElGamal 暗号で強秘匿性の条件を考えてみましょう。 $g, y := g^x$ で a を $a = m$ か $a = m'$ のどちらかとして $(c_1, c_2) := \text{Enc}(a) = (g^r, ay^r) = (g^r, ag^{rx})$ を受け取ります。 $z := c_2/m$ か c_2/m' のどちらかは g^{rx} になります。CDH 仮定から g, g^x, g^r から g^{rx} は求まりませんが、 z が g^{rx} かどうかは判るのでしょうか。答えを教えてもらっている可能性があるのだから攻撃者にとってかなり有利に思えます。つまり、

「 g, g^a, g^b が与えられ、更に $z_1 := g^{ab}$ がランダムな c に対する $z_2 := g^c$ のどちらかの z_i が与えられたとき i は 1 と 2 のどちらか判定せよ。」

これを判定 DH 問題 (DDH : Decisional Diffie-Hellman) といい、今のところ DDH も十分難しいと考えられています。DDH 仮定のもとで ElGamal 暗号は強秘匿性を持っています。DLP が解ければ CDH が解け、CDH が解けるなら DDH が解けます。つまり DLP が一番難しくて、次に

CDH, DDH の順で易しくなります。

ちょっと混乱しやすいので別の例をあげると任意の 3 次方程式が解けるなら 2 次方程式が解け、2 次方程式が解けるなら 1 次方程式が解けます。一番難しい問題は 3 次方程式を解くことです。逆に「相手が解けない」ということを仮定するなら「1 次方程式が解けない」という仮定が一番強い仮定です。これは「3 次方程式が解けない」という仮定を含むからです。

よって DLP 仮定, CDH 仮定, DDH 仮定の中では DDH 仮定が一番強い仮定です。これからいろいろな問題の困難さを仮定した暗号を紹介しますが、大抵は値そのものを求める計算問題と、それに対応した判定問題があります。

3.13 ElGamal 暗号の弱点

ElGamal 暗号は DDH 仮定の元で強秘匿性を持つでした。秘密鍵を知らない人は全然平文の情報を得られないのだから絶対安全と思えます。しかし、強秘匿性を持っていたとしても状況によっては必ずしも安全とは限らないのです。第三者は復号できないけれどもメッセージの値を改変できるかもしれませんからです。

たとえば秘密の通信をしている二人の間に攻撃者がいて $\text{Enc}(m) = (c_1, c_2)$ を盗聴し、 $(c_1, 10c_2)$ と変更して相手に渡したと考えてみましょう。すると受信者は $\text{Dec}(c_1, 10c_2) = 10my^r/g^{rx} = 10m$ と復号してしまいます。

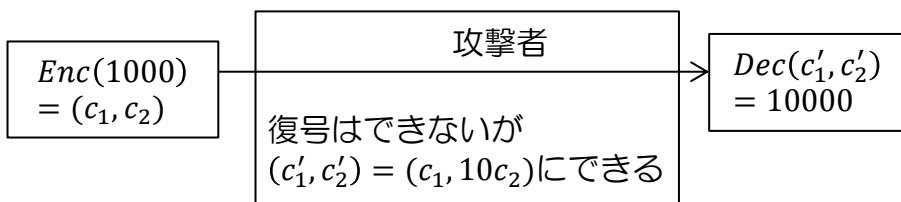


図 3.6 ElGamal 暗号に対する攻撃者による平文の改竄

もともと m という値だったのが $10m$ という値に変わりました。もし m が商品の値段だったら 10 倍の値段になってしまったのです。これでは困ります。平文が分からぬだけでなく、その値を操作されてもいけないので。そのような変更ができないことを頑強性（がんきょうせい）とか非展性（ひてんせい）といいます。

3.14 より強力な攻撃

公開鍵暗号では、攻撃者はいつでも自分の好きな平文を選んで暗号化することで情報を得る攻撃 – 選択平文攻撃 (CPA) – ができました。これに対して、選択暗号文攻撃 (CCA : Chosen Ciphertext Attack) と呼ばれる攻撃があります。攻撃者は平文ではなく暗号文を選びます。

まず攻撃者が暗号文 c を攻撃する際, c とは異なる好きな暗号文を選んで攻撃対象者に渡します。そして攻撃対象者に復号してもらい, その平文を取得します（これを問い合わせとよぶことがあります）。問い合わせで得た情報を元に c を解読する攻撃です。「え, 問い合わせなんてできないでしょ」という疑問は少し脇に置いてください。

CCA には 2 種類あります。問い合わせが攻撃対象の暗号文 c が与えられる前にしかできないものを CCA1 といいます。 c が与えられてからも, それに応じた戦略を立てて問い合わせする攻撃を適応的選択暗号文攻撃 CCA2 といいます。当然 CPA よりも CCA1, CCA2 の方が攻撃者の攻撃能力が強い状況です。3.12 節の当てっこ問題で CCA を考えると図 3.7 の様に点線で囲まれた部分が増えます。

喻えるなら, CCA1 は本試験の前に模擬試験の問題を考えて先生に答えを聞く状況です。CCA2 は本試験が始まってからも試験中にしつこく類題を出して先生に答えを聞いてから試験問題を考えている状況です。CCA1 はまだしも CCA2 はかなりずるい？です。こんな攻撃に対しても安全, つまり CCA 版当てっこ問題が解けないとき IND-CCA1, IND-CCA2 安全といいます。IND は識別不可能性 (indistinguishability) を意味します。当てっこ問題でどちらか識別できないからです。

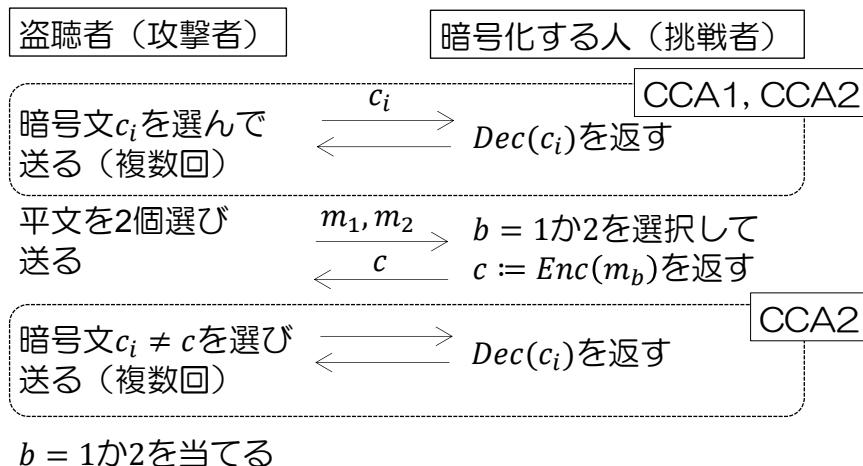


図 3.7 CCA 版当てっこ問題

なお, 3.13 節では暗号文を改竄されない頑強性も必要だと書きました。ElGamal 暗号は IND-CCA2 安全ではありません。なぜなら $c := Enc(m)$ を攻撃するとき, 問い合わせで対象暗号文 c の 2 倍の値（これは c とは異なるので可能です）を渡して $m' := Dec(2c)$ の値をもらいます。すると, それから $m'/2 = m$ を求めることで元の暗号文を復号できてしまうからです。

このように一般に IND-CCA2 安全な暗号は頑強性を持っていなければならないことが示されます。現在は IND-CCA2 安全な公開鍵暗号が最も安全であると考えられています。1998 年, ElGamal 暗号とハッシュ関数を組み合わせた Cramer-Shoup 暗号が提案されました。Cramer-Shoup 暗号は IND-CCA2 安全で実用的な暗号の一つです。

当てっこ問題は専門的にはゲームと呼ばれます。今まで「○○の問題が難しいという仮定の元で安全だと示される」という言い方をしていました。これはより正確には、その対偶である「もしそのゲームに勝つ攻撃者がいたとすると、その問題の答えを見つけることができる」という形で示されます。その厳密な証明手法は『公開鍵暗号の数理』(森山大輔, 西巻陵, 岡本龍明) [森山11] がとても詳しいです。

ところで実際のところこのような暗号文を復号してもらえるような状況は考えにくいです。そのためCCAは主に理論的観点から研究されていました。ところが1998年、BleichenbacherはCCAが現実的にありえる攻撃であることを示します[Ble98]。次節で述べるRSA暗号の標準フォーマット(PKCS #1)では平文にヘッダとパディング情報を付与して暗号化します。盗聴者(攻撃者)が普通の通信のふりをして暗号文 c_i を渡して攻撃対象者(挑戦者)に渡します。挑戦者が復号しようとして、ヘッダがおかしかったのでエラーを出します。そのエラー情報から平文の情報が少し漏れるのです。つまり問い合わせをエラー情報で代用するのです。攻撃者はこれを数十万回繰り返すことで解読します。1.10節で触れた攻撃と同様の手法ですね。2012年、Romain, Riccardo, 川本氏, Lorenzo, Graham, Joe-Kaiたちは、この手法よりもずっと攻撃回数が少なくてすむパディングオラクル攻撃を提案します[BFK⁺12][BFK⁺13]。このため、現在ではCCAに対しても安全であることが望ましいです。また暗号プロトコルを実装する際、復号時のエラーメッセージに詳しい情報を載せてはいけません。

3.15 攻撃と安全性のまとめ

公開鍵暗号に対する攻撃者のいろいろな攻撃や、暗号が持つべき安全性に関する性質がでてきました。ここで少しまとめておきましょう。

暗号文が与えられたときに平文に戻せてはいけません。これは当然ですね。暗号が持つべきこの性質を一方向性といいます。それ以外にも次の性質を持つのが望ましいことを紹介しました。

強秘匿性：暗号文からもとの平文のどんな情報もえられないこと。

頑強性：暗号文をいじって、もとの平文と関係のある別の暗号文を作れないと。

一般に強秘匿性よりも頑強性を持つ暗号がより安全です。たとえばElGamal暗号は強秘匿性を持ちましたが、頑強性は持ちませんでした。

それから公開鍵暗号の攻撃には攻撃者が持つ能力に応じていくつかの種類があります。攻撃したい暗号文を c とします。

選択平文攻撃(CPA)：暗号文 c を受けとる前後に自分で選んだ平文に対応する暗号文を得られる。

選択暗号文攻撃(CCA1)：暗号文 c を受け取る前に、自分で選んだ c 以外の暗号文に対応する平文を得られる。

適応的選択暗号文攻撃 (CCA2) : 暗号文 c を受け取る前後に、自分で選んだ c 以外の暗号文に対応する平文を得られる。

公開鍵暗号では、攻撃者は公開鍵を使って自分で選んだ平文に対応する暗号文を作れるので CPA 安全であることは最低限の要件です。後者になるほど攻撃者の能力は高いです。

CPA, CCA1, CCA2 に対して強秘匿性を持つ暗号を IND-CPA 安全, IND-CCA1 安全, IND-CCA2 安全といいます。同様に頑強性に対してもこれらの攻撃に対して安全な性質を考えられます。つまり CCA2 に対して頑強性を持つ暗号が最も安全と考えられます。ただ CCA2 に対して強秘匿性を持てば頑強性を持つことが知られています。したがって強秘匿性を持つ IND-CCA2 安全な（つまり同時に頑強性も備える）公開鍵暗号が理想です。

3.16 RSA 暗号

RSA 暗号は ElGamal 暗号とは別の公開鍵暗号です。1977 年に Rivest, Shamir, Adleman たちにより初めて公開鍵暗号を実現したものとして広く知られています。ただイギリスの通信電子セキュリティグループ (CESG) によると 1970 代初頭に Cocks たちが彼らよりも先に公開鍵暗号の概念や RSA 暗号などを考えていたそうです [Coc73]。

p, q を相異なる素数とし $n := pq$ とします。整数 e を選び

$$de \equiv 1 \pmod{(p-1)(q-1)}$$

となる整数 d を求めます。 (n, e) が公開鍵で d が秘密鍵です。RSA 暗号は平文 m に対して暗号文 c を

$$c = \text{Enc}(m) := m^e \pmod{n},$$

で計算します。復号は暗号文 c に対して

$$\text{Dec}(c) := c^d \pmod{n}$$

とします。後半の 16.3 節で紹介する定理を使うと任意の整数 x に対して $x^{p-1} - 1$ は p で割り切れ、 $x^{q-1} - 1$ は q で割り切れます。つまり $m^{(p-1)(q-1)} - 1 = (m^{q-1})^{p-1} - 1 = (m^{p-1})^{q-1} - 1$ は p でも q でも割り切れます。 p と q は互いに素な素数なので $m^{(p-1)(q-1)} - 1$ は $n = pq$ で割り切れます。すなわち

$$m^{(p-1)(q-1)} \equiv 1 \pmod{n}.$$

d, e の作り方から、ある整数 s が存在して

$$de = 1 + s(p-1)(q-1).$$

よって

$$\text{Dec}(c) \equiv c^d \equiv (m^e)^d = m^{de} = m^{1+s(p-1)(q-1)} = m \left(m^{(p-1)(q-1)} \right)^s \equiv m \pmod{n}$$

となり RSA 暗号は正しく復号できることが分かります。 $(n, e, c = m^e)$ から m を求める問題を RSA 問題といいます。 n が十分大きくていくつかの細かい条件を満たすとき、RSA 問題は難しいと考えられています。RSA 問題が困難であるという仮定を RSA 仮定といいます。

$n = pq$ の p, q が求められれば d は容易に求めることができます。RSA 問題は n が与えられたときに n を素因数分解する問題とほぼ同等の難しさを持っていると考えられています。

RSA 暗号のアルゴリズムには乱数が登場しないので決定的なアルゴリズムです。したがってこのままでは安全な公開鍵暗号とはいえません。また、 $c_1 = m_1^e, c_2 = m_2^e$ が与えられたとき $c_1 c_2 = (m_1 m_2)^e$ が成り立つので復号しなくても $m_1 m_2$ の暗号文を作ることができます。ElGamal 暗号と同じく頑強性も持っています。

1994 年、Bellare, Rogaway たちはある種の関数があると、いくつかの条件の元で IND-CCA2 安全な暗号を作ることができる枠組みを提案します。その枠組みは OAEP (Optimal asymmetric encryption padding) と呼ばれます [BR94]。2001 年、藤崎氏、岡本氏、Pointcheval, Stern たちは OAEP が安全であることを正しく証明します [FOPS04]。OAEP を RSA 暗号に適用することで IND-CCA2 安全な暗号 RSA-OAEP を作れます。

3.17 公開鍵基盤と認証局

さて、現在最も強力な IND-CCA2 安全な公開鍵暗号があればもう何も問題なく安心して秘密に通信できるのでしょうか。残念ながらそうとは限らないので困ってしまいます。

A さんが B さんと公開鍵暗号を使って通信するとします。A にとって B の公開鍵は本当に B のものなのでしょうか。A が B とは知り合いで普段からメールや web などでやりとりをしていて、そのとき B 自身が常に自分の公開鍵をそれらに添付していればほぼ本人のものと考えて問題ないでしょう。実際に会って公開鍵を手渡しすると確実です。B にとっての A も同様です。

しかし B がショップの運営者で、A は初めて B と通信をするときを考えます。A は B の公開鍵を持っていないので A は B から B の公開鍵をもらわなければなりません。もし、その通信路が DH 鍵共有のところで述べた中間者攻撃を受けているとその公開鍵を使った通信は安全になりません。つまり A が B の公開鍵をもらうところで何か安心できる方法が必要になります。

公開鍵が正しいことや、通信経路で改竄されていないことを検証するには 4 章で説明するメッセージ認証符号 (MAC) やデジタル署名を使います。しかしデジタル署名には公開鍵が必要です。その公開鍵鍵はどうやって送るのでしょう。またもや元に戻ってしまいました。互いが互いを必要としている循環しています。

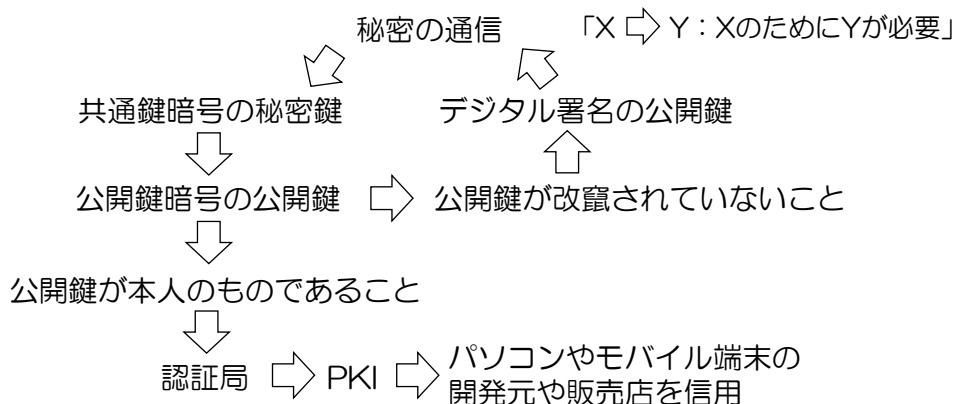


図 3.8 公開鍵とデジタル署名と認証局

インターネット上の B のサイトが本当にその B のものなのは暗号技術では知りようがありません。B の公開鍵が本当に B のものであると証明するものを公開鍵証明書といいます。一般的には国際電気通信連合（ITU : International Telecommunication Union）が定めた X.509 という規格が使われます。

公開鍵証明書の発行や失効の手続きをするところを認証局（CA : Certificate Authority）といいます。認証局にはたとえば VeriSign を買収した Symantec や GlobalSign があります。

認証局によって証明書を扱う枠組みが公開鍵基盤（PKI : Public Key Infrastructure）です。PKI では、信用できる認証局が保証している認証局は信用できるという形で信用の輪を広げます。

B が認証局 X に証明書の発行依頼をすると、認証局 X は B の身元確認をして B の公開鍵証明書（エンドエンティティ証明書とかサーバ証明書といいます）を作成します。B のサーバ証明書は認証局 X の公開鍵でその正しさが検証されます。認証局 X の証明書（CA 証明書）は別の認証局 Y の公開鍵で検証されます…と連鎖が続きます。

最終的に証明書の検証を自分自身の公開鍵で行うものにたどり着きます。その証明書をルート証明書、その証明書を持つ認証局をルート認証局といいます。ルート証明書はその認証局のブランドや実績によって信頼されます。暗号的には無条件に受け入れるので非常に重要です。いくつかのルート証明書はブラウザに最初から入っています。私たちがブラウザで安心して公開鍵暗号を使えるのは、初めて行くサイトでもこの枠組みによってきちんと検証されているからです。

ブラウザに必要なルート証明書が含まれていないとサーバ証明書の検証が正しく行えず、警告が出ることがあります。たとえば 2015 年 3 月の時点で Firefox というブラウザで、総務省が管理する政府認証基盤（GPKI）のサイト <https://www.gPKI.go.jp/> に接続すると「接続の安全性が確認できない」と表示されました。こういった場合『自己署名証明書をダウンロードして「信頼する」にチェックしてください』と指定されることがあります、不用意に OK と押すのはよくありません。

なぜならそういう証明書を http 経由で取得した場合、通信経路は暗号化されず平文のままでです。

そのためその証明書が改竄されていないことを検証できず、本物である保証を得られないからです。振り込め詐欺の電話で「おれは本物だから安心してね」と言われて信用するのと変わらないことを理解してください。オレオレ詐欺に掛けて「オレオレ証明書」と呼ばれることがあります。

証明書には fingerprint (フィンガープリント：拇印) という印がついています。その fingerprint が指定されたものと同一であれば、証明書が改竄されていないことが保証されます。もちろんその fingerprint は同じサイトからとってきてはいけません（それも改竄されているかもしれない！）。別の正しいと信じられるサイトを見るか、上記 GPKI の証明書の fingerprint なら官報でも確認できます。

最後に無効化の話をします。サーバ証明書が不要になれば認証局に無効化の依頼をします。すると認証局はその証明書を失効し、失効リスト (CRL : Certificate Revocation List) と呼ばれる失効した証明書一覧に登録します。実はブラウザが証明書を検証するときは証明書自体の検証の他に、最新の CRL を取得して証明書が失効されていないことを確認しています。そうしないと失効した証明書を受け入れてしまう危険性があるからです。

このように公開鍵の正しさの確認には面倒な問題がつきまといます。PKI による正しさを維持する運営が必要不可欠です。

3.18 PKI への攻撃

PKI における認証局の証明書は重要な役割を果たします。ですから、そこを攻撃されると大きな被害が出ます。

2011 年 COMODO や DigiNotar という認証局への攻撃があり、認証局が偽のサーバ証明書を発行してしまう事件がありました。DigiNotar は信頼を失い自身のルート証明書を無効化され、最終的に自己破産しました。

2012 年 Flame というマルウェアは偽物の Microsoft の Windows Update を本物に見せかける証明書を作って攻撃をしました。証明書の検証に MD5 という安全ではないハッシュ関数を使っていたのが原因の一つです [Mic12]。

2015 年 Lenovo 製のパソコンの一部にプレインストールされていた Superfish というソフトウェアの問題点が明らかになりました。Superfish はブラウザで見ているサイトに広告を挿入するためのものです。そのために独自の証明書を正しいルート証明書としてシステムに認識させていました。つまり中間者攻撃と同じ原理で動作していたのです。しかも独自の証明書の公開鍵と秘密鍵は全てのパソコンで同一でプログラムに埋め込まれていました。そのため秘密鍵を知っている第三者はそのパソコンにとって正規の証明書と区別のつかない証明書を容易に作れます。購入したパソコンが最初からそのような状態になっているとユーザは防御しようがなく、PKI の信頼性を根本から損なわせるものです。同時に現在の PKI が「何かを信用する」ということの上に成り立っている技術だということを再認識させられます。

3.19 この章のまとめ

整数を素数で割った余りの世界を有限体といいます。有限体では普通の足し算、引き算、掛け算の他に割り算も定義できます。有限体上で巾乗の計算は易しいです。しかしその逆、何乗したらその値になるかを求めることが（離散対数問題）は難しいです。暗号にとってこの一方向性が非常に重要な性質です。

離散対数問題が難しいという性質を用いて DH 鍵共有や ElGamal 暗号という公開鍵暗号を作ります。これにより二者間で秘密に通信できます。公開鍵暗号の安全性にはいくつか段階があり IND-CCA2 安全なものが一番よいとされています。

公開鍵暗号はその公開鍵が本人のものであることを別の手段で示す必要があり、現在は公開鍵基盤（PKI）を用いて検証されています。公開鍵を安全に使うためには PKI の絶え間ない維持が必要です。

第4章

認証

公開鍵基盤では、公開鍵の証明書が必要でした。与えられた公開鍵が正しいこと示すにはなんらかの認証機構が必要です。認証には、あるメッセージが書き換えられていないことを検証するメッセージ認証や、メッセージを作った人が当人であることを検証するデジタル署名などがあります。

この章ではそれらの技術で使われるハッシュ関数を紹介します。その後、ハッシュ関数や乱数を共通鍵暗号や公開鍵暗号と組み合わせることで認証や署名を作る方法を紹介します。ブラインド署名という一風変わった署名も紹介します。

4.1 ハッシュ関数

公開鍵暗号などを使うときには、その通信データが途中で改竄されていないことを確認する必要があります。送信したデータのどこか1箇所でも変更されれば検出できるようにしたいのです。そのため必要な技術の一つがハッシュ関数です。

一般にハッシュ関数とは任意の大きさのデータ m に対して、ある決まった手順にしたがって固定ビットサイズの値 $H(m)$ を出力する関数 H です。どんなに大きいデータであっても同じサイズ（たとえば 20 バイト）の値です。同じ入力に対してはいつも同じ値を返す決定的な関数です。

ハッシュ関数と聞くとプログラミング言語で使われる連想配列や辞書と呼ばれるデータ構造に使われるものを思い浮かべるかもしれません。そのような用途のハッシュ関数は、高速に計算できて、ランダムな入力データに対して出力値がある程度一様に分布すれば十分です。ただ 2011 年、Hashdos と呼ばれる攻撃が提案されました [aKW11]。これは web サーバにハッシュ値が同じ値になる多数のデータを与えることで CPU のリソースを奪う攻撃です。したがって近年は与えられたハッシュ値になる多数のデータを求めていくハッシュ関数が望まれています（別の手法で Hashdos を回避することもできます）。2012 年に提案された SipHash [AB12] は、ある程度そのような性質を持ち、かつ短いメッセージを高速に計算できるので注目されています。

暗号プロトコルで使われるハッシュ関数は、そのようなハッシュ関数よりもより強い性質が求められます。普通のハッシュ関数と区別するために、暗号学的ハッシュ関数ということがあります。

たとえば 1 バイトごとの値を加算していき、最後に合計（チェックサムと呼ばれることがあります）を出力する関数を考えてみます。どこか 1箇所値を変更するとチェックサムが変わるので変更されたと判断できます。しかしデータのどこか 2 箇所の値を交換した場合、合計は変わらないのでチェックサムを確認するだけでは変更されたことを検知できません。暗号学的ハッシュ関数はそんなことができないように設計されています。詳しくいうと、暗号学的ハッシュ関数は次の性質を満たします。

1. 原像計算困難性：与えられた h に対して $H(m) = h$ となる m を見つけることが困難である。
 2. 第二原像計算困難性： m_1 が与えられたときに $H(m_1) = H(m_2)$ となる $m_2 (\neq m_1)$ を求めするのが困難である。
 3. 衝突困難性：相異なる m_1 と m_2 で、 $H(m_1) = H(m_2)$ となるものを見つけることが困難である。
1. はハッシュの値から元のデータを再現できないことを要求しています。2. と 3. は同じように見えるかもしれませんが全然違います。喻えるなら 2. は学校のクラスの中で自分と同じ誕生日の人を見つける問題です。3. はクラスの中で同じ誕生日である人の組を見つける問題です。

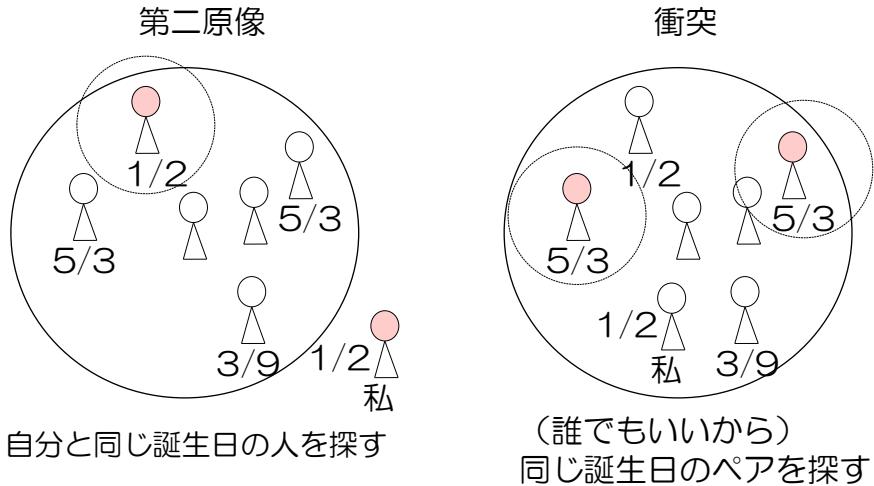


図 4.1 第二原像と衝突の違い

1 年が 365 日で誕生日はどの日も等確率、クラスの人数が 40 人としましょう。自分と同じ誕生日の人が見つかる確率 p_1 は、1 から自分と同じ誕生日の人が誰もいない確率を引けばよいので

$$p_1 = 1 - \left(\frac{364}{365} \right)^{40} \approx 0.10$$

となります。それに対してクラスの中に同じ誕生日の人の組がいる確率 p_2 は、1 から全員の誕生

日が異なる確率を引けばよいので

$$p_2 = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{365 - 40 + 1}{365} \approx 0.89$$

となります。これは上の 0.10 に比べて非常に高い確率ですね。この現象は誕生日のパラドックスと呼ばれます。

仮にある現象の確率が $1/2$ を越えると「その現象が発生する」と定義することにします。ハッシュ関数の出力の長さが n ビットのときはクラスの人数が 2^n であることに相当します。すると第二原像が発生する、つまりあるハッシュ値と同じ値になるものが見つかる確率が $1/2$ となるのは全体の半分近く集まつたときです。これは $O(2^n)$ の計算量です。

それに対して衝突する 2 個の値を見つけるには、計算過程を省略しますが $O(2^{n/2})$ 個集めればよいです。第二原像に比べてそのルートの計算量ですむのをずっと易しいのです。誕生日のパラドックスと同じ現象です。2 番目の問題より 3 番目の問題の方が易しいので、ハッシュ関数に要求される条件としては第二原像計算困難性よりも衝突困難性の方が求められます。したがって出力が n ビットの理想のハッシュ関数の強度は $O(2^{n/2})$, $n/2$ ビットセキュリティです。

2004 年、ハッシュ関数 MD5 は衝突困難性を持っていないことが示されました。現在は暗号学的ハッシュ関数の性質を満足していないため使ってはいけません。また SHA-1 も本来なら 80 ビットのセキュリティであるはずが、それよりもずっと小さい強度しかないことが判明しています。そのため日本の暗号技術検討会及び関連委員会（CRYPTREC）では 2013 年に SHA-1 を運用監視リストに移行しました。今は SHA-2 (SHA-256 以上) を使うことが推奨されています。その次の規格である SHA-3 は 2012 年に選定され、まもなく標準規格として登場するところです。

なおファイルをダウンロードする web ページでファイルへのリンクと同じページに SHA-1 の値が書かれていることがあります。しかし、これはセキュリティ的には何の安全も保証しないことに注意してください。もし、サーバが乗っ取られてページを書き換えられたり、通信経路で改竄されたりしているなら、ファイルだけでなく SHA-1 の値もいっしょに書き換えられるからです。ダウンロード中にファイルが壊れてないかを確認する程度にしか意味はありません。メッセージやデータが改竄されていないことを確認するには 4.3 節で紹介する技術を用います。

4.2 ハッシュ関数の構成

ハッシュ関数は（殆ど）任意の長さのメッセージに対して固定長の長さを出力します。いきなり任意の長さに対応したハッシュ関数を作るのは難しいので、まず固定長の長さの入力を少しだけ小さくする圧縮関数を用意します。圧縮関数という名前ですが通常のデータの圧縮とは違い元のデータを復号できるわけではありません。

メッセージ m が与えられたとき、まず m をある固定サイズ l のブロックに分割します。分割するときはパディングと呼ばれる処理をします。これは m にビット 1 とビット 0 を必要なだけ追加

し、最後に m の長さをくっつけて全体の長さが丁度 l の倍数になるように調節する作業です。

$$m = m_1 || m_2 || \dots || m_n.$$

ここで $||$ はデータをそのまま連結することを示します。初期値 h_0 を固定し、圧縮関数 f を用いて $h_1 := f(h_0, m_1)$ を求めます。次に $h_2 := f(h_1, m_2)$ を求めます。これを繰り返し適用することで最終的に $h_n := f(h_{n-1}, m_n)$ を求め、 h_n をハッシュ関数の値として出力します。圧縮関数を繰り返し適用するので逐次的に処理する反復型ハッシュ関数（iterated hash function）と呼ばれます。SHA-1などのよく使われるハッシュ関数はこの形をしています。

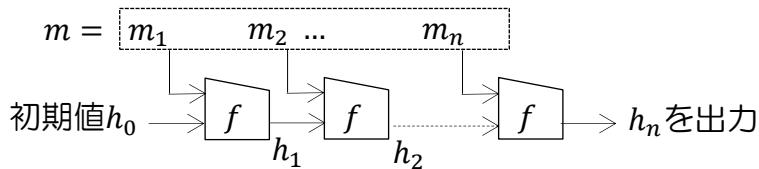


図 4.2 反復型ハッシュ関数

4.3 メッセージ認証符号

メッセージが改竄されていないかどうかを確認する技術をメッセージ認証符号（MAC : Message Authentication Code）といいます。MAC は秘密鍵 k と任意のメッセージ m に対してある値 $t = \mathcal{M}(k, m)$ を出力するアルゴリズムです。 \mathcal{M} を MAC を計算する関数、 t のことを MAC 値といいます。A さんと B さんの間で秘密の鍵 k を共有しておき、A が B にメッセージ m を送ります。そのとき B には $(m, t = \mathcal{M}(k, m))$ を送ります。B は受け取った (m, t) に対して

$$t = \mathcal{M}(k, m)$$

を確認します。もし等号が成り立たなければメッセージが改竄されたと分かります。

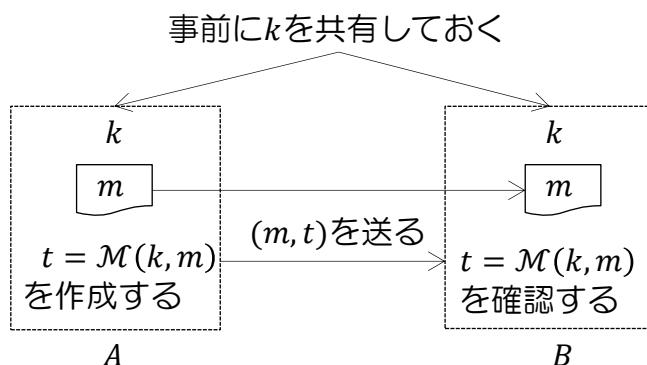


図 4.3 MAC によるメッセージの検証

k を秘密に共有しておいて、異なる m に対して何度も MAC 値を計算して使います。だから盗聴者は沢山の $\{(m_i, \mathcal{M}(k, m_i))\}$ を集められます。この情報から k が漏洩すると困ります。また仮に秘密鍵 k が漏洩しなかったとしても、A が作っていないメッセージに対する MAC 値 t を勝手に作られても困ります。B はそのメッセージを A が作成したものと判断してしまうからです。すなわち MAC には

「攻撃者が好きに選んだメッセージに対する沢山の MAC 値を取得できたとしても、攻撃者が取得していないメッセージに対する MAC 値を偽造できない。」

という性質が求められます。これを適応的選択平文攻撃に対して存在的偽造困難 (EUF : existentially unforgeable) であるといいます。

MAC の構成法はいくつかあります。一つは 1.9 節で紹介したブロック暗号を CBC モードで使う方法です。初期化ベクトル IV は 0 固定で利用します。メッセージ m の先頭に m の長さを連結したものを秘密鍵 k で暗号化します。できた暗号文の最終ブロックを MAC 値として出力します。

$$\text{CBC-MAC}(k, m) := \text{CBC-Enc}(k, \text{IV} = 0, m \text{ の長さ } ||m) \text{ の最終ブロック}.$$

暗号化のときと違って IV をランダムに作るべきではありません。なぜなら 0 固定でないと、IV を送る必要があり、IV を改竄する攻撃が存在するからです。また CBC-MAC にはメッセージの先頭に長さを付加しないと 2 組のメッセージと MAC 値のペアから別のメッセージと MAC 値のペアを生成する攻撃があることも知られています。そのため可変長メッセージに対する CBC-MAC は使うべきではありません。固定長メッセージを扱う場合は安全ですが、2013 年の CRYPTREC の運用監視リストに入っています。

2003 年、岩田氏と黒澤氏は CBC-MAC を改良して OMAC (One-Key CBC MAC) を開発しました [岩田 06]。これは 2005 年、CMAC という名前で NIST の MAC の標準となっています [Dwo05]。

4.4 HMAC

MAC にはブロック暗号を使ったのもの以外にハッシュ関数を使う方法もあります。ハッシュ関数を使った MAC を HMAC (Hash-based MAC) といいます。HMAC は n ビット出力のハッシュ関数 $H(x)$ と n ビットの鍵 k に対して

$$\mathcal{H}(k, m) := H((k \oplus opad) || H((k \oplus ipad) || m)) \quad (4.1)$$

と定義されます。ここで $opad$ や $ipad$ はある定数、 $||$ はデータの連結を意味します。このようにハッシュ関数を 2 回使うことで安全な MAC を構成できることが知られています。

これを面倒だと思って

$$\mathcal{H}^\times(k, m) := H(k || m)$$

をしてしまいたくなるかもしれません。しかしこれは可変長メッセージの場合に安全ではないの

でやってはいけません。SHA-1などの反復型ハッシュ関数に対して伸長攻撃（length-extension attack）と呼ばれる攻撃を受けます。

どういう攻撃か少し紹介しましょう。今秘密の k に対する、あるメッセージ m とその HMAC もどき $h := H(k||m)$ を手に入れたとします。ハッシュ関数の内部の最終ブロック m_n は $k||m$ の最後にパディング p をつけたものです。 p は m の長さだけから決まる値です。

$$m_n := (k||m) \text{ の最後の残り } ||p.$$

反復型ハッシュ関数の出力はこの m_n と、一つ前の圧縮関数の値 h_{n-1} から圧縮関数 f によって決まっていました。

$$h = H(k||m) = f(h_{n-1}, m_n).$$

$m||p$ の後ろに更に適当なデータ c (1 バイトでいい) をつけてメッセージを作ります。すると $m' := m||p||c$ に対する HMAC もどき $H(k||m')$ は h と c から求められます。

$$\mathcal{H}^\times(k, m') = H(k||m||p||c) = f(h, c \text{ にパディングしたもの})$$

となって k の値を知らなくても m とは異なる m' に対する MAC を構成できてしまいました。

したがって HMAC もどき $H(k||m)$ は安全ではないのです。じゃあ $H(m||k)$ ならうまくいくのではと思われるかもしれません、この場合はまた別の攻撃方法が知られています。したがって式(4.1)で定義された HMAC を使うべきです。

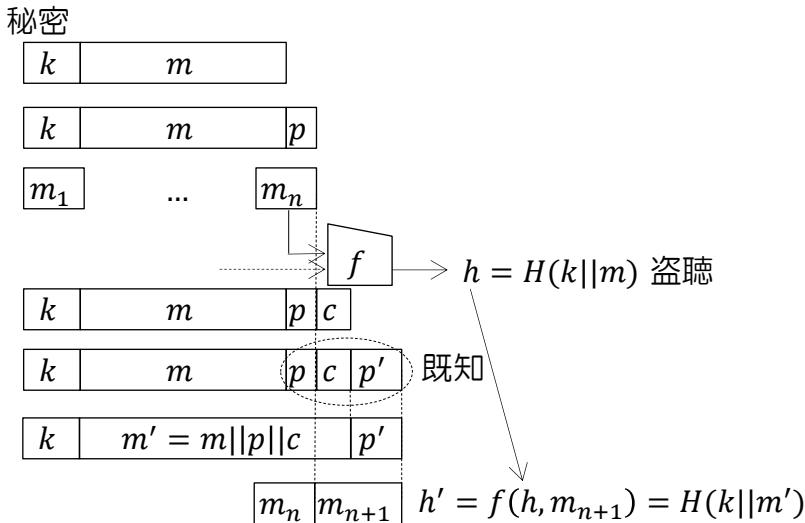


図 4.4 伸長攻撃

4.5 認証付き暗号

データを安全に秘密に通信するには暗号化と認証の両方が必要です。暗号化と MAC との組み合わせは様々な方法が考えられます。しかし組み合わせ方によっては安全性が損なわれることがあります。1.9 節で触れた TLS3.0 に対する攻撃 POODLE も CBC モードの不適切な扱いをついた攻撃の一つです。そのため、暗号化と認証を行なう認証付き暗号（AEAD : Authenticated Encryption with Associated Data あるいは単に AE）が考えられています。

AEAD の一つに GCM (Galois/Counter Mode) があります。GCM は 2.4 節で紹介した CTR モードに認証機構を組み込んでいます。CBC モードよりも性能がよく安全なため広く使われています。SSL の後継の TLS (Transport Layer Security) のバージョン 1.2 からは GCM をサポートしています。2013 年からよりよい AEAD を目指した暗号方式のコンテストが始まっています^{*1}。

4.6 デジタル署名

デジタル署名とは、あるデータがその作者によって作られたことを検証する仕組みです。電子署名や単に署名ということも多いようです。他人がその作者になりすまして署名を作ったり、別のデータに対する偽の署名を作ったりできないようになっています。そのため署名したことを後で否認することができません。またそのデータが改竄されていないことも確認できます。

基本アイデアは公開鍵暗号の原理を使う点にあります。公開鍵暗号は、公開鍵で暗号化されたメッセージは秘密鍵を持っている人だけが復号できるのでした。逆にもし秘密鍵でメッセージを暗号化したらどうなるでしょう。その暗号文は公開鍵を使えば誰でも復号できます。もちろんこれは暗号の意味をなしていません。しかしそのメッセージを暗号化できるのはその秘密鍵を持っている人だけなのです。つまり A さんがメッセージ m を自分の秘密鍵 K によって暗号化し $(m, \text{Enc}(K, m))$ を公開したとします。すると、A さんの公開鍵 K' を知っている人は誰でも

$$\text{Dec}(K', \text{Enc}(K, m)) = m$$

を確認することで $(m, \text{Enc}(K, m))$ は A さんが作成したと判断することができます。

MAC はメッセージが改竄されていないかを確認するためだけに使われます。MAC は二人の間で秘密の鍵を共有する必要がありました。人が増えるとそれぞれの組の間で秘密の鍵を共有する必要があります、管理する鍵が膨大になってしまいます。デジタル署名は秘密の鍵を共有することなく誰でも署名を検証できます。MAC とデジタル署名の関係は、共通鍵暗号と公開鍵暗号の関係に似ています。

実際のところ、公開鍵暗号をそのままデジタル署名として使えるわけではありません。通常、RSA 暗号以外の公開鍵暗号では平文の空間と暗号文の空間が異なるからです。しかし公開鍵暗号

^{*1} <http://competitions.cr.yp.to/caesar.html>

である RSA 暗号, ElGamal 暗号, Cramer-Shoup 暗号に対応して, RSA 署名, ElGamal 署名, Cramer-Shoup 署名などが作られています.

4.6.1 RSA-FDH 署名

まずは式が比較的簡単な RSA 暗号とハッシュ関数を組み合わせて作られた RSA-FDH (FDH : Full Domain Hash) 署名を紹介しましょう [BR93].

RSA 暗号の公開鍵を (n, e) , 秘密鍵を d とします. 更に Full Domain ハッシュ関数 H を用意します. Full Domain とはハッシュ値が公開鍵 n と同じサイズとなるハッシュ関数のことです.

メッセージ m に対する署名 :

$$s := H(m)^d \bmod n.$$

メッセージ m に対する s の検証 :

$$H(m) \equiv s^e \bmod n$$

が成り立つとき署名を受理, そうでないとき棄却 (ききやく) します.

ハッシュ関数を使うところ以外は RSA 暗号とほぼ同じです. この署名は RSA 仮定とランダムオラクルモデルの元で安全です. ランダムオラクルモデルというのはハッシュ関数を出力が完全にランダムになる理想の関数であるという仮定を置いたモデルのことです. n が 2048 ビットならハッシュ関数も 2048 ビット必要なのであまり効率がよいとはいえない. この署名を構成した Bellare と Rogaway は確率的な要素を追加し, RSA-FDH よりも効率のよい RSA-PSS (Probabilistic Signature Scheme) も提案しています [BR96].

4.6.2 DSA

次に ElGamal 署名の改良系であり, 現在標準的に使われている DSA (Digital Signature Algorithm) [KSD13] という署名を紹介しましょう.

鍵生成 :

1. ハッシュ関数 H を決める.
2. (p, q, g) を DH パラメータ (3.3 節参照) とする. ただし p は 2048 ビットの素数で, q はハッシュ関数の出力ビット長と同じ (たとえば 256 ビット) 大きさのものを選ぶ. g は生成元で $g^q \equiv 1 \pmod p$. $p - 1$ は q で割り切れる.
3. $0 < x < q$ となる x をランダムに選び $y := g^x \bmod p$ とする. (g, p, q, y) が公開鍵で x が秘密鍵.

メッセージ m に対する署名 :

1. k をランダムに選び $r := (g^k \bmod p) \bmod q$ とする.

2. $s := (H(m) + xr)/k \bmod q$ とする. (r, s) を m に対する署名とする.

メッセージ m に対する (r, s) の検証 :

1. $0 < r, s < q$ を確認する. 範囲外なら棄却する.

2.

$$\begin{aligned} w &:= s^{-1} \bmod q, \\ u_1 &:= H(m)w \bmod q, \\ u_2 &:= rw \bmod q, \\ v &:= (g^{u_1}y^{u_2} \bmod p) \bmod q \end{aligned}$$

を求める.

3. $v = r$ なら署名を受理し, そうでないなら棄却する.

p と q の二種類の剩余を使います. これは p が大きいので q で割ってよいところは q を使って効率化するためです. 正しいメッセージに対して署名が受理されることを確認しましょう. 式がやや複雑なのですが, 一つ一つは難しくはありません. 上記手順にしたがって m に対する署名 (r, s) を作ります. すると,

$$\begin{aligned} w &= s^{-1} = k/(H(m) + xr), \\ g^{u_1}y^{u_2} &= g^{H(m)w}(g^x)^{rw} = g^{(H(m)+xr)w} = g^k = r \end{aligned}$$

となって検証が通ります.

4.7 ブラインド署名

1982年, Chaum はブラインド署名 [Cha83] という面白い概念を提案しました. これは署名者がどんなメッセージに署名しているか分からぬまま署名させるというものです.

実際にするならこんな感じでしょうか. Aさんがメッセージ m を Bさんに署名して欲しいとします. まず A はメッセージを封筒に入れます. 封筒には署名欄のところだけ切り抜いて穴を開けておきます. B は封筒に入ったままメッセージを読まずに署名欄に署名します. A は封筒から B の署名が入ったメッセージをとりだします.

現実世界で何に署名しているか分からぬまま署名するのはとても怖いことです. 詐欺にあうかもしれません. しかし, ブラインド署名は電子投票や電子マネーなど匿名性を担保したままその内容を保証したいときの利用が考えられています. 選挙管理委員に投票内容を教えないまま投票用紙としての正当性を選挙管理委員に保証してもらったり, 電子マネーの紙幣の番号を教えずに銀行にそのマネーの保証をしてもらったりします. 電子投票については 14.4 節でもう少し詳しく考察します.

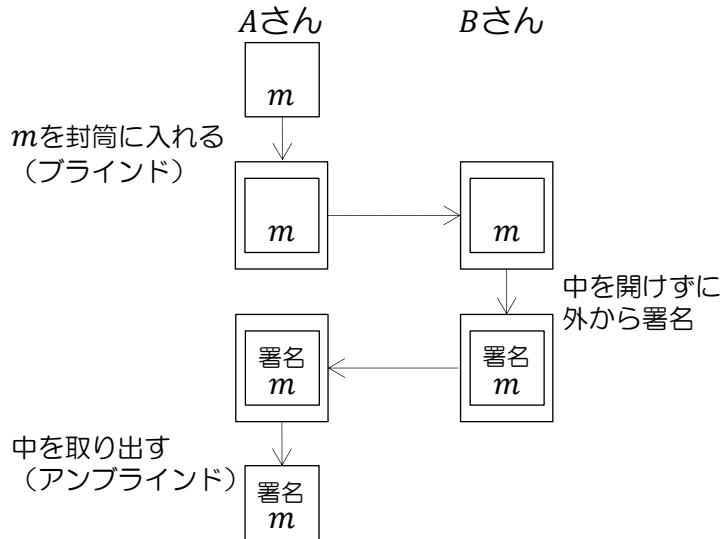


図 4.5 ブラインド署名の概念処理

ブラインド署名の枠組みを考えます。まず B が使う電子署名の公開鍵と秘密鍵を生成します。A はメッセージ m を誰も読めない形 $x := \text{Blind}(m)$ にして B に渡します（ブラインド処理）。B は x に署名をして $y := \text{Sign}(x)$ を A に返します（署名処理）。A は y から $\text{Unblind}(y)$ を求めます（アンブラインド処理）。

$$\text{Unblind}(y) = \text{Unblind}(\text{Sign}(\text{Blind}(m))) = \text{Sign}(m)$$

なっていると A が m を公開すれば誰もが $(m, \text{Sign}(m))$ は B が署名したということを確認できます。RSA 署名を使ったブラインド署名を紹介しましょう。

鍵生成 : B は RSA-FDH 署名の署名鍵 d と検証鍵 (n, e, H) を用意する。

ブラインド : A はメッセージ m に対して乱数 r を用いて

$$x := \text{Blind}(m) := r^e H(m) \bmod n$$

を計算し x を B に渡す。

署名 : B は x に対して署名 y を計算して A に返す。

$$y := \text{Sign}(x) := x^d \bmod n.$$

アンブラインド : A は y に対して

$$s := \text{Unblind}(y) := y/r \bmod n$$

を計算する。 (m, s) がメッセージ m に対する B の署名である。

署名になっていることの確認 :

$$y \equiv x^d \equiv (r^e H(m))^d = r^{ed} H(m)^d \equiv r H(m)^d \pmod{n}$$

なので

$$s \equiv y/r \equiv H(m)^d \pmod{n}$$

となり，A は m に対する B の RSA-FDH 署名を取得できていることが分かります。

4.8 秘匿共通集合計算

秘匿共通集合計算 (PSI : Private Set Intersection) とは，あるデータの集合に対して A さんと B さんがそれぞれその部分集合を持っているとき，お互いに何を持っているのか秘密にしたまま共通する要素を特定する技術です。

物騒な例ですが，政府が極秘でテロリストの容疑者の一覧を持っているとします。飛行機会社は搭乗者の中にテロリストがないか調べたいとします。PSI を使うことによって搭乗者の情報を政府に知らせることなく，また政府はテロリストのリストを飛行機会社に知らせずにその要求を満たせます。

他にも通信情報，遺伝子情報など様々なプライベートな情報に対する操作で PSI が使える部分はあるでしょう。13.2 節では化合物データベースの検索への応用例を紹介します。PSI は様々な実現方法が提案されていますが，ここではブラインド署名を使った方法を紹介しましょう。

A さんが $S_A := \{ a_i \mid i = 1, \dots, N_A \}$, B さんが $S_B := \{ b_j \mid j = 1, \dots, N_B \}$ を持っているとします。A が B に問い合わせることで A は $S_A \cap S_B$ を求めます。

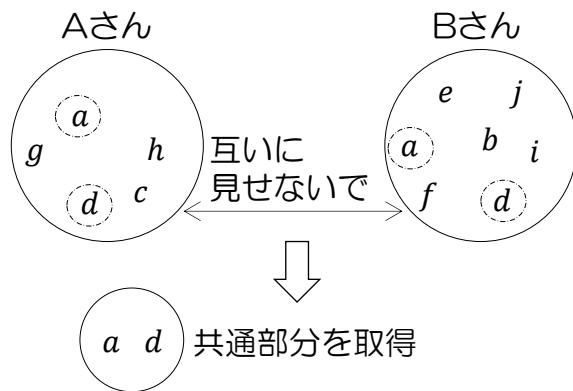


図 4.6 PSI

1. B はブラインド署名の初期化を行う。通常のハッシュ関数を H とする。
2. A は各 a_i に対してブラインド署名プロトコルを使って $s_i := \text{Sign}_B(a_i)$ を取得する。
3. B は各 b_j に対して署名をしてからハッシュをとり $\{ b'_j := H(\text{Sign}_B(b_j)) \}$ を A に送る。

4. A は s_i から $a_i := H(\text{Sign}_B(a_i))$ を求めて $\{a'_i\}$ と $\{b'_j\}$ を比較し、同じ値があるかを探す。それが共通部分集合である。

この方式は演算量が要素の個数に比例し効率的です。ただしこの方式だと B は共通部分の情報を得られないので A と B の立場が対等ではありません。また A は $H(\text{Sign}_B(b_j))$ の値をもらっています。これから b_j の情報を引き出すのは無理ですが、この値は確定的です。そのためこのプロトコルを複数回行った場合、もし B の集合 S_B に変化があると「どの番号の要素が変わった」という情報が得られてしまいます。これらの欠点の改良案もいろいろ提案されています。

なお、ブラインド署名を使わずに B が A にハッシュ関数 $\{b'_j := H(b_j)\}$ を渡してもよいのではと思われるかもしれません。しかし、この方法では b_j の予想できる種類が多くないとき、A は自分で手当たり次第に $H(b_k)$ を計算して b'_j と比較できます。そうするとどの b_k をもらったか分かってしまいます。だからハッシュ関数だけでは駄目なのです。

4.9 部分ブラインド署名

ブラインド署名は大変有用な技術ですが問題点もあります。それは署名者がどんなメッセージに署名をしているか全く分からぬ点にあります。この性質がまさにブラインド署名の特長なのですが、署名をさせる人が悪意を持っている場合に対処できません。そのため部分ブラインド署名という概念が提案されました。

これはメッセージの一部を両者が共有することで署名者が署名してよいか判断できるようにした方式です。たとえば共有するメッセージの部分に有効期限を入れておく、署名要求者の名前を入れるなどが想定されます。2000 年に阿部氏、岡本氏により提案された方式 [AO00] を紹介しましょう。これは離散対数問題を安全性の根拠に置く Schnorr 署名の拡張になっています。A さんが署名をしてほしい人、B さんが署名者です。メッセージを m 、両者が共有するメッセージを m' とします。

鍵生成 (p, q, g) を DH パラメータ（3.3 節参照）とする。 p と q は $p - 1$ が q の倍数となる素数で $g^q \equiv 1 \pmod{p}$ である。B は秘密鍵 x を決めて $y := g^x \pmod{p}$ を公開鍵とする。更にハッシュ関数 H を決める。

1. B は m' を確認してブラインド署名することに決める。 u, s, d をランダムに選ぶ。 $z := H(m')$, $a := g^u$, $b := g^s z^d$ を計算して (a, b) を A に送る。
2. A は $z = H(m')$ を確認し t_1, t_2, t_3, t_4 をランダムに選び

$$\begin{aligned}\alpha &:= ag^{t_1} y^{t_2}, \\ \beta &:= bg^{t_3} z^{t_4}, \\ \epsilon &:= H(\alpha, \beta, z, m), \\ e &:= \epsilon - t_2 - t_4\end{aligned}$$

を計算して e を B に送る。

3. B は $c := e - d$, $r := u - cs$ を計算し A に (r, c, s, d) を送る。
4. A は $\rho := r + t_1$, $\omega := c + t_2$, $\sigma := s + t_3$, $\delta := d + t_4$ を計算し $(\rho, \omega, \sigma, \delta)$ を m に対する署名とする。
5. 署名の検証は

$$\omega + \delta = H(g^\rho y^\omega, g^\sigma z^\delta, z, m)$$

が成り立つかどうかで判定する。

正しく計算しているなら

$$\begin{aligned}\omega + \delta &= c + t_2 + d + t_4 = e - d + t_2 + d + t_4 = (\epsilon - t_2 - t_4) + t_2 + t_4 = \epsilon, \\ g^\rho y^\omega &= g^{r+t_1} (g^x)^\omega = g^{(u-cx)+t_1+cx+t_2x} = g^{u+t_1+t_2x} = g^u g^{t_1} y^{t_2} = \alpha, \\ g^\sigma z^\delta &= g^{s+t_3} z^{d+t_4} = (g^s z^d) g^{t_3} z^{t_4} = \beta \text{ より} \\ H(g^\rho y^\omega, g^\sigma z^\delta, z, m) &= H(\alpha, \beta, z, m) = \epsilon\end{aligned}$$

となることを確認できます。この方式は DLP 仮定のもとで安全だと証明されています。

4.10 この章のまとめ

公開鍵基盤に必要な認証について説明しました。ハッシュ関数は任意のメッセージを固定長の値に変換する関数です。ハッシュ関数はその関数の値（ハッシュ値）から元の値を求められないだけでなく、同じハッシュ値を持つ異なるメッセージが見つかってはいけません。この性質を衝突困難性といいます。

認証には主に、メッセージが改竄されていないかを検証するメッセージ認証と、メッセージを書いた人が本人であることを検証するデジタル署名があります。メッセージ認証は共通鍵暗号、デジタル署名は公開鍵暗号と関係が深いです。

第 5 章

楕円曲線暗号

この章では楕円曲線暗号について紹介します。楕円曲線暗号は同程度の安全性を持つ有限体上の暗号に比べて鍵サイズが小さいという特徴があります。

楕円曲線暗号は住民基本台帳カードなどの IC カードや、パソコンで Google や twitter などのサービスを利用するときに使われています。新しい暗号通貨、仮想通貨として話題になったビットコインにも楕円曲線を使った電子署名が使われています。近い将来キャッシュカードなどでも使われるようになるでしょう。

この章ではイメージしやすいように、楕円曲線の幾何学的な描写を重視して説明します。暗号の計算で使いやすい代数的な側面については第 III 部で紹介します。

5.1 楕円曲線暗号

楕円曲線暗号とは楕円曲線上で組み立てられた暗号プロトコル全般のことを指します。楕円曲線は実現したい暗号プロトコルの手段の一つです。公開鍵暗号や共通鍵暗号といった実現したい機能を表すグループの名称とは「暗号」の使い方がやや異なるので注意してください。「楕円曲線を使った暗号」というニュアンスです。

楕円曲線についてはこれからゆっくりと説明します。なお、楕円という言葉が入っていますが中学校や高校で習う円がつぶれた形の楕円とは違います。楕円曲線という一つの用語です。

今までにいろいろな暗号プロトコルが出てきたので整理するためにまとめてみました。図 5.1 は実現したい機能（横軸）と、実現手段（縦軸）の 2 軸で示しています。「鍵共有」や、「公開鍵暗号」など長方形の枠で囲ったものが機能面の名前、左の縦に使っている主な道具で分けています。有限体を使った暗号と楕円曲線を使った暗号はほぼ対応していることが分かります。後半の章で紹介するペアリングを使うとそれらとは違う機能を実現できます。

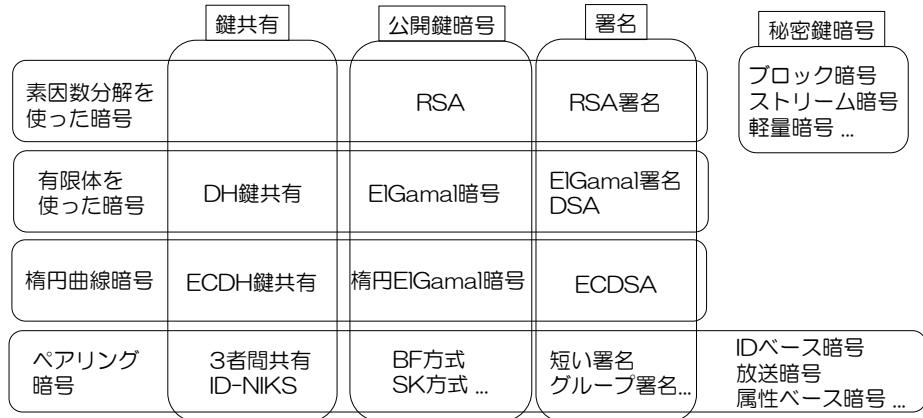
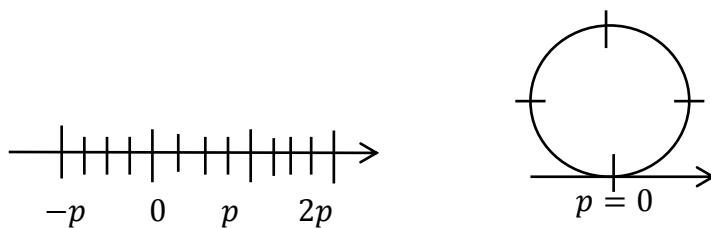


図 5.1 機能と道具の観点からみた主な暗号プロトコル

まだ説明していない用語やこのテキストでは紹介しないものも含んでいます。もちろんこの枠に入らない暗号もたくさんあります。また、ある機能を持つ暗号がその道具を使わないとできないというわけでもありません。

5.2 2次元の世界へ

それでは楕円曲線のはなしに入ります。ElGamal 暗号は有限体を使って構成されていました。有限体 \mathbb{F}_p の世界は数直線上を p 進むと元に戻ります。これは 0 と p のところをつなげると円周をぐるぐる回っていることになります。円周という線の上を動いているので 1 次元の世界です。

図 5.2 \mathbb{F}_p 上の加算は円周をぐるぐる回っている

これを 2 次元にすると別の暗号を作れるでしょうか。1 次元では円周だったので 2 次元ならビーチボールのような球面を考えればよさそうです。ちょっと考えるとぐるぐる回ってうまくいきそうに思います。しかし残念ながら実は球面上ではうまくいかないことが知られています。もう少し数学的にいうと 2 次元球面にはきれいな足し算の構造を入れられないのです。「きれいな足し算の構造」を厳密に説明するのはこのテキストの範囲を超えるため省略しますが感覚的な説明をしましょう。

球面状にきれいな足し算の構造が入ったとします。ある点にとても小さな値を少しづつ足していくと、球面の上を点が少しづつ動いてその軌跡は水の流れのようなものになります。しかし、どう流れを作ってもどこかに涌きだし口や渦のような吸い込み口という特殊な点ができてしまします。これは Hairy ball (毛玉) の定理という幾何学の定理により示されています。球面を頭、水の流れを髪の毛にたとえると、特別な点は頭のつむじに相当します。頭には（髪の毛があれば）つむじが必ずあるという定理です。

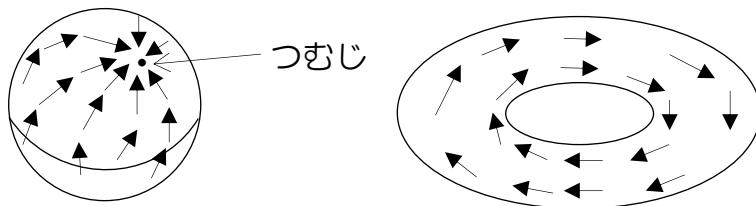


図 5.3 球面と浮輪の上の水の流れ

そしてその特別な点ではきれいな足し算の構造が壊れることができます。したがって球面上にはきれいな足し算の構造を入れられません。球面ではなく浮輪の表面だと涌きだし口や吸い込み口のない水の流れを作れます。つまりきれいな足し算の構造が入ります。

5.3 ドラクエ 3 の世界

きれいな足し算の構造が入る 2 次元の世界を考えるとき、球面では駄目で浮輪の表面だとうまくいくのでした。でも浮輪の表面で足し算ってイメージしにくいですね。

ここで唐突ですがドラゴンクエストというゲームを紹介します。ゲームのキャラクターたちは長方形の世界地図の中を移動します。パッと見ると普通の世界地図なのですが、右へ進んで右端に来ると左端から出てきます。上に進んで上端に着くと下端から出てきます。

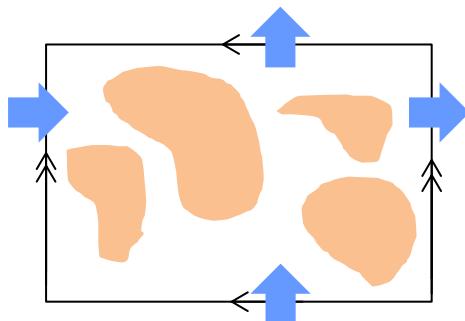


図 5.4 ドラクエ 3 の世界地図

これは、この惑星？がもし 3 次元空間の中に存在していたら、どんな形をしているでしょう。右

端と左端が同じなのでくっつけると円筒になります。メルカトル図法による普通の世界地図の両端をくっつけたのとそっくりです。ただ世界地図と違ってドラクエ3では上端と下端も同じなので、それらもくっつける必要があります。すると（多少曲げないといけませんが）浮輪の表面の形になります。なんとドラクエは球面ではなく、浮輪の表面の世界だったのです。

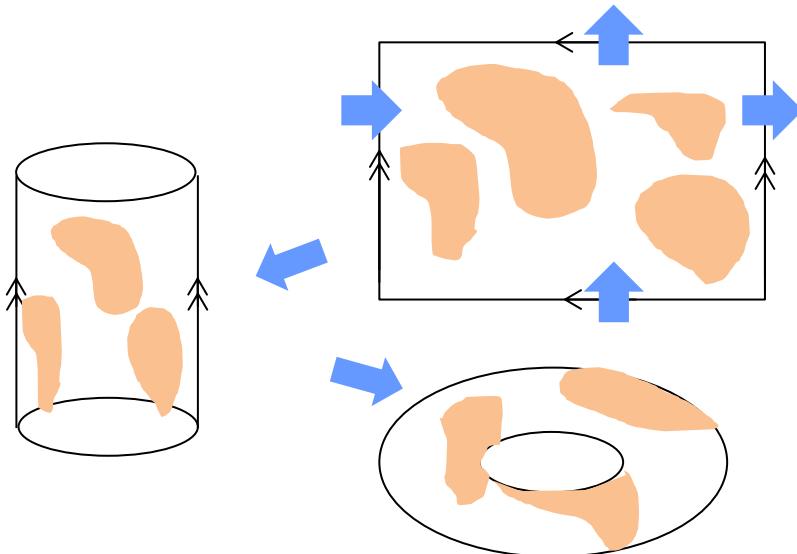


図 5.5 ドラクエ3の世界地図の端を張り付ける

浮輪の表面を数学の言葉でトーラス (torus) といいます。トーラス上の足し算は分かりにくくてもドラクエ世界の足し算は容易に考えられます。世界を長方形 $OABC$ とします。

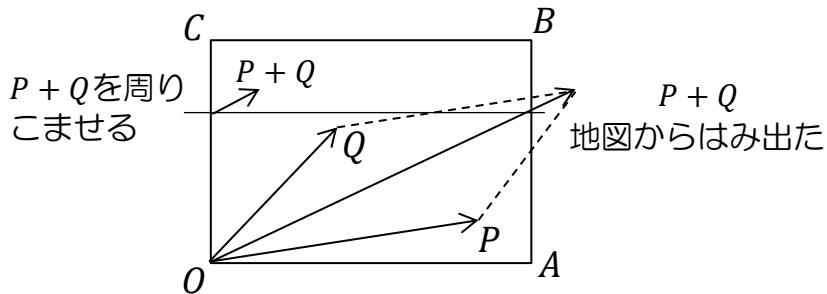


図 5.6 トーラス上の足し算

2点 P, Q に対してその足し算 $P + Q$ を点 P から \overrightarrow{OQ} だけ進めということにします。つまり $O, P, P + Q, Q$ が平行四辺形になるように $P + Q$ を決めます。そのとき、 $P + Q$ が長方形 $OABC$ をはみ出たら回り込ませます。点 P から \overrightarrow{OQ} だけ進んでも、逆に点 Q から \overrightarrow{OP} だけ進んでも同じ

場所にたどり着きます。つまり

$$P + Q = Q + P$$

です。 $-P$ は P と反対方向の点としましょう。このようにして両端がつながった長方形の世界、つまりトーラス上に足し算を定義できます。

足し算の入ったトーラスを楕円曲線といいます。数学的には楕円曲線はトーラスとは別のもので定義されます。しかし、実は根っこでトーラスと同じものであることが分かっています。第 III 部の 17 章でその関係を紹介します。

足し算ができるのならある点 P の整数倍も考えられます。 nP を P を n 回足すことで定義します。

$$nP := \underbrace{P + P + \cdots + P}_{n \text{ 回足す}}$$

n 倍の計算は有限体のときの巾乗の計算と同様にバイナリ法（3.8 節参照）を使うことで効率よく求められます。

5.4 楕円離散対数問題

楕円曲線上でゲームのキャラクターが歩き回ることを考えましょう。

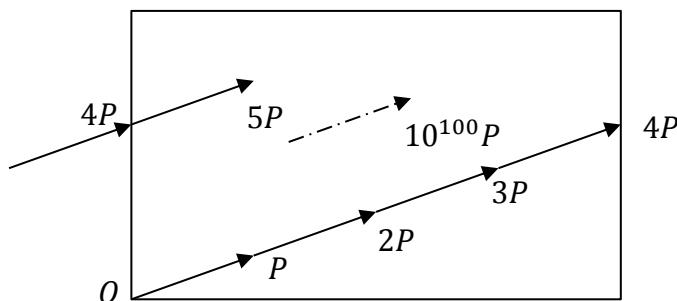


図 5.7 楕円曲線上を歩く

キャラクターの一歩が P だとします。原点から 2 歩歩くと $2P$ の位置にいます。どんどん進んで端に到達すると逆から出てきます。10 歩でも 100 歩でも 10^{100} 歩でも進めます。そして歩いて到達した位置 $10P$ や $100P$, $10^{100}P$ は容易に求められます。

さて、移動していたキャラクターはうっかり何歩歩いていたか忘れてしまいました。現在地と自分の一歩から何歩歩いたのか知りたいのです。実はこれはとても難しいことが知られています。

一歩 P と歩数 n から現在地 nP を求められる \iff 一歩 P と現在地 nP から n を求められない有限体のときと同じく、この非対称性が重要です。

「 P, nP が与えられたときに n を求めよ.」

という問題を楕円離散対数問題 (ECDLP : Elliptic Curve DLP) といいます。楕円曲線暗号は ECDLP が難しいことを仮定して構成されます。 nP と P から n を求めるのだから除算問題じゃないのか、なぜ対数と呼んでいるのかという疑問をもたれるかもしれません。それについては 6.2 章で説明します。

5.5 楕円 DH 鍵共有と楕円 ElGamal 暗号

楕円曲線暗号の中で基本的な楕円 DH 鍵共有を紹介しましょう。3.3 節で紹介した Diffie-Hellman 鍵共有の楕円曲線版です。

1. P を固定する（これは公開情報です）。
2. Aさんは秘密の数字 a を決めて $K_A := aP$ を計算して公開する。
3. Bさんは秘密の数字 b を決めて $K_B := bP$ を計算して公開する。
4. Aさんは a と K_B を使って $aK_B = abP$ を計算する。
5. Bさんは b と K_A を使って $bK_A = abP$ を計算する。

これにより二人の間で秘密の数値を共有できます。盗聴者は P, aP, bP を知りますが、それから a や b を求められないで abP が分かりません。有限体のときと同様、正確には楕円 DH 問題

「 P, aP, bP が与えられたときに abP を求めよ.」

が困難であるとき、この鍵共有は安全に行えます。

同様に有限体上の ElGamal 暗号を楕円曲線上に焼き直した楕円 ElGamal 暗号も作れます。

セットアップ：楕円曲線 E とその上の点 P を固定してみなに公開する。ユーザ A は整数 a をランダムに選び $K_A := aP$ とする。 a が秘密鍵で K_A が公開鍵である。

暗号化：平文 M を E の点として表現する（たとえば楕円曲線の点の x 座標を入れる）。整数 r をランダムに選び公開鍵 K_A を用いて

$$\text{Enc}(M) := (rP, M + rK_A)$$

とする。

復号： $(C_1, C_2) := \text{Enc}(M)$ を受け取った人は秘密鍵 a をもつ人は

$$\text{Dec}(C_1, C_2) := C_2 - aC_1$$

として平文を復号する。

正しく復号できることは

$$\text{Dec}(\text{Enc}(M)) = \text{Dec}(rP, M + rK_A) = (M + rK_A) - a(rP) = M + raP - arP = M$$

から分かります。

5.6 楕円曲線暗号の利点

1次元のような有限体から2次元の世界を考えることで楕円曲線暗号にたどり着きました。楕円曲線は普通の有限体上の暗号に比べて鍵のサイズが小さくてよいという利点があります。

両暗号が同程度の安全性を持つときの鍵のビット長を『楕円曲線暗号とRSA暗号の安全性比較』(富士通株式会社) [富士 10] をもとに表5.1にしました。

表5.1 同じ安全性のRSA暗号、楕円曲線暗号、共通鍵暗号の鍵のビット長の比較

RSA暗号	1024	1219	2048	2832	11393
楕円曲線暗号	138	152	206	245	497
共通鍵暗号	72	80	108	128	256

1024ビットのRSA暗号は近い将来解読されると推測されています。したがって現在は2000ビット以上のRSA暗号が推奨されています。高い安全性が要求されると必要な鍵長の差は広がり、楕円曲線暗号の利点が大きくなります。これはRSA暗号や有限体上の暗号に使われる効率のよい攻撃アルゴリズムが楕円曲線暗号には使えないからです。国家安全保障局(NSA)が利用する暗号リストSuite BにはRSAを使ったアルゴリズムは含まれていません[NSA09]。

5.7 ECDSA

楕円曲線を使うと暗号の他にデジタル署名も作れます。4.6節で紹介した有限体上のデジタル署名DSAの楕円曲線版を紹介しましょう。暗号通貨、仮想通貨という名前で有名になったビットコインは256ビット長のECDSAを利用しています[漆薫14]。

鍵生成：楕円曲線 E と点 P とハッシュ関数 h を選び、 $q := \min \{ n \mid nP = 0, n > 0 \}$ とする。整数 x をランダムに選び $Q := xP$ として (P, Q) を公開鍵、 x を秘密鍵とする。

平文 m に対する署名：整数 k をランダムに選び kP の x 座標を r とする。

$$s := (h(m) + xr)/k \bmod q$$

を求めて (r, s) を m に対する署名とする。

平文 m に対する (r, s) の検証：公開鍵 P, Q と与えられた m と (r, s) に対して

$$R := (h(m)/s)P + (r/s)Q$$

を計算し、 R の x 座標が r に等しいければ署名を受理し、そうでなければ棄却する。

正しい平文に対しては正しく受理できることを確認します。

$$(h(m)/s)P + (r/s)Q = \frac{h(m)}{s}P + \frac{r}{s}xP = \frac{h(m) + xr}{s}P = kP$$

となり kP の x 座標は r に等しいです。

5.8 前方秘匿性

身近なところで椭円曲線暗号が使われている例を紹介しましょう。AさんとBさんが秘密の通信をしています。通信内容を共通鍵暗号AESで暗号化し、AESの鍵 K を公開鍵暗号RSAで暗号化して渡していました。 K はセッション毎に異なる値です。盗聴者はその二人の間の暗号化された通信の中身を読めないけれども、ずっと記録していたとしましょう。ある日、AとBの間で使われていた公開鍵暗号の秘密鍵が漏洩（ろうえい）してしまいました。すると盗聴者は過去に遡って記録していた暗号文を復号し、内容を見られます。AやBが重要な人物の場合、通信の記録が保持されていてもおかしくはありません。

このような事態になっても安全性を担保するには、ある段階で使われていた秘密鍵が漏洩したとしても、それよりも過去の暗号文を容易に復号できないようにする必要があります。公開鍵暗号のような比較的長期に渡って使われる秘密鍵が漏洩したとしても、その時に使われていた共通鍵暗号に使われる秘密鍵が漏洩しないようにしたいのです。これを前方秘匿性（ぜんぽうひとくせい：Forward Secrecy : FS）といいます。PFS（Perfect Forward Secrecy）ともいわれます。様々な暗号に対してどのようにしたら前方秘匿性の機能を持たせられるか研究されています。

先ほどのRSA暗号で K を暗号化している場合はFSは保たれません。

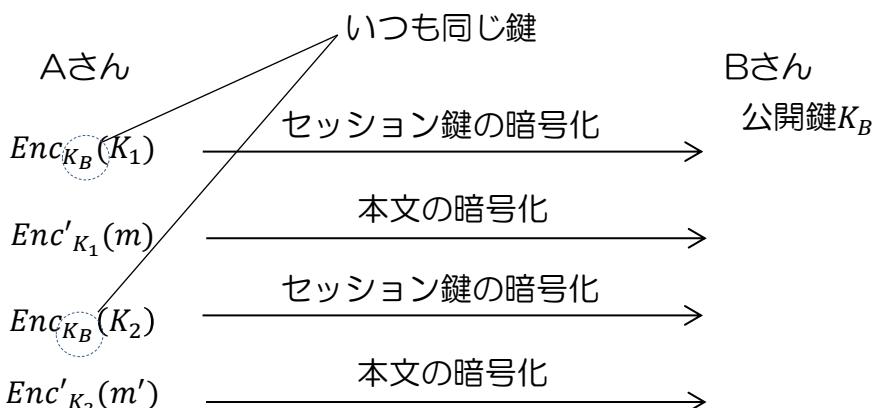


図 5.8 RSA 暗号を用いた ClientKeyExchange

しかし毎回DH鍵共有で秘密鍵 K を作ればRSA暗号の秘密鍵とは無関係なので解読できません。

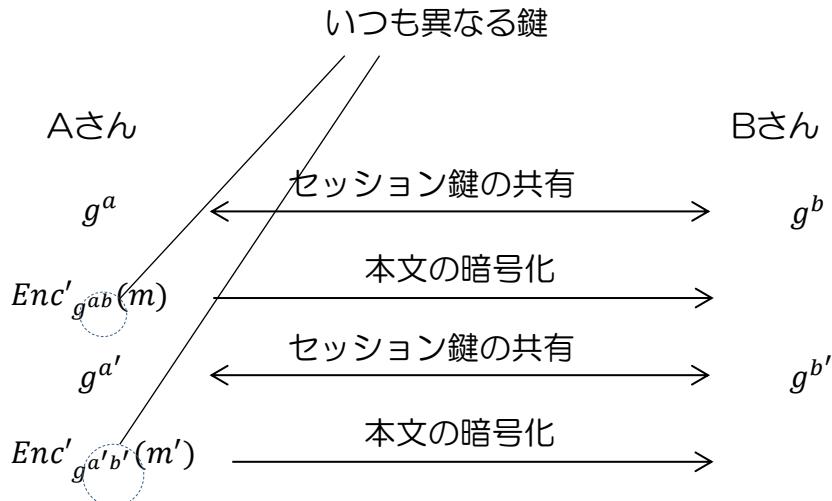


図 5.9 DH 鍵共有を用いた ClientKeyExchange

つまり、PFS が保たれます。有限体上の DH 鍵共有は処理が重いためあまり使われていませんでした。しかし、橿円 DH 鍵共有は RSA 暗号に比べてもそれほど重たくはないため徐々に普及しています。

Google の Gmail は 2011 年に PFS に対応しました。2013 年アメリカ国家安全保証局（NSA）がインターネットの通信を盗聴、保存、解析していたことが明らかになります。当時局員だった Snowden が告発したのです。そのとき FBI が追求のため Snowden が利用していたメールサービス提供者に秘密鍵の提供を求めます [Gua13] [ukk13]。それにより秘密鍵が漏洩するとそれまでの通信が復号できる状況が現実的なものとして認識されます。そのため twitter などの大手のサービス提供社も次々と PFS に対応しました。

図 5.10 2014 年 3 月に <https://www.google.co.jp/> に接続したときの情報

Firefox や Chrome などのブラウザで <https://www.google.co.jp/> に接続し、鍵アイコンをクリックすると接続に使われた暗号アルゴリズムを確認できます。図 5.10 を見ると ECDHE_ECDSA という鍵交換メカニズムが使われていることが分かります。ECDHE は椭円曲線版 DH 鍵共有、ECDSA は前節で紹介した椭円曲線版 DSA のことです。ECDHE の最後の E は Ephemeral (短命の) という意味を表しています。

5.9 この章のまとめ

椭円曲線暗号とは椭円曲線を使った暗号全般を指します。椭円曲線とは上下左右がつながった長方形世界のことです。椭円離散対数問題 (ECDLP) とは、その世界のある点に何歩歩けばたどり着けるかという問題です。

ECDLP が困難であることを利用して、鍵共有や公開鍵暗号を構成できます。椭円曲線暗号は有限体を使った暗号に比べて鍵のサイズが小さいという利点があります。

鍵漏洩時の影響を減らす前方秘匿性の需要の高まりに応じて椭円曲線暗号の使われるシーンが増えています。

第 6 章

群

この章はやや抽象的な話をします。有限体上の ElGamal 暗号と楕円 ElGamal 暗号は異なる式ですが、なんとなく似ていました。この「似ている」という感じを抽象化すると群上の演算というものにたどりつけます。また楕円離散対数問題は $P, Q (= nP)$ が与えられたときに P を何倍すれば Q になるのかという問題でした。「何倍か」という問い合わせ、「対数」という名前が入っているのが気持ち悪いかもしれません。群の概念を理解するとその気持ち悪さも解消されると思います。

6.1 演算の抽象化

群とは足し算や掛け算といった演算の抽象化です。演算の抽象化とは聞き慣れないですが普段意識しないところでしています。たとえば 1 個 50 円のお菓子を 4 個買うときは、

$$50 \times 4 = 200$$

と計算して 200 円と求めます。また縦 50cm で横 4cm の長方形の面積は、

$$50 \times 4 = 200$$

と計算して 200cm^2 と求めます。お菓子の金額と面積とは異なる概念です。それなのに同じ計算式で 200 という値を計算しています。これはよく考えると不思議です。なぜ本来は別のものなのに同じ演算で記述できるのでしょうか。

これはお菓子の金額を 1 円玉が何個あるか、また面積を一辺 1cm の正方形が何個あるか、と考えることで、「○個あるか」という問い合わせるために計算手段を抽象化しているからです。そもそも 50 円と 50cm という違うものを同じ 50 という値で表しているのも抽象化の一つです。

余りの世界では足し算や引き算を「足したり引いたりしてから余りをとる」という操作で定義しました。最初は $(\text{mod } p)$ を書いていましたが、慣れてくるといちいち $(\text{mod } p)$ を書かなくても + や - を見たときに余りの計算とみなせるようになりました。また割り算は普通の整数の世界と異なる方法で導入しました。ただその割り算に対して / という普通の割り算と同じ記号を使っても

違和感はありませんでした。そうするとそもそも足し算や引き算とはなんなのか、どういう性質を持つていたら足し算と思えるのかという問い合わせが生まれます。その問い合わせに対する答えの一つが群です。群とは掛け算（あるいは足し算）が満たすべき性質を最小限に絞って抽象化したものなのです。

6.2 群の定義

それでは群の定義から始めましょう。前節で触れたように群とは足し算や掛け算のみに着目してそれを抽象化した概念です。一般に、ある集合 G の 2 個の元に対してその集合の元を対応させる写像 μ を（2 項）演算といいます。

$$\begin{array}{ccc} \mu: & G \times G & \longrightarrow & G \\ & \Downarrow & & \Downarrow \\ & (x, y) & \longmapsto & \mu(x, y). \end{array}$$

集合 G 上の演算 $\mu: G \times G \rightarrow G$ が次の性質を満たすとき G は演算 μ に関して群であるといいます。記号の簡略化のため、 $\mu(x, y)$ を $x \cdot y$ と書きます。

結合則：任意の $x, y, z \in G$ に対して $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

単位元：単位元 $e \in G$ と呼ばれるものが存在して任意の $x \in G$ に対して $x \cdot e = e \cdot x = x$.

逆元：任意の $x \in G$ に対して x の逆元 $x^{-1} \in G$ と呼ばれるものが存在して $x \cdot x^{-1} = x^{-1} \cdot x = e$.

結合則は実数の足し算や掛け算について $a + (b + c) = (a + b) + c$ や $a(bc) = (ab)c$ が成り立っていたことを群にも要求しています。単位元は演算が足し算なら $0 + a = a + 0 = a$ 、掛け算なら $1 \cdot a = a \cdot 1 = a$ を意味します。逆元は足し算なら $-a$ 、掛け算なら $1/a$ です。

G の元の個数が有限個のとき有限群といいます。そのときの群の元の個数を位数（いすう）といい $|G|$ と書きます。具体例を挙げます。実数全体の集合 \mathbb{R} 上の足し算は 2 項演算です。つまり $x, y \in \mathbb{R}$ に対して $\mu(x, y) := x + y \in \mathbb{R}$ です。 \mathbb{R} は足し算に関して群となります。なぜならまず任意の \mathbb{R} の元 x, y, z に対して結合法則 $(x + y) + z = x + (y + z)$ が成り立ちます。 $0 \in \mathbb{R}$ は単位元で任意の x に対して $x + 0 = 0 + x = x$ です。そしてその符号を反転させた $-x$ が x の逆元であり、 $x + (-x) = (-x) + x = 0$ が成り立つからです。同様に有限体 \mathbb{F}_p も足し算に関して群となります。 \mathbb{F}_p は元の数が有限個なので有限群です。

\mathbb{R} は掛け算に関しては群にはなりません。 $\mu(x, y) := xy$ とすると結合法則が成り立ち、任意の x に対して $x \cdot 1 = 1 \cdot x = x$ ですが、 0 の逆元（逆数）は存在しないからです。しかし \mathbb{R} から 0 を取り除いた集合 $\mathbb{R}^* := \mathbb{R} \setminus \{0\}$ は掛け算に関して群になります。 1 が単位元です。同様に有限体 \mathbb{F}_p は掛け算に関して群にはなりませんが、 0 を除いた集合 $\mathbb{F}_p^* := \mathbb{F}_p \setminus \{0\}$ は掛け算に関して群となります。楕円曲線上の点の集合も足し算に関して群となります。点 P に原点 O を足しても値は変わらないので O が単位元です。 $P + O = O + P = P$. 点 P の逆元はそのベクトルの向きを逆にしたものでした。 $P + (-P) = (-P) + P = O$. これが逆元に相当します。楕円曲線上の点同

士の演算には掛け算に相当するものはありません。

これらの例はどれも 2 項演算 μ に対し全ての x, y について

$$\mu(x, y) = \mu(y, x)$$

という関係が成り立っていました。このような関係が成り立つ演算を可換（かかん）であるといいます。群の演算が可換なとき可換群とか加法群といいます。

可換でない群の例として逆行列を持つ 2 行 2 列の実正方行列全体 $GL_2(\mathbb{R})$ があります。 $GL_2(\mathbb{R})$ は行列の掛け算に関して群となります。単位元は

$$I_2 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

です。行列の積は

$$A := \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad B := \begin{pmatrix} p & q \\ r & s \end{pmatrix} \text{ について } AB := \begin{pmatrix} ap + br & aq + bs \\ cp + dr & cq + ds \end{pmatrix}.$$

逆元は

$$A^{-1} := \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

行列 A, B に関して常に $AB = BA$ が成り立つとは限りません。

6.3 巾乗

もう少し抽象的な話が続きます。（可換とは限らない）群 G があると結合則により任意の $x \in G$ に対して $(x \cdot x) \cdot x = x \cdot (x \cdot x)$ が成り立ちます。つまり

$$x \cdot x \cdot x$$

を考えたとき、この値は前から先に演算しても後ろから先に演算しても同じであることを示しています。結果が演算の順序に依存しないので括弧を省略しても曖昧にはなりません。これを $x^3 := x \cdot x \cdot x$ と書くことにしましょう。次に 4 個の x の演算

$$x \cdot x \cdot x \cdot x$$

を考えます。これもどこから先に演算するかで様々な組み合わせがありますが、結合則によりどの場合も最終的な演算結果は同じになります。たとえば

$$((x \cdot x) \cdot x) \cdot x = (x \cdot x) \cdot (x \cdot x) = x \cdot (x \cdot (x \cdot x)).$$

括弧を省略して $x^4 := x \cdot x \cdot x \cdot x$ と書きましょう。一般に n 個の x の演算を

$$x^n := \underbrace{x \cdot x \cdots \cdot x}_{n \text{ 回演算する}}$$

と書き x の n 乗と呼びます。 $x^0 = e$ (単位元), n がマイナスのときは $x^n := (x^{-1})^{-n}$ と定義しましょう。これらの演算や記号は実数などにおける巾乗の自然な類推です。

楕円曲線上の演算は点の足し算でした。このときは・の代わりに + を使うので点 P を n 回演算するときは P^n ではなく nP と書くことが多いです。群の演算をどういう記法で表すかの違いだけで点 (元) 同士の演算を繰り返ししているのは同じであることに注意してください。

6.4 巡回群

巾乗を定義したので巾乗の元が作る集合を考えてみます。群 G の元 g を一つとり, $\langle g \rangle := \{g^i \mid i \in \mathbb{Z}\}$ とします。この集合は元の群の演算に関して群になります。群の定義を見ながら確認しましょう。まず結合則は G で成り立っているので部分集合である $\langle g \rangle$ でも成り立ちます。 $\langle g \rangle$ は単位元 $g^0 = e$ を含んでいます。任意の g^i に対してその逆元 g^{-i} も $\langle g \rangle$ の元です。よって $\langle g \rangle$ が群であることを確認できました。また任意の g^i, g^j について $g^i \cdot g^j = g^{i+j} = g^j \cdot g^i$ ですから、元の群 G が非可換群であっても $\langle g \rangle$ は可換群となります。 $\langle g \rangle$ を巡回群, g を $\langle g \rangle$ の生成元といいます。

G が有限群のときは $\{g^0, g^1, g^2, \dots\}$ も有限集合のはずです。ということはこれらの値がのどちらかは同じ値になります。つまり $g^i = g^j$ となる 2 個の異なる i と j が存在します。もし $i < j$ なら i と j を入れ換えることで $i > j$ とします。すると等式の両辺に g^{-j} をかけると $g^{i-j} = e$ となります。 $s := i - j > 0$ とおくと $\langle g \rangle = \{g^0, g^1, \dots, g^{s-1}\}$ であることが分かりました。無限回巾乗しているつもりでしたが、実は途中で単位元にもどってぐるぐる回っていたということですね。巡回群のイメージができたでしょうか。

$\mathbb{F}_7^* = \{1, 2, 3, 4, 5, 6\}$ について $x = 1, \dots, 6$ に対して $\langle x \rangle$ を計算してみましょう。

$x \setminus i$	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

$\langle x \rangle$ の表

\mathbb{F}_7^* の部分集合 $\langle 1 \rangle = \{1\}$ は要素が 1 しかありませんがこれも群です。 $1 \cdot 1 = 1$ ですし 1 の逆元は 1 です。単位元のみからなる一番簡単な群です。

$\langle 2 \rangle = \{1, 2, 4\}$ は 3 個の元からなる群です。 $2^2 = 4$, $4^2 = 2$, $2 \cdot 4 = 1$ です。 $\langle 4 \rangle$ や $\langle 6 \rangle$ も $\langle 2 \rangle$ と同じ巡回群になります。 $\langle 3 \rangle$ や $\langle 5 \rangle$ は全部の値がでて \mathbb{F}_7^* となっています。一般に有限体 \mathbb{F}_p^* の生成元となる要素を原始元といいます。

$\langle 6 \rangle = \{1, 6\}$ は要素が 2 個の群です。 $1 \cdot 6 = 6$ で $6 \cdot 6 = 1$ なので $6^{-1} = 6$ です。これは単位元とそうじゃない元一つからなる群ですね。

このように生成元によっていろいろな形の巡回群ができます。

6.5 群に対する離散対数問題

巡回群の説明が終わったのでようやく暗号の話に戻れます。有限群 G に対してその元 g を一つとり巡回群 $\langle g \rangle$ を考えます。 $h \in \langle g \rangle$ が与えられたときに $h = g^n$ となる n を求める問題を巡回群 $\langle g \rangle$ に関する離散対数問題といいます。

暗号の論文では G を取り直して最初から巡回群であるとしていることが多いです。

有限体の演算と楕円曲線上の演算を並べてみましょう。

対象物	有限体 (\mathbb{F}_p)	楕円曲線 (E)
演算	掛け算 ($g \cdot h$)	足し算 ($P + Q$)
整数 n に対して	g の n 乗 (g^n)	P の n 倍 (nP)
離散対数問題	g と g^a から a	P と aP から a
CDH 問題	g と g^a と g^b から g^{ab}	P と aP と bP から abP

有限体と楕円曲線の比較

表にすると有限体での掛け算は楕円曲線の足し算、有限体での n 乗算は楕円曲線上の n 倍算に対応していることが分かります。有限体と楕円曲線という別のものであっても、群という抽象的な枠組みで考えると、ある巡回群上の離散対数問題を扱っているという観点では同じなのです。これが、楕円曲線上でも離散”対数”問題と呼ばれている理由です。

こういう抽象化はどういうメリットがあるのでしょうか。ある暗号を巡回群の上で構成できたなら、その巡回群として具体的に有限体や楕円曲線を選ぶことでいろいろな実装を用意できます。新たに巡回群になるものを見つければ別の暗号を構成できるわけです。実際、超楕円曲線やアーベル多様体という別の数学的な対象物を巡回群と見て暗号を構成できます。

超楕円曲線を使うと楕円曲線よりも鍵サイズの小さい暗号を構成できることが知られています。

6.6 離散対数問題に対する攻撃

この節では Polar (ポラード) の ρ 法といわれる攻撃方法を紹介します。これはほぼ一般の巡回群の離散対数問題に対して可能な攻撃方法です。総当たり攻撃の計算量が $O(\text{巡回群の位数})$ であるのに対して、この攻撃の計算量は $O(\sqrt{\text{巡回群の位数}})$ です。

有限体上の離散対数問題に対してはもっと効率のよい方法が提案されています。しかし、楕円曲線上の離散対数問題に対してはこれよりよい方法は今のところ知られていません。楕円曲線暗号の鍵サイズが有限体上の暗号に比べて小さくできる理由の一つです。

加法に関するある巡回群 $G := \langle P \rangle$ とその点 $Q \in G$ が与えられたとします. $Q = aP$ となる a を見つけましょう.

分割数と呼ばれる固定パラメータ $m \in \mathbb{N}$ を一つ決めます. m 個の整数の組 (u_i, v_i) をランダムに選び,

$$M_i := u_i P + v_i Q$$

とします. G の点を動かす関数 $f : G \rightarrow G$ を

$$\begin{aligned} f(R) &:= R + M_{g(R)}, \\ g(R) &:= (R \text{ から一意に決まるある整数値}) \bmod m \end{aligned}$$

とします. 関数 g は点から $0, \dots, m - 1$ への適当に分布する関数なら特になんでもかまいません. たとえば楕円曲線上の点ならその座標を使います. それ以外の群であってもそれぞれの群に応じて適切に作ることができるでしょう. このようにして作った f をランダムウォーク関数と呼びます.

点 $R_0 = P$ とし, $R_{i+1} = f(R_i)$ と R_i に繰り返し f を適用して点の集合列 $X := \{R_i\}$ を作ります.

$$\begin{aligned} R_0 &= P, \\ R_1 &= P + M_{i_0}, \\ R_2 &= P + M_{i_0} + M_{i_1}, \\ &\vdots \end{aligned}$$

と M_i を一つ選んで次々と足していくことになります. M_i の定義から各 R_i は P と Q の線形和の形をしています. つまり $a_i, b_i \in \mathbb{Z}$ として

$$R_i = a_i P + b_i Q$$

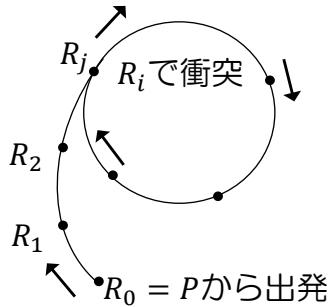
とかけます. X に R_i を追加して更新するとき, X に $R_i = R_j$ となる $j < i$ が見つかったとしましょう. そのとき

$$R_i = a_i P + b_i Q = a_j P + b_j Q = R_j$$

です. すると

$$Q = \frac{a_i - a_j}{b_j - b_i} P$$

となって離散対数問題が解けました (運悪く $b_i = b_j$ だったときはやり直します).

図 6.1 ρ 法による R_i の点列のイメージ

このアルゴリズムは集合 X にランダムに点を増やして、たまたま X の中に同じ点が見つかるまで繰り返します。4.1 節のハッシュ関数のところで紹介したように、見つかるのに必要なステップ数は $O(\sqrt{G \text{ の位数}})$ です。 ρ 法と呼ばれる名前の由来は、 ρ の字の形がしっぽのところから登ってぐるぐる回っているうちに同じところに衝突した図を表しているからだそうです。

6.7 巡回群の位数

この節では巡回群の位数が合成数の場合、DLP が解きやすくなることを示します。このため同じビット長の位数の巡回群を使うなら、素数位数の巡回群であることが望ましいです。

加法巡回群 $G := \langle P \rangle$ の位数が $n := ab$ (a と b は互いに素な自然数) とします。 $Q := xP \in G$ をとります。 x を今から求めます。

aP は b 倍すると単位元になるので $\langle aP \rangle$ の位数は b です。 $\langle aP \rangle$ の中で aP に対する aQ の DLP が解けたとしましょう。つまりある x_1 が見つかって $aQ = x_1(aP)$ 。 $Q = xP$ より

$$x_1(aP) = aQ = axP, \quad \text{つまり } a(x_1 - x)P = 0.$$

P の位数が $n = ab$ なので

$$a(x - x_1) \equiv 0 \pmod{ab}.$$

a と b は互いに素なので a で割って $x - x_1 \equiv 0 \pmod{b}$ 。つまりある整数 k があって

$$x = x_1 + bk$$

と表せます。同様に $\langle bP \rangle$ の中で bP に対する bQ の DLP が解けて x_2 が見つかると

$$x_2(bP) = bQ = bxP.$$

つまり $x \equiv x_2 \pmod{a}$ となり、ある整数 h があって

$$x = x_2 + ah$$

と表せます。

a と b が互いに素なので 16.1 節で紹介する拡張 Euclid の互除法を使うことで

$$sa + tb = 1, \quad \text{つまり } sa = 1 - tb$$

となる整数 s と t が見つかります。

$$x = x_1 + bk = x_2 + ah, \quad \text{つまり } x_1 - x_2 = ah - bk.$$

よって

$$\begin{aligned} s(x_1 - x_2) &= s(ah - bk) = sah - sbk = (1 - tb)h - sbk = h - (th + sk)b \quad \text{つまり}, \\ h &= s(x_1 - x_2) + (th + sk)b. \end{aligned}$$

よって

$$x = x_2 + ah = x_2 + a(s(x_1 - x_2) + (th + sk)b) = x_2 + as(x_1 - x_2) + (th + sk)ab.$$

よって

$$x \equiv x_2 + as(x_1 - x_2) \pmod{ab}$$

と x を求めることができます。位数 ab の DLP は位数 a の DLP と位数 b の DLP を解くと求められることが分かりました。

もし位数 n の DLP を解く演算量が $O(\sqrt{n})$ で、 $n = ab$ の a, b が同じぐらいだったとします。すると $a \simeq b \simeq \sqrt{n}$ です。それぞれの DLP を解く演算量は $O(\sqrt{a}) = O(\sqrt{\sqrt{n}})$ です。つまり n が素数なら DLP の演算量は $O(\sqrt{n})$ なのに、 a, b それぞれで解けるなら演算量は $O(2n^{1/4})$ とずっと小さくなります。

また素数巾の形 p^e の位数を持つ DLP も p 位数の DLP を解くことに帰着できることも知られています。つまり位数が素因数分解できればできるほど DLP が容易になります。有限体 \mathbb{F}_p の DLP の場合、16.4 節で示すように巡回群の位数は $p-1$ の約数です。2 以外の素数は奇数なので $p-1$ は必ず偶数です。よって暗号で使われる素数は $p-1 = 2 \times \text{素数}$ という形であることが望されます。

6.8 この章のまとめ

足し算や掛け算の抽象化である群という概念を紹介しました。有限体上の暗号と楕円曲線上的暗号は巡回群で考えると同じ式で記述できます。離散対数問題の解きやすさを考えると巡回群の大きさは素数であることが望ましいです。

第 II 部

新しい暗号技術

II 部の流れ

I 部は共通鍵暗号や公開鍵暗号などの現在使われている技術の紹介でした。II 部は主に 2000 年以降に登場した技術の紹介をします。

クラウドサービスが普及しています。しかし全面的にクラウドを信用したいわけではありません。できれば暗号化してからデータをクラウドに置くことで情報漏洩のリスクを減らしたいです。その場合従来の暗号技術ですと単にクラウドをデータストレージとしてしか利用できません。暗号化したままクラウドのリソースを使って検索したい、計算したい、と言った要求が生まれます。この II 部で紹介する暗号技術はそれらの要求に答えようとするものです。

7 章ではペアリングと呼ばれる楕円曲線上の 2 点から有限体への写像を紹介します。このテキストで紹介する新しい暗号技術の殆どはペアリングを用いて実現されています。最初にペアリングの性質を紹介してから ID ベース暗号を作ります。ID ベース暗号は公開鍵として自由な数値 (ID) を選べる暗号です。それから後はいくつかの暗号について個別に紹介します。各章は概ね独立しているので興味あるところを眺めてみるのもよいでしょう。

8 章は暗号化したまま検索できる検索可能暗号を紹介します。

9 章は暗号文を復号することなく別の人向けた暗号文を作り直すプロキシ暗号を紹介します。

10 章は沢山の人に安全に効率よく暗号文を配信する放送型暗号を紹介します。

11 章は復号できる条件を自由に指定できる属性ベース暗号を紹介します。

12 章は属性ベース暗号をより一般化した関数型暗号を紹介します。共通鍵暗号や公開鍵暗号は暗号を使う鍵と復号に使う鍵が 1 対 1 でした。属性ベース暗号や関数型暗号は一つ暗号鍵に複数の復号鍵が対応したり、複数の暗号鍵が一つの復号鍵に対応したりします。公開鍵暗号を拡張した暗号技術です。

13 章は暗号化したまま計算できる準同型暗号を紹介します。暗号化したまま足し算だけできる加法的準同型暗号の応用例を紹介してから足し算と掛け算の両方ができる完全準同型暗号を紹介します。完全準同型暗号は理論上任意の計算ができるため近年研究が進んでいる分野です。

14 章はゼロ知識証明を紹介します。ゼロ知識証明は自分がある情報を知っているということをその情報を相手に教えないまま納得してもらう技術です。ゼロ知識証明は暗号ではありませんが暗号と一緒に使うことで非常に有用な技術です。

15 章はこの部で紹介した暗号に使われている様々安全性仮定をまとめて紹介しています。

第 7 章

ペアリングと ID ベース暗号

2000 年以降、従来の共通鍵暗号や公開鍵暗号とは異なる機能を持った暗号が提案されてきました。ID ベース暗号、放送型暗号、属性ベース暗号、関数型暗号などです。この章ではそれらの暗号で使われる数学の道具の一つ、ペアリングという演算について説明します。ペアリングを使った暗号は活発に研究され、実用化に向けた標準策定も進められています。

7.1 ペアリング

ペアリングは楕円曲線 E 上の 2 個の点の組からある有限体 F_q への写像です。

$$e : E \times E \longrightarrow F_q.$$

厳密な定義はかなりの数学的な準備を必要とするため 18 章で改めて紹介します。ここでは直感的に理解しやすい幾何学的な描写をします。

楕円曲線上の 2 点 P, Q が与えられると原点 O , $P, Q, R := P + Q$ で作られた平行四辺形 $OPRQ$ が決まります。

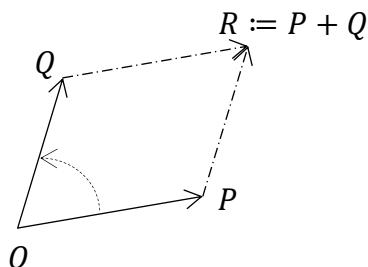


図 7.1 原点 O と点 P, Q から定まる平行四辺形 $OPRQ$

この平行四辺形 $OPRQ$ の面積は 2 点 P, Q で決まるので $S(P, Q)$ と書くことにします。なお、面積は符号付きで考えることにして Q が P の反時計周りにあるときを正、逆向きにあるときを負

とします。面積 $S(P, Q)$ を、有限体のある固定値 g の指数に乗せたものがペアリングの値です。 g については後ほど触れます。

$$\begin{array}{ccc} e : & E \times E & \longrightarrow F_q \\ & \Downarrow & \Downarrow \\ & (P, Q) & \longmapsto g^{S(P, Q)}. \end{array}$$

ペアリングの性質を調べましょう。まず $P = Q$ のときは平行四辺形がつぶれて面積が 0 になるのでペアリングの値は 1 になります。

$$e(P, P) = 1.$$

点 P, Q が与えられたとき、2 点 $2P, Q$ で決まる平行四辺形の面積は P, Q で決まる平行四辺形の面積の 2 倍です。

$$S(2P, Q) = 2S(P, Q).$$

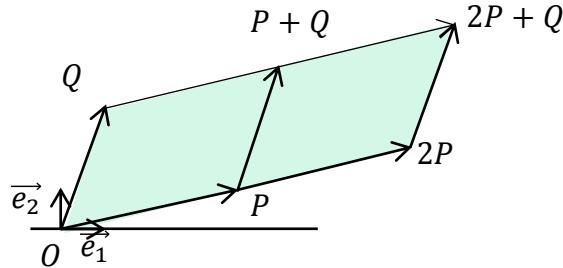


図 7.2 2 倍点の作る平行四辺形

ペアリングは面積を対応させる写像なので

$$e(2P, Q) = g^{S(2P, Q)} = g^{2S(P, Q)} = \left(g^{S(P, Q)}\right)^2 = e(P, Q)^2$$

と元のペアリングの値の 2 乗になります。一般に点 P を整数 a 倍すると面積は a 倍になるのでペアリングの値は a 乗になります。この性質は点 Q の側についても成り立ちます。よって一般に整数 a, b に対して

$$e(aP, bQ) = e(P, Q)^{ab}$$

が成り立ちます。ペアリングを暗号を使うときに一番重要な関係式です。

7.2 ID を使った鍵共有

ID (identifier) とは自分を示す何らかの識別子のことです。たとえば名前やメールアドレス、学生番号などが識別子になります。

1999 年に大岸氏たちにより提案されたペアリングを使った ID 鍵共有方式を紹介しましょう。ID はユーザを区別する識別子なので ID X をもつユーザをユーザ X ということにします。

セットアップ：鍵生成局が楕円曲線 E と、ID の集合から E への中への 1 対 1 の関数

$$H : \{ \text{ID} \} \rightarrow E$$

を決めて公開する。ユーザ X に対して $P_X := H(X)$ とおく。 P_X は誰でも計算できる。整数 s をマスター秘密鍵とする。

秘密鍵の生成：生成局は、各ユーザ X に対して $K_X := sP_X = sH(X)$ を秘密鍵として渡す。

鍵共有：

A は秘密鍵 K_A と、B の ID から求まる P_B を使って $s_A := e(K_A, P_B)$ を計算する。

B は秘密鍵 K_B と、A の ID から求まる P_A を使って $s_B := e(P_A, K_B)$ を計算する。

ペアリングのとり方の順序に注意。ID に順序関係を入れて小さい方の添え字を前にする（この場合は $A < B$ ）などの取り決めが必要。

$$s_A = e(K_A, P_B) = e(sP_A, P_B) = e(P_A, P_B)^s = e(P_A, sP_B) = e(P_A, K_B) = s_B$$

が成り立つので二人の間で秘密の値を共有できる。

この ID 鍵共有の安全性については後ほど触れるとして、ECDH 鍵共有との違いを比べます。

表 7.1 ECDH 鍵共有と ID 鍵共有の比較

	自分の秘密鍵	相手の公開情報	共有鍵
ECDH 鍵共有	a	bP	abP
ID 鍵共有	$sH(A)$	$H(B)$	$e(H(A), H(B))^s$

ECDH 鍵共有では秘密鍵 a や b を先に（自由に）決めてから、公開鍵 aP や bP を作ります。したがって、 aP や bP を自分の狙った数値にできません。それに対して、ペアリングを使った ID 鍵共有では公開情報 A や B を先に決めてから秘密鍵 $sH(A)$ や $sH(B)$ を作ります。そのため公開情報として自由な ID を使うことができます。

7.3 3 者間 DH 鍵共有

2000 年に Joux により提案されたペアリングを用いた 3 者間での鍵共有 [Jou00] を紹介しましょう。殆どの鍵共有法は 2 者間で行われます。これは 3 者間で鍵共有を行う珍しいものです。

楕円曲線上の 2 点 P, Q を固定して皆で共有します。3 人 A, B, C はそれぞれ整数 a, b, c を秘密鍵として $(aP, aQ), (bP, bQ), (cP, cQ)$ を公開します。 A は自身の秘密鍵 a と B, C の公開鍵 bP, cQ を使って

$$e(bP, cQ)^a = e(P, Q)^{abc}$$

を計算します。 B, C も同様に計算します。

$$\begin{aligned} B : \quad & e(aP, cQ)^b = e(P, Q)^{abc}, \\ C : \quad & e(aP, bQ)^c = e(P, Q)^{abc}. \end{aligned}$$

3人同時に $e(P, Q)^{abc}$ を共有することができました。この方法は ECDH 鍵共有の素直な拡張に見えますね。より一般に整数 a, b, \dots と何かの点 P, Q, \dots に対して

$$f(aP, bQ, cR, dS, \dots) = f(P, Q, R, S, \dots)^{abcd\dots}$$

となる関数 f （多重線形写像といいます）が見つかれば多人数での鍵共有ができそうです。実際多重線形写像があると、効率のよい放送暗号や様々な興味深い暗号プロトコルが構成されることが知られています。しかし残念ながら暗号に使えるそのような関数は今のところ構成されていません。ただ 2012 年、Garg, Gentry, Halevi たちがイデアル格子というものを使って多重線形写像の候補となるものを構成しました [GGH13]。また Coron, Lepoint, Tibouchi たちは別的方式で構成しています [CLT13]。2015 年初頭では処理速度や鍵サイズの観点からまだまだ実用的とはいえませんが今後の発展が望まれます。

7.4 双線形と非対称

これからペアリングを使った暗号プロトコルを紹介します。ただその前にもうしばらくペアリングの性質について調べましょう。楕円曲線上の点を 2 個の基底ベクトル \vec{e}_1, \vec{e}_2 を使って表現します。

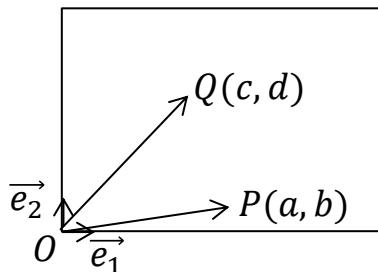
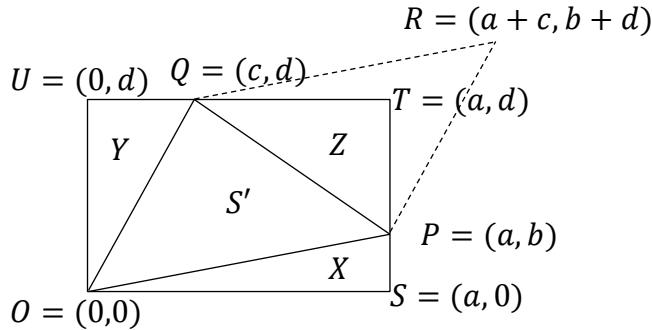


図 7.3 楕円曲線上の点の基底ベクトルを使った表現

任意の 2 個の点 P, Q を整数 a, b, c, d を使ってそれぞれ

$$\begin{aligned} P &:= a\vec{e}_1 + b\vec{e}_2, \\ Q &:= c\vec{e}_1 + d\vec{e}_2 \end{aligned}$$

とします。すると平行四辺形の 4 個の座標は $(0, 0), (a, b), (a+c, b+d), (c, d)$ です。この平行四辺形の面積 $S(P, Q)$ を求めます。

図 7.4 3 角形 OPQ と平行四辺形 $OPRQ$

まず 3 角形 OPQ をぴったり囲む長方形 $OSTU$ を考えます。ここで $S = (a, 0)$, $T = (a, d)$, $U = (0, d)$ です。3 角形 OPQ の面積 S' は、長方形 $OSTU$ の面積から 3 個の 3 角形 $X := \triangle OSP$, $Y := \triangle OQU$, $Z := \triangle PTQ$ の面積を引いたものです。3 個の 3 角形の面積はそれぞれ

$$X \text{ の面積} = \frac{1}{2}ab, \quad Y \text{ の面積} = \frac{1}{2}cd, \quad Z \text{ の面積} = \frac{1}{2}(a-c)(d-b).$$

よって

$$S' = ad - \frac{1}{2}ab - \frac{1}{2}cd - \frac{1}{2}(a-c)(d-b) = \frac{1}{2}(ad - bc)$$

となり、平行四辺形 $OPRQ$ の面積はその 2 倍なので $S(P, Q) = 2S' = ad - bc$ です。単位ベクトル \vec{e}_1 と \vec{e}_2 のペアリングの値を $g := e(\vec{e}_1, \vec{e}_2)$ とすると P と Q のペアリングの値は

$$e(P, Q) = g^{ad-bc}$$

となります。

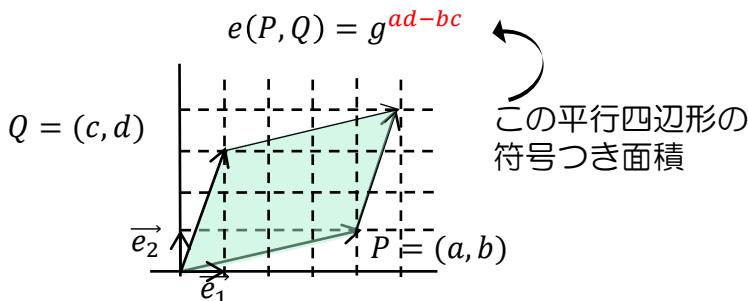


図 7.5 ペアリングの図形的な意味

特に点 $P_1 = a_1\vec{e}_1 + b_1\vec{e}_2$, $P_2 = a_2\vec{e}_1 + b_2\vec{e}_2$ について $P_1 + P_2 = (a_1 + a_2)\vec{e}_1 + (b_1 + b_2)\vec{e}_2$ よって

$$e(P_1 + P_2, Q) = g^{(a_1+a_2)c-(b_1+b_2)d} = g^{(a_1c-b_1d)+(a_2c-b_2d)} = e(P_1, Q)e(P_2, Q)$$

が成り立ちます。点の和のペアリングはそれぞれのペアリングの積になっています。点 Q_1, Q_2 に関する同様に

$$e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)$$

が成り立ちます。これらの性質を双線形 (bilinear) といいます。 P についても Q についても線形 (linear) だからです。右辺が値の和 $e(P, Q_1) + e(P, Q_2)$ ではなく値の積 $e(P, Q_1)e(P, Q_2)$ なので慣習的に線形と呼んでいます。それでペアリングのことを双線形写像 (bilinear map) ともいいます。

最後に点 P と Q を入れ換えるとどうなるか調べます。 P と Q を入れ換えると (a, b) と (c, d) がひっくり返るので

$$e(Q, P) = g^{cd-ab} = (g^{ad-bc})^{-1} = e(P, Q)^{-1}.$$

図形的には平行四辺形の面積がひっくり返って符号が変わったと解釈できます。

7.5 対称ペアリングと非対称ペアリング

前述で紹介したペアリングは Weil ペアリングと呼ばれ任意の点 P に対して $e(P, P) = 1$ です。

ペアリングが暗号を構成するのに重要な分かると、どんな楕円曲線やペアリングなら効率よく計算できるのか研究されました。 P と Q が異なる楕円曲線上の点となるペアリングも構成されています。ある種の楕円曲線には楕円曲線のある点 P の巡回群 $\langle P \rangle := \{nP \mid n \in \mathbb{Z}\}$ に含まれない別の点 Q に移す線形写像 ψ が存在します。 ψ は線形写像なので整数 a に対して

$$\psi(aP) = a\psi(P) = aQ$$

が成り立ちます。

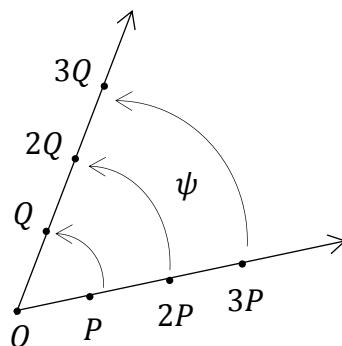


図 7.6 distortion 写像

ψ を distortion 写像といいます。2001 年 Boneh, Franklin が提案しました [BF03]。

ψ を使って任意の P の倍数 aP, bP に対して写像 $e' : \langle P \rangle \times \langle P \rangle \rightarrow F_q$ を

$$e'(aP, bP) := e(aP, \psi(bP))$$

と定義します。片方の点を distortion 写像で移してからペアリングをとるのです。すると $e'(P, P) = e(P, Q) \neq 1$ です。また ψ の線形性と e の双線形性から

$$e'(aP, bP) = e(aP, \psi(bP)) = e(aP, b\psi(P)) = e(P, \psi(P))^{ab} = e'(P, P)^{ab}$$

という関係式が成り立ちます。特に $e'(aP, bP) = e'(bP, aP)$ です。 aP と bP を交換してもペアリングの値は同じなので e' を対称ペアリング、あるいは type I のペアリングといいます。

それに対して distortion 写像のような片方からもう片方への写像がないペアリングを type III といいます。distortion 写像があっても逆向きの写像が簡単には計算できないときは type II ペアリングです。type II と type III を合わせて非対称ペアリングといいます。

非対称ペアリングは異なる楕円曲線の 2 点 P, Q が必要なのに対して、対称ペアリングは 1 点 P だけあればよく、必要なパラメータの数が少なくなります。ペアリングを使った暗号プロトコルではパラメータの数が少ない方がなにかと便利なので対称ペアリングがよく使われます。

たとえば 7.3 節で紹介した 3 者間 DH 鍵共有は対称ペアリング e' を使うと次のようになります。楕円曲線上の点 P を固定してみなで共有します。3 人 A, B, C はそれぞれの整数 a, b, c を秘密鍵として aP, bP, cP を公開します。3 人は

$$\begin{aligned} A : \quad & e'(bP, cP)^a = e'(P, P)^{abc}, \\ B : \quad & e'(aP, cP)^b = e'(P, P)^{abc}, \\ C : \quad & e'(aP, bP)^c = e'(P, P)^{abc} \end{aligned}$$

により鍵共有を行います。共有、転送すべきパラメータが減ったのと、ペアリングの順序を気にしなくてよいのですっきりしますね。このテキストでも以降、対称ペアリングを前提とし記号も e' ではなく e と書くことにします。

ただ 2012 年頃から対称ペアリングでよく使われていた曲線に対する DLP を解く方法が大きく進展しました [AMORH14]。したがって実際に使う場合は非対称ペアリングを用いる方が多いと思われます。

なお暗号の教科書や論文では、巡回群 $\langle P \rangle$ を G と書き、楕円曲線上の点の足し算演算を乗法表記するのが主流です。これは有限体や楕円曲線などの具体的なものではなく、巡回群 G があれば暗号プロトコルを考えることができるという抽象化の流れの一環です。たとえば対称ペアリングの関係式は a, b を整数、 $g \in G$ として

$$e : G \times G \ni (g^a, g^b) \longmapsto e(g, g)^{ab} \in G$$

と書きます。このテキストでは演算が楕円曲線の足し算というのを強調したいのと、乗法表記すると指指数が小さくなつて読みにくくなることが多いと思うので加法表記にしています。

7.6 DLP と ECDLP とペアリング

ペアリングは有限体と楕円曲線をつなぐ架け橋になっています。ペアリングが満たす関係式を再度記してそのことを確認します。 P を楕円曲線上の点、 a, b を整数、 $g := e(P, P)$ としたとき

$$e(aP, bP) = g^{ab}$$

が成り立つのでした。 P と b を固定して a だけ動かしてみましょう。

$$f : E \ni X \mapsto e(X, bP) \in \mathbb{F}_q$$

という写像を考えることになります。この f を $e(\cdot, bP)$ と書くことにします。一つ目の \cdot の部分に楕円曲線の点を入れて値を確定させてねという意味です。すると

$$f(aP) = e(aP, bP) = e(P, bP)^a = f(P)^a$$

なので楕円曲線上の a 倍が有限体上の a 乗に対応していることが分かります。

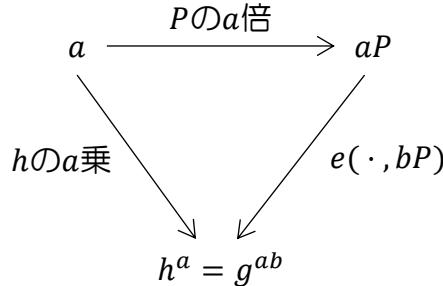


図 7.7 点の a 倍と a 乗とペアリングの関係

ここで P と aP が与えられたとします。するとペアリング $e(\cdot, bP)$ を作用させることで $h := e(P, bP) = g^b$ と $e(aP, bP) = g^{ab} = h^a$ を求められます。もし、 h と h^a から a を求められたなら、もとの楕円曲線上の離散対数問題が解けたことになります。

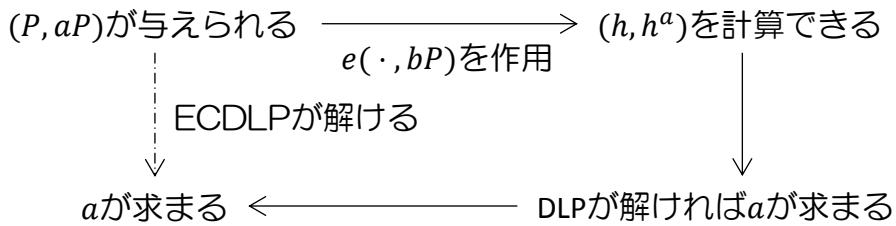


図 7.8 MOV リダクション

つまり ECDLP を DLP に帰着できるということです。1990 年、この考え方による ECDLP の攻撃が提案されました。MOV リダクション（Menezes, Okamoto, Vanstone）と呼ばれます。もともとペアリングは楕円暗号を攻撃するための道具として利用されたのでした。

私は 1999 年頃暗号に興味を持ち、当時の京都工芸繊維大学の笠原研究室にお伺いしていました。暗号のことを教えていただきながら代わりに私はペアリングの数学的な定義などの解説をしました。そうした中で研究室の方がこのペアリングを暗号に使うことを思いつきます。暗号を破るために使われた手法を、新しい暗号を作るために使おうという発想です。ECDLP が DLP に帰着されるということですが、逆に言えばその DLP が困難なら ECDLP も困難でしょう。

話を戻してもう一度式を見直します。 aP や bP があったときに ECDH 仮定により abP を求めるのは困難です。また g, g^a, g^b が与えられても DH 仮定により g^{ab} を求めるのは困難です。

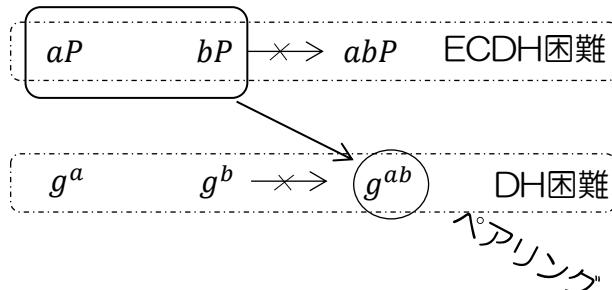


図 7.9 DH, ECDH, ペアリングの関係

しかし、 aP と bP があったときにペアリングを使うとまるで DH 問題を解いたかのように g^{ab} を計算できます。つまりペアリングを使うと一度だけ DH 問題を解く手段を手に入れたと解釈できます。

7.7 ID ベース暗号

ID ベース暗号とは公開鍵に自分の好きな文字列 ID を選べる暗号のことです。たとえば自分のメールアドレスを自分の公開鍵として使うことができます。他の人が A さんに暗号文を送信したければ、A さんのメールアドレスと自分の秘密鍵を使って暗号文を生成できます。従来の公開鍵暗号では公開鍵が本当にその本人のものであるという認証が必要でした。メールアドレスは普段使っているものですから、間違えようがありません。

ID ベース暗号の概念自体は 1980 年代からあったのですが、実際にどうすれば効率のよいものが作れるかわかつていませんでした。ペアリングを使った手法が提案されることで実際に使えるものとなり始めたのです。

まずペアリングを使った ID ベース暗号の一つを紹介します。これは大岸、境、笠原、Boneh, Franklin たちによるものです。7.2 節で紹介した ID を使った鍵共有のアイデアを使っています。

システムセットアップ: 鍵生成局が橿円曲線 E , ID の集合から E へのハッシュ関数 $H_1 : \{\text{ID}\} \rightarrow E$, E から有限体へのハッシュ関数 $h_2 : E \rightarrow \mathbb{F}_q$ を決めて公開する。ユーザ X に対して $P_X := H_1(X)$ とおく。 P_X は誰でも計算できる。整数 s をマスター秘密鍵とする。橿円曲線 E 上の点 P と sP を公開する。

秘密鍵の生成: 生成局はユーザ X に対して $K_X := sP_X$ を秘密鍵として渡す。

ユーザ A に対して平文 m を暗号化: 亂数 r をとり $\text{Enc}(m) := (rP, m \oplus h_2(e(P_A, sP)^r))$ を暗号文とする。

ユーザ A の復号: 暗号文 $C = (U, v)$ を受け取り $\text{Dec}(U, v) := v \oplus h_2(e(K_A, U))$ とする。

ペアリングの性質から

$$e(P_A, sP)^r = e(P_A, P)^{sr} = e(sP_A, rP) = e(K_A, rP)$$

が成り立ちます。すると

$$\begin{aligned} \text{Dec}(\text{Enc}(m)) &= \text{Dec}((U, v)) = v \oplus h_2(e(K_A, U)) = (m \oplus h_2(e(P_A, sP)^r)) \oplus h_2(e(K_A, rP)) \\ &= (m \oplus h_2(e(K_A, rP))) \oplus h_2(e(K_A, rP)) = m \end{aligned}$$

となり正しく復号できます。この ID ベース暗号は、CBDH (Computational Bilinear Diffie-Hellman) 仮定, 他にいくつかの仮定の元で安全とされています。CBDH 仮定とは次の CBDH 問題と呼ばれる問題が難しいという仮定です。単に BDH 問題ともいわれます。

「橿円曲線上の点 P と整数 a, b, c に対して (P, aP, bP, cP) が与えられたときに $e(P, P)^{abc}$ を求めよ。」

その後、より安全なバージョンや効率のよい方法などがいろいろ提案されています。CRYPTREC 報告書の 2008 年度「ID ベース暗号に関する調査報告書」[CRYPT08] の 1 章では様々な方式が、2 章では様々な安全性仮定が紹介されています。付録 A でもこのテキストに登場する安全性仮定をまとめています。

7.8 ID ベース暗号と公開鍵暗号の違い

ID (公開鍵) としてメールアドレスを使えば公開鍵が本人であることは間違いないと思われます。すると公開鍵暗号で面倒な PKI は不要となり、これからは ID ベース暗号に置き換わっていくものなのでしょうか。実はそんなに簡単なものではありません。

まず、ID ベース暗号だと鍵生成局はマスター鍵を持っているのでどの人の秘密鍵も作成できます。これは相当強い権限ですね。またマスター鍵が漏洩すると全ての人の鍵が分かってしまいまます。この問題に対しては生成局を分散化する方式が提案されています。そうすると、たとえ一つの生成局が不正をしても安全性は保たれます。

また鍵生成局はユーザから依頼されたときにその人の ID に対する秘密鍵を渡します。それには ID に対する本人確認と秘密鍵を渡すための安全な通信経路が必要でしょう。

鍵の失効に関する問題もあります。何らかの不手際で万一自分の秘密鍵が漏れてしまったとします。従来の公開鍵暗号では新しい鍵を作り直して認証局に鍵の失効手続きをすればよいです。しかし、IDベース暗号で自分のアドレスを使っていた場合はどうすればよいでしょうか。自分のメールアドレスを変えるのはなかなか不便です。銀行の通帳や車の免許証のようにメールアドレスの末尾に発行回数を追加しておくという手はあります。あるいは鍵の有効期間をIDに含めてよいです。いずれにせよ古い番号のものが失効されたかどうかを確認するにはPKIのような鍵の管理システムが必要になります。そこでPKIとIDベース暗号の長所を組み合わせた証明書ベースの暗号方式や証明書不要の方式が提案されています。

というように実際に運用を考えるといろいろな枠組みが必要であることに気づきます。とはいっても、IDベース暗号は従来の公開鍵暗号では実現できない機能をいろいろ持っています。たとえばメールアドレスをIDとするIDベース暗号が運用されていたとすると、ある人のメールアドレスだけでその人に秘密の文章を送れます。その人がまだIDベース暗号のシステムに加入していないとも暗号化できるのです。受け取った人は、その文章を読みたければそこで初めて鍵生成局に自身の秘密鍵を申請して復号します。なんだか加入が強制され気持ちは悪いですが暗号化が先にできるというのは面白いですね。

7.9 タイムリリース暗号

IDベース暗号のIDとして時刻を使うと何ができるでしょうか。前節最後の秘密鍵がなくても暗号化が先にできるという話が生きてきます。タイムリリース暗号というものを紹介しましょう。これは決められた時刻になるまでは復号できないことを保証する暗号方式です。

会社の投資家向けの情報（IR情報）について考えてみます。IR情報は株価を左右するかもしれない重要な情報です。IR情報の公開前に証券会社がその情報を利用してインサイダー取引などの不公平な株の取引が行われてる事件があります。これに対処するために会社はその情報をたとえば午後3時を指定して暗号化し直接投資家に配布する方法が考えられます。

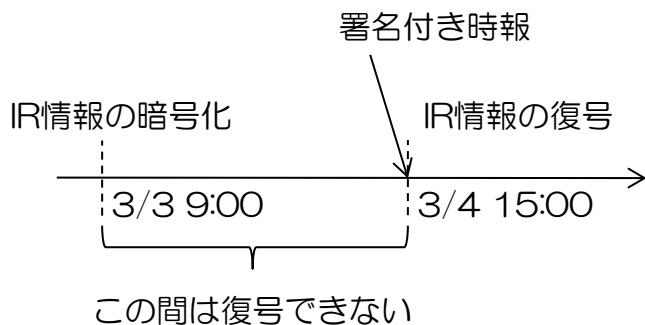


図 7.10 タイムリリース暗号

既存の枠組みでこのような機能を持ったシステムはどうやって作るか考えてみましょう。まず、復号側で決められた時刻になったか否かを知る必要があります。そのデバイスの内蔵時計を見ればよい？その時刻を勝手に変更されてしまう危険性があります。とするとインターネットか電波経由で正しい時刻になったことをその都度確認しなければなりません。これは論理的に必要な条件です。

さて、正しい時刻になったことを確認すれば復号を開始するわけですが、復号鍵を事前に持っていると悪意ある復号者が時刻とは無関係にその鍵を見つけてしまう恐れがあります。DRM（デジタル著作権管理）ソフトはそのようなことができないよう、耐タンパー性技術を用いて復号鍵を隠し持たせることができます。しかし耐タンパー性を信用できなければ、結局その時刻になったら復号鍵をもらうという単純な方式に落ちつけます。

これに対してタイムリリース暗号は次のような形になります。鍵生成局は正しい時刻を配信する時刻センターが担います。

セットアップ：時刻センターは整数 s と楕円曲線 E とその点 P を選び (P, sP) を公開する。 s を秘密鍵とする。時刻センターは一定間隔（たとえば1秒）で現在時刻 T と $sH(T)$ の組 $(T, sH(T))$ を配送する。

暗号化：あるユーザが時刻 T_0 になるまで平文 m を復号させたくないとする。乱数 r を選び $s_{T_0} = e(H(T_0), rsP)$ を計算する。 s_{T_0} で m を暗号化した $\text{Enc}(s_{T_0}, m)$ と rP を配送する。

復号：時刻 T_0 に時刻センターから $sH(T_0)$ を受け取るので $e(sH(T_0), rP) = s_{T_0}$ を計算し m を復号する。

時刻 T に時刻センターから鍵に相当する $sH(T)$ をもらうのですから、一見先ほど考察した従来の方式とあまり変わらないように見えます。しかしその場合、暗号化するひとそれぞれが鍵を復号したい人に渡す必要がありました。タイムリリース暗号では暗号化する人は時刻 T に何か行動を起こす必要はありません。時刻センターが不特定多数に一度に $sH(T)$ を配信するだけです。復号時刻が同じ複数の暗号文を一つの $(T, sH(T))$ で復号できます。この点が大きな違いです。

また時刻 T に受け取った $Q = sH(T)$ が正しいものか否かを確認するには公開情報 P と sP を使って

$$e(Q, P) = e(H(T), sP)$$

の等号成立を確認すればできます。つまり時刻センターは署名付きの時刻を配信しているだけと考えることもできます。既に電子取引のために正確な時刻を提供するサービスはあります。そのサービス提供者がこの署名付きの時刻配信を担うとよいように思います。

この形のタイムリリース暗号では時刻 T_0 になると不特定多数の人が復号できます。最初に述べた会社のIR情報を配信したい場合はこの機能で十分でしょう。しかし特定の人にのみ復号したい場合はこの方式を拡張する必要があります。

7.10 ペアリングを使ったデジタル署名

タイムリリース暗号である時刻に対する復号鍵が正しいか否かを確認できて署名の役割を果たしました。一般にIDベース暗号ではIDとしてメッセージを選ぶとその復号鍵は署名の役割を持つことが分かります。この考え方でいくつか効率のよいデジタル署名が構成されています。ここでは一つを紹介しましょう。

セットアップ：楕円曲線上の点 P と整数 x, y をランダムに決めて

$$Q := xP, \quad R := yP$$

とする。 (P, Q, R) が公開鍵で (x, y) が秘密鍵となる。

文章 m に対する署名： r をランダムにとり

$$S := \frac{1}{x+m+yr}P$$

として (S, r) を署名とする。

検証：公開鍵 (P, Q, R) と署名 (S, r) に対して

$$e(S, Q + mP + rR) = e(P, P)$$

が成り立てば署名を受理する、そうでなければ棄却する。

正しい署名を検証した場合は、

$$\begin{aligned} e(S, Q + mP + rR) &= e\left(\frac{1}{x+m+yr}P, xP + mP + ryP\right) \\ &= e\left(\frac{1}{x+m+yr}P, (x+m+yr)P\right) \\ &= e(P, P) \end{aligned}$$

により検証が通ります。この署名は15.5節で紹介する n -SDH仮定の元で安全であることが示されています。

7.11 この章のまとめ

ペアリングと呼ばれる楕円曲線の2点から有限体のある値が決まる関数を紹介しました。この関数は片方の点を2倍にすると関数の値が2倍になる性質があります。この性質を双線形といいます。

ペアリングを使うとIDベース暗号を構成できます。IDベース暗号はユーザの好きな文字列(ID)を公開鍵にできる暗号方式です。ペアリングはIDベース暗号の他にもデジタル署名やこれから紹介する様々な暗号の構成に使われます。

第8章

検索可能暗号

クラウドサービスの発展に伴ってデータを外部のサーバに預ける機会が増えています。重要なデータは万一のデータ漏洩に備えて手元で暗号化してからサーバに置きたいものです。そうすると安全性は高まりますが、データを暗号化したままサーバにそれらの処理をさせる仕組みが望まれます。この章では暗号化したまま検索を行う方法を紹介しましょう。

8.1 キーワード検索

いきなり任意の検索ができるものを考へるのは難しいのでまずキーワード検索から始めます。ここでキーワード検索とは、各文章データに対して予めいくつかのキーワードを割り当てておき、ユーザがキーワードを入力すると、そのキーワードを含む文章を提示してくれる検索機能とします。

データ	キーワードリスト		
文章1	社外秘	予算	計画
文章2	旅行	予算	
文章3	社外秘	面接	
文章4	アルバイト	面接	
文章 ...	キーワード ...		

図 8.1 キーワード検索

たとえば図8.1では、ユーザが「社外秘」というキーワードを提示すると、サーバ側でキーワードリストからそのキーワードを探査します。そして見つかったキーワードリストに対応する文章1と文章3をユーザに伝えます。

文章などの各データは、検索時に必要がないので従来の共通鍵暗号で暗号化しておきます。各キーワードも暗号化しましょう。キーワード全体を $\{m_1, \dots, m_n\}$ とします。ユーザは初期化ベクトルIV（1.9節参照）を固定して共通鍵暗号を使ってキーワードを暗号化します。秘密鍵 K で全てのキーワードを暗号化してキーワードリストを作りサーバに置きます。そしてキーワード m を暗号化して $\text{Enc}(K, m)$ をサーバに送ります。サーバは $\text{Enc}(K, m)$ と $\text{Enc}(K, m_i)$ が一致するものを見つけて対応する文章を返します。

この方法はサーバが文章やキーワードの中身を見られないという利点があります。しかしIVを固定しているので同じキーワードは同じ暗号文になります。するとサーバはどのキーワードリストとどのキーワードリストが同じキーワードを持っているか分かります。たとえば図8.1の例では「文章1」と「文章3」は同じキーワード（サーバはどんなキーワードかは分からぬ）を持っている、「文章1」と「文章2」も「予算」（やはりサーバには中身は分からぬ）という同じキーワードを持っているということは分かります。

そこで安全性を高めるために同じキーワードを暗号化しても異なる暗号文になってほしいです。単にキーワードごとにIVを変えると同じキーワードでも異なる暗号文になります。しかし、それではサーバが二つのキーワードが同じか否かを判定することができません。

したがって、暗号化する度に同じキーワードでも異なる暗号文になるのだけれども、それを復号しなくともとのキーワードが一致しているかどうかを判別することだけはできるような暗号方式が望ましいです。こういった性質を満たすものを検索可能暗号といいます。

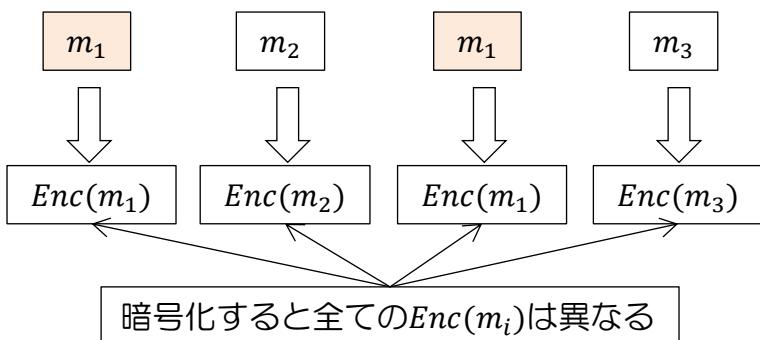


図8.2 異なる暗号文

なお、キーワードが一致するかを判別するのはサーバです。だから、せっかく同じキーワードでも異なる暗号文にしたとしてもサーバにとってはどの暗号化キーワードとどの暗号化キーワードが同じか分かってしまいます。ただ、外部の人にとってはどの暗号化されたキーワードが同じかは判らないので、うっかりサーバのデータが流出してもキーワードが推測されるリスクは減っています。

す。完全に秘匿する方法も提案されていますが、効率は悪いようです。このように検索可能暗号は効率と安全性の落とし処が難しいです。

8.2 キーワード検索公開鍵暗号

2004 年に提案されたキーワード検索公開鍵暗号 (PEKS : Public key Encryption with Keyword Search) を紹介します [BCOP04]。このアルゴリズムでは、まず A さんが公開鍵と秘密鍵を用意します。暗号化されたキーワードが一致しているかどうかを確認する人（たとえばサーバ）を S とします。A は秘密鍵を用いてあるキーワード w に対して落とし戸と呼ばれる特別な値 T_w を作ります。 T_w を S に渡します。それから A は自身の公開鍵を用いていろいろなキーワード w を暗号化して p_w を作ります。同じ w でも暗号化するたびに異なる p_w になります。 p_w は公開鍵だけで作れるので他の人が作っても構いません。S は与えられた p_w と T_w が同じ w から作られたものかどうかの確認ができます。どんなキーワード w を暗号化したのかは分かりません。詳細なアルゴリズムは次の通りです。

鍵生成：椭円曲線 E とその上の点 P を決める。A は乱数 a をとり秘密鍵 K_{priv} とする。 $Q := aP$ として $K_{\text{pub}} := (P, Q)$ を公開鍵とする。 H_1 をキーワードから E へのハッシュ関数、 h_2 を有限体から整数へのハッシュ関数とする。

PEKS：乱数 r をとり、公開鍵 K_{pub} を用いてキーワード w を暗号化する。

$$\text{PEKS}(K_{\text{pub}}, w) := (rP, h_2(e(H_1(w), rQ))) \in E \times \mathbb{Z}.$$

落とし戸関数：A は秘密鍵 K_{priv} を用いてキーワード w に対して落とし戸を作る。

$$\text{Trapdoor}(K_{\text{priv}}, w) := aH_1(w).$$

テスト： $p_w := \text{PEKS}(K_{\text{pub}}, w) = (R, h) \in E \times \mathbb{Z}$ と落とし戸 $T_w := \text{Trapdoor}(K_{\text{priv}}, w)$ を用いて

$$h_2(e(T_w, R)) = h$$

が成り立てば p_w と T_w は同じ w に対応するものだったことが分かる。

テストの正当性は

$$h_2(e(T_w, R)) = h_2(e(aH_1(w), rP)) = h_2(e(H_1(w), arP)) = h_2(e(H_1(w), rQ)) = h$$

から分かります。

8.3 問題点と改良

前述のアルゴリズムは BDH 仮定の元で安全であることが示されます。ただこのアルゴリズムの落とし戸関数は秘密鍵 a とキーワード w だけから決まる決定的な関数です。したがって、落とし

戸の値を暗号化されていない状態でやりとりしていると、盗聴者はそのデータを集めてキーワードの分布からキーワードの重要度などを推測することができるかもしれません。そのため、落とし戸関数も確率的関数にする方法が提案されています。

また別の問題もあります。サーバがあるキーワード w_0 に対する落とし戸 T_{w_0} をもらったとします。サーバはその値から w を求めることはできないとされています。しかし、PEKS は公開鍵さえあれば任意のキーワード w に対して求めることができます。

したがって、サーバがキーワードに使われそうな単語リストを用意して次の方法で辞書攻撃ができます。まず、そのリストの各 w に対して PEKS p_w を求めます。落とし戸 T_{w_0} と p_w に対してテストをして $w = w_0$ かを確認します。 $w = w_0$ になるような w が見つかるまで w を次々と変えてやり直します。

このような攻撃ができないような方法も研究されています。また、共通鍵を使った方式 SSE (Searchable Symmetric Encryption) も研究されています。

8.4 この章のまとめ

データを暗号化したまま検索する検索可能暗号の紹介をしました。キーワードを秘匿にしたまま検索する研究では、そもそも何を隠したいのか、演算効率、想定される攻撃などについて様々なモデルが提案されています。キーワードの完全一致ではなく部分一致に対応したものも研究されています。これらの手法では事前に決められたキーワードのリストを使います。そのリストが固定では使い勝手がよくありません。リストの更新が容易にできるものが望ましいため、それらの手法も研究されています。

第9章

プロキシ暗号

情報漏洩を心配して手元でデータを暗号化してからクラウドサーバに置きます。一人でデータを利用しているのならよいのですが、複数人でデータをやりとりする場合はその暗号化に使った鍵を別途共有する必要があります。この問題を解決する一つの方法がプロキシ暗号です。

9.1 概要

プロキシ暗号は、代理暗号、代理人暗号、再暗号化、代理人再暗号化などともいう暗号技術です。英語では Proxy Re-Encryption (PRE) といいます。

Aさんが仕事で扱っているメールがあります。AのメールサーバにあるデータはAの公開鍵で暗号化されています。Aの休暇中にBが代わりにそのデータを扱うことになりました。

Aの秘密鍵をBに渡すのは好ましくありません。そうするとAはサーバからデータをダウンロードし、自身の秘密鍵で復号してBの公開鍵で暗号化し、再度サーバにアップロードしなくてはなりません。データを共有したい人が増えるとこれは大変です。

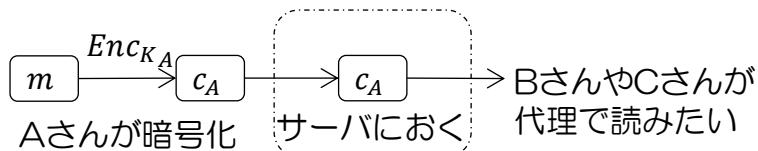


図 9.1 暗号化データを共有したい

プロキシ暗号を使うとサーバ上でデータを復号しなくてもBが復号できるように変換できます。その作業をサーバに委ねることができるのでプロキシ（代理人）暗号といいます。

AはBの公開鍵暗号 K_B と自身の秘密鍵 S_A から再暗号化鍵 $R_{A \rightarrow B}$ を作ります。再暗号化鍵を使うと、平文 m をAの公開鍵で暗号化した暗号文 $c := \text{Enc}(K_A, m)$ をBの秘密鍵で復号できる

ように変換できます。

$$\begin{aligned} \text{ReEnc}(R_{A \rightarrow B}, \text{Enc}(K_A, m)) &= \text{Enc}(K_B, m), \quad \text{すなわち} \\ \text{ReEnc}(R_{A \rightarrow B}, c) &= \text{Enc}(K_B, \text{Dec}(P_A, c)). \end{aligned}$$

あたかも、一度 A の秘密鍵で復号して B の公開鍵で暗号化したかのように見えるのがポイントです。なお、後述するように再暗号化する前とした後で暗号方式が異なることがあります。

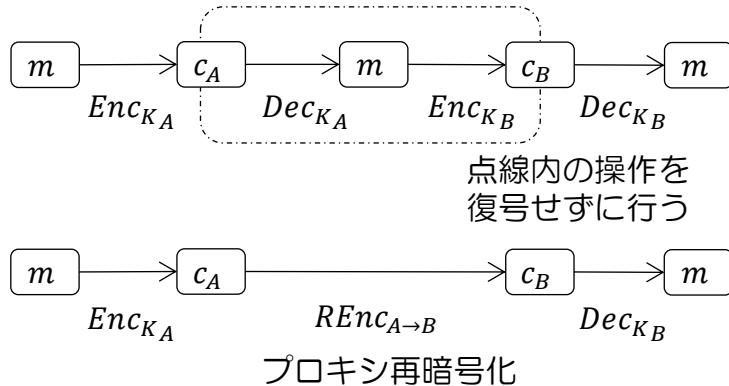


図 9.2 プロキシ暗号

9.2 最初の方式

1996 年、満保雅浩氏、岡本栄司氏が代理署名という概念を提案し、1998 年、Blaze, Bleumer, Strauss が原子的プロキシ暗号 (Atomic Proxy Cryptography) を提案します [BBS98]。原子的というのは復号と再暗号化を分離せずに一つの処理として行うという意味です。これは ElGamal 暗号を変形したものでした。

鍵生成：乗法的巡回群 $G := \langle g \rangle$ をとります。 g は公開されます。A は整数 a をランダムに選んで秘密鍵とし g^a を公開鍵とします。B も同様に b を秘密鍵、 g^b を公開鍵とします。 $r_{A \rightarrow B} := b/a$ を再暗号化鍵とします。

暗号化：A は平文 m に対して整数 r をランダムに選び

$$c_a := (mg^r, g^{ar})$$

を暗号文とします。

通常の復号：暗号文 $c_a = (u, v_a)$ に対して

$$\frac{u}{(v_a)^{1/a}} = \frac{mg^r}{(g^{ar})^{1/a}} = \frac{mg^r}{g^r} = m$$

と復号する。

再暗号化：暗号文 $c_a = (u, v_a)$ に対して $r_{A \rightarrow B}$ を用いて

$$c_b := (u, v_a^{r_{A \rightarrow B}})$$

とします。

再暗号化された $c_b = (u, v_b)$ は

$$v_b := v_a^{r_{A \rightarrow B}} = (g^{ar})^{b/a} = g^{br}$$

となり B の秘密鍵 b を用いて

$$u/(v_b)^{1/b} = m^{gr}/g^r = m$$

と復号できます。

この方式はいくつか欠点があります。まず再暗号化鍵 $r_{A \rightarrow B} = b/a$ をどうやって作るかという問題です。A や B が $r_{A \rightarrow B}$ を知ってしまうと相手の秘密鍵が分かってしまうので別の誰かに a と b をそれぞれ教えて b/a を計算してからプロキシにのみこっそり教えるしかありません。あまり嬉しくありませんね。

それからプロキシと A が結託すると B の秘密鍵 b が分かります（A と B を入れ換ても同様）。それからプロキシは b/a から a/b を計算できるので B の暗号文を A が復号できるようにも再暗号化できます。これは利点のときもありますが、B が許可したつもりのない操作かもしれません。

9.3 改良方式

2006 年 Ateniese たちは前節の方式を改良し、DBDH 仮定のもとで結託攻撃ができない方式を提案します [AFGH06]。これは秘密鍵の受け渡しが不要なものでした。

鍵生成： E を椭円曲線、 P をその上の点、 $e : E \times E \rightarrow G$ を対称ペアリングとします。

$g := e(P, P) \in G$ とします。 P や g は公開されます。A は整数 a をランダムに選び秘密鍵とし、 aP を公開鍵とします。B も b を秘密鍵、 bP を公開鍵とします。

再暗号化鍵生成：A は自身の秘密鍵 a と B の公開鍵 bP を用いて

$$r_{A \rightarrow B} := (1/a)(bP) = (b/a)P$$

を求めます。

暗号化：A は整数 r をランダムに選び

$$c_a := (mg^r, r(aP))$$

を暗号文とします。

通常の復号：暗号文 $c_a := (u, V_a)$ に対して

$$\frac{u}{e(V_a, (1/a)P)} = \frac{mg^r}{e(raP, (1/a)P)} = \frac{mg^r}{e(P, P)^r} = \frac{mg^r}{g^r} = m$$

と復号する。

再暗号化：暗号文 $c_a := (u, V_a)$ に対して

$$e(V_a, r_{A \rightarrow B}) = e(raP, (b/a)P) = e(P, P)^{rb} = g^{rb}$$

とし，再暗号化文を (u, g^{rb}) とします。二つ目の元はペアリングをとることで橿円曲線の点から有限体の元になったことに注意してください。

再暗号化された暗号文の復号：再暗号化された $c_b = (u, v_b)$ に対して B の秘密鍵 b を用いて

$$\frac{u}{v_b^{1/b}} = \frac{mg^r}{(g^{rb})^{1/b}} = \frac{mg^r}{g^r} = m$$

とします。

この方式は最初の方式に比べていろいろ改善されています。まず，再暗号化鍵は自分の秘密鍵と相手の公開鍵のみから作成できるので余計な秘密の通信がいりません。それからプロキシと B が結託しても $(b/a)P$ と b からは a を求めることができません。

更にプロキシは P と $(b/a)P$ から $(a/b)P$ をつくることができない（15.3 節参照）ので逆向きに利用されることはありません。なお，これと異なりクラウドが $r_{A \rightarrow B}$ と $r_{B \rightarrow A}$ を相互に変換できる方式も提案されています。どちらがよいかはアプリケーションによるでしょう。

また通常の暗号化と再暗号化で方式が異なるので再暗号化は一度しかできません。たとえばプロキシが A さんから B さんへの再暗号化鍵 $r_{A \rightarrow B}$ ，B さんから C さんへの再暗号化鍵 $r_{B \rightarrow C}$ を持っていても，A さんの暗号文を C さんが復号できるようにはできないということです。これは復号可能な人を細かく制御できるというメリットがあります。

9.4 その後の進展

プロキシ暗号は様々な拡張が考えられます。たとえば再暗号化鍵に有効期間を設ける，メンバーの削除時に再暗号化鍵を無効化する，別の暗号方式にプロキシ暗号の機能を持たせるなどです。

ただプロキシ暗号は通常の暗号化，復号の他に再暗号化鍵，再暗号化する前後の暗号文など多くの情報があります。結託攻撃も複数のユーザによる結託だけでなく，プロキシが関わる結託攻撃もあります。したがって安全性を厳密に定義し証明することが難しく，複雑な安全性仮定を置くこともあります。そのため公開鍵暗号，電子署名や秘密分散といった基本的な技術を組み合わせてプロキシ暗号を構成して安心感を高める設計方式も提案されています [花岡 14]。

9.5 この章のまとめ

プロキシ暗号を紹介しました。プロキシ暗号は，暗号文を復号することなく別の人気が復号できる暗号文に変換する暗号です。その再暗号の操作は変換鍵を委託された第三者が行えます。そのためクラウドサービスと相性がよく，他の暗号と組み合せた方式が考えられています。

第 10 章

放送型暗号

この章では放送型暗号を紹介します。放送型暗号とは名前の通り、デジタル放送などの放送時ににおけるコンテンツの保護を目的とした暗号方式です。

10.1 放送型暗号の目的

CS デジタル放送には映画やスポーツ中継などの番組を有料で提供するサービスがあります。この様なサービスではコンテンツを暗号化しておき、契約をした人にのみが視聴できる限定受信を実現したいです。限定受信は、たとえば契約者ごとに異なる秘密鍵を配布してコンテンツを各異なる秘密鍵で暗号化して配布する方法が考えられます。

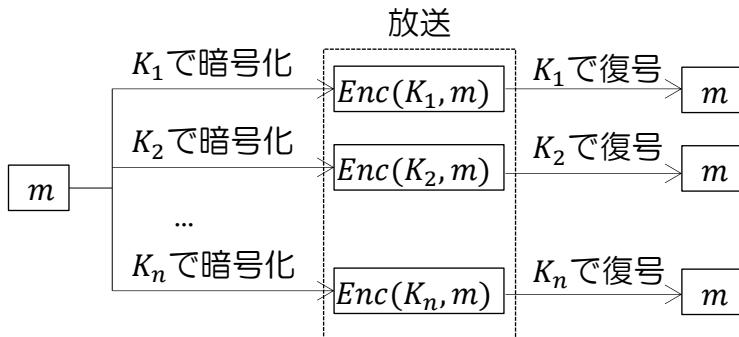


図 10.1 平文を別々に暗号化して放送

この方法はユーザが n 人いると、放送すべきサイズがコンテンツの n 倍になってしまう問題があります。たとえばコンテンツが 1GB でユーザが 1 万人いたら 10TB です。これでは放送の帯域がいくらあっても足りません。

そこでコンテンツ m をある秘密鍵 K で暗号化し、その秘密鍵 K を各ユーザ鍵 K_i で暗号化して、 $(\text{Enc}(K, m), \text{Enc}(K_1, K), \text{Enc}(K_2, K), \dots)$ を配布することになるでしょう。この場合、 n 倍

になるのは秘密鍵 K の大きさなので放送すべきデータがかなり削減できます。とは言え、ユーザが増えると送信すべきデータのサイズが増えるのは変わりません。また K が漏洩したときに素早く対処するため K は短いタイミングで新しいものに更新されているそうです。デジタルテレビで使われる B-CAS カードはそういった通常の共通鍵暗号と DRM を組み合わせて限定受信機能を実現しています。ただ 2012 年に、DRM 周りが破られ不正なカードが出回るなどの社会問題になりました。

放送型暗号はこのような方法よりも安全で効率のよい方法を目指して研究されています。次節以降で放送型暗号に求められるいくつかの機能を紹介しましょう。

10.2 ユーザの排除機能

放送型暗号は暗号文のサイズを小さくするだけが目的ではありません。通常のサービスでは加入者は増えたり減ったりします。特定のユーザを復号できなくするしくみを排除（revocation）といいます。

IPSec のマルチキャスト配信などで使われている木構造を使った排除機能の方法を紹介しましょう。説明を簡単にするために $n = 8$ とします。8人のユーザを2分木で管理します。各ノードに秘密鍵をおき、各リーフにいるユーザにはそのリーフからノードをたどってルートにまでいく途中のノードにある秘密鍵を渡します。つまりユーザ u_1 には (K_1, K'_1, K''_1, K''') 、ユーザ u_2 には (K_2, K'_1, K''_1, K''') 、…、ユーザ u_8 には (K_8, K'_4, K''_2, K''') を秘密鍵として配布します。

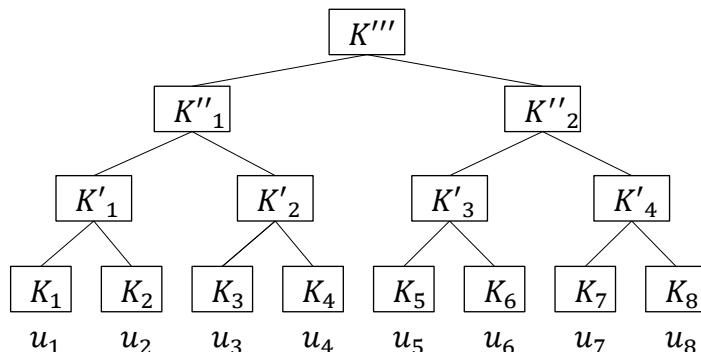
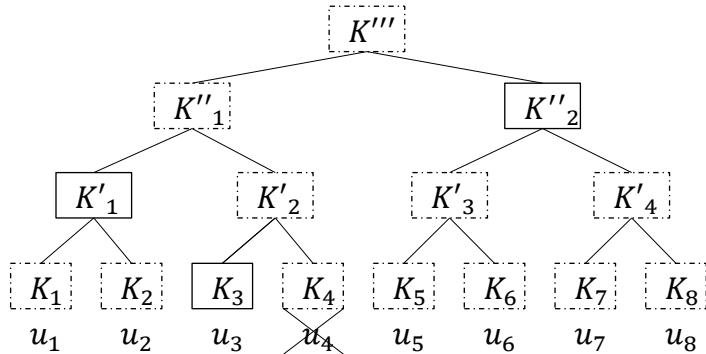


図 10.2 木構造による鍵管理

最初全員に秘密の通信をしたいときは K''' を使って平文 m を暗号化し $\text{Enc}(K''', m)$ を配信します。するとユーザ全員復号できます。さて、ユーザ u_4 を排除したくなりました。

図 10.3 u_4 を排除するときに使う秘密鍵

このときは秘密鍵 K'_1, K_3, K''_2 を使って m を暗号化し,

$$(\text{Enc}(K'_1, m), \text{Enc}(K_3, m), \text{Enc}(K''_2, m))$$

を配信します。すると $(K''', K''_1, K''_2, K_4)$ しか秘密鍵を持っていない u_4 は復号できません。他のユーザはどれかの鍵を持っているので復号できます。この方式はユーザ数 n に対して排除する人数が少ないときは $O(\log(n))$ だけの鍵を使うので効率的です。しかし、排除すべき人数が増えてくると $O(n)$ に近くなってしまいます。また放送型暗号にこのまま直接利用できるわけではありません。より効率よくするための木の作り方、または木構造以外の方法が提案されています。また鍵をどうやって保持、更新するかなどの改良も研究されています。

10.3 不正者追跡

放送型暗号では、海賊版復号器対策が必要になります。海賊版復号器（以降海賊版と略記します）とは許可されたユーザ以外が復号してコンテンツを見られる装置です。この章の冒頭で触れた不正な B-CAS カードは海賊版に相当します。

あるユーザが自身の秘密鍵を不正に利用して海賊版を作りました。海賊版の存在を知ったコンテンツ配信者がそれを入手します。押収した海賊版を調べてその鍵がどのユーザのものだったか分かれば、そのユーザ（不正者といいます）に対してアカウント停止や賠償請求などの対策ができます。

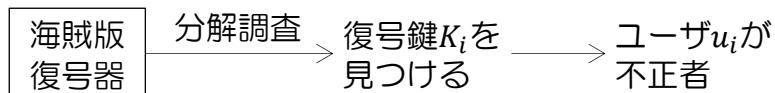


図 10.4 不正者追跡

これを不正者追跡 (traitor tracing) と呼びます。

不正者はもしかしたら複数人いて、互いの鍵から別のユーザの秘密鍵を構成しているかもしれません。複数のユーザが互いの秘密鍵の情報を用いて、その中にいない別のユーザの秘密鍵を作ることを結託攻撃（Collusion Attack）といいます。不正者追跡で間違えて他の人を不正者と特定してしまうと困ります。放送型暗号では結託攻撃に対して安全であることが望ましいです。

また不正者は慎重で、海賊版を押収した人が装置を分解して中の秘密鍵を取り出そうとすると秘密鍵を消去してしまう対策をとっているかもしれません。あるいは回路に直接秘密鍵が埋め込まれると見つけ出すのは困難です。そこで海賊版を分解しなくとも正常なフォーマットのコンテンツを入力し、それが復号できるか否かを判定することで不正者を特定できると便利です。装置の中がどうなっているか調べなくてもよいからです。そのような手法をブラックボックス型不正者追跡といいます。

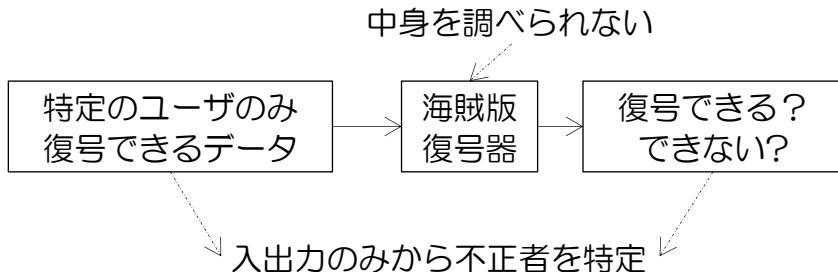


図 10.5 ブラックボックス不正者追跡

海賊版を作っている不正者が一人ならユーザ排除機能を持っている放送型暗号はブラックボックス型不正者追跡ができます。あるユーザ u だけが復号できないようにしたデータ D_u を海賊版に与えます。復号できれば別の u に対する D_u を与えます。これを繰り返して、復号できないデータ D'_u が見つかれば、 u' が不正なユーザです。この方法はユーザが n 人いるとき最大 n 回テストする必要があります。

放送型暗号が任意の複数のユーザを同時に排除できるなら、もっと効率よくブラックボックス不正者追跡ができます。まず n 人のユーザのうち半分のユーザ集合 S が復号できないデータを海賊版に与えます。復号できれば S の更に半分、できなければ残りのユーザの半分に対して同様のことを行います。この方法だと $O(\log(n))$ の回数で不正者を特定できます。

10.4 放送型暗号の始まり

1994 年に Fiat と Naor が初めて放送型暗号の考えを提案しました¹。同じ年に彼らと Chor により不正者追跡の論文も出ています。1998 年に黒澤氏と Desmedt が、11.3 節の秘密分散を用いた k 人までの結託攻撃に対しては安全で効率のよい不正者追跡の手法を提案しました [KD98]。2001

¹ http://www.wisdom.weizmann.ac.il/~naor/PAPERS/broad_abs.html

年、私と境氏、笠原氏でペアリングを使った手法を初めて提案します。ただ残念ながらこの方式は安全ではなく1年後に自分で別の攻撃を見つけてしました。しかしその後ペアリングを使った様々な方式の放送型暗号が提案されています。次節でその一つを紹介しましょう。

10.5 効率のよい放送型暗号

2005年Boneh, Gentry, Watersにより任意の不正者による結託攻撃に対して安全な方式が提案されました[BGW05]。これは秘密鍵の大きさが定数サイズで、公開鍵はユーザ数 n に比例する方式です（以降BGW方式と呼びます）。

鍵生成 : P を楕円曲線上の点、整数 a, r をランダムにとる。 $P_i := a^i P, Q := rP$ として

$$(P, P_1, \dots, P_n, P_{n+2}, \dots, P_{2n}, Q)$$

を公開鍵とする (P_{n+1} だけ抜けているのに注意)。各ユーザ u_i の秘密鍵を $K_i := rP_i = ra^i P$ とする。

暗号化 : 整数 t をランダムにとり $s := e(P_1, P_n)^t$ をセッション鍵としてコンテンツ m を暗号化する。復号したいユーザの集合 $S \subset \{1, \dots, n\}$ に対して

$$H := (tP, t(Q + \sum_{j \in S} P_{n+1-j}))$$

を $(S, H, \text{Enc}(s, m))$ を送信する。

復号 : ユーザ u_i は $(S, K_i, H = (C_0, C_1))$ を使って

$$s = e(P_i, C_1) / e(C_0, K_i + \sum_{j \in S \setminus \{i\}} P_{n+1-j+i})$$

を求めて $\text{Dec}(s, \text{Enc}(s, m)) = m$ を得る。復号時に必要な $P_{n+1-j+i}$ は $j \neq i$ となる範囲で動くので P_{n+1} は不要なことに注意してください。

かなり複雑ですね。正規のユーザが復号時に求めた s が暗号化の s と等しいことを確認しておきましょう。まず

$$\begin{aligned} X &:= e(P_i, C_1) = e(a^i P, t(Q + \sum_{j \in S} P_{n+1-j})) = e(a^i P, tQ) \cdot e(a^i P, t \sum_{j \in S} a^{n+1-j} P) \\ &= e(a^i P, trP) \cdot \prod_{j \in S} e(a^i P, a^{n+1-j} P)^t = e(a^i P, P)^{tr} \cdot \prod_{j \in S} e(P, a^{n+1-j+i} P)^t. \end{aligned}$$

次に

$$\begin{aligned} Y &:= e(C_0, K_i + \sum_{j \in S \setminus \{i\}} P_{n+1-j+i}) = e(tP, ra^i P) \cdot e(tP, \sum_{j \in S \setminus \{i\}} a^{n+1-j+i} P) \\ &= e(a^i P, P)^{tr} \cdot \prod_{j \in S \setminus \{i\}} e(P, a^{n+1-j+i} P)^t. \end{aligned}$$

この2個の式を比べると \prod をとっている部分は $e(P_i, C_1)$ の方が $j = i$ のところだけ多いです。よって、割り算をするとそれ以外の部分はキャンセルして

$$X/Y = e(P, a^{n+1-i+i}P)^t = e(P, a^{n+1}P)^t = e(aP, a^n P)^t = e(P_1, P_n)^t = s$$

と s を復元できます。

BGW 方式は暗号化と復号時にユーザ数 S に比例した和が必要なので計算が大変なように見えます。しかし予め和 X を計算しておけば S が変化しないときは使い回すことができます。また、あるユーザ u_i の排除が必要になったときは $X' := X - P_i$ と更新差分だけ計算すればよいので問題ありません。ユーザの追加に関しては一人追加する毎に n を増やすと鍵生成からやり直しになり鍵の配布や更新が大変です。ユーザ数よりもある程度 n を大きめにしておき、その人への秘密鍵配布と全員の和の差分更新だけするとよいでしょう。

10.6 BGW 方式の安全性

結託攻撃をしようとする不正なユーザの集合を S とします。彼らの秘密鍵 $\{P_i = a^i P \mid i \in S\}$ から P_{n+1} が求まってしまうと困ります。BGW 方式は l -BDHE 仮定 (BDH Exponent) のもとで、そんな結託攻撃ができないことが示されます。

(計算) l -BDHE 問題：「楕円曲線上的点 P, Q とある整数 a に対して

$$(P, aP, a^2P, \dots, a^lP, a^{l+2}P, \dots, a^{2l}P, Q)$$

が与えられたときに $e(P, Q)^{a^{l+1}}$ を求めよ。」

判定 l -BDHE 問題：「楕円曲線上的点 P, Q とある整数 a とペアリングの行き先の適当な点 h に対して

$$(P, aP, a^2P, \dots, a^lP, a^{l+2}P, \dots, a^{2l}P, Q, h)$$

が与えられたときに $h = e(P, Q)^{a^{l+1}}$ かどうかを判定せよ。」

安全性の根拠となる問題もかなり複雑になっていますね。一見、問題が難しいのかどうかも分かりません。このあたりの問題については15章で改めて取り上げましょう。

10.7 より一般の不正者追跡

10.3節では、海賊版復号器の中身を調べることなく、入力データに対する反応から不正者を特定するブラックボックス不正者追跡という考え方を紹介しました。放送型暗号がユーザ排除機能を持っているとそれを実現できると書きましたが、それには不正者が一人で海賊版を作っているという仮定が入っていました。

実際の海賊版は複数の鍵が入っているかもしれません。復号時にそれらの復号鍵をランダムに選んで復号しているかもしれません。そうするとブラックボックス型不正者追跡は極めて困難になります。

2006 年 Boneh, Waters たちは初めて任意の不正者に対する耐性を持った方式を提案します [BSW06]. これは n 人のユーザに対して暗号文が $O(\sqrt{n})$, 秘密鍵が $O(1)$ の大きさのものです. 方式はかなり複雑で, より効率のよい方式も研究されています.

10.8 この章のまとめ

放送型暗号を紹介しました. 放送型暗号は沢山の人に一度に効率よく暗号化されたコンテンツを配信することを目指して作られた暗号です. 放送型暗号はユーザの加入や退会に柔軟に対応でき, 海賊版復号器から不正者を特定できるのが望ましいです. 海賊版復号器中身を見なくても特定できるブラックボックス型不正者追跡の改良も進められています.

第 11 章

属性ベース暗号

共通鍵暗号は暗号鍵と復号鍵が同じでした。公開鍵暗号は暗号鍵と復号鍵を別のものにすることで利用範囲が広がりました。しかし、暗号鍵と公開鍵が 1 対 1 の関係であることには変わりません。属性ベース暗号は暗号鍵と公開鍵が 1 対 n や n 対 1 の関係になる暗号です。公開鍵暗号の一般化になっています。

11.1 属性ベース暗号とは

ある企業内で社内の特定の人だけが見られるようにした文章を配布したいとしましょう。ここで特定の条件は、たとえば「人事部であり、かつ部長である」とか「開発部である」というものだったとします。「人事部である」といった条件を属性と呼びます。

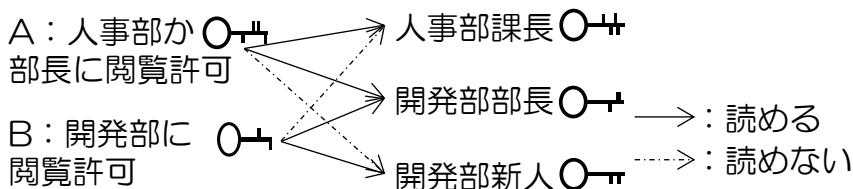


図 11.1 復号する人を属性で制御する

もちろんその特定の人を全て列挙し、それぞれの人に暗号化して文章を送れば従来の暗号方式でも実現できます。ただ対象人数が多いと煩雑になったり、対象者が属性でしか分からなかったりするかもしれません。個別に暗号化するのではなく属性に対するアクセス権をシステムのアクセス権を用いて実現することもあります。この方式なら一般的に行われているでしょう。しかしクラウド提供者のシステム管理者なら、それらのアクセス権を越えて全てを見られます。

属性ベース暗号とは特定の属性を持った対象者だけが復号できる暗号方式です。暗号方式にアクセス権の制御に相当するものが組み込まれているため従来の方式に比べて安全性が高まります。2005 年に Sahai と Waters により初めて提案されました。属性ベース暗号は秘密分散と ID ベース

暗号と組み合わせて作られています。秘密分散は 1979 年に提案された昔からあるもので、まずはこれから紹介しましょう。

11.2 密密分散

秘密分散（Secret Sharing）とは一つの情報を複数に分割して管理する技術です。たとえば会社の重要なパスワードを社長だけが管理していたとしましょう。社長がうっかりそのパスワードを忘れるとき困ります。だからと言って秘書や副社長に渡しておいて勝手に使われてしまうのも困ります。

秘密分散では、複数人の取締役にパスワードをそのままではなく分割して配布します。分割されたパスワードは取締役のうち何人かが同意すると元のパスワードを復元できるようになります。一人で使うことはできません。こうすると情報の紛失や、情報漏洩に対する防御を強められます。

まず一番簡単な秘密分散である秘密を二人で分散管理し、二人が集まると復元できる方法を紹介しましょう。といっても簡単です。以下 \mathbb{F}_p 上の平文 s について考えます。乱数 r を生成し、 r を A, $s + r$ を B に渡します。A や B それぞれ単独では s の情報を得られませんが、二人集まると $(s + r) - r = s$ により復元できます。

これってどこかで見た覚えがありますね。そうです 1.3 節の Vernam 暗号です。Vernam 暗号は一つの平文を暗号文と秘密鍵に分割した秘密分散とみなせます。

この場合、A や B の単独の情報 r だけ、あるいは $s + r$ だけでは s の情報を一切もらっていない点に注意しましょう。Vernam 暗号は情報理論的安全な暗号なのでした。通常、秘密分散ではその分割された平文から元の情報の一部を復元できてしまいません。平文 s を単に上半分 s_1 と下半分 s_2 の二つに切った場合、その二つを $s = s_1||s_2$ と連結すれば元の s を復元できます。しかし、 s_1 や s_2 は元の平文の半分の情報をそのまま保持しているので駄目なのです。

3 個に分割し、3 個集めれば復元できるようにするには乱数 r_1, r_2 を生成し、A に r_1 を、B に $r_1 + r_2$ を、C に $r_2 + s$ を渡せばよいでしょう。A と B、あるいは B と C や C と A だけでは s の情報は何も得られません。しかし 3 人集まると s を復元できるのはすぐ分かると思います。次の節では Shamir が提案した多項式を用いる秘密分散法を紹介しましょう。

11.3 (k, n) 閾値法

秘密を何個にでも分散できるように拡張します。またそのうち全員でなくても一部の人が集まれば復元できるようにしましょう。 (k, n) 閾値法とは情報を n 個に分割し、そのうちの k 個を集めると元の情報を復元できる方式です。 $k - 1$ 個しか集められなかったときはもとの情報は何も復元できません。先ほどの一番簡単な秘密分散は情報を 2 個に分割し、2 個集めると復元できるので $(2, 2)$ 閾値法といえます。二つ目の例は $(3, 3)$ 閾値法ですね。

秘密の数 s に対して (k, n) 閾値法をするには次のようにします。まず $k - 1$ 個の乱数 $r_1, \dots, r_{k-1} \in \mathbb{F}_p$ を作ります。 $r_0 := s$ とし、 $k - 1$ 次多項式

$$f(x) := r_0 + r_1 x + \dots + r_{k-1} x^{k-1} = \sum_{u=0}^{k-1} r_i x^u$$

を定義します。この $f(x)$ は秘密の情報です。適当な n 個の異なる番号 $x_1, \dots, x_n \in \mathbb{F}_p \setminus \{0\}$ を用意し ($x_j := j$ でもかまいません)， $y_j := f(x_j)$ を計算して

$$(x_j, y_j)$$

を j 番目の人に渡します。渡し終わった後は、元の s や $f(x)$ を消去します。

$f(x)$ は $k - 1$ 次多項式ですから、 n 個のうち k 個の異なる (x_j, y_j) が決まるとその点を通る $f(x)$ が一意に決まります。 $f(x)$ が確定するので特に $f(0) = r_0 = s$ を復元できます。

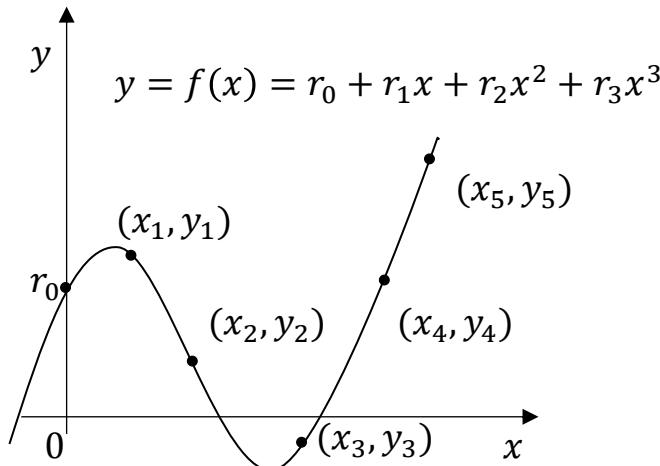


図 11.2 3 次多項式による $(4, n)$ 密分散

11.4 (k, n) 閾値法の復号

k 個の異なる (x_j, y_j) を集めたときの復号方法について詳しく調べます。まず $k = 2$ のときを考えます。 $f(x)$ は 1 次多項式、つまり直線です。

$$f(x) = r_0 + r_1 x.$$

$r_0 (= s)$, r_1 が秘密です。2 点 (x_1, y_1) , (x_2, y_2) の組が与えられると

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} r_1 x_1 + r_0 \\ r_1 x_2 + r_0 \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$

という関係式が成り立ちます。 x_1, x_2, y_1 が既知なのでこの連立方程式を解くと r_0, r_1 が求まります。

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix}^{-1} = \frac{1}{x_2 - x_1} \begin{pmatrix} x_2 & -x_1 \\ -1 & 1 \end{pmatrix}$$

なので

$$\begin{pmatrix} r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix}^{-1} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{1}{x_2 - x_1} \begin{pmatrix} x_2 & -x_1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

となり、とくに

$$s = r_0 = \frac{-x_2}{x_1 - x_2} y_1 + \frac{-x_1}{x_2 - x_1} y_2 \quad (11.1)$$

です。次に $k = 3$ のときを考えてみましょう。 $f(x) = r_0 + r_1 x + r_2 x^2$ は 2 次多項式です。3 点 $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ の組が与えられると

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & x_1 & {x_1}^2 \\ 1 & x_2 & {x_2}^2 \\ 1 & x_3 & {x_3}^2 \end{pmatrix} \begin{pmatrix} r_0 \\ r_1 \\ r_2 \end{pmatrix}$$

という関係式が成り立ちます。3 次行列の逆行列を求めて右から掛けて

$$s = r_0 = \frac{x_2 x_3}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{x_1 x_3}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{x_1 x_2}{(x_3 - x_1)(x_3 - x_2)} y_3$$

と求められます。

一般に k 個の異なる (x_j, y_j) を集めたときは k 次行列を

$$X := \begin{pmatrix} 1 & x_1 & \cdots & {x_1}^{k-1} \\ 1 & x_2 & \cdots & {x_2}^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_k & \cdots & {x_k}^{k-1} \end{pmatrix}$$

とおいて

$$\begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} = X \begin{pmatrix} r_0 \\ \vdots \\ r_{k-1} \end{pmatrix}$$

という連立方程式ができます。 X の形の行列を Vandermonde (ファンデルモンド) 行列といいます。 $S := \{x_1, \dots, x_k\}$ としたとき、 $u \in S$ に対しする Lagrange (ラグランジュ) 係数を

$$\Delta_{u,S}(x) := \prod_{j \in S \setminus \{u\}} \frac{x - j}{u - j}$$

と定義します。 $\Delta_{u,S}(x)$ は $k - 1$ 次多項式です。すると任意の $k - 1$ 次多項式 $f(x)$ に対して

$$f(x) = \sum_{u \in S} f(u) \Delta_{u,S}(x) \quad (11.2)$$

が成り立ちます。式 (11.2) を Lagrange の補間多項式といいます。何やら恐ろしげな式ですが、具体例を見るとそれほどでもありません。たとえば $k = 2$ で $S = \{x_1, x_2\}$ のときは

$$\begin{aligned}\Delta_{x_1, S}(x) &= \frac{x - x_2}{x_1 - x_2}, \\ \Delta_{x_2, S}(x) &= \frac{x - x_1}{x_2 - x_1}\end{aligned}$$

で $f(x_1) = y_1, f(x_2) = y_2$ なので

$$f(x) = y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x - x_1}{x_2 - x_1}$$

となります。これに $x = 0$ を代入すると先ほどの式 (11.1) になります。

このように k 個の異なる (x_j, y_j) を集めたときはきっと s を復元できますが、 $k - 1$ 個しか集められないときは方程式の数が足りないので s はどんな値であってもいい、つまり s の情報を得られない点に注意してください。

11.5 Fuzzy ID ベース暗号

秘密分散の紹介が終わったので属性ベース暗号について説明しましょう。最初の属性ベース暗号は Fuzzy ID ベース暗号という名前がつけられていました [SW05]。この暗号は、

1. 最初のパラメータ設定をするセットアップアルゴリズム。指定された属性集合のうち何個属性を持っていると復号できるかという下限を k とする。
2. 属性集合 ω_u を持つユーザ u に対して秘密鍵 K_{ω_u} を生成する鍵生成アルゴリズム。
3. 平文 m を属性集合 ω で暗号化するアルゴリズム $\text{Enc}(m, \omega)$ 。
4. ユーザ u の属性集合 ω_u と暗号化に使われた属性集合 ω に対して $|\omega_u \cap \omega| \geq k$ ならば復号するアルゴリズム。

$$\text{Dec}(\text{Enc}(m, \omega), \omega_u) = m.$$

$|\omega_u \cap \omega| < k$ ならば復号できない。

という 4 個のアルゴリズムの組からなります。Fuzzy ID ベース暗号は属性集合 ω のうち k 個合致する属性をもっていれば復号できる暗号です。 $|\omega|$ 個全てもっているのではなく、それよりも少ない k 個のどれかでよいというのが要点です。条件を概ね満足していれば復号できるという意味で Fuzzy と名付けられたのでしょう。

そして Fuzzy ID ベース暗号は結託攻撃に対して安全であることが要求されます。ここで結託攻撃とは復号要件を満たしていない人たちが複数集まって本来復号できてはいけないものを復号することです。この場合は、4. の符号条件を満たさない人、つまり $|\omega_u \cap \omega| < k$ である属性 ω_u を持つユーザが何人集まって互いの秘密鍵 K_{ω_u} を持ち寄ったとしても $\text{Enc}(m, \omega)$ を復号できてはなりません。以下順に詳しく見ていきます。

11.5.1 セットアップ

楕円曲線 E と巡回群 G とペアリング $e : E \times E \rightarrow G$ を用意します。 $P, Q \in E$ と $y \in \mathbb{F}_p$ をとり、 $R := yP$ とします。属性を「人事部」や「部長」などの文字列とし、属性の集合から E へのハッシュ関数を H とします。 y が秘密鍵で P, Q, R が公開鍵です。

11.5.2 鍵生成

属性集合 ω_u を持つユーザ u に対して秘密鍵 K_{ω_u} を作ります。まず $k - 1$ 次の多項式で定数項が y である $q_u(x)$ をランダムに作ります。 $q_u(0) = y$ です。 ω_u の i 番目の属性 a_i に対してランダムに r_i をとり、 $P_i := r_i P$, $D_i := q_u(i)Q + r_i H(a_i)$ とします。

$$K_{\omega_u} := \{ P_i, D_i \}_{\{ i=1, \dots, |\omega_u| \}}$$

だけをユーザ u に渡します。

11.5.3 暗号化

属性集合 ω を選び、平文 $m \in G$ を暗号化します。 ω の中のどれか k 個の属性を持っている人だけが復号できます。 s をランダムに選び

$$\text{Enc}(m, \omega) := (\omega, t := m \cdot e(R, Q)^s, T := sP, E_i := sH(a_i)_{\{ i=1, \dots, |\omega| \}})$$

と暗号化します。

11.5.4 復号

属性集合 ω_u を持つユーザ u は $|\omega \cap \omega_u| \geq k$ を確認します。成り立たなければ復号できません。成り立つ場合は共通部分の中から k 個適当に属性を選びその集合を S とします。 $S \subseteq \omega \cap \omega_u$, $|S| = k$ です。やや複雑ですが頑張って計算しましょう。

$$\begin{aligned} z &= \prod_{a_i \in S} \left(\frac{e(T, D_i)}{e(P_i, E_i)} \right)^{\Delta_{i,S}(0)} = \prod_{a_i \in S} \left(\frac{e(sP, q_u(i)Q + r_i H(a_i))}{e(r_i P, sH(a_i))} \right)^{\Delta_{i,S}(0)} \\ &= \prod_{a_i \in S} \left(\frac{e(P, Q)^{sq_u(i)} e(P, H(a_i))^{sr_i}}{e(P, H(a_i))^{sr_i}} \right)^{\Delta_{i,S}(0)} = \prod_{a_i \in S} \left(e(P, Q)^{sq_u(i)} \right)^{\Delta_{i,S}(0)} \\ &= e(P, Q)^{\left(\sum_{a_i \in S} q_u(i) \Delta_{i,S}(0) \right) s} \quad \text{Lagrange の補間多項式 (11.2) を使って} \\ &= e(P, Q)^{q_u(0)s} = e(P, Q)^{ys}. \end{aligned}$$

$t = m \cdot e(yP, Q)^s = m \cdot e(P, Q)^{ys} = mz$ なので

$$t/z = m$$

と復号できます。

11.6 安全性

Fuzzy ID ベース暗号は、攻撃者が攻撃対象のユーザを決めたときに DBDH (Decisional Bilinear Diffie-Hellman) 仮定の元で結託攻撃に対して安全であることが示されます。DBDH 仮定とは 7.7 節で紹介した CBDH 仮定に対する判定問題版で、次の DBDH 問題が難しいという仮定です。

「楕円曲線上の点 P と整数 a, b, c , 有限体の値 d に対して (P, aP, bP, cP, d) が与えられたときに $d = e^{abc}$ か否かを判定せよ。」

この仮定は有限体上の CDH 仮定、楕円曲線上の CDH 仮定を含みます。直感的にはユーザ毎に異なる多項式を持っていて、その多項式の値は Q に掛かっているので攻撃者は値を直接求められそうにありません。

11.7 属性ベース暗号

Fuzzy ID ベース暗号は属性が k 個以上という固定の条件でした。たとえば属性が 5 個 A, B, C, D, E で $k = 3$ の場合は $(A, B, C), (A, B, D), (C, D, E)$ などの条件を持っている人のみという制約です。これを属性 A と B なら 2 個だけもっていれば OK、そうでなければ 3 個というような条件にも対応できるようにしたいです。秘密分散のところを見直してみましょう。 $(3, 5)$ 閾値法では、

$$X = \begin{pmatrix} 1 & x_1 & {x_1}^2 \\ 1 & x_2 & {x_2}^2 \\ 1 & x_3 & {x_3}^2 \\ 1 & x_4 & {x_4}^2 \\ 1 & x_5 & {x_5}^2 \end{pmatrix}$$

という行列のうち、どの 3 行を取り出してもその逆行列は Vandermonde 行列の形をしていて逆行列を求められたのでした。この性質が 3 個集まると復号できるという性質になっています。

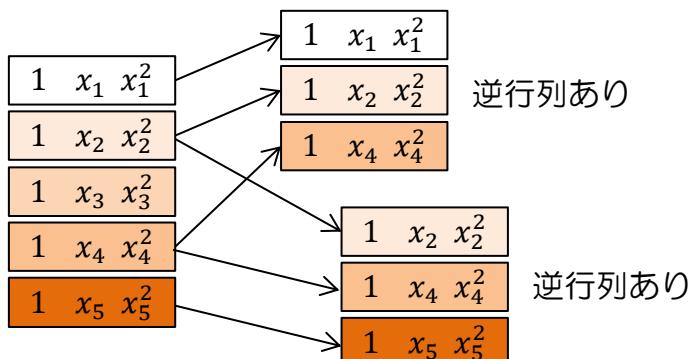


図 11.3 どの 3 個の行を取り出しても逆行列がある

ではこの行列の形をもっと別のものに換えたらどうなるでしょう。たとえば s, r_0, r_1, r_2 を秘密とし

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s \\ r_0 \\ r_1 \\ r_2 \end{pmatrix}$$

の関係式でえられた y_1, \dots, y_5 を配布したとします。展開すると

$$(y_1, y_2, y_3, y_4, y_5) = (r_0, s + r_0, r_1, r_1 + r_2, s + r_2)$$

です。これは $(r_0, s + r_0)$ と $(r_1, r_1 + r_2, s + r_2)$ に分けるとそれぞれ $(2, 2)$ 閾値法、 $(3, 3)$ 閾値法ですね。つまり y_1 と y_2 の条件があれば復元、 y_3, y_4, y_5 があれば復元できるという性質を実現できそうです。一般的の条件は and と or の組み合わせで成り立っています。 s を $(A \wedge B) \vee ((C \vee D) \wedge E)$ という条件が成り立つように秘密分散してみましょう。ここで \wedge は and 条件、 \vee は or 条件を表すとします。or に対してはそれぞれに s を割り当てます。すると最初の項は A と B で s を復元できればよいので $(2, 2)$ 閾値法を使います。実際には乱数 r_1 を用意して A には r_1 を、 B には $r_1 + s$ を渡せばよいですね。

後ろの条件については $C \vee D$ と E に対して $(2, 2)$ 閾値法を使います。乱数 r_2 を用意して $C \vee D$ に r_2 を、 E に $r_2 + s$ を渡します。最後 $C \vee D$ についてはどちらががあれば OK なのでそれぞれに r_2 を渡します。最終的に $(A, B, C, D, E) = (r_1, r_1 + s, r_2, r_2, r_2 + s)$ を渡せばよいです。実際にこのように配布すると A と B を持っているか、「 C または D 」かつ E を持っているときに s を復元できます。

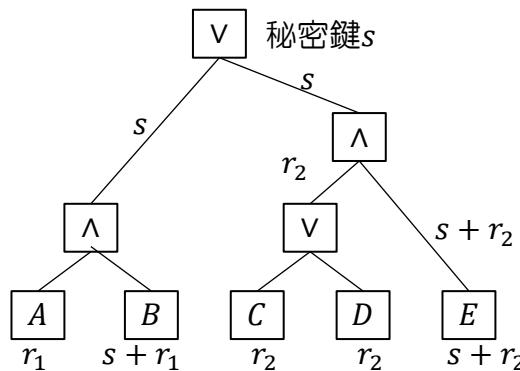


図 11.4 アクセス構造 $(A \wedge B) \vee ((C \vee D) \wedge E)$ を表す木

属性の and と or の任意の組み合わせによるアクセス木 (access tree) で表された復号条件をポリシー (Policy) といいます。ポリシーはどのような鍵を持っていれば復号できるかを決める情報です。ポリシーを元に秘密分散を行うことを線形秘密分散法 (LSSS : Linear Secret Sharing

Scheme) といいます。アクセス木を用いた秘密分散と Fuzzy ID ベース暗号を組み合わせると属性ベース暗号 (ABE : Attribute Based Encryption) を構成できます [GPSW06]。

暗号文にポリシー、各自の秘密鍵に属性を埋め込む方式を CP-ABE (CipherText Policy ABE)，逆に暗号文に属性、各自の秘密鍵にポリシーを埋め込む方式を KP-ABE (Key Policy ABE) といいます。事前に「人事部である」，「部長である」といった属性を復号する人に合わせた秘密鍵を作成し、暗号化するときに「部長 or 人事部」といったポリシーを決めるのが CP-ABE です。

KP-ABE の例としては動画などのコンテンツ配信が考えられます。事前に「アニメ」，「SF」，「映画」，「ホラー」などの属性を決めてコンテンツを暗号化します。ユーザは「SF and アニメ」を見たいというポリシーで契約し、対応する秘密鍵をもらいます。そうするとプリペイド形式でポリシーにマッチしたコンテンツだけを自由に見られます。

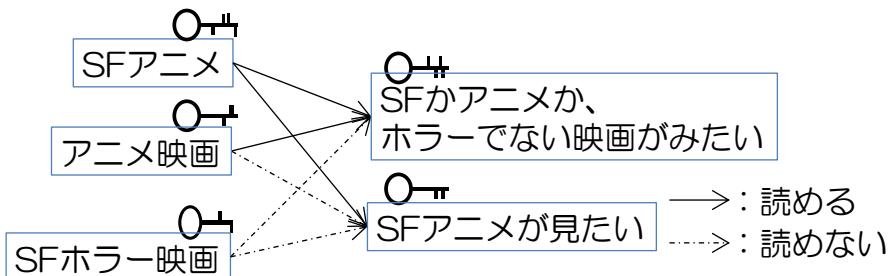


図 11.5 条件指定によるコンテンツ受信

11.8 not をサポート

前節の属性ベース暗号ではポリシーの条件に and や or の組み合わせを使えました。ただこれだけでは不十分なことがあります。たとえば、趣味の属性を考えます。

$$\mathcal{A} = \{ \text{読書}, \text{ピアノ}, \text{テニス}, \text{ハイキング}, \text{バイク} \}$$

このとき、ポリシーで「ピアノではない」としたくなつたとしましょう。単純に「ピアノ」の属性がない人が復号できるようにすると、「ピアノ」属性の人が対応する鍵を持っていないふりをして復号できてしまうかもしれません。それでは困るのでポリシーは「読書」 or 「テニス」 or 「ハイキング」 or 「バイク」とします。しかし属性の数が増えるとこれは面倒です。そのため 2007 年にポリシーの組み合わせに and と or だけでなく not に対応した属性ベース暗号が提案されました [OSW07]。

その論文で提案された not を実現するアイデアを簡単に紹介します。それは 11.3 節の秘密分散を用いた多項式による排除の方法です。放送型暗号で特定のユーザを排除したいときに使われる手法の一つです。 d 次多項式 $f(x)$ を適当に決めます。 $f(x)$ は $d+1$ 個の点が定まると確定し、Lagrange 補間により $f(0)$ を計算できます。つまり d 個の相異なる点を x_1, \dots, x_d 、残りの一つの

点を t として,

$$f(x_1), \dots, f(x_d), f(t)$$

が渡されたとき, $f(0)$ を計算できる必要十分条件は $t \notin \{x_1, \dots, x_d\}$ です. $\{f(x_i)\}$ を暗号文の情報に埋め込み, 各ユーザに $f(t)$ 相当のものを渡します. すると, t が x_i のどれかの場合, その人は Lagrange 補間ができず, $f(0)$ を計算できません. $f(0)$ を復号に必要な値にしておくと, 属性 $t = x_i$ に相当する人は復号できないことになります. この方法を使って and, or, not の組み合わせができる属性ベース暗号を構成します. その後, もっと効率のよい方法が提案されています.

11.9 この章のまとめ

ID ベース暗号の一種の拡張である属性ベース暗号を紹介しました. 属性ベース暗号は鍵や復号者の属性を指定して復号できるかできないかを制御できる暗号です. 属性ベース暗号は ID ベース暗号と秘密分散を組み合わせて作られます. 属性の条件は and や or を組み合わせたアクセス木と呼ばれる木構造を用います.

アクセス制御が暗号文に組み込まれているため, 従来のユーザ認証による制御に比べてより安全になります.

第 12 章

関数型暗号

属性ベース暗号はポリシーと属性がマッチしたときに復号できる仕組みでした。ポリシーと属性、およびマッチするということをより一般的な枠組みで考えようという動きがでます。また応用によっては属性自体を隠したいこともあります。述語暗号、関数型暗号と呼ばれる暗号で、これはIDベース暗号や属性ベース暗号の更なる拡張になっています。

12.1 述語暗号

2007 年 Katz, Sahai, Waters たちは属性ベース暗号を拡張した述語暗号(Predicate Encryption)を提案します [KSW08]。後述の関数型暗号と呼ばれることもあります。秘密鍵をあるパラメータ v を用いて作成し、暗号文は別のあるパラメータ x を用いて作成します。そして v と x がある関係式 $R(\cdot, \cdot)$ を満たすときのみ復号できるという考え方です。 $R(\cdot, \cdot)$ はパラメータ v, x を与えたときに true か false が決まり、true のときは v と x は関係があるとみなします。

もう少し正確に言うと次の形です。まず全体をつかさどるシステムはマスター秘密鍵 K_{prv} と公開鍵 K_{pub} を用意します。それからシステムはパラメータ v にしたがって決まる秘密鍵 k_v を作ります。

$$k_v := \text{KeyGen}(K_{\text{prv}}, K_{\text{pub}}, v).$$

ユーザは平文 m を公開鍵 K_{pub} と別のあるパラメータ x を使って暗号化します。

$$c_x := \text{Enc}(K_{\text{pub}}, x, m).$$

復号できるのは $R(v, x)$ が true になる秘密鍵 k_v を持っている人だけであり、そのとき

$$m = \text{Dec}(K_{\text{pub}}, k_v, c_x)$$

と復号できます。

従来の属性ベース暗号では平文 m を暗号化するときに指定した属性パラメータ x は誰でも知ることができました。述語暗号ではどんなパラメータ x を使ったか分からないようにすること

が求められます。たとえばある暗号文に「社内秘」、「極秘」といった属性がつけられていることを知られないようにできます。属性を隠すのでそれを強調したいときは attribute-hiding あるいは private-index といいます。対応して従来の属性ベース暗号は payload-hiding あるいは public-index といわれます。

それからただ単に「関係がある」というのはちょっと一般的すぎます。彼らの論文では 2 個のパラメータ v と x をベクトルとし、その 2 個のベクトルが直交するときに v と x は関係があるとして暗号を構成しました。直交する条件は内積を用いて確認できるので内積暗号 (IPE : Inner Product Encryption) ともいわれます。内積というのは 2 個の n 次元ベクトル $v := (v_1, \dots, v_n)$, $x := (x_1, \dots, x_n)$ があったときに

$$v \cdot x := \sum_{i=1}^n v_i x_i$$

で定義される値のことです。2 個のベクトル v, x が直交するときのみ内積は 0 になり、そのとき関係があるとするのです。

$$R(v, x) := \begin{cases} \text{true if } v \cdot x = 0, \\ \text{false if } v \cdot x \neq 0. \end{cases}$$

$n = 2$ のとき図 12.1 を見ながら内積の意味を確認しておきましょう。

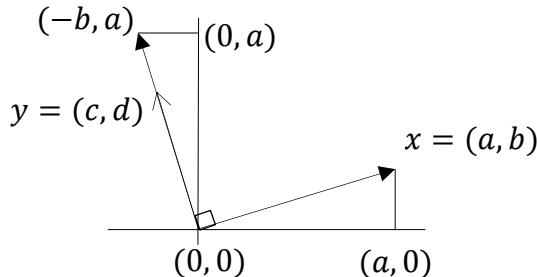


図 12.1 直交する 2 次元ベクトル

$v := (a, b)$, $x := (c, d)$ とします。点 (a, b) を、原点を中心にして 90 度半時計周りに回すと $(-b, a)$ になります。 (a, b) と (c, d) が直角なら $(-b, a)$ と (c, d) は並行です。よって

$$\frac{-b}{a} = \frac{c}{d}.$$

すなわち

$$ac + bd = 0.$$

よって

$$(a, b) \cdot (c, d) := ac + bd = 0$$

です。より高い次元でも同様になります。

たとえば $v := (1, a)$, $x := (b, -1)$ という 2 個の 2 次元パラメータを持つ内積暗号を考えます。この内積暗号が復号できるのは $v \cdot x = b - a = 0$, つまり $a = b$ のときのみです。パラメータ a に対してユーザに秘密鍵を渡し、ユーザはパラメータ b を使って暗号文を作ります。復号できるのが $a = b$ のときのみということなのですから a を ID と思うと、これは ID ベース暗号を意味します。つまり内積暗号は ID ベース暗号の拡張になっています。

また 3 個のパラメータ t, a, b を用いて

$$\begin{aligned} v &:= (1, t, t^2), \\ x &:= (ab, -(a+b), 1) \end{aligned}$$

という 2 個の 3 次元ベクトルを考えます。この内積は

$$v \cdot x = ab - (a+b)t + t^2 = (t-a)(t-b)$$

となります。内積が 0 になるのは $t = a$ または $t = b$ のときです。つまりこれは属性 t が a または b のとき復号できるというポリシーを表します。属性ベース暗号を考えることもできますね。

もう一つ、今「または」という条件を実現できることが確認できました。次のようにすると「かつ」もできます。4 個のパラメータ s, t, a, b と乱数 r, r' を用いて

$$\begin{aligned} v &:= (1, s, t), \\ x &:= (-ar - br', r, r') \end{aligned}$$

という 2 個の 3 次元ベクトルを考えます。この内積は

$$v \cdot x = -ar - br' + sr + tr' = r(s-a) + r'(t-b)$$

となります。 r, r' は乱数なのでこの内積が 0 になるのは無視できる確率を除いて $s = a$ かつ $t = b$ のときとなります。より一般に d 次多項式 $f(t)$ を 2 個の $d+1$ 次元ベクトルを用いて

$$f(t) := \sum_{i=0}^d f_i t^i = (f_0, f_1, \dots, f_d) \cdot (1, t, t^2, \dots, t^d)$$

と表現することでさまざまな条件を内積が 0 という条件に埋め込みます。

Katz たちは内積暗号を合成数位数ペアリングと呼ばれるものを用いて構成しました。合成数位数のペアリングは今まで紹介していた素数位数のペアリングに比べて 10~100 倍遅いため [Gui13]、素数位数のペアリングを用いた手法が望まれます。2010 年、岡本氏、高島氏は素数位数のペアリングを用いて効率がよく、安全性の高い述語暗号を構成します [OT10]。更に条件の否定を、ある属性集合内での not という日常的な感覚にあった形で扱えます。この方式は複数の楕円曲線上の点をまとめて扱う双線形写像ベクトル空間 (DPVS : Dual Pairing Vector Space) を用いています。DVPS については次節で紹介しましょう。

理論的な進展としては 2014 年に Garg, Gentry, Halevi, Zhandry たちが一般の回路を条件に利用できる適応的安全な属性ベース暗号を提案します [GGHZ14]。これは 7.3 節で少し触れた多重線

形写像を前提として構成しています。今のところ多重線形写像自体が実用的と呼べるものではありませんが興味深いところです。

12.2 ベクトルの基底変換と内積

DPVS の紹介の前に線形代数の基本をおさらいします。 x, y を有限体 \mathbb{F}_p の n 個の点からなる横ベクトルとします。

$$\begin{aligned} x &:= (x_1, \dots, x_n) \in \mathbb{F}_p^n, \\ y &:= (y_1, \dots, y_n) \in \mathbb{F}_p^n. \end{aligned}$$

$B := (b_{ij})$ を行列式が 0 でない n 次正方行列とし、 $b_i := (b_{i1}, \dots, b_{in})$ とすると $\mathbb{B} := \{b_i\}$ は V の基底となり $x_{\mathbb{B}} := xB^{-1} := (x'_1, \dots, x'_n)$ は $\{b_i\}$ に関する x の表現ベクトルです。すなわち

$$x = xB^{-1}B = x_{\mathbb{B}}B = \sum_{i=1}^n x'_i b_i.$$

同様に $C := (B^T)^{-1} = (c_{ij})$ (B^T で B の転置を表す) として、

$$b_i^* := (c_{i1}, \dots, c_{in})$$

とすると $\mathbb{B}^* := \{b_i^*\}$ も V の基底となり $y_{\mathbb{B}^*} := yB^T$ は $\{b_i^*\}$ に関する y の表現ベクトルです。 x と y の内積を考えると

$$x \cdot y = xy^T = (xB^{-1})(By^T) = (xB^{-1})(yB^T)^T = x_{\mathbb{B}} \cdot y_{\mathbb{B}^*}.$$

これは x と y の内積は x の \mathbb{B} に関する表現ベクトル $x_{\mathbb{B}}$ と y の \mathbb{B}^* に関する表現ベクトル $y_{\mathbb{B}^*}$ の内積に等しいことを意味しています。

具体例で考えると難しいことはしていません。

$$x := (1, 4), \quad y := (2, 3), \quad B := \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}, \quad B^{-1} = \begin{pmatrix} 5 & -3 \\ -3 & 2 \end{pmatrix}$$

とします。 $b_1 := (2, 3), b_2 := (3, 5)$ となり x を b_1, b_2 の線形和で表すと

$$x = (xB^{-1})B = (-7, 5) \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = -7b_1 + 5b_2.$$

同様に $b_1^* := (5, -3), b_2^* := (-3, 2)$ となり y を b_1^*, b_2^* の線形和で表すと

$$y = (yB^T)((B^T)^{-1}) = (13, 21) \begin{pmatrix} b_1^* \\ b_2^* \end{pmatrix} = 13b_1^* + 21b_2^*.$$

内積は

$$x \cdot y = (1, 4) \cdot (2, 3) = 1 \cdot 2 + 4 \cdot 3 = 14 = -7 \cdot 13 + 5 \cdot 21 = (-7, 5) \cdot (13, 21)$$

が成り立つことを確認できます。

12.3 DPVS

さて G を素数 p 個の楕円曲線上の点からなる巡回群, その生成元を P , V を G の n 個の点からなる集合とします.

$$V := \underbrace{G \times \cdots \times G}_{n \text{ 個}}.$$

V の点の加算や \mathbb{F}_q 倍を各要素に対する操作として定義することで V は \mathbb{F}_q -ベクトル空間になります. V の n 個の点 a_i を

$$a_i := (0, \dots, \overset{i \text{ 番目}}{\underset{\text{P}}{\tilde{P}}}, \dots, 0)$$

と定義します. そうすると V の任意の点 x は \mathbb{F}_q の元 x_i を用いて

$$x = (x_1 P, \dots, x_n P) = (x_1 P, 0, \dots, 0) + (0, x_2 P, 0, \dots, 0) + \cdots = \sum_{i=1}^n x_i a_i$$

と一意に表せます. よって $\mathbb{A} := \{a_1, \dots, a_n\}$ は V の基底です. この係数 x_i を並べたものを基底 \mathbb{A} に関する x の表現ベクトル

$$x_{\mathbb{A}} := (x_1, \dots, x_n) \in \mathbb{F}_q^n$$

と書くことにします. 特に a_i の表現ベクトルは

$$a_{i\mathbb{A}} = (0, \dots, \overset{i \text{ 番目}}{\underset{\text{1}}{\tilde{1}}}, \dots, 0)$$

です. V の 2 点 $x = (x_1 P, \dots, x_n P)$, $y = (y_1 P, \dots, y_n P)$ に対して $V \times V$ 上のペアリングを

$$e(x, y) := \prod_{i=1}^n e(x_i P, y_i P)$$

と定義します. 右辺のペアリングは従来のペアリング $e : G \times G \rightarrow G_T$ (G_T は位数 p の巡回群) です. すると

$$e(x, y) = \prod_{i=1}^n e(P, P)^{x_i y_i} = e(P, P)^{\sum_{i=1}^n x_i y_i} = e(P, P)^{x_{\mathbb{A}} \cdot y_{\mathbb{A}}}$$

となります. 右辺の $x_{\mathbb{A}} \cdot y_{\mathbb{A}}$ は $x_{\mathbb{A}}$ と $y_{\mathbb{A}}$ の内積です. $(p, V, G_T, \mathbb{A}, e)$ の組を DPVS といいます.

基底の変換を考えましょう. 行列式が 0 でない n 次正方行列 $B = (b_{ij})$ に対して

$$b_i := (b_{i1} P, \dots, b_{in} P) \in V$$

とすると $\mathbb{B} := \{b_i\}$ は V の基底です. $C := (B^T)^{-1} = (c_{ij})$ として,

$$b_i^* := (c_{i1} P, \dots, c_{in} P) \in V$$

とすると $\mathbb{B}^* := \{b_i^*\}$ も V の基底になります。 b_i と b_j^* についてペアリングをとると、

$$e(b_i, b_j^*) = \prod_{k=1}^n e(b_{ik}P, c_{jk}P) = e(P, P)^{\sum_{k=1}^n b_{ik}c_{jk}}.$$

指数部分を見ると、 $C = (B^T)^{-1}$ より

$$\sum_{k=1}^n b_{ik}c_{jk} = (BC^T)_{ij} = (BB^{-1})_{ij} = (I_n)_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

したがって

$$e(b_i, b_j^*) = \delta_{ij}.$$

よって $x, y \in V$ について、

$$e(x, y) = e(P, P)^{x_{\mathbb{B}} \cdot y_{\mathbb{B}}} = e(P, P)^{x_{\mathbb{B}} \cdot y_{\mathbb{B}}^*}$$

が成り立ちます。基底を取り替えるとその表現ベクトルの内積が変わらないためペアリングの値は同じです。

12.4 DPVS を用いた内積暗号

DPVS を用いて構成した attribute-hiding な内積暗号を紹介します [OT12]。

暗号の最初のセットアップでは前節の最後の行列式 B を使います。ただ n 次元ではなく $4n+2$ 次元です。また乱数 $r \in \mathbb{F}_p$ をとり、 $C = (B^T)^{-1}$ の代わりに $C := r(B^T)^{-1}$ を使います。全体が r 倍されるだけで、直交性などの性質は変わりません。 $g_T := e(P, P)^r$ とします。行列 \mathbb{B} と g_T がシステムの公開鍵で \mathbb{B}^* がシステムの秘密鍵です。

パラメータ $v \in \mathbb{F}_p^n$ に対する鍵生成を次のようにします。 σ, η を $\mathbb{F}_q, (\mathbb{F}_q)^n$ からランダムに選び、秘密鍵 k^* を

$$k^* := (\underbrace{1}_{1}, \underbrace{\sigma v}_{n}, \underbrace{0}_{2n}, \underbrace{\eta}_{n}, \underbrace{0}_{1})_{\mathbb{B}^*} \text{ に対応する } V \text{ の元}$$

としてユーザに渡します（基底 \mathbb{B}^* を用いて計算します）。ここで $1, n, 2n, n, 1$ はそれぞれの次元を表します。

平文 m とパラメータ $x \in \mathbb{F}_p^n$ に対する暗号化は、 ω, ϕ, ζ をランダムに選び、

$$c_1 := (\underbrace{\zeta}_{1}, \underbrace{\omega x}_{n}, \underbrace{0}_{2n}, \underbrace{0}_{n}, \underbrace{\phi}_{1})_{\mathbb{B}} \text{ に対応する } V \text{ の元,}$$

$$c_2 := g_T^\zeta m,$$

$$\text{Enc}(m, x) := (c_1, c_2)$$

とします。

復号は $(c_1, c_2) := \text{Enc}(m, x)$ に対して

$$\text{Dec}(c_1, c_2) := c_2 / e(c_1, k^*)$$

です。 $e(c_1, k^*)$ の指数部分は c_1, k^* のベクトル表現の内積に等しいでした。

$$e(c_1, k^*) = g_T^{(\zeta, \omega x, 0, 0, \phi) \cdot (1, \sigma v, 0, \eta, 0)} = g_T^{\zeta + \omega \sigma v \cdot x}.$$

$v \cdot x = 0$ が成り立つなら

$$c_2 / e(c_1, k^*) = g_T^\zeta m / g_T^\zeta = m$$

より復号できます。

基底の変換はベクトルを隠すために使われます。たとえば xyz -3次元空間の中であるベクトル $v = (a, b, c)$ が xy 平面にあるか否かは z 成分 c が 0 かそうでないかですぐ分かります。では適当な基底 b_1, b_2, b_3 をとったときに v が b_1 と b_2 で作られる平面の中にあるか否かはどうでしょう。15.1節で紹介する DLIN 仮定の元ではこの場合は難しいことが知られています。

一見 k^* や c_1 の最初の成分 $(1, \sigma v)$ や $(\zeta, \omega x)$ の後半部分は不要に見えます。しかし $3n + 1$ だけ余計につけることでこの方式が DLIN 仮定の元で適応的安全であり、パラメータ x の情報をもらさない attribute-hiding な方式であることが証明されます。 k^* や c_1 は $4n + 2$ 次元全体ではなくその部分空間を動くのですが、DLIN 仮定の元ではランダムに選んだ点と区別できないということを非常に複雑な手順で証明しています。

12.5 関数型暗号

述語暗号はもう少しだけ一般化されます。それは復号すると元の平文ではなく、指定した関数に平文を入れた値が出てくるというものです。述語暗号と同様に定式化すると次のようになります。まず 2 個のパラメータ v と X にしたがって決まる関数 f を決めます。全体をつかさどるシステムはマスター秘密鍵 K_{prv} と公開鍵 K_{pub} を用意します。それからシステムはパラメータ v にしたがって決まる秘密鍵 k_v を作ります。

$$k_v := \text{KeyGen}(K_{\text{prv}}, K_{\text{pub}}, v).$$

ユーザは平文 X を公開鍵 K_{pub} を使って暗号化します。

$$c_X := \text{Enc}(K_{\text{pub}}, X).$$

そのとき

$$\text{Dec}(K_{\text{pub}}, k_v, c_X) = f(v, X)$$

となります。 X がそのままでのではなく計算された結果がでるので、このような形の暗号を関数型暗号 (FE : Functional Encryption) といいます。 X をパラメータ x と従来の平文 m の組

$X := (x, m)$ とし,

$$f(v, (x, m)) := \begin{cases} m & \text{if } R(v, x) = \text{true}, \\ \text{出力しない} & \text{if } R(v, x) = \text{false} \end{cases}$$

とすると関係 $R(\cdot, \cdot)$ に関する述語暗号となるので関数型暗号は述語暗号の一般化です。ただどんな関数 f が使えるとなると一般論の構成はとても難しいことが想像できます。

識別不能難読化 (iO) と呼ばれる手法を用いた一般的な構成法が提案されています [Wat14]、一般的すぎて効率のことは考えられていません。この分野は今しばらく理論面を進める動きが強いようです。

12.6 この章のまとめ

属性ベース暗号をより一般化した述語暗号や関数型暗号を紹介しました。これらの暗号は秘密鍵と公開鍵の他に様々なパラメータを持ち、それらのパラメータにある関係があるときのみ復号できる暗号です。述語暗号では暗号化に使われたパラメータを秘密にできます。

第 13 章

準同型暗号

この章では準同型暗号と呼ばれる暗号とその応用について紹介します。準同型という言葉は日常ではありませんなじみがないため難しそうに見えますが、手短に言うと暗号化したまま平文の足し算や掛け算ができる性質のことです。足し算ができる準同型暗号を加法準同型暗号、掛け算ができる暗号を乗法準同型暗号と呼びます。2で割った余りの世界 (\mathbb{F}_2) で足し算は排他的論理和 (xor), 掛け算は論理積 (and) に相当します。この二つの操作ができると理論的に任意の演算ができます。

加法準同型暗号や乗法準同型暗号の例は昔からいくつか知られていましたが、両方の性質を満たす完全準同型暗号と呼ばれる暗号の例は長らく知られていませんでした。2009年に初めて完全準同型暗号の構成方法が提示され、それ以降研究が活発に行われています。

13.1 ElGamal 暗号再び

ElGamal 暗号は g を公開パラメータ、秘密鍵を x 、公開鍵 $y := g^x$ を固定し、平文 m に対して乱数 r を用いて

$$\text{Enc}(m) := (c_1, c_2) = (g^r, my^r)$$

という形の暗号化をしていました。3.13節で述べたように ElGamal 暗号は秘密鍵や平文を知らなくても暗号文を何倍かできてしまうという弱点がありました。その性質についてもう少し考えます。2個の平文 m_1, m_2 があってそれぞれ別の乱数 r_1, r_2 を用いて暗号化していたとします。

$$\begin{aligned} \text{Enc}(m_1) &:= (g^{r_1}, m_1 y^{r_1}), \\ \text{Enc}(m_2) &:= (g^{r_2}, m_2 y^{r_2}). \end{aligned}$$

暗号化されたものの要素をそれぞれ掛けると $(g^{r_1+r_2}, m_1 m_2 y^{r_1+r_2})$ となります。これは平文 $m_1 m_2$ を乱数 $r_1 + r_2$ で暗号化したものとみなせます。これを $\text{Enc}(m_1) \text{Enc}(m_2)$ と書くことになると

$$\text{Enc}(m_1) \text{Enc}(m_2) = \text{Enc}(m_1 m_2)$$

が成り立ちます。平文 m_1, m_2 を暗号化したまま 2 個の平文の積 m_1m_2 を得られたので ElGamal 暗号は乗法準同型暗号です。

また ElGamal 暗号をちょっとひねってみましょう。適当な整数 h をとり m の代わりに h^m を暗号化してみます。

$$\text{Enc}'(m) := \text{Enc}(h^m) = (g^r, h^m y^r).$$

すると 2 個の平文 m_1, m_2 に対しては

$$\text{Enc}'(m_1) \text{Enc}'(m_2) = \text{Enc}(h^{m_1}) \text{Enc}(h^{m_2}) = \text{Enc}(h^{m_1} h^{m_2}) = \text{Enc}(h^{m_1+m_2}) = \text{Enc}'(m_1 + m_2)$$

が成り立ちます。ここでこの式の右辺を形式的に $\text{Enc}'(m_1) + \text{Enc}'(m_2)$ と書くと

$$\text{Enc}'(m_1) + \text{Enc}'(m_2) = \text{Enc}'(m_1 + m_2)$$

となります。こちらは暗号化したまま 2 個の平文の和 $m_1 + m_2$ を得られたので Enc' は加法準同型暗号です。ここで $m_1 = m_2 = m$ とすると $\text{Enc}'(2m) = 2 \text{Enc}'(m)$ が成り立ち、一般に任意の整数 n に対して $\text{Enc}'(nm) = n \text{Enc}'(m)$ も成り立ちます。

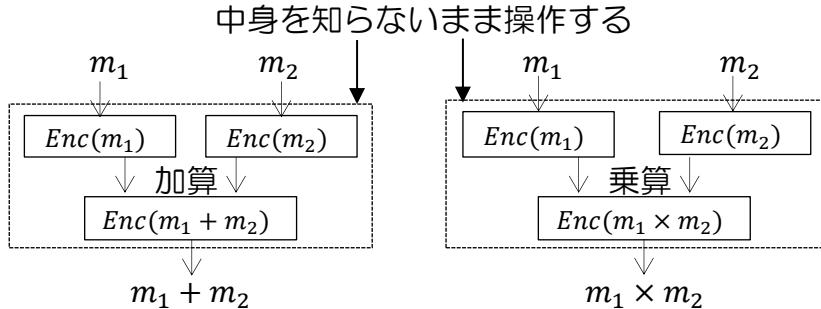


図 13.1 準同型暗号

このひねった ElGamal 暗号を以降では lifted-ElGamal 暗号と呼びます。lifted-ElGamal 暗号には一つ注意点があります。それは復号する場合、ElGamal 暗号と同様にすると h^m までは求められますが、ここから m を求めるには離散対数問題を解けなければなりません。これは一般には無理です。それが難しいことを仮定して暗号が作られているのですから！ただし m がたとえば 0 以上 1 万以下の数値であるという制限があれば秘密鍵を知っている人だけが総当たりなどの手法を使うことで復号できます（値の制限をつけても復号できるのは秘密鍵を知っている人だけです。知らない人は依然として復号できません）。

このように lifted-ElGamal 暗号は平文になんらかの制約を入れないと使えません。そのような制約のない実用的な加法準同型暗号としては 1998 年に提案された岡本・内山暗号や 1999 年に提案された Pailler 暗号などが知られています。加法性と乗法性の両方を備えた準同型暗号を完全準同型暗号といいます。完全準同型暗号は鍵サイズが数十 MB から GB 単位と巨大になるため今のところ実際に使うのは難しいです。

なお準同型暗号は $\text{Enc}(m)$ と公開鍵から $\text{Enc}(2m)$ を作れます。したがって 3.13 節の ElGamal 暗号の弱点と同様頑強性を持っています。公開鍵暗号と同じモデルで考える限り IND-CCA1 安全なものが最良ということになります。

13.2 化合物データベースの秘匿検索

古典的ではありますが加法準同型暗号の別の応用例として 2011 年に提案された化合物データベースの秘匿検索技術を紹介しましょう^{*1}。まず背景から説明します。新薬の開発では効果がありそうと思われる新しい化合物に対して、それとよく似た既存の化合物の性質を調べることがよくあります。化合物はいくつかの原子の固まり（基と呼ばれる）の組み合わせからなっています。

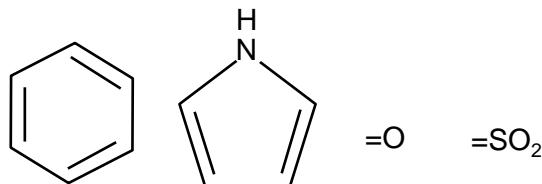


図 13.2 基の例

図 13.2 は基の一部を書いたものです。こういったものが複雑に組み合わさって大きな化合物になっている図を見られた方も多いでしょう。その一部を別のものに変えるとどうなるかという膨大な資料（データベース）があります。様々な用途に特化した高価なデータベースが市販されています。

さて、ここでデータベースを使いたい研究者と、データベースを売りたい販売会社との間に相反する要求があります。研究者がどんな薬を開発しているかは企業秘密で、自分は何の化合物を調べたいか外部の人に知らせたくはありません。しかしあるデータベースを購入する前に、そのデータベースが自分の用途にあってはいるかはちょっと確認してから買いたいものです。

逆に販売会社としては購入前にできるだけ中身の情報を見せたくはありません。自分の用途にあってはいるかを判断する材料は、ここでは対象とした化合物の類似物がデータベース内にいくつあるかという情報とします。つまり

- (A) クライアントは自分が何を検索しているかは知らせずにデータベース内の類似物の総数を知りたい
- (B) サーバは類似物の総数以外の情報をクライアントに与えたくない

ということです。条件 (A) を諦めてクライアントが対象化合物をサーバに教えれば条件 (B) は容

^{*1} 秘密計算による化合物データベースの検索技術：

http://www.aist.go.jp/aist_j/press_release/pr2011/pr20111101/pr20111101.html

易に実現できます。逆に (B) を諦めてサーバがデータベースの情報を全て公開すれば条件 (A) は容易に実現できます。いかにして二つの条件を両立するかが問題です。

13.2.1 Tversky 指数

まずこの問題を考える前に類似物の定義をしましょう。化合物は前述したように基からなっています。それをいくつかまとめたもう少し大きいまとまり（部分構造）を規定します。部分構造は数百から数万種類になるときもあるそうです。そしてある化合物が与えられたときに、その化合物が i 番目の部分構造をもっていれば 1, 持っていないければ 0 とすることで 01 のビット列をえられます。これを化合物のフィンガープリントと呼びます。MacCS key と呼ばれる方法では 166 ビットのビット列として表現するそうです。

化合物をビット列で表現できれば、2 個のビット列に対して近い、遠いを定義できます。ここでは Tversky 指数と呼ばれる指数を紹介します。 $\alpha, \beta > 0$ を固定し、2 個のビット列 \mathbf{x}, \mathbf{y} に対して、

$$f(\mathbf{x}, \mathbf{y}) := \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cap \mathbf{y}| + \alpha |\mathbf{x} \setminus \mathbf{y}| + \beta |\mathbf{y} \setminus \mathbf{x}|}$$

とします。ここで \cap や \setminus などの操作は \mathbf{x}, \mathbf{y} を 1 がたっている部分構造の集まりで表される集合に対する演算とします。式で書くと

$$\begin{aligned} |\mathbf{x}| &:= \#\{i \mid x_i = 1\} = \sum_i x_i, \\ |\mathbf{x} \cap \mathbf{y}| &:= \#\{i \mid x_i = 1 \text{かつ } y_i = 1\} = \mathbf{x} \cdot \mathbf{y}, \\ |\mathbf{x} \setminus \mathbf{y}| &:= \#\{i \mid x_i = 1 \text{かつ } y_i = 0\} = |\mathbf{x}| - |\mathbf{x} \cap \mathbf{y}| \end{aligned}$$

です。Tversky 指数 f は 0 以上 1 以下の値をとり、2 個の化合物が一つも同じ部分構造を持っていなければ 0, 完全に一致すれば 1 となります。そうすると、たとえば「指定した化合物 \mathbf{x} に対して Tversky 指数 0.9 以上のものがいくつあるか」を明確に定義できます。 $\alpha = \beta = 1$ のときは Tversky 指数の定義の分母は

$$\begin{aligned} \text{分母} &= |\mathbf{x} \cap \mathbf{y}| + (|\mathbf{x}| - |\mathbf{x} \cap \mathbf{y}|) + (|\mathbf{y}| - |\mathbf{x} \cap \mathbf{y}|) \\ &= |\mathbf{x}| + |\mathbf{y}| - |\mathbf{x} \cap \mathbf{y}| = |\mathbf{x} \cup \mathbf{y}| \end{aligned}$$

なので

$$f(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|}$$

となりますね。

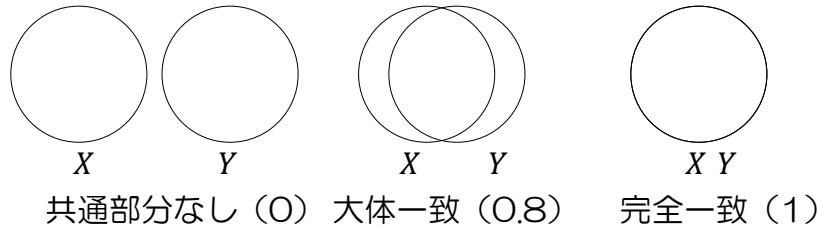


図 13.3 Tversky 指数の例

もう一度やりたいことをおさらいしましょう。クライアントがビットベクトル \mathbf{x} , サーバが N 個のビットベクトル \mathbf{y}_i を持っています。

- (A) クライアントは \mathbf{x} をサーバに教えずに類似度が t ($0 \leq t \leq 1$) 以上の \mathbf{y}_i の個数 $C = \#\{i \mid f(\mathbf{x}, \mathbf{y}_i) \geq t\}$ を知りたい。
- (B) サーバは C 以外の情報をクライアントに与えたくない。

ここまで定式化するとこのプロトコルは化合物の類似度検索だけでなく、様々な応用に使えそうですね。4.8節で紹介した秘匿共通集合計算の応用例ととらえることもできます。

ただ $t = 1$ (完全一致) としてこのプロトコルを走らせるとクライアントはサーバが \mathbf{x} を持っているかどうかの情報を得られます。すると何度も繰り返すとサーバの情報の一部を取得できてしまいます。ですので実際には t の値に何らかの制約を入れるか問い合わせ回数の上限を入れるなどの運用が必要でしょう。

13.2.2 暗号化したまま Tversky 指数を計算する

プロトコルの実現方法を考えましょう。まずクライアントが検索対象として \mathbf{x} を選び、サーバ側のある \mathbf{y} に対して Tversky 指数を計算しようと思います。今やりたいことはクライアント側で \mathbf{x} を何らかの暗号化をしてサーバに送り、その状態でサーバに計算してもらうことです。その操作に加法準同型暗号を使います。

ただこのままで小数を含むため暗号の計算には向きません。整数のみの計算ですむように式変形をしましょう。まず $\alpha = a/c$, $\beta = b/c$, 指定した Tversky 指数を t/c と分数で近似します。 a, b, t は 0 以上の整数, c は正の整数とします。 $f(\mathbf{x}, \mathbf{y}) \geq t/c$ という関係式の分母を払って $|\mathbf{x} \setminus \mathbf{y}| = |\mathbf{x}| - |\mathbf{x} \cap \mathbf{y}|$ などを使うと

$$R := (c^2 - tc + ta + tb)|\mathbf{x} \cap \mathbf{y}| - ta|\mathbf{x}| - tb|\mathbf{y}| \geq 0 \quad (13.1)$$

となります。目標は暗号化された \mathbf{x} が与えられたときに任意の \mathbf{y} と式 (13.1) の左辺を計算することです。平文 m の lifted-ElGamal 暗号を $\text{Enc}(m)$ とします。 $\mathbf{x} := (x_1, \dots, x_n)$ の各ビットをそれぞれ暗号化したもの $\text{Enc}(\mathbf{x}) := (\text{Enc}(x_1), \dots, \text{Enc}(x_n))$ と書き、クライアントは $\text{Enc}(\mathbf{x})$ をサーバに送ります。

加法準同型の性質を使うと

$$\text{Enc}(|\mathbf{x}|) = \text{Enc}\left(\sum_i x_i\right) = \sum_i \text{Enc}(x_i)$$

が成り立ち、サーバは $\text{Enc}(|\mathbf{x}|)$ を計算できます。同様に $|\mathbf{x} \cap \mathbf{y}|$ は $y_i = 1$ となる i についての x_i の総和なので

$$\text{Enc}(|\mathbf{x} \cap \mathbf{y}|) = \text{Enc}\left(\sum_{y_i=1 \text{ となる } i} x_i\right) = \sum_{y_i=1 \text{ となる } i} \text{Enc}(x_i)$$

が成り立ちます。 $|\mathbf{y}|$ の値をサーバは知っているので $\text{Enc}(|\mathbf{y}|)$ を計算できます。よって式 (13.1) の左辺のそれぞれの項を暗号化したまま計算できてそれらの線形和である R_i に対する $\text{Enc}(R)$ をサーバ側で計算できることが分かりました。

特に $|\mathbf{x} \cap \mathbf{y}| = \mathbf{x} \cdot \mathbf{y}$ は $0, 1$ ベクトルである \mathbf{x} と \mathbf{y} の内積なので $\text{Enc}(|\mathbf{x} \cap \mathbf{y}|)$ は片方のベクトルを暗号化したまま内積を求めたことになります。そう考えるといろいろな応用ができそうですね。

$\text{Enc}(R)$ をクライアントに返して復号すれば $R \geq 0$ か否かを判別できます (R は \mathbb{F}_p の元なので $0 < x < p/2$ なら正, $p/2 < x < p - 1$ なら負とする)。これがこのプロトコルの主要な部分です。

13.2.3 サーバ側の安全性向上

13.2.2 節の方法でクライアントの情報を暗号化したまま送りました。ただこのままではクライアントはサーバから返された各 $\text{Enc}(R_i)$ の情報から i 番目のデータがどれくらい x に近いが分かってしまいます。より安全性を高めるためにデータの攪乱（かくらん）を行います。まず 0 以上の値 u_i を N_p 個、負の値 v_i を N_n 個（クライアントが復号できる範囲で）ランダムにとり、それらを暗号化します。そして $\{\text{Enc}(u_i), \dots, \text{Enc}(v_i), \dots, \text{Enc}(R_i), \dots\}$ の順序をランダムに入れ換えて、これらの集合と N_p の値をクライアントに返します。

クライアントは受け取った値の集合を復号し 0 以上である個数を数え、それから N_p を引くと \mathbf{x} に類似する個数を得えられます。こうするとどの y_i に R_i が対応するのか分からなくなるためサーバから漏れる情報を減らせます。

13.2.4 悪意あるクライアントへの対処

13.2.3 節のダミーデータの追加と攪乱により、サーバとクライアントの相反する要求に答えました。ただしもしくはクライアントがある x_i を 0 でも 1 でもない極端に大きい値を入れていたらどうでしょう。対応する R_i も極端な値となります。たとえ $\{R_i\}$ がシャッフルされていたとしてもどの i に対応していたのか分かってしまうかもしれません。これを防ぐにはサーバ側で受け取った $\text{Enc}(x_i)$ に対して x_i が 0 か 1 のどちらかであることを確認し、そうでなければ攻撃されていると解釈してエラーにする必要があります。もちろん 0 でも 1 でもないということは分かっても 0 か 1 のどちらかであることは分かってはいけません。そんなことが可能なのでしょうか。

それはゼロ知識証明という手法を使うとできます。したがって悪意あるクライアントに対処できます。ゼロ知識証明については章を改めて紹介しましょう。この節で秘匿検索の紹介についてはひとまず終わり、残りの節では完全準同型暗号の構成の一つについて紹介します。

13.3 LWE 問題

今まで紹介した楕円曲線やペアリングとは全く異なる種類の暗号を紹介します。それは格子をベースにした暗号です。格子とはベクトル空間の基底をとったとき、その整数係数の線形結合で表される部分空間のことです。近年、格子を使って完全準同型暗号や、多重線形写像、関数型暗号などが構築され注目を集めています。

LWE (Learning with Errors) 問題とは誤差が入った多変数線形連立方程式を解く問題です。以下は有限体 \mathbb{F}_p 上で考えます。 s を n 次元ベクトル、 e をある確率にしたがって選ばれる m 次元ベクトル（誤差と呼ぶ）、 A を $n \times m$ 次行列 ($m \geq n$) とします。このとき

「 $A, As + e$ が与えられたときに s を求めよ。」

という問題です。今までと同様、ランダムに選んだ m 次元ベクトル b をとり、

「 $(A, As + e)$ と (A, b) が与えられたときにどちらが与えられたか判別せよ。」

という判別問題もあります。誤差 e が無い ($e = 0$) なら問題は単なる線形代数です。しかし誤差があると簡単にはいきません。誤差は Gauss 分布（正規分布：釣鐘形の分布）からとることが多いようです。

たとえば $n = 2, m = 4$,

$$A := \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 4 & 1 \\ 7 & 4 \end{pmatrix}, \quad s := \begin{pmatrix} 4 \\ 2 \end{pmatrix}, \quad e := \begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \end{pmatrix}, \quad x := As + e = \begin{pmatrix} 11 \\ 9 \\ 18 \\ 37 \end{pmatrix}$$

として、 A, x が与えられたとして $s := (a, b)^T$ (同時に $e := (e_1, e_2, e_3, e_4)^T$ も) を求めることを考えましょう。つまり

$$\begin{aligned} a + 3b + e_1 &= 11, \\ 2a + b + e_2 &= 9, \\ 4a + b + e_3 &= 18, \\ 7a + 4b + e_4 &= 37 \end{aligned}$$

を解く問題です。通常なら未知数が $2 + 4 = 6$ 個で方程式が 4 個なので解けません。しかし e の成分が ± 1 か 0 しかないという情報があったとしましょう。すると 3 番目の式から 2 番目の式を引くことで

$$2a + e_3 - e_2 = 9$$

となり、 $e_3 - e_2$ のとりうる範囲は $-2, -1, 0, 1, 2$ なので a は 4 か 5.

4番目の式より $4b + e_4 = 37 - 7a$ なので $a = 4$ なら $b = 2, e_4 = 1$. $a = 5$ なら $4b + e_4 = 2$ となる b はありません. よって $a = 4, b = 2$ となりました. これぐらいの規模なら s を求められます. しかし変数の数が多くなると組み合わせの数が多くなります.

p, n, m を適切にとると LWE 問題は格子上の短い長さのベクトルを求める問題に帰着されることが知られています. その問題の解法は現在知られている最良のものでも指数時間かかります. 量子コンピュータが登場しても解くのは難しいと思われています. 今まで紹介した楕円曲線やペアリングを使った暗号は量子コンピュータだと解けることが知られているためこれは大きな利点です.

LWE のうち演算効率がよいものとして ring-LWE (RLWE) があります [LPR10]. これは行列の演算を環（かん）上の演算に置き換えたものです. 環とは足し算, 引き算, 掛け算はできるけど割り算ができるとは限らない対象物です. 体になり損ねたものという感じです. たとえば 6 は素数ではないのでその余りの世界 $\mathbb{Z}/6\mathbb{Z} := \{0, 1, 2, 3, 4, 5\}$ は体ではなく環になります. 『イデアル格子暗号入門』(有田正剛) [有田 14] は具体的な小さい値について例を出しながら詳しく解説しています.

RLWE では n を 2 の巾乗の整数として $R := \mathbb{Z}[x]/(x^n + 1)$ を考えます. $\mathbb{Z}[x]$ は x を変数とする整数係数の多項式全体で, $\mathbb{Z}[x]/(x^n + 1)$ は多項式を $x^n + 1$ で割った余りを考えるという意味です (16.5 節参照). 整数係数の代わりに \mathbb{F}_p 係数の場合を $R_p := \mathbb{F}_p[x]/(x^n + 1)$ と書くことにします. R の元 y は $x^n + 1$ で割った多項式ですから高々 $n - 1$ 次の多項式となり, n 個の $a_i \in \mathbb{Z}$ を使って

$$y = \sum_{i=0}^{n-1} a_i x^i$$

と表せます. y の大きさを $|y| := \max |a_i|$ とします. R の元をある分布にしたがって選ぶというのを, n 個の \mathbb{Z} の元をその分布から選ぶことと定義します. 特に平均 0, 標準偏差 σ の Gauss 分布から y を選ぶことを $y \leftarrow \chi$ と書くことにします. R_p の元についても同様です.

RLWE は次の形になります. R_p からランダムに s をとり, 適当な回数だけ $a_i \in F_p$ をランダムに, $e_i \in R_p$ をある確率にしたがって選ぶ誤差としたとき,

$$\{(a_i, a_i s + e_i)\}$$

が $(R_p)^2$ からランダムに選んだ (a_i, b_i) と区別が付けられるかという問題です. パラメータを適切に選ぶと RLWE は難しいと考えられています.

13.4 RLWE 仮定による完全準同型暗号

RLWE を使って準同型暗号 (HE : Homomorphic Encryption) を構成します.

鍵生成 : n を 2 の巾, p を $p - 1$ が $2n$ の倍数であるような素数として環 R, R_p を作る. p より小さい素数を t とする. $R_t := (\mathbb{Z}/t\mathbb{Z})[x]/(x^n + 1)$ が平文空間となる.

$s \leftarrow \chi$ を秘密鍵とする。 R_p から a_1 をランダムに選び、 $e \leftarrow \chi$ を誤差と呼ぶ。 $a_0 := -(a_1 s + te)$ と a_1 を公開鍵とする。

暗号化：平文 $m \in R_t$ に対して、 $e_1, e_2, e_3 \leftarrow \chi$ を選び、

$$\text{Enc}(m) := (m + a_0 e_1 + t e_3, a_1 e_1 + t e_2)$$

とする。

復号：暗号文が $c := (c_0, c_1, \dots, c_k)$ の形のとき

$$\text{Dec}(c) := \sum_{i=0}^k c_i s^i \in R_q$$

とする。上記暗号化では要素が 2 個なのになぜここでは k 個あるかというと、乗算するたびに暗号文の要素が増えるからです。

暗号文の加算：暗号文 $c := (c_0, \dots, c_k)$, $c' := (c'_0, \dots, c'_{k'})$ に対して $k'' := \max(k, k')$ とし、短い方のベクトルに 0 を追加して同じ長さ k'' のベクトルにする。

$$\text{Add}(c, c') := (c_0 + c'_0, \dots, c_{k''} + c'_{k''})$$

とする。

暗号文の乗算：暗号文 $c := (c_0, \dots, c_k)$, $c' := (c'_0, \dots, c'_{k'})$ に対して

$$\hat{c}_i := \sum_{j=0}^i c_j c'_{i-j} \quad (13.2)$$

とする。これは y を変数として R_p 係数の多項式としての乗算：

$$\left(\sum_{i=0}^k c_i \right) \left(\sum_{i=0}^{k'} c'_i v^i \right) = \sum_{i=0}^{k+k'} \hat{c}_i v^i$$

を計算したことになる（係数の畳み込みと呼ばれます）。そして

$$\text{Mul}(c, c') := (\hat{c}_0, \dots, \hat{c}_{k+k'})$$

とする。

まず正しく復号できることを確認します。

$$(c_0, c_1) := (m + a_0 e_1 + t e_3, a_1 e_1 + t e_2)$$

のとき

$$\begin{aligned} \text{Dec}(c) &= c_0 + c_1 s = (m + a_0 e_1 + t e_3) + (a_1 e_1 + t e_2)s \\ &= m + (-(a_1 s + te))e_1 + t e_3 + a_1 e_1 s + t e_2 s = m + t(-e e_1 + s e_2 + e_3) \\ &= m + t e' \text{ ただし } e' := -e e_1 + s e_2 + e_3. \end{aligned}$$

ここで s, e, e_1, s, e_2, e_3 は全て標準偏差 σ の Gauss 分布から選んでいますのでそれらの係数は σ 程度の大きさです。したがって ee_1 や se_2 も σ^2 程度になります（もちろん正確には n にも依存します。厳密な評価はここでは省略します）， p を十分大きくとっておけば $|te'| < p/2$ とできます。そうすると、それらの係数を t で割って余りの m を取り出せます。

このように正しい値に誤差を加えて暗号化し、復号するときにその誤差を取り除くのが格子を使った暗号の特徴です。暗号文， $c := (c_0, \dots, c_k)$ について

$$\varphi(c) := \sum_{i=0}^k c_i s^i$$

としたとき， $\varphi(c) = m + te$ (e は適当な誤差で $|te| < q/2$) という形になったとします。すると、暗号文 c と c' に対して

$$\begin{aligned}\varphi(c)\varphi(c') &= (m + te)(m' + te') = mm' + t(te' + em' + e'm) = mm' + te'', \\ \text{ただし } e'' &:= te' + em' + e'm.\end{aligned}$$

e'' の中に t が入っているので足し算のときよりも値が大きいですが $|te''| < p/2$ となるように p を十分大きく取っておけば t で割って mm' を取り出せます。これにより暗号化したまま乗算もできます。

この暗号方式では計算途中に現れる式の係数が $p/2$ を越えてしまうと復号できなくなるので各種パラメータを慎重に決めなければなりません。

ここで紹介した方式は演算（特に乗算）を繰り返すと誤差の項が指数的に増大します。誤差が $p/2$ を超えないようにするためにには p を非常に大きくしなければなりません。あるいは演算回数に上限を設けます。こういう制約がある準同型暗号を SHE (Somewhat HE) といいます。

SHE を使って演算回数の上限がない本当の完全準同型暗号 (FHE : Fully HE) を作る方法の一つに bootstrap があります。それはなかなかトリッキーな手法です。

復号回路をそれ自身の SHE を使って作ります。復号回路を作るのに必要な乗算回数が可能なようにパラメータ設定をしておきます。暗号文 c を復号はできるが誤差が溜まった状態とします。そして c を暗号化して復号回路に入れると誤差がリセットされた状態の暗号文が出てきます。この暗号文に対して引き続き演算をするのです。

容易に想像できますが、この処理は非常に重たいです。しかも復号回路に必要な秘密鍵を自身の秘密鍵で暗号化して入れる必要があります、問題がややこしくなります。したがって上限回数を極力減らす方式や誤差を取り除く回路の効率化が研究されています。

ただ実際にクラウドで使いたい場合、複数回乗算を繰り返す操作というのは多くありません。たとえば n 個の暗号化された値の平均値を求めるなら加算は n 回必要ですが乗算は不要です。 n で割るのはクライアントですればよいからです。この場合は加法準同型暗号だけで実現できます。

分散を求めるのは加法準同型暗号だけではできません。しかし乗算は各要素については 1 回ずつ必要ですが後は総和を取るだけです。そのため SHE で実現できます。ベクトルの内積も分散と同じです。

表 13.1 準同型暗号の比較

	加算回数	乗算回数	処理性能	計算
従来の暗号	×	×	◎	暗号化のみ
加法 HE	任意回	×	○	平均など
SHE	十分な回数	数回	△	分散, 内積など
FHE	任意回	任意回	×	任意の演算

そこで SHE を使った応用例がいろいろ考えられています。実際にどのようなパラメータをとるうまくいくのかについて [NLV11] では 128 ビット AES 相当のセキュリティパラメータを使って実装実験をしています。それによると数十 KB の鍵サイズで実行時間も数十ミリ秒で可能とのことです。

たとえば富士通は指紋などの生体特徴データを暗号化したまま照合するシステムを提案しています [YSK⁺13]。ここで提案された手法は面白いテクニックを使っています。

SHE の暗号の乗算である式 (13.2) に着目します。2 個のベクトル $c := (c_0, \dots, c_k)$, $c' := (c'_0, \dots, c'_{k'})$ に対して畳み込みをしていました。ここで c' の要素を逆向きに並べたものを $d := (c'_{k'}, c'_{k'-1}, \dots, c'_0)$ とします。すると

$$\hat{c}_i := \sum_{j=0}^i c_j c'_{i-j} = \sum_{j=0}^i c_j d_j = c \cdot d$$

とベクトル c と d の内積で表せます。SHE の暗号文の乗算を暗号化された 2 個のベクトルの内積演算とみなせるのです。このアイデアを使って生体特徴データの照合を高速に処理しています。

また LWE 仮定ではなく、最大公約数 (GCD) を求める問題に誤差を入れて難しくした近似 GCD 問題に基づく SHE も提案されています [vDGHV10] [CLT14]。

13.5 この章のまとめ

暗号化されたデータの中身を知らないままで暗号化したまま足したり掛けたりできる準同型暗号を紹介しました。準同型暗号はクラウドサービスの形態と相性がよいので盛んに研究されています。ここでは加法準同型を用いた秘匿検索技術を紹介しました。加法と乗法の両方が可能な完全準同型暗号 (FHE) はまだ実用的とは言い難いです。乗算回数に制約がある Somewhat 準同型暗号 (SHE) でも応用例は多く、こちらはそれなりの速度で動くものが登場しています。

第 14 章

ゼロ知識証明

私はある方程式の答えを知っています。そしてあなたに私が答えを知っているということを示すことができます。でもあなたはその答えを知らないままなのです。ゼロ知識証明を使うとそんなことができます。

クラウドサービスでは暗号化したまま処理を行う準同型暗号が注目されています。準同型暗号では暗号化したまま処理をするのでクラウド側が処理をさぼって適当な計算をしてもクライアントは検証できません。正しくチェックしようとすると全ての計算を手元でしなくてはならず、クラウドに計算をさせた意味がなくなります。少ないコストでクラウドが不正をしていないか検証できないといけません。

また、逆にクラウドは中身を見ずに計算するためクライアントが不正な値を入れているか検証できません。そのような検証にゼロ知識証明の技術が使われることがあります。

14.1 ゼロ知識証明

ゼロ知識証明とはある人が自分の持っているある命題が真であることを、それ以外の知識を何も教えずに証明する手法です。1985 年に Shafi たちが初めて定式化しました [GMR89]。

ゼロ知識証明には大きく分けて二つの方式があります。一つは対話式ゼロ知識証明と呼ばれるもので、証明したい人と、確認したい人の間で何度もデータのやりとりをすることで納得する方法です。たとえばある合成数の素因数分解の結果を知っているということを相手に納得させる、離散対数問題の答えを知っていることを納得させるなどのやり方が提案されています。

もう一つは非対話ゼロ知識証明と呼ばれるもので、こちらは一度データを提示するだけで相手に納得させるものです。一般にこちらの方が効率よいです。

A さんが命題 P を知っていて B さんがそれを検証するとします。A さんが本当に命題 P を知っているときに検証が通ることを完全性 (completeness) といいます。逆に、検証が通れば本当に A さんが命題 P を知っていることを保証するのが健全性 (soundness) です。そして、その検証仮定で命題 P に関する情報が漏れていなことをゼロ知識性 (zero knowledge) といいます。ゼロ知

識証明はこの三つの性質を持っていなければなりません。

14.2 離散対数問題のゼロ知識証明

対話式ゼロ知識証明の例として離散対数問題の答えを知っていることを示す方法を紹介しましょう。 g, p, y は公開情報として私 A は「 $y := g^x \in \mathbb{F}_p$ となる x を知っている」ことを x を教えることなく他人 B に証明する方法です。

二人の間で次の対話をします。

A (証明したい人)		B (検証したい人)
Step 1 :	r を選び $a := g^r$ を送る	\longrightarrow
Step 2 :	$b = 0$ または 1 を受け取る	\longleftarrow
Step 3 :	$c := r + bx$ を計算して送る	\longrightarrow
Step 4 :		$g^c = ay^b$ が成り立てば OK

なぜこれで証明できたと思えるのか順を追って考えてみましょう。A が x を知っていれば上記手順を踏むと

$$g^c = g^{r+bx} = g^r(g^x)^b = ay^b$$

ですから Step 4 をパスします。

A が x を知らないのに答えを知っている振りはできるでしょうか。Step 2 で $b = 0$ が来ると予想していれば、Step 1 で適当に r を作って $a := g^r$ を渡します。すると Step 2 で期待通り $b = 0$ を受け取れば x を知らなくても $c = r$ なのでこれを送ると Step 4 の

$$g^c = g^r = a = ay^0 = ay^b$$

が成り立ちます。しかし予想が外れて $b = 1$ が来ると Step 4 をパスするには

$$g^c = (g^r)y^1 = g^{r+x}$$

でなくてはなりません。つまり Step 3 で $c = r + x$ を送らないといけないのですが、これは x を知らない人にとっては無理です。

逆に Step 2 で $b = 1$ が来ると予想していたとしましょう。Step 1 では適当な r' を作って $a := g^{r'}/y$ を渡します。Step 2 で期待通り $b = 1$ を受け取れば Step 3 では c として r' を渡します（相手をだませさえすればよいので計算手順に従う必要はないのです）。すると

$$g^c = g^{r'} = (g^{r'}/y)y = ay = ay^b$$

となり Step 4 をパスします。しかし予想が外れて $b = 0$ が来ると Step 4 をパスするには

$$g^c = ay^b = a = g^{r'}/y = g^{r'}/g^x = g^{r'-x}$$

でなくてはなりません。つまり Step 3 で $c := r' - x$ を送らないといけないのですが、やはりこれは x を知らない人にとっては無理です。

というわけで、この対話を一度だけしたときに x を知らない A がたまたま B の検証をパスする確率は $1/2$ と分かりました。するとこの対話を n 回して全てパスする確率は $(1/2)^n$ となります。十分な回数の対話を繰り返せば x を知らない人がパスする確率は無視できるほど小さくなります。B は自分が納得できる回数だけ対話を繰り返せばよいでしょう。

最後に B は納得できる以外に何か x の情報をえられたか確認しましょう。 $b = 0$ を送ったときは r の情報をもらうだけです。 $b = 1$ を送ったときは g^r と $r + x$ をもらいますが r が分からぬので x も分かりません。というわけで大丈夫なようです。本当は「知っていること以外何も情報を与えていない」とはどういうことかを厳密に定義する必要があります。そしてそのこのプロトコルがその定義を満たしていることも示す必要があります。このテキストでは省略します。

またこの方法では b の選択肢が 0 と 1 の 2 種類しかありませんでしたが、選択肢を増やすことで 1 回でパスする確率をずっと小さくしたり、非対話形式にしたりする方法もあります。情報セキュリティ大学院大学公開講座の『ゼロ知識証明入門』(土井洋) はそれらの方法について丁寧に解説しています [土井 07]。

14.3 秘匿検索におけるゼロ知識証明

ゼロ知識証明の簡単な紹介が終わったところで 13.2.4 節の悪意あるクライアントに対処する方式を紹介しましょう。2013 年に坂井氏たちにより lifted-ElGamal 暗号に対する平文が 0 か 1 のどちらかであることを非対話式ゼロ式証明で確認できる方法が提案されました [SEH⁺13]。クライアントがハッシュ関数といくつかの指數計算の組み合わせの情報を一度だけ送り、サーバ側で検証します。前節のように検証側が納得できるまで何度もやりとりが必要なプロトコルに比べてずっと効率的です。

lifted-ElGamal 暗号は x を秘密鍵、 g, h と $y := g^x$ を公開鍵としました。平文 m に対して乱数 r を用いて

$$\text{Enc}(m) = (c_1, c_2) = (g^r, h^m y^r)$$

と暗号化しています。 $m = 0$ または $m = 1$ に対するゼロ知識証明を次のように構成します。 H をハッシュ関数とします。

- $m = 0$ のとき : r_0, t_1, s_1 をランダムに選び,

$$R_{0,1} := g^{r_0}, \quad R_{0,2} := y^{r_0}, \quad R_{1,1} := g^{s_1}/c_1^{t_1}, \quad R_{1,2} := y^{s_1}/(c_2/h)^{t_1}$$

とする。

$$t := H(R_{0,1}, R_{0,2}, R_{1,1}, R_{1,2}, c_1, c_2, h, g, y)$$

として $t_0 := t - t_1, s_0 := r_0 + t_0 r$ とする。

- $m = 1$ のとき : r_1, t_0, s_0 をランダムに選び,

$$R_{0,1} := g^{s_0}/c_1^{t_0}, \quad R_{0,2} := y^{s_0}/c_2^{t_0}, \quad R_{1,1} := g^{r_1}, \quad R_{1,2} := y^{r_1}$$

とする.

$$t := H(R_{0,1}, R_{0,2}, R_{1,1}, R_{1,2}, c_1, c_2, h, g, y)$$

として $t_1 := t - t_0, s_1 := r_1 + t_1 r$ とする.

$(c_1, c_2, t_0, t_1, s_0, s_1)$ をゼロ知識証明の検証に使う値とします.

$m = 0$ か $m = 1$ であることは次のように検証します. $(c_1, c_2, t_0, t_1, s_0, s_1)$ が与えられたとき,

$$R'_{0,1} := g^{s_0}/c_1^{t_0}, \quad R'_{0,2} := y^{s_0}/c_2^{t_0}, \quad R'_{1,1} := g^{s_1}/c_1^{t_1}, \quad R'_{1,2} := y^{s_1}/(c_2/h)^{t_1}$$

を計算する.

$$t' := H(R'_{0,1}, R'_{0,2}, R'_{1,1}, R'_{1,2}, c_1, c_2, h, g, y)$$

を求めて $t = t_0 + t_1$ が成り立てば受理, そうでなければ棄却します.

式がとても複雑です. ここでは完全性だけ確認しましょう. $y = g^x, c_1 = g^r, c_2 = h^m y^r$ で $(c_1, c_2, t_0, t_1, s_0, s_1)$ が与えられたとします.

$m = 0$ のとき $t_0 = t - t_1, s_0 = r_0 + t_0 r$ なので

$$\begin{aligned} R'_{0,1} &= g^{s_0}/(g^r)^{t_0} = g^{s_0-t_0 r} = g^{r_0} = R_{0,1}, \\ R'_{0,2} &= y^{s_0}/(h^m y^r)^{t_0} = y^{s_0-t_0 r} = y^{r_0} = R_{0,2}, \\ R'_{1,1} &= R_{1,1}, \\ R'_{1,2} &= R_{1,2}. \end{aligned}$$

$m = 1$ のとき $t_1 = t - t_0, s_1 = r_1 + t_1 r$ なので

$$\begin{aligned} R'_{0,1} &= R_{0,1}, \\ R'_{0,2} &= R_{0,2}, \\ R'_{1,1} &= g^{s_1}/(g^r)^{t_1} = g^{s_1-t_1 r} = g^{r_1} = R_{1,1}, \\ R'_{1,2} &= y^{s_1}/((h^m y^r)/h)^{t_1} = y^{s_1-t_1 r} = y^{r_1} = R_{1,2}. \end{aligned}$$

いずれのときも $t' = t$ が成り立つので受理します.

ここでは lifted-ElGamal 暗号で平文が 0 か 1 のどちらかであることのみを検証する方法を紹介しました. クライアントが不正をしていないか確認するために使いました. 逆にクラウドに 13 章のような準同型演算をさせるときは, クラウドが不正をしていないか検証したいです. ある種の一般的な回路に対して正しく計算していることを検証できる枠組みを作る研究もあります [BSCG⁺13].

14.4 電子投票

この節では電子投票について考えてみます。秘密投票を電子的に行うにはどうすればよいでしょうか。4.7節ではブラインド署名が電子投票に使われる書きました。ここではブラインド署名を使わない方式を紹介しましょう。

Aさんの投票 m を暗号化 $\text{Enc}(m)$ しても、どこかのタイミングで復号しなければなりません。復号する人が $\text{Enc}(m)$ が Aさんの投票だと分かっていれば、その人は Aさんがどういう投票をしたのか分かってしまいます。したがって誰がどの暗号文を作ったのか分からないようにしなければなりません。

現実の期日前投票では秘密保持のために二重封筒を使います。投票用紙を内封筒に入れて封をし、それを外封筒に入れて封をして署名します。外封筒を開ける人は署名者が当人であること、当日に選挙権があることなどを確認します。外封筒のみを開封して内封筒をとりだした状態で皆の封筒を混ぜます。そうすると、どれが誰の封筒だったか分からなくなります。その後内封筒を開封して集計します^{*1}。

この操作と同様のことをするために皆の暗号文を集めてからかき混ぜる操作を行います。これをミックスネット（Mix-net）といいます。より安全にするためにミックスネットを複数回適用することも考えられます。

また復号鍵を複数個に分割し全てが揃わないと復号できないようにすることで開票者が一人で不正できないようにすることもできます。

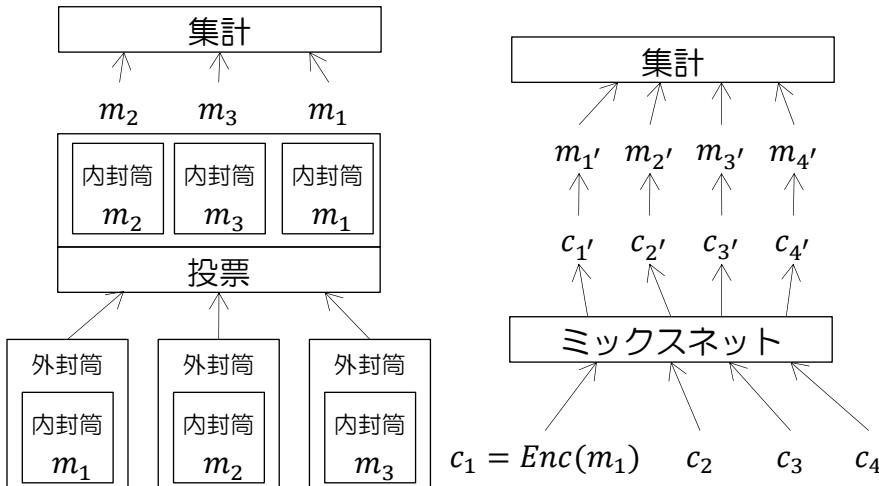


図 14.1 不在投票と電子投票

*1 「町田市 投票日に用事ができた」マメ知識「2重封筒のその後」：

<https://www.city.machida.tokyo.jp/shisei/senkyo/tohyo/tohyo03.html>

ミックスネットに求められる性質を考えてみましょう。まず誰もどの暗号文が誰のだったのか分からないようにかき混ぜなければなりません。それからミックスネットが不正をしないようにしなければなりません。誰かの投票を消して別の人の投票にしたりするとそれを第三者が検出できるようになります。これはゼロ知識証明などを使って行います。

14.5 シャッフル

暗号文をかき混ぜる（シャッフルする）ときに使われる再暗号化という方法を紹介します。ElGamal 暗号にはちょっと面白い性質があります。公開鍵 (g, y) と乱数 r を使って平文 m を暗号化します。

$$\text{Enc}(m) := (c_1, c_2) = (g^r, my^r).$$

この暗号文を受けた人が新たに乱数 r' を使って c_1 と c_2 から

$$(c'_1, c'_2) := (g^{r'} c_1, y^{r'} c_2)$$

を作ります。これは結局

$$(c'_1, c'_2) = (g^{r+r'}, my^{r+r'})$$

となり乱数 $r + r'$ を使って m を暗号化したのと同じです。元の平文を知らなくても他人が暗号化したものを見直すことができるのです。あるいは ElGamal 暗号の乗法準同型性を用いて暗号文 $\text{Enc}(m)$ に平文 1 の暗号文 $\text{Enc}(1) = (g^{r'}, y^{r'})$ を掛けた考えることもできます。これを ElGamal 再暗号化といいます。再暗号化すると元の暗号文がなんだったか分からなくなります。そうすると、与えられた ElGamal 暗号の暗号文 c_1, \dots, c_n に対してそれぞれを再暗号化してシャッフルすることでミックスネットを実現できます。

14.6 シャッフルのゼロ知識証明

ElGamal 暗号でシャッフルを用いたミックスネットは第三者がゼロ知識証明を用いて不正をしていないかチェックすることができます。2001年古川氏と佐古氏により提案されました [FS01]。

シャッフルは数学的には置換と呼ばれています。 n 個の値の置換には n 次正方行列 A が対応します。たとえば (a, b, c, d) を置換して (d, c, a, b) にする場合、

$$A := \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} d \\ c \\ a \\ b \end{pmatrix} = A \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

とれます。この行列 A がこの置換に対応します。置換行列 A はどの成分も 0 か 1 であり、どの行、どの列でも 1 になる値は一つしかありません。

さて (g, y) を公開鍵とする n 個の ElGamal 暗号文 (g_i, t_i) が与えられたとします。 n 次置換行列 A と n 個の乱数 r_1, \dots, r_n を用いると ElGamal 再暗号化してシャッフルする操作は

$$(g'_i, t'_i) := \left(g^{r_i} \prod_{j=1}^n g_j^{a_{ji}}, y^{r_i} \prod_{j=1}^n t_j^{a_{ji}} \right)$$

と表せます。 \prod が出ていて一見仰々しいですが、 a_{ji} は j が走ったとき 1 個を除いて全て 0 なのに注意してください。一つ目の成分の式を取り出すと

$$g'_i = g^{r_i} \prod_{j=1}^n g_j^{a_{ji}}. \quad (14.1)$$

です。 $\{y, g_i, t_i\}$ が与えられるとミックスネットは乱数 r_i と置換行列 A を使って (g'_i, t'_i) を計算して返します。値を受け取った人は正しくシャッフルされていることを確認します。つまりゼロ知識証明を使って r_i と A を教えてもらわずにシャッフルに使われた A が置換行列であることを確認するのです。実際のプロトコルはとても複雑なのでキーアイデアを紹介します。

まずある行列 $A = (a_{ij})$ が置換行列になる必要十分条件は

$$\sum_{h=1}^n a_{hi} a_{hj} = \delta_{ij}, \quad (14.2)$$

$$\sum_{h=1}^n a_{hi} a_{hj} a_{hk} = \delta_{ijk} \quad (14.3)$$

です。ここで δ_{ij} は $i = j$ のときのみ 1、それ以外は 0、 δ_{ijk} は $i = j = k$ のときのみ 1、それ以外は 0 を表す記号です。

置換行列が上記式を満たすことを示すのはそれほど難しくありません。 $i = j$ のとき $a_{hi} a_{hj} = a_{hi}^2$ 。 A の要素は 0 か 1 なので $a_{hi}^2 = a_{hi}$ 。 h を動かしたとき 1 になるのは一つだけなので

$$\sum_{h=1}^n a_{hi} = 1.$$

二つの異なる行において、1 になる列の位置は異なるので $i \neq j$ のとき $a_{hi} a_{hj} = 0$ 。よって式 (14.2) が示されます。式 (14.3) や逆向きの証明はここでは省略します。

さてミックスネットを検証したい人は $\{c_j\}$ をランダムにとりミックスネットに送ります。ミックスネットは

$$s := \sum_{j=1}^n r_j c_j, \quad (14.4)$$

$$s_i := \sum_{j=1}^n a_{ij} c_j \quad (14.5)$$

を計算して返します。検証者は

$$\sum_{i=1}^n s_i^2 = \sum_{j=1}^n c_j^2, \quad (14.6)$$

$$g^s \prod_{i=1}^n g_i^{s_i} = \prod_{i=1}^n g_i^{c_i} \quad (14.7)$$

を確認します。式(14.7)に式(14.6)、式(14.4)、式(14.5)を代入すると成立することが分かります。式(14.6)に式(14.5)を代入して式変形すると

$$\begin{aligned} \sum_{i=1}^n s_i^2 - \sum_{j=1}^n c_j^2 &= \sum_{i=1}^n \left(\sum_{j=1}^n a_{ij} c_j \right) \left(\sum_{k=1}^n a_{ik} c_k \right) - \sum_{j,k} c_j c_k \delta_{jk} \\ &= \sum_{j,k} \left(\sum_i a_{ij} a_{ik} - \delta_{jk} \right) c_j c_k = 0 \end{aligned}$$

となります。最後の等号は式(14.2)からしたがいます。逆にこの等式を満たす行列があったとき勝手な c_i に対してこの等式が成り立つためには無視できる確率を除いて

$$\sum_i a_{ij} a_{ik} - \delta_{jk} = 0$$

でなければなりません。同様に3乗の式

$$\sum_{i=1}^n s_i^3 = \sum_{j=1}^n c_j^3$$

を考えると二つ目の式(14.3)を満たす必要があることが示されます。したがって検証者が式(14.6)と式(14.7)の成立を確認することで A が置換行列であることを検証できます。実際には s を直接使うと a_{ij} の情報が漏れるため s や s_i にランダムな要素を加えてプロトコルを構成します。このように構成することで判定DH問題が困難ならミックスネットが不正をしていないことを確認できることが示されます。

まとめるとDDH仮定の元で検証可能なミックスネットを構築できました。その後対称多項式を使ったより効率のよい方法も提案されています。

14.7 この章のまとめ

自分が知っていることだけを相手に示すことができるゼロ知識証明という概念を紹介しました。ゼロ知識証明を用いると、あるDLPの答えを知っていることを相手に証明したり、暗号化された値の中身を見ずに値がある条件を満たしているかを確認したりできます。

ゼロ知識証明はクライアントやサーバが不正をしていないか確認する際にも有用な技術です。

第 15 章

ペアリングの安全性仮定

ペアリングを使った暗号プロトコルでは、様々な問題の困難さが仮定されています。この章ではそれらのいくつかを俯瞰します。

15.1 DDH 仮定と DLIN 仮定

3.12 節で ElGamal 暗号が強密匿性を持つことを紹介しました。強密匿性というのは暗号文からもとの平文のどんな情報もえられないという性質でした。その強密匿性を示すには DDH 問題が解けないという DDH 仮定が必要です。DDH 問題を復習すると楕円曲線上の点 P について、

「 P, aP, bP が与えられ、更に $Q_1 := abP$ からランダムな c に対する $Q_2 := cP$ のどちらかの Q_i が与えられたとき i は 1 と 2 のどちらか判定せよ。」

という問題です。DDH 仮定は暗号の安全性仮定で使われる基本的な仮定の一つです。ところがペアリング暗号では使い勝手が悪いことがあります。なぜなら aP と bP から $e(aP, bP) = e(P, P)^{ab}$ が得られます。

$$\begin{aligned} e(P, Q_1) &= e(P, abP) = e(P, P)^{ab}, \\ e(P, Q_2) &= e(P, cP) = e(P, P)^c \end{aligned}$$

ですから、これらの値と $e(P, P)^{ab}$ を比較することで DDH 問題を解けるからです（7.6 節参照）。

そこでペアリング暗号では DDH 仮定を変形した DLIN 仮定（Decisional LINear assumption）が使われることがあります。それは変数が増えてちょっと分かりにくいくらいですが次の問題が困難であるという仮定です。

「 P, xP, yP, axP, byP が与えられたときに $(a+b)P$ とランダムな c に対する cP のどちらが与えられたか判定せよ。」

DLIN 仮定はペアリングにおける基本的な仮定の一つと考えられています。なお DDH 問題について楕円曲線上の点 2 個からなる 2 次元ベクトル (P, aP) を考えて b 倍すると

$$b(P, aP) = (bP, abP)$$

です。つまり2点 $(0,0)$ と (P,aP) を結ぶ直線上に (bP,Q) があるか判定せよと解釈できます。

同様にDLIN問題は3次元ベクトル $(xP,0,P)$ と $(0,yP,P)$ について適当な a,b をとり、 Q が $(a+b)P$ とランダムな c に対する cP のどちらかのときに

$$a(xP,0,P) + b(0,yP,P) = (axP,byP,Q)$$

とできるか、すなわち3点 $(0,0,0)$, $(xP,0,P)$, $(0,yP,P)$ を含む平面上に (axP,byP,Q) があるか判定せよという問題です。

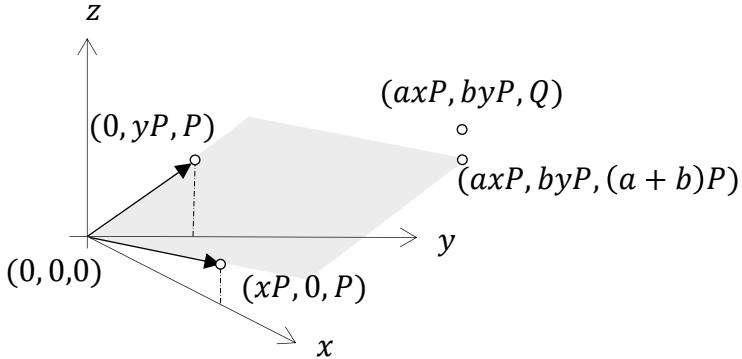


図 15.1 DLIN 仮定の解釈

15.2 パラメータつき問題

有限体や楕円曲線のDLPやDH問題は入力値が2個や3個、ペアリングのCBDH問題は4個と入力値の個数は一定でした。しかし、10.6節で紹介した l -BDHE問題は入力値の個数が l によって変わりました。このような入力値の個数がパラメータによって変化する問題をパラメータつき問題といいます。

パラメータつき問題は様々なものが提案されています。ルーツは私たちの提案した（安全では無かった）放送型暗号のプロトコルです。仕組みは簡単ですので紹介します。楕円曲線の点 P と整数 a を秘密に選び、各ユーザ u に対して

$$K_u := \frac{1}{u+a}P$$

を秘密鍵として秘密裏に渡します。整数 r をランダムにとり、 $Q := rP$ として、 $s := e(P,Q)$ をセッション鍵としてコンテンツを暗号化します。

$$(Q, aQ)$$

をセッション鍵の暗号化したものとして配信します。ユーザ u は自身の秘密鍵 K_u を用いて

$$e(K_u, uQ + aQ) = e\left(\frac{1}{u+a}P, (u+a)Q\right) = e(P, Q) = s$$

とペアリングを計算してセッション鍵を得られます。ユーザの秘密鍵に使われるパラメータが消え、单一の s が出てくるこの関係式を見つけたときは、これは綺麗だと思ったものです。最初に不正者追跡に使ってしまったのは失敗でしたが。

それはともかく、ここでユーザの部分集合 S が結託攻撃をして $\{K_{u_i}\}_{u_i \in S}$ に入っていないユーザ u に対する K_u を作るのは難しいだろうと感じました。つまり

「 $\left(\frac{1}{u_1+a}P, \dots, \frac{1}{u_n+a}P\right)$ が与えられたときに $u \notin \{u_1, \dots, u_n\}$ に対する $\frac{1}{u+a}P$ を求めよ。」

この問題（仮に n -CAA と呼びましょう）は難しい。この直感は今でも有効です（安全でなかったのはこれとは別の部分です）。

ただ発表するときに n -CAA が難しいから安全というのでは説得力が足りません。2000年頃はまだペアリングを使った暗号自体が認知されていませんでした。初めてこの方式を発表したのは、2001年3月の先端技術フォーラム（FACT）という研究集会で、これは Boneh-Franklin たちの発表でペアリング暗号が広まった CRYPTO2001 が開催された 2001 年 8 月より前です。

ペアリング自体が珍しいのに、その上よく分からぬ安全性仮定をつけても戸惑うだけです。そこで n -CCA って難しいよね、という根拠を見せなければなりません。それには DH 問題などの既存の問題との関係を示せるとよいです。いろいろ式をこね繰り回して納得のいくものを作ります。

15.3 ちょっとしたクイズ

さて突然ですが次の問題の中で一番難しいのはどれでしょう。

- I. P, aP, bP から abP を求める。
- II. P, aP から a^2P を求める。
- III. P, aP から $(1/a)P$ を求める。

I は通常の DH 問題です。II はその特別版なので I の方が難しそうです。III は逆数版ですね。

実はどれも同じ程度の困難さなのです。まず I が解けるなら II が解けることを示します。 P, aP が与えられたとします。 $b = a$ として P, aP, aP の三つ組が与えられたとすると I が解けるならこれから $aaP = a^2P$ が求められます。つまり I が解けるなら II が解けます。

この逆、II が解けるなら I が解けることも示せます。 P, aP, bP が与えられると $aP + bP = (a+b)P$ を求められます。 $(P, aP), (P, bP), (P, (a+b)P)$ のそれぞれの組に対して II を適用して $a^2P, b^2P, (a+b)^2P$ を求められます。すると

$$(a+b)^2P - a^2P - b^2P = 2abP$$

が分かり 2 で割って abP を求められます。つまり I が解けました。

というわけで I の難しさと II の難しさは同程度だと分かりました。最後に II が解けると III が解けることを示します。 P, aP が与えられたとき、これを $(aP), (a^{-1})(aP)$ が与えられたとみな

します。すると II を使うと $(a^{-1})^2(aP) = (1/a)P$ が求まり、III が解けました。逆も同様です。

15.4 ヒントの多い問題

さて、一見異なる難しさに見えた II や III は DH 問題と同じ難しさであることが分かりました。I より II や III はパラメータの数が少ないのでこちらで考えます。

II や III の問題、すなわち DH 問題が難しいということを、与えられたデータの線形和しか作れないという解釈をしてみます。 P と aP という“ベクトル”から（多項式時間で）計算できるのは $\langle P, aP \rangle := \{xP + y(aP) \mid x, y \in \mathbb{Z}\}$ という形だけで、その中に a^2P や $(1/a)P$ という“ベクトル”は入ってないというイメージです。もちろん巡回群 $\langle P \rangle$ を考えると実際には入っていますが、計算できる範囲には含まれていないというニュアンスです。

すると P, aP から $(1/a)P$ を求めるのが難しいなら、「 P, aP, a^2P が与えられたときに $(1/a)P$ を求める問題」、「 P, aP, a^2P, a^3P が与えられたときに $(1/a)P$ を求める問題」などもヒントは増えるけれども十分難しいのではないかという気がしてきます。2 個のときは DH 問題と等価なのもありがたいです。すなわち

n -weak DH 問題 (n -DHI 問題) 「 P, aP, a^2P, \dots, a^nP が与えられたときに $(1/a)P$ を求めよ」あるいは同値なことですが

n -weak DH 問題 2 「 P, aP, a^2P, \dots, a^nP が与えられたときに $a^{n+1}P$ を求めよ」

という問題を提起します。そして $(n-1)$ -weak DH 問題は n -CCA と等価です。 n -CCA が解けるなら $(n-1)$ -DH 問題が解けることを示しましょう。

$$\{P, aP, \dots, a^{n-1}P\}$$

が与えられたとします。 $\{u_i\}$ を全て異なるとして、 $b := a - u_0$, $Q := (b + u_1) \cdots (b + u_n)P$ とおくと

$$\frac{1}{b + u_j} Q = \prod_{i=1, i \neq j}^n (a - u_0 + u_i)P$$

の積の部分は a についての $n-1$ 次多項式で係数は全て既知です。仮定より a^iP が与えられているので

$$\left(\frac{1}{b + u_1} Q, \dots, \frac{1}{b + u_n} Q \right)$$

は全て計算できます。よって n -CCA が解けるなら

$$\frac{1}{b + u_0} Q = \frac{1}{a} Q = (1/a) \prod_{i=1}^n (a - u_0 + u_i)P$$

を求められます。

$$\prod_{i=1}^n (a - u_0 + u_i) = \prod_{i=0, n} d_i a^i P,$$

と積を展開してできる係数を d_i とすると, $\{u_i\}$ は全て異なるので定数項 d_{-1} は 0 ではありません。この両辺を a で割ると左辺と右辺の $a - iP$ ($i = 0, \dots, n-1$) がわかっているので $(1/a)P$ も計算できます。よって $(n-1)$ -DH 問題が解けました。□

逆に $(n-1)$ -weak DH 問題が解けるなら n -CCA が解けることも示せます。これにより逆数の形の問題は DH 問題の類似だとみなせます。この解釈で私の中では逆数の形の秘密鍵の位置づけが明確になりました。 n -weak DH 問題は今は n -DHI (DH Inversion) 問題と呼ばれます。

15.5 n -DHI 問題と n -SDH 問題

逆数の形を秘密鍵に使うと 7.7 節とは異なる構造の ID ベース暗号を構成できます。

鍵生成 : 楕円曲線 E とその点 P をとる。 $p := |\langle P \rangle|$ とし, ハッシュ関数 $h : \{ID\} \rightarrow \mathbb{F}_p$ を選ぶ。整数 s と点 $Q \in \langle P \rangle$ をランダムにとり (P, s) を秘密鍵, $(Q, sQ, g := e(P, Q))$ を公開鍵とする。

ユーザ鍵生成 : 各ユーザ u に対して $h_u := h(ID_u)$ として

$$S_u := \frac{1}{h_u + s} P$$

を秘密鍵として渡す。

ユーザ u に対する暗号化 : 平文 m に対して整数 r をランダムにとり,

$$(C_1, c_2) := (r(h_u Q + sQ), mh(g^r))$$

を暗号文とする。

復号 : ユーザ u は暗号文 (C_1, c_2) に対して h を使って $c_2/h(e(S_u, C_1))$ により平文を求める。

うまく復号できることは

$$\frac{c_2}{h(e(S_u, C_1))} = \frac{mh(g^r)}{h\left(e\left(\frac{1}{h_u+s}P, r(h_u+s)Q\right)\right)} = \frac{mh(g^r)}{h(e(P, Q)^r)} = m$$

となることで確認できます。この暗号 (SK 暗号) は 2005 年に Chen, Cheng たちにより n -BDHI 問題 (n -Bilinear DH Inversion) が困難な仮定のもので安全と示されます [CC05]。 n -BDHI 問題は n -DHI 問題とペアリングを合成した形の問題です :

「 $P, aP, a^2P, \dots, a^n P$ が与えられたときに $e(P, P)^{1/a}$ を求めよ」

n -DHI が解けるなら n -BDHI は解けます。求めた値 $(1/a)P$ と P のペアリングをとればよいからです。DLP と DH 問題の関係のように, $(1/a)P$ を直接求められなくても $e(P, P)^{1/a}$ を求められる可能性はありますが、今のところその方法は見つかっていません。

さて、2004 年に Boneh, Boyen はペアリングを使った短い署名を提案します。方式は 7.10 節で紹介しました。この署名の安全性は前節の n -DHI 問題を少し修正したものを根拠にしています。

「 $P, aP, a^2P, \dots, a^n P$ が与えられたときにある c に対する $\frac{1}{c+a}P$ を求めよ」

これは n -SDH 問題 (n -Strong DH) と呼ばれています。先程の n -weak DH 問題は n -SDH 問題で c を 0 に固定したと類似物とみなせます。 $c = 0$ じゃなくてもよいからいいから、とりあえずどれかの c について見つければよいと問題自体を緩くしました。また “weak” が “strong” に変わりました。DH 問題より易い問題なので私は “weak” と名付けましたが、その問題を安全性仮定に使うときは仮定が強いです。そのため “strong” とするのが適切だったのかもしれません。

最後に 10.6 節の n -BDHE 問題を振り返りましょう。 n -DH 問題は与えられた “ベクトル” $P, aP, \dots, a^n P$ の右隣、 n -DHI 問題は左隣の値を求める問題と解釈します。すると n -BDHE 問題は左右に連続する “ベクトル” が与えられたときに真ん中の値を求める問題とペアリングを合成した形にみえます。

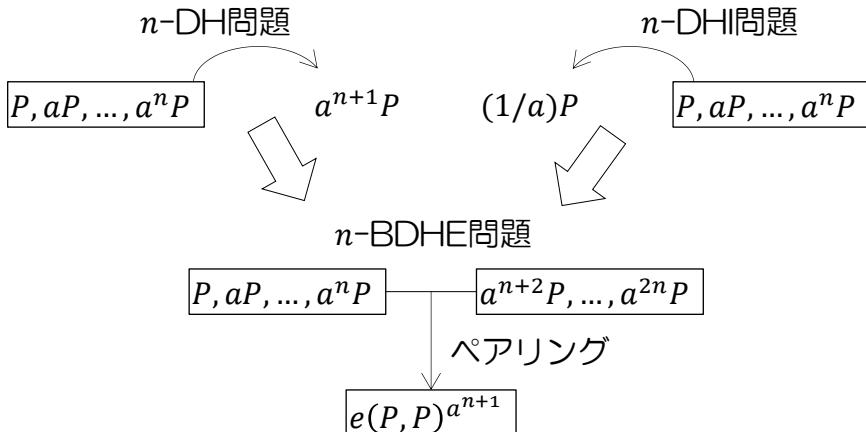


図 15.2 n -DH と n -DHI と n -BDHE の関係

これらの問題の派生系もいくつかあります。一般的にできるだけ単純でよく知られた問題を安全性の根拠におくのがよいです。

15.6 補助入力付き DLP

パラメータつき問題は従来の DH 問題よりも沢山の情報が付与されています。それらの情報を使って問題を解くことはできないのでしょうか。2006 年 Cheon が、ある条件を満たせば 6.6 節で紹介した ρ 法よりも効率よく求める方法があることを示します [Che06]。それは次の問題です。

楕円曲線上の点 P に対する巡回群 $G := \langle P \rangle$ の大きさを素数 p とし、 $p - 1$ の約数の一つを d とします。このとき

「 P, aP, a^dP が与えられたとき、 a を求めよ。」

通常の離散対数の入力 P, aP の他に a^dP の情報が与えられているので、この問題は補助入力付き離散対数問題と呼ばれます。この問題は $O(\sqrt{p/d} + \sqrt{d})$ の計算量で求められます。特に d が

\sqrt{p} ぐらいの大きさのときは

$$O\left(\sqrt{p/\sqrt{p}} + \sqrt{\sqrt{p}}\right) = O(2p^{1/4}) = O(p^{1/4})$$

となります。

Cheon の攻撃方法は 2 ステップからなります。まず \mathbb{F}_p^* の生成元を g とします。 $e := (p-1)/d$ とし、 $g_d := g^d$, $g_e := g^e$ とおきます。 g_d は e 乗すると 1, g_e は d 乗すると 1 になります。さて、 $h_1 := a^d$ とおくと e 乗すると 1 なので $h_1 \in \langle g_d \rangle$ 。よって

$$h_1 = g_d^{k_1}$$

となる整数 $0 \leq k_1 < e$ が存在します。未知の整数 x に対して $h_1 P = g_d^x P$ という方程式を考えると、 $x = k_1$ がこの方程式の解です。この解を ρ 法と同様のやり方で探します。この計算量は $O(\sqrt{e}) = O(\sqrt{p/d})$ です。

次に $h_2 := g^{-k_1} a$ とすると

$$h_2^d = g^{-dk_1} a^d = g_d^{-k_1} g_d^{k_1} = 1.$$

d 乗して 1 になったので

$$h_2 = g_e^{k_2}$$

となる整数 $0 \leq k_2 < d$ が存在します。未知の整数 x に対して $h_2 P = g_e^x P$ という方程式を考えると、 $x = k_2$ がこの方程式の解です。この解を探す計算量は $O(\sqrt{d})$ です。

このとき

$$(g^{-k_1} a) P = h_2 P = g_e^{k_2} P = g^{ek_2} P,$$

つまり

$$a P = g^{k_1 + ek_2} P$$

となり $a = g^{k_1 + ek_2}$ が求まりました。この演算量は $O(\sqrt{e})$ と $O(\sqrt{d})$ の和なので $O(\sqrt{p/d} + \sqrt{d})$ です。

伊豆氏、酒見氏たちは Chen のアルゴリズムと実装を改良することで 2012 年に 160 ビットの補助入力付き DLP を解いています [SHI⁺12]。ペアリングのパラメータつきの問題はパラメータの選び方によっては Cheon の攻撃を受ける場合があるので慎重に設計しなければなりません。

15.7 この章のまとめ

ペアリングを使った暗号で使われる安全性仮定の一部を紹介しました。DBDH や DLIN 仮定が基本的です。安全性の定義に使われる要素の数が変動するタイプのものも使われます。そのようなタイプの問題は要素の数がかなり大きいと攻撃されやすくなることがあります。

第 III 部

数学的なはなし

III 部の流れ

III 部は数学を中心とした内容です。

16 章では有限体での割り算のやり方を詳しく紹介します。それからペアリングの計算で現れる拡大体の紹介もします。多項式を使って拡大体を作る方法は、13.4 節で紹介した ring-LWE ベースの準同型暗号の構成にも使われる重要なものです。

17 章ではトーラスと楕円曲線の関係を紹介します。 \wp 関数と呼ばれる複素数上の二重周期関数を用いてトーラスから Weierstrass の方程式の解（楕円曲線）への写像を構成します。それからトーラスの上で自然に考えられた足し算が楕円曲線上ではどのような演算になるのか、幾何学的な定義と具体的な式を導出します。群の単位元になる無限遠点は射影空間を導入すると統一的に扱えます。最後に演算を効率的に行うための手法をいくつか紹介します。

18 章ではペアリングの定義と具体的な計算方法について述べます。そのために楕円曲線上の点からなる因子と呼ばれるものを定義し、因子と楕円曲線上の関数の密接な関係を紹介します。

第 16 章

有限体

II 部で紹介した暗号プロトコルは有限体や楕円曲線、ペアリングなどの数学的対象物が登場しました。楕円曲線は浮輪の表面（トーラス）という幾何学的な描写で紹介し、ペアリングも同じ枠組みで説明しました。それはできるだけ少ない数学的知識でイメージしやすいものにしたかったからです。

基本的な性質を認めれば、それらを利用した暗号プロトコルの理解はできると思います。しかし、楕円曲線やペアリングを使った暗号をコンピュータ上で実装しようとそれだけでは不十分です。これ以降の章では数学的な準備を追加しつつ、有限体の補足や楕円曲線の代数的な側面、およびペアリングの具体的な計算方法について触れたいと思います。

16.1 Euclid の互除法と有限体の逆数

有限体の逆数は 3.10 節で定義しましたが素数 p に対して \mathbb{F}_p が体になることは示していませんでした。この節では \mathbb{F}_p が体になること、すなわち 0 でない任意の元 $x \in \mathbb{F}_p$ に対して逆数が存在することを具体的に計算して示します。

そのための準備としてまず任意の自然数 a, b の最大公約数 (greatest common divisor) $\gcd(a, b)$ の求め方を説明します。たとえば 72 と 27 の最小公倍数は $72 = 2^3 \cdot 3^2$, $27 = 3^3$ ですから $\gcd(72, 27) = 3^2 = 9$ です。

因数分解しないで求める方法として次の方法があります。もしかしたら小学校で習った方がいらっしゃるかもしれません。

$$72/27 = 2 \text{ 余り } 18.$$

$$27/18 = 1 \text{ 余り } 9.$$

$$18/9 = 2.$$

$18/9 = 2$ で割り切れたので $\gcd(72, 27) = 9$ です。これを一般化します。

まず最小公倍数の性質を確認しましょう。

- 0 はどんな数の倍数でもあるので $\gcd(a, 0) = a$.
- $\gcd(a, b) = \gcd(a - b, b)$.

等号を確認するために $c := \gcd(a, b), c' := \gcd(a - b, b)$ とおきます。 $a = ca', b = cb'$ とかけるので $a - b = c(a' - b')$ 。よって c は $a - b$ と b の公約数となり、 c' の最大性から $c \leq c'$ 。

同様に $a - b = c'x, b = c'y$ と書くと $a = c'(x + y)$ 。よって c' は a と b の公約数となり、 c の最大性から $c' \leq c$ 。よって $c = c'$. \square

- $b \neq 0$ のとき $\gcd(a, b) = \gcd(a \bmod b, b)$.

二つ目の性質を繰り返し適用すれば OK です。

これらの性質を踏まえて $r_0 > r_1 \geq 0$ となる整数の $c := \gcd(r_0, r_1)$ を求めます。 $r_1 = 0$ なら $c = r_0$ です。 $r_1 > 0$ なら r_0 を r_1 で割って $r_0 = r_1 q_2 + r_2, 0 \leq r_2 < r_1$ とします。

$$\gcd(r_0, r_1) = \gcd(r_0 \bmod r_1, r_1) = \gcd(r_2, r_1).$$

$r_2 = 0$ なら $\gcd(r_0, r_1) = r_1$ です。 $r_2 > 0$ なら今度は r_1 を r_2 で割ります。 $r_1 = r_2 q_3 + r_3, 0 \leq r_3 < r_2$ とおくと

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \gcd(r_2, r_3).$$

これを繰り返し適用すると $r_0 > r_1 > r_2 > \dots$ という自然数の減少列ができます。 $r_i \geq 0$ なのでこれは有限回で止まります。つまり $r_s = 0$ となる s があり、 $\gcd(r_0, r_1) = r_{s-1}$ となります。これを Euclid の互除法といいます。

先ほどの例を見直すと、

$$\begin{aligned} \gcd(72, 27) &= \gcd(72 \bmod 27, 27) = \gcd(18, 27) = \gcd(27, 18) = \gcd(27 \bmod 18, 18) \\ &= \gcd(9, 18) = \gcd(18, 9) = \gcd(18 \bmod 9, 9) = \gcd(0, 9) = 9 \end{aligned}$$

を計算していたのでした。

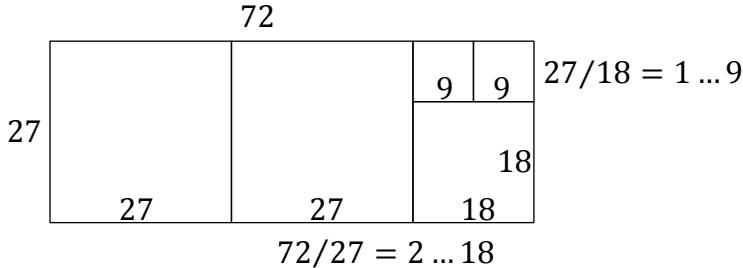


図 16.1 72 と 27 の最小公倍数の求め方

横 72, 縦 27 の長方形を大きな正方形から敷きつめていく感じです。

さて、たとえば $s = 5$ だったとすると、

$$\begin{aligned} r_0 &= r_1 q_2 + r_2, \\ r_1 &= r_2 q_3 + r_3, \\ r_2 &= r_3 q_4 + r_4, \\ r_3 &= r_4 q_5 + r_5 (= 0) \end{aligned}$$

という関係式が成り立っています。すると

$$\begin{aligned} \gcd(r_0, r_1) &= r_4 = r_2 - r_3 q_4 = r_2 - (r_1 - r_2 q_3) q_4 = r_1(-q_4) + r_2(1 + q_3 q_4) \\ &= r_1(-q_4) + (r_0 - r_1 q_2)(1 + q_3 q_4) \\ &= r_0(1 + q_3 q_4) + r_1(-q_4 - q_2(1 + q_3 q_4)). \end{aligned}$$

これは a, b となる整数を用いて

$$ar_0 + br_1 = \gcd(r_0, r_1)$$

とできることを示しています。しかも a, b はこの手続きで具体的に計算できます。同様の手順をたどることで一般の s に対しても整数 a, b を具体的に計算し

$$ar_0 + br_1 = \gcd(r_0, r_1)$$

とできることが分かります。この方法を拡張 Euclid の互除法といいます。

さてようやく有限体の逆数を求める方法の準備が整いました。任意の $x \in \mathbb{F}_p^* := \mathbb{F}_p \setminus \{0\}$ に対して逆数を求めます。 p は素数で $0 < x < p - 1$ なので x と p は互いに素です。つまり $\gcd(x, p) = 1$ です。拡張 Euclid の互除法を用いることで

$$ax + bp = \gcd(x, p) = 1$$

となる整数 a, b を見つけることができます。このとき $ax \equiv 1 \pmod{p}$ なので a は x の逆数です。これにより \mathbb{F}_p が有限体になることが分かりました。

16.2 Euclid の互除法の効率

前節で拡張 Euclid の互除法を用いて $x \in \mathbb{F}_p^*$ の逆元を具体的に求めることができました。この計算効率はどの程度なのでしょう。3.10 節で試した一つずつ逆数か確認する総当たり方法より効率はよいのでしょうか。大雑把ですが考えてみましょう。

拡張 Euclid の互除法では整数の減少列

$$r_0 > r_1 > r_2 > \cdots > r_s = 0$$

を作ります。 r_i がどれぐらいの速さで小さくなるかを見積もりましょう。 r_{i+1} が $r_i/2$ より大きいかそうでないかで場合分けします。

- $r_{i+1} \geq r_i/2$ のとき, $r_i > r_{i+1} \geq r_i/2$ なので r_i を r_{i+1} で割ると商は $q_{i+2} = 1$ で余りは

$$r_{i+2} = r_i - r_{i+1} \leq r_i/2.$$

- $r_{i+1} < r_i/2$ のときは $r_{i+2} < r_{i+1} < r_i/2$.

いずれの場合も $r_{i+2} \leq r_i/2$ です。つまり 2 ステップ進めると必ず半分未満になることが分かりました。したがって $r_s = 0$ となる s は高々 $2\log_2(r_0)$ となります。 $x < p$ ですので逆元を求める計算は $O(\log(p))$ となります。よって総当たりよりもずっと効率がよいアルゴリズムであると分かりました。

16.3 Fermat の小定理と有限体の逆数

有限体の逆数を求める別の方法を紹介しましょう。

2 項定理から始めます。変数 x, y と 0 以上の整数 n に対して

$$(x+y)^n = \sum_{k=0}^n {}_n C_k x^k y^{n-k}$$

が成り立ちます。ここで

$${}_n C_k := \frac{n!}{k!(n-k)!}$$

は n 個の中から k 個取り出す組み合わせの数で 2 項係数といいます。2 項定理を証明しましょう。

$n = 0$ のときは両辺ともに 1 で成り立ちます ($x^0 = 0$ とします)。

$n = k$ のとき成り立つとすると $n = k + 1$ のとき,

$$\begin{aligned} (x+y)^{n+1} &= (x+y)(x+y)^n = (x+y) \left(\sum_{k=0}^n {}_n C_k x^k y^{n-k} \right) \\ &= \left(\sum_{k=0}^n {}_n C_k x^{k+1} y^{n-k} \right) + \left(\sum_{k=0}^n {}_n C_k x^k y^{n+1-k} \right) \\ &\quad (\text{一つ目の項で } k=n \text{ を分ける。二つ目の項で } k=0 \text{ を分ける}) \\ &= x^{n+1} + \left(\sum_{k=0}^{n-1} {}_n C_k x^{k+1} y^{n-k} \right) + \left(\sum_{k=1}^n {}_n C_k x^k y^{n+1-k} \right) + y^{n+1} \\ &= x^{n+1} + \left(\sum_{k=1}^n {}_n C_{k-1} x^k y^{n+1-k} \right) + \left(\sum_{k=1}^n {}_n C_k x^k y^{n+1-k} \right) + y^{n+1} \\ &= x^{n+1} + \left(\sum_{k=1}^n ({}_n C_{k-1} + {}_n C_k) x^k y^{n+1-k} \right) + y^{n+1}. \end{aligned}$$

ここで

$$\begin{aligned} {}_nC_{k-1} + {}_nC_k &= \frac{n!}{(k-1)!(n-k+1)!} + \frac{n!}{k!(n-k)!} = \frac{n!k}{k!(n+1-k)!} + \frac{n!(n+1-k)}{k!(n+1-k)!} \\ &= \frac{n!(k+n+1-k)}{k!(n+1-k)!} = \frac{(n+1)!}{k!(n+1-k)!} = {}_{n+1}C_k \end{aligned}$$

を使うと

$$(x+y)^{n+1} = x^{n+1} + \left(\sum_{k=1}^n {}_{n+1}C_k x^k y^{n+1-k} \right) + y^{n+1} = \sum_{k=0}^{n+1} {}_{n+1}C_k x^k y^{n+1-k}$$

が成り立つことが分かりました。よって任意の n について 2 項定理が成り立ちます。 \square

2 項係数を図 16.2 のように 3 角形に並べると、ある列の隣り合う 2 個の数字の和が一つ下の列の値になっていることが分かります。これは上で示した ${}_nC_{k-1} + {}_nC_k = {}_{n+1}C_k$ の関係を示しています。パスカルの 3 角形といいます。この図から任意の ${}_nC_k$ が自然数であることを見てとれます。

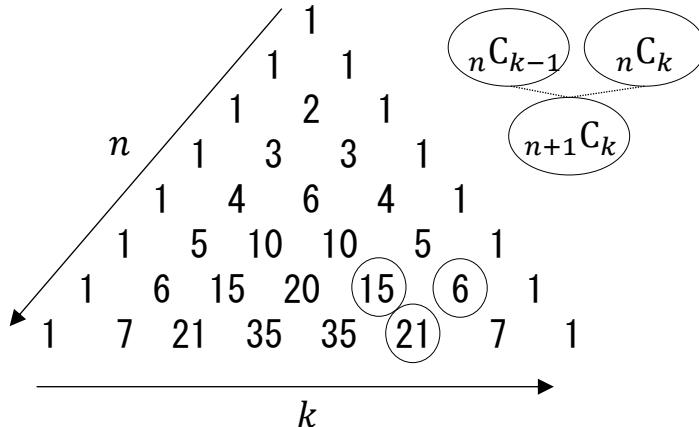


図 16.2 パスカルの 3 角形

補題 1 素数 p と $0 < k < p$ となる自然数 k に対して ${}_pC_k$ は p の倍数である。

なぜなら ${}_pC_k = \frac{p!}{k!(p-k)!}$ の分子 $p!$ は p の倍数で、分母 $k!(p-k)!$ に p は出てこないので p が約分されないからです。 \square

図 16.2 では $n = 2, 3, 5, 7$ のとき両側の 1 以外は n の倍数であることを確認できます。 n が 4 や 6 などの合成数のときは n の倍数になっていないですね。

補題 2 素数 p と整数 x, y に対して

$$(x+y)^p \equiv x^p + y^p \pmod{p}.$$

なぜなら左辺を 2 項展開すると補題 1 より k が 0 や p 以外では係数が p の倍数となるからです。 \square

補題 3 素数 p と整数 x に対して

$$x^p \equiv x \pmod{p}.$$

なぜなら補題 2 を使うと

$$x^p = (1+(x-1))^p \equiv 1^p + (x-1)^p = 1 + (x-1)^p \equiv 1 + 1 + (x-2)^p \equiv \cdots \equiv \underbrace{1 + \cdots + 1}_{x \text{ 個}} = x \pmod{p}.$$

□

定理 4 (Fermat (フェルマー) の小定理) 有限体 \mathbb{F}_p^* の元 x に対して

$$x^{p-1} = 1.$$

なぜなら補題 3 の右辺を左辺に移項して $x(x^{p-1} - 1) \equiv 0 \pmod{p}$. $0 < x < p$ なら x は p と互いに素なので $x^{p-1} - 1 \equiv 0 \pmod{p}$. つまり有限体の元として $x^{p-1} - 1 = 0$. □

Fermat の小定理を示せたので有限体の逆数の求め方の準備が整いました. 有限体 \mathbb{F}_p^* の元 x に対して,

$$x^{p-1} = x \cdot x^{p-2} = 1$$

が成り立ちます. これは x^{p-2} が x の逆数であることを示しています. また, 内乗の計算は 3.8 節で示したように $O(\log(p))$ で計算できます. よってこの方法は拡張 Euclid の互除法と同程度の計算量でできることが分かりました.

16.4 巡回群の大きさ

6.4 節で確認したように $\mathbb{F}_7^* := \mathbb{F}_7 \setminus \{0\}$ の要素からなる乗法的巡回群について, その大きさは 1, 2, 3, 6 の 4 種類ありました. これは \mathbb{F}_7^* の位数 $7 - 1 = 6$ の約数です. 一般に \mathbb{F}_p^* の要素 g に対して巡回群 $\langle g \rangle$ の位数は $p - 1$ の約数になります.

なぜなら g の位数を a とします. 定義により a は $g^a = 1$ となる最小の a で $a \leq p - 1$ です. また Fermat の小定理より $g^{p-1} = 1$ です. a と $p - 1$ について拡張 Euclid の互除法により

$$as + t(p-1) = \gcd(a, p-1)$$

となる整数 s, t があります. すると

$$g^{\gcd(a, p-1)} = g^{as+t(p-1)} = (g^a)^s \cdot (g^{p-1})^t = 1.$$

$\gcd(a, p-1) \leq a$ で a の最小性より $\gcd(a, p-1) = a$. つまり a は $p - 1$ の約数です. □

□

16.5 拡大体

実数に対する複素数のように有限体 \mathbb{F}_p の複素数版, あるいはそれ的一般化である拡大体を紹介します. 拡大体とはいくつかの \mathbb{F}_p をまとめたものに新しい四則演算を定義して一つの体と見なしものです. 拡大体はペアリングの計算で必要になります.

定義に入る前に、まずは複素数の作り方を確認しましょう。複素数 \mathbb{C} の点 z は実数 \mathbb{R} の 2 点 a, b を使って

$$z = a + bi$$

と表しました。ここで i は 2 乗して -1 になる値です。2 点 $z_1 = a + bi, z_2 = c + di$ に対して加減算は

$$z_1 \pm z_2 := (a \pm c) + (b \pm d)i$$

です。乗算は $i^2 = -1$ という規則により

$$z_1 z_2 := (ac - bd) + (ad + bc)i$$

となります。 $z = x + yi$ の逆数は分子、分母に $x - yi$ を掛けることで

$$\frac{1}{z} = \frac{1}{a + bi} = \frac{a - bi}{(a + bi)(a - bi)} = \frac{a - bi}{a^2 + b^2} = \frac{a}{a^2 + b^2} - \frac{b}{a^2 + b^2}i$$

とできました。

2 乗して -1 になる値は \mathbb{R} の中には存在しません。しかしその値を形式的に i とおいて $r_0 + bi$ に対して i が変数であるかのように計算し、 i^2 がでてきたら -1 に置き換えるという作業をしています。この作業により \mathbb{C} は体になりました。このとき \mathbb{C} は \mathbb{R} の 2 次拡大体であるといいます。 \mathbb{C} は 2 次元 \mathbb{R} ベクトル空間に掛け算と割り算を導入したものと考えることもできます。

今 \mathbb{R} に対してこの作業をしましたが、これを \mathbb{F}_7 に置き換えて考えてみましょう。すると実数に対する複素数と同じようなことができる事が分かります。係数はそれぞれ mod 7 します。

たとえば $(3 + 2i) + (6 + 5i) = 9 + 7i = 2$ とか $(3 + 2i)(6 + 5i) = (18 - 10) + (15 + 16)i = 8 + 31i = 1 + 3i$ ということです。

このように \mathbb{F}_7 に i を追加してできた $r_0 + bi$ の形の集合を \mathbb{F}_{7^2} と書いて \mathbb{F}_7 の 2 次拡大体といいます。 $7^2 = 49$ で \mathbb{F}_{49} と間違えそうですが、全然違うものであることに注意してください。 \mathbb{F}_p は p が素数のときでないと逆数が存在しないことがあるので体にならないのでした。

\mathbb{F}_p に i を追加する操作はいつでもうまくいくのでしょうか。実はそうではないのです。たとえば \mathbb{F}_5 に i を追加してもうまくいきません。なぜならたとえば

$$(1 + 2i)(1 - 2i) = (1 + 4) + (2 - 2)i = 5 = 0 \text{ mod } 5$$

という等式が成り立ちます。これは 0 でない 2 個の数を掛けて 0 になることを示しています。つまり $1 + 2i$ の逆数は存在しないため $\mathbb{F}_5[i]$ は体ではありません。

実数 \mathbb{R} や \mathbb{F}_7 に $i = \sqrt{-1}$ を追加してうまく拡大体を作れたのは、その値が \mathbb{R} や \mathbb{F}_7 に含まれていなかつたからなのでした。つまり方程式 $t^2 = -1$ の解が \mathbb{R} や \mathbb{F}_7 に解を持たないということです。実際、 \mathbb{F}_7 の中で $0, 1, 2, \dots, 6$ を 2 乗すると $0, 1, 4, 2, 2, 4, 1$ となり $-1 = 6 \text{ mod } 7$ になるものはありません。

\mathbb{F}_5 ではどうでしょう。 $2^2 = 4 = -1 \text{ mod } 5$ なので \mathbb{F}_5 の中で $t^2 = -1$ は解を持ちます。つまり $t^2 + 1 = (t + 2)(t - 2)$ と因数分解できるということです。方程式 $t^2 + 1 = 0$ の解が \mathbb{F}_p の中で解

すると \mathbb{F}_p に $i = \sqrt{-1}$ を追加しても拡大体はできません。解けない場合は拡大体を作れることが分かります。方程式が解けないとき、つまり因数分解できないとき $t^2 + 1$ を \mathbb{F}_p 上既約であるといいます。

まとめると $f(t) = t^2 + 1$ について i を $f(i) = 0$ となる形式的な値として

$$\begin{aligned} f(t) &= (t+2)(t-2) \bmod 5 \text{ なので } \mathbb{F}_5 \text{ に } i \text{ を追加しても拡大体にならない} \\ f(t) &\text{は } \mathbb{F}_7 \text{ 上既約なので } \mathbb{F}_7 \text{ に } i \text{ を追加すると 2 次拡大体 } \mathbb{F}_{7^2} \text{ になる} \end{aligned}$$

ということです。

ところで「形式的な値を追加する」というのがやや気持ち悪いですね。そこで、もう少し別の表現をしてみます。今 t を変数とする多項式で、その係数が K (K は \mathbb{C} や \mathbb{F}_p など) であるもの全体を $K[t]$ と書くことにします。

$$K[t] := \left\{ f(t) := \sum_i^n a_i t^i \mid a_i \in K, n = 0, 1, 2, \dots \right\}.$$

この多項式を $t^2 + 1$ で割った余り全体の集合を $R := K[t]/(t^2 + 1)$ と書くことにします。2次の多項式で割っているので余りは1次式の集まりです。つまり

$$R := K[t]/(t^2 + 1) = \{ a + bt \mid a, b \in K \}$$

となります。 R の2個の元 $a + bt, c + dt$ をとっても足し算や引き算は

$$(a + bt) \pm (c + dt) := (a \pm c) + (b \pm d)t$$

となります。掛け算をすると2次の項が出ますが $t^2 + 1$ で割った余りを考えているので

$$\begin{aligned} (a + bt)(c + dt) &:= ac + (ad + bc)t + bdt^2 \\ &= ac + (ad + bc)t + bd(t^2 + 1 - 1) \\ &= (ac - bd) + (ad + bc)t \pmod{t^2 + 1} \end{aligned}$$

となります。これらの計算は複素数の計算で i を t に置き換えたものと同じ形をしています。つまり複素数全体は実数係数の多項式全体を $t^2 + 1$ で割った余り $\mathbb{R}[t]/(t^2 + 1)$ とみなせます。これにより「形式的な値 i を追加する」という操作を厳密に表現できます。

一般的には t に関する \mathbb{F}_p 係数の n 次多項式 $f(t)$ が既約なとき、 $\mathbb{F}_p[t]/(f(t))$ は係数が \mathbb{F}_p で高々 $n-1$ 次の多項式の集まりであり、この中で四則演算を定義できます。この集合を \mathbb{F}_{p^n} と書いて \mathbb{F}_p の n 次拡大体といいます。拡大する前の \mathbb{F}_p を素体（そたい）ということがあります。

$$\mathbb{F}_{p^n} := \mathbb{F}[p]/(f(t)) := \left\{ \sum_{i=0}^{n-1} a_i t^i \mid a_i \in \mathbb{F}_p \right\}.$$

拡大体 \mathbb{F}_{p^n} の標数は基礎体 \mathbb{F}_p の標数と同じ p と定義されます。

16.6 拡大体の元の逆元

前節で素数 p に対して \mathbb{F}_p の n 次拡大体 \mathbb{F}_{p^n} を既約多項式 $f(x)$ を用いて定義しました。拡大体 \mathbb{F}_{p^n} の 0 以外の元 $a = a(t)$ ($a(t)$ は高々 $n - 1$ 次の多項式) の逆元の求め方を紹介します。これは 16.1 節で紹介した Euclid の互除法と同様のやり方が使えます。16.1 節の r_0, r_1 の最大公約数を求める手続きを $r_0 := f(t), r_1 := a(t)$ としてまねをします。

$$r_0 := r_1 q_2 + r_2.$$

ここで q_2 と r_2 は t に関する多項式で r_2 は $\deg a(t)$ より小さい次数です。割り算を繰り返すと、次数に関する

$$\deg r_0 > \deg r_1 > \deg r_2 > \cdots \geq 0$$

という自然数の減少ができます。この操作はいつか有限回で止まります。つまり $\deg r_s = 0$ となるある s があり、

$$\gcd(r_0, r_1) = r_s$$

となります。 r_s は定数であり、 $f(t)$ の既約性から 0 ではありません。適当に定数倍することである多項式 $u(t), v(t)$ で

$$a(t)u(t) + f(t)v(t) = 1$$

となるものが見つかったことになります。すると

$$a(t)u(t) \equiv 1 \pmod{f(t)}$$

となり、 \mathbb{F}_{p^n} の中で $u(t)$ が $a(t)$ の逆元です。

16.7 この章のまとめ

有限体上の逆元計算に必要な Euclid の互除法や Fermat の小定理を紹介しました。また有限体と既約多項式を組み合わせることで、その有限体を含む拡大体を構成できました。拡大体の元の逆元も素体の元の逆元と同様、多項式に関する Euclid の互除法を利用することで求められます。

第 17 章

楕円曲線

この章では暗号で使われる楕円曲線の具体的な計算式について解説します。まず 5.3 節で与えたトーラスが Weierstrass（ワイエルシュトラス）の方程式と呼ばれる 2 変数の 3 次方程式で表されることを示します。次に Weierstrass の方程式で表現された楕円曲線に対して具体的な足し算操作を導入し、群になることをみます。更に暗号で使われているいくつかより効率のよい計算式を紹介します。

17.1 トーラス上の関数

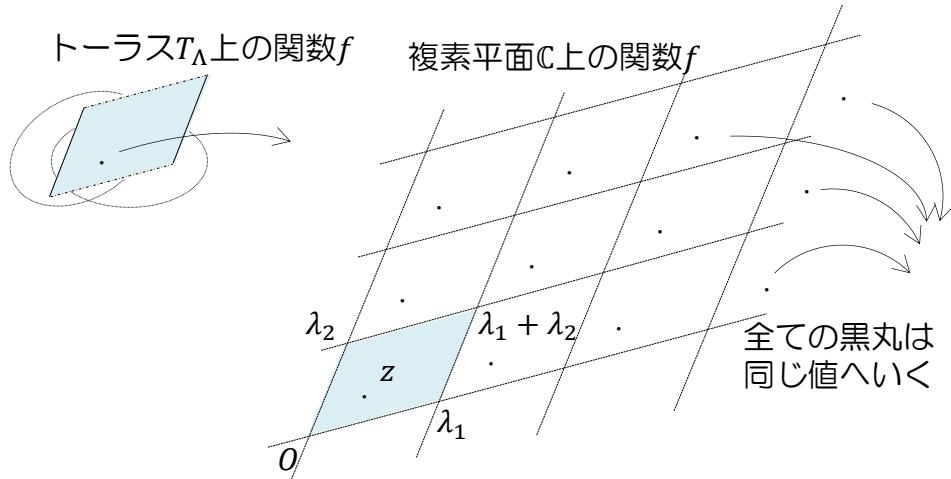
5.3 節で紹介したトーラスは長方形を一つ決めてその両端を張り付けて作りました。複素平面上 \mathbb{C} で長方形（より一般に平行四辺形）は原点 O と 2 点 $\lambda_1, \lambda_2 \in \mathbb{C}$ を決めると決まります。その長方形の頂点は $0, \lambda_1, \lambda_1 + \lambda_2, \lambda_2$ で、向かい合う辺同士を張り合わせて作ったトーラスを T_Λ と書くことにします。

さて平行四辺形を張り合わせるということは、複素平面上に同じ平行四辺形を隙間無く敷きつめたと考えることもできます。平行四辺形の右端の線を越えるとぐるっと回って左端から出たのではなく、コピーされた隣の平行四辺形に飛び込んだと考えるわけです。SF でいうところのパラレルワールドのようですね。ただしここでは本当にオリジナルと同じコピーでそれぞれは区別できないのですが。

ここでトーラス T_Λ 上の関数 f を考えてみましょう。 f を複素平面上の関数と思うと、どのコピーされた平行四辺形の上でも同じ値をとることになります。つまりある z に対して $y := f(z)$ なら $y = f(z + \lambda_1), y = f(z + \lambda_2)$ が成り立ちます。したがって $\Lambda := \{ n\lambda_1 + m\lambda_2 \mid n, m \in \mathbb{Z} \}$ とすると全ての $z \in \mathbb{C}, \lambda \in \Lambda$ に対して

$$f(z + \lambda) = f(z + n\lambda_1 + m\lambda_2) = f(z).$$

図 17.1 で説明すると平行四辺形の中にあるどこの黒丸に対しても同じ値になるということです。

図 17.1 トーラス T_Λ 上の関数

一般に関数 f が、複素数 $T \neq 0$ と任意の z に対して

$$f(z + T) = f(z)$$

となるとき f は周期 T を持つといいます。たとえば三角関数 $\sin(x)$ や $\cos(x)$ は周期 2π を持つ周期関数です。トーラス T_Λ 上の関数は \mathbb{C} 上の λ_1 と λ_2 の 2 種類の周期を持つ周期関数とみなせます。2 種類の周期を持つため 2 重周期関数と呼ばれます。

17.2 \wp 関数

トーラス T_Λ 上の関数、すなわち λ_1, λ_2 を周期に持つ \mathbb{C} 上の 2 重周期関数の例として

$$f(z) := \sum_{\lambda \in \Lambda} \frac{1}{(z - \lambda)^3} = \sum_{n,m \in \mathbb{Z}} \frac{1}{(z - n\lambda_1 - m\lambda_2)^3}$$

があります。無限和で一見複雑です。ただよく見るとなるほどなと思えます。なぜなら $f(z + \lambda_1)$ を考えると n の代わりに $n + 1$ で和をとることになるのですが、全ての n を動くので結局は同じ値になるからです。

$$f(z + \lambda_1) = \sum_{n,m} \frac{1}{((z + \lambda_1) - n\lambda_1 - m\lambda_2)^3} = \sum_{n'=n+1, m} \frac{1}{(z - n'\lambda_1 - m\lambda_2)^3} = f(z).$$

同様に $f(z + \lambda_2) = f(z)$ となるのでこの f は Λ についての 2 重周期関数です。和をとるときに 3 乗分の 1 になっているのは n, m が全ての整数を渡るのでうまく収束するようになります。また $\sum_{n=1}^{\infty} 1/n^2$ は収束しますが、 $\sum_{n=1}^{\infty} 1/n$ は収束しないのです。

ちなみに同様の手法を用いて実数 \mathbb{R} 上の周期関数を作ることもできます。たとえば

$$g(x) := \sum_{n=-\infty}^{\infty} \frac{1}{(x-n)^2}$$

とすると、これは任意の $x \in \mathbb{R} \setminus \mathbb{Z}$ について収束します。そして

$$g(x+1) = \sum_n \frac{1}{(x+1-n)^2} = \sum_n \frac{1}{(x-(n-1))^2} = \sum_{n'} \frac{1}{(x-n')^2} = g(x)$$

が成り立ち、 $g(x)$ は周期 1 の周期関数です。実は

$$\sum_n \frac{1}{(x-n)^2} = \frac{\pi^2}{\sin^2(\pi x)}$$

となることが知られています。

話を戻しましょう。2重周期関数には他には次の形もあります。

$$\wp(z) := \frac{1}{z^2} + \sum_{\lambda:=n\lambda_1+m\lambda_2 \in \Lambda \setminus \{0\}} \left(\frac{1}{(z-\lambda)^2} - \frac{1}{\lambda^2} \right).$$

先ほどと違って 2乗分の 1 で和をとっていますが、括弧の中で引き算をしているのでうまく収束することが示されます。 $n = m = 0$ のときは引き算する値が $1/0$ の形になってしまって取り除いて、その代わりに先頭に $1/z^2$ を足しています。この関数も先ほどと同様に $\wp(z + \lambda_i) = \wp(z)$ を示すことができ、 Λ についての 2重周期関数であることが分かります。この関数は Weierstrass の \wp 関数（ペー関数）と呼ばれ、トーラスにとってとても重要な関数です。 \wp 関数を微分してみましょう。

$$\wp'(z) = -2 \frac{1}{z^3} - \sum_{\lambda \in \Lambda \setminus \{0\}} \frac{2}{(z-\lambda)^3} = -2 \sum_{\lambda \in \Lambda} \frac{1}{(z-\lambda)^3} = -2f(z).$$

先ほど定義した $f(z)$ は $-(1/2)\wp'(z)$ だったのでした。

17.3 \wp 関数の関係式

\wp 関数の性質をもう少し観察してみましょう。 \wp の定義から $\wp(z) - 1/z^2$ は $z = 0$ になります。 $\wp(z) - 1/z^2$ を級数展開します。級数展開とは $z = 0$ の付近で対象となる関数を無限個の z の巾乗の和で表現することです。

$$\wp(z) - 1/z^2 = c_0 + c_1 z + c_2 z^2 + c_3 z^3 + \dots.$$

ここで c_i は z によらないある定数です（実は平行四辺形 Λ が与えられると具体的に計算できます）。今述べたように $z = 0$ で左辺が 0 なので $c_0 = 0$ です。次に \wp 関数は、その定義式の形で z を $-z$ に置き換えて同じ値になるので偶関数です。

$$\wp(-z) = \wp(z).$$

したがって奇数の項 c_{2i+1} は 0 になります.

$$\wp(z) = 1/z^2 + c_2 z^2 + c_4 z^4 + c_6 z^6 + \dots$$

よって \wp 関数を微分すると

$$\wp'(z) = -2/z^3 + 2c_2 z + 4c_4 z^3 + 6c_6 z^5 + \dots$$

となります. $\wp'(z)^2$ と $4\wp(z)^3$ を級数展開したものはどちらも $4/z^6$ で始まります. その差は何になるでしょうか. 展開して少し頑張って計算しましょう.

$$\begin{aligned}\wp(z)^3 &= 1/z^6 + 3c_2/z^2 + 3c_4 + (3c_2^2 + 3c_6)z^2 + \dots, \\ \wp'(z)^2 &= 4/z^6 - 8c_2/z^2 - 16c_4 + (-24c_6 + 4c_2^2)z^2 + \dots.\end{aligned}$$

よって

$$\wp'(z)^2 - 4\wp(z)^3 = -20c_2/z^2 - 28c_4 - (36c_6 + 8c_2^2)z^2 + \dots.$$

$-20c_2/z^2$ の項があるので $20c_2\wp(z)$ を足してみます. すると

$$\wp'(z)^2 - 4\wp(z)^3 + 20c_2\wp(z) = -28c_4 + (12c_2^2 - 36c_6)z^2 + \dots.$$

こうすると右辺は z の負の巾はなくなり z の正の巾の和だけになりました. すると, \wp 関数の周期性と複素解析で習う Liouville の定理（リウビル）を用いると右辺が定数であることが示されます. つまり右辺の z の係数が全て消えます（たとえば $c_6 = c_2^2/3$ が成り立つ）. よって

$$\wp'(z)^2 - 4\wp(z)^3 + 20c_2\wp(z) = -28c_4.$$

辺を移項すると,

$$\wp'(z)^2 = 4\wp(z)^3 - 20c_2\wp(z) - 28c_4. \quad (17.1)$$

これは \wp 関数が満たす非常に重要な関係式です.

$E_\Lambda := \{(x, y) \mid y^2 = 4x^3 - 20c_2x - 28c_4\} \cup \infty$ という集合を考えます. するとトーラス T_Λ 上の点 z に対して,

$$\phi : T_\Lambda \ni z \mapsto (\wp(z), \wp'(z)) \in E_\Lambda \quad (17.2)$$

という写像を考えることができます. $\wp(z)$ と $\wp'(z)$ は式 (17.1) を満たすからです. $z = 0$ は ∞ に対応させます. ここで証明はできませんが, この写像は 1 対 1 の関係があることが示されます. つまりトーラス T_Λ は E_Λ という集合と同じものであることが分かるのです. 2 変数 3 次方程式の解の集合 E_Λ を橢円曲線といいます. 以降, 今まで浮輪の表面という幾何学的な描写をしていた対象物 T_Λ を ϕ を通すことによって方程式の解 E_Λ という代数的なものとして扱えるようになりました.

17.4 Weierstrass の方程式

改めて用語を定義しましょう。 k を複素数体か標数が 2 でも 3 でもない有限体とします。 $4a^3 + 27b^2 \neq 0$ となる $a, b \in k$ を選びます。

$$y^2 = x^3 + ax + b$$

を Weierstrass の方程式あるいは Weierstrass の標準形と呼びます。

前節で紹介したように任意のトーラス E は $\{(x, y) \in k \mid y^2 = x^3 + ax + b\}$ に無限遠点 O を追加した集合として表現できます。グラフを書くときによく使われる 2 次元平面 (x, y) のことをアフィン座標といいます。アフィン座標では無限遠点を表現できません。そのためアフィン座標を拡張した射影座標というものを導入します。射影座標と無限遠点の正確な意味は 17.9 節で説明します。

17.5 判別式

Weierstrass の標準形の定義にある $4a^3 + 27b^2 \neq 0$ という条件は 3 次方程式の解に関わるものです。

補題 5 3 次方程式 $x^3 + ax + b = 0$ が重解を持たない必要十分条件は $4a^3 + 27b^2 \neq 0$ です。この関係式を 3 次方程式の判別式といいます。

なぜなら $x^3 + ax + b = 0$ の 3 個の解を α, β, γ とすると

$$(x - \alpha)(x - \beta)(x - \gamma) = x^3 - (\alpha + \beta + \gamma)x^2 + (\alpha\beta + \beta\gamma + \gamma\alpha)x - \alpha\beta\gamma = x^3 + ax + b = 0.$$

つまり解と係数の関係は

$$\begin{aligned} \alpha + \beta + \gamma &= 0, \\ \alpha\beta + \beta\gamma + \gamma\alpha &= a, \\ \alpha\beta\gamma &= -b. \end{aligned}$$

重解を持たない条件は α, β, γ が全て異なるので $\alpha - \beta \neq 0$ かつ $\beta - \gamma \neq 0$ かつ $\gamma - \alpha \neq 0$ 。式変形しやすいようにそれぞれ 2 乗して掛けると

$$D := (\alpha - \beta)^2(\beta - \gamma)^2(\gamma - \alpha)^2 \neq 0.$$

これが重解を持たない条件です。解と係数の関係を使って D を式変形します。まず一つ目の式から $\alpha + \gamma = -\beta$ 。二つ目の式から $\gamma\alpha = a - \beta(\alpha + \gamma)$ 。これに代入して $\gamma\alpha = a + \beta^2$ 。よって

$$(\alpha - \beta)(\beta - \gamma) = -\beta^2 + (\alpha + \gamma)\beta - \alpha\gamma = -\beta^2 + (-\beta)(\beta) - (a + \beta^2) = -(3\beta^2 + a).$$

同様に $(\beta - \gamma)(\gamma - \alpha) = -(3\gamma^2 + a)$, $(\gamma - \alpha)(\alpha - \beta) = -(3\alpha^2 + a)$. よって

$$\begin{aligned} D &= (\alpha - \beta)(\beta - \gamma) \cdot (\beta - \gamma)(\gamma - \alpha) \cdot (\gamma - \alpha)(\alpha - \beta) = -(3\beta^2 + a)(3\gamma^2 + a)(3\alpha^2 + a) \\ &= - (a^3 + 3(a^2 + \beta^2 + \gamma^2)a^2 + 9((\alpha\beta)^2 + (\beta\gamma)^2 + (\gamma\alpha)^2)a + 27(\alpha\beta\gamma)^2). \end{aligned}$$

ここで

$$\begin{aligned} \alpha^2 + \beta^2 + \gamma^2 &= (\alpha + \beta + \gamma)^2 - 2(\alpha\beta + \beta\gamma + \gamma\alpha) = -2a, \\ (\alpha\beta)^2 + (\beta\gamma)^2 + (\gamma\alpha)^2 &= (\alpha\beta + \beta\gamma + \gamma\alpha)^2 - 2\alpha\beta\gamma(\alpha + \beta + \gamma) = a^2 \end{aligned}$$

を使うと

$$-D = a^3 - 6a^3 + 9a^3 + 27b^2 = 4a^3 + 27b^2.$$

つまり重解を持たない必要十分条件は $4a^3 + 27b^2 \neq 0$. \square

なお, Weierstrass の方程式で制約している標数が 2 でも 3 でもないという条件は緩めることができますが, 式が多少複雑になります. ここでは省略します.

17.6 楕円曲線の加法の図形的な定義

さて, 楕円曲線が暗号にとって重要なのは加法群になるからでした. まず加法群の図形的な定義から説明しましょう.

楕円曲線 E 上に点 O をどこでもよいので（無限遠点でなくてよい）一つ選んで固定します. これが単位元となります. 楕円曲線 E は 3 次曲線なので一般に直線とは 3 点で交わります. E 上の点 P をとり P と O を通る直線 l_1 を考えると, それは E とまた別の点で交わります. その点を点 P の逆元 $-P$ と定義します.

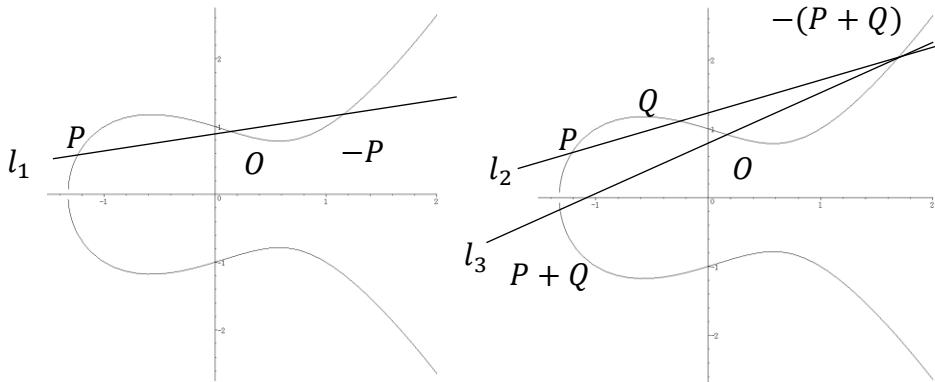


図 17.2 楕円曲線上の逆元と加法

次に E 上の点 P, Q に対しては P と Q を通る直線 l_2 が, E と交わるもう一つの点を $-(P+Q)$ と定義します. すなわち $P+Q$ はその点 $-(P+Q)$ と O を通る直線 l_3 が, E と交わる更にもう

一つの点です。 $P = Q$ のときは E における接線をとります。このようにすると橋円曲線 E は O を単位元とする加法群になります。

O が単位元となるのを確認しましょう。 P と O を通る直線 l_1 が交わる点は $-(P + O)$ です。これは $-P$ と同じ点なので $P + O = P$ です。 l_2 のとり方から交換法則 $P + Q = Q + P$ が成り立つのはすぐ分かります。

結合法則は自明ではありません。図に書きながら様子を見ましょう。

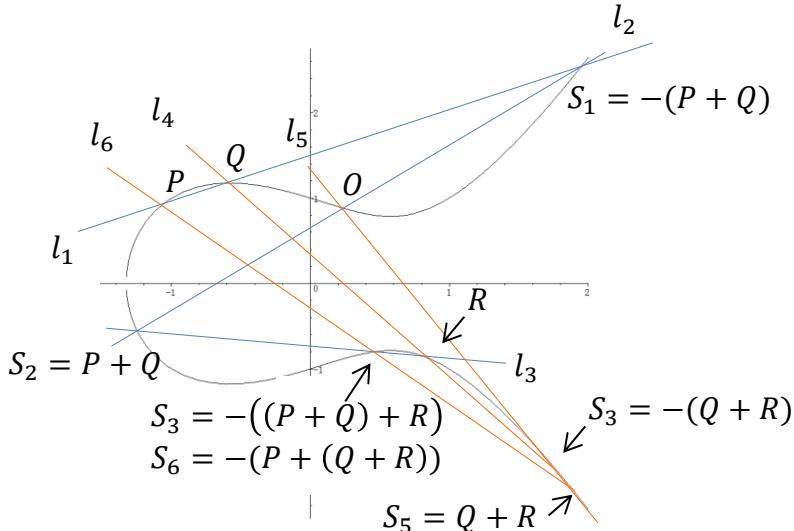


図 17.3 橋円曲線の結合法則

橋円曲線 E 上の点 P, Q, R, O を任意にとります。 P と Q を通る直線 l_1 が E と交わった点は $-(P + Q)$ です。点 $-(P + Q)$ と O を通る直線 l_2 が E と交わった点は $P + Q$ です。それから $P + Q$ と R を通る直線 l_3 が E と交わった点は $S_3 := -((P + Q) + R)$ です。

同様に Q と R を通る直線 l_4 が E と交わった点が $-(Q + R)$ で、 $-(Q + R)$ と O を通る直線 l_5 が E と交わった点が $Q + R$ です。最後に $Q + R$ と P を通る直線 l_6 が E と交わった点が $S_6 := -(P + (Q + R))$ です。結合法則は $S_3 = S_6$ なることです。結合法則は、より一般の 3 次曲線

$$x^3 + c_1x^2y + c_2xy^2 + c_3y^3 + c_4x^2 + c_5xy + c_6y^2 + c_7x + c_8y + c_9 = 0$$

でも成り立ちます。この定理は異なる 2 個の 3 次曲線は 9 個の点で交わるという Bézout (ベズー) の定理と、3 次曲線は 9 個の係数を用いて表されるという事実を組み合わせて証明されます。詳細はたとえば『橋円曲線論入門』(J. H. シルヴァーマン, J. テイト著 足立恒雄他訳) を参照ください。なお、3 次曲線の特殊なケース、曲線の方程式が 3 本の直線 L_1, L_2, L_3 の積に分解できるときは Pappus (パッパス) の 6 角形定理に相当します。

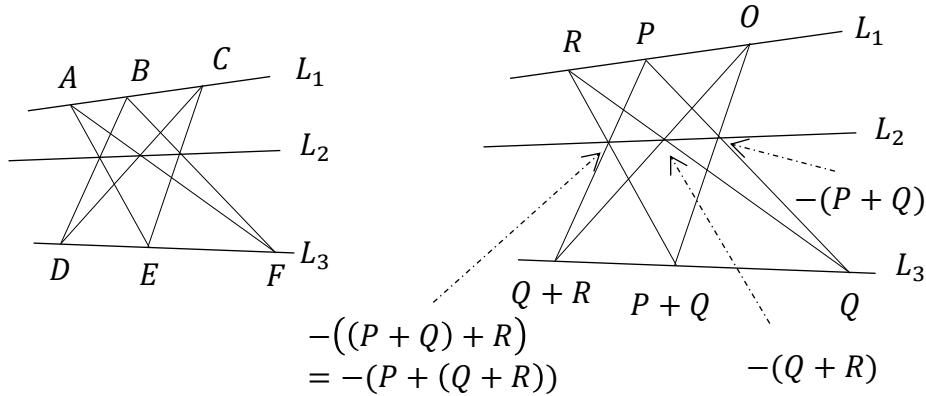
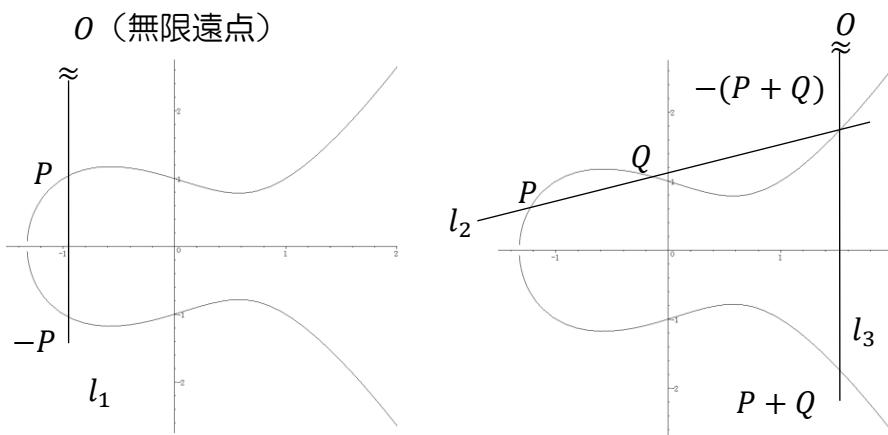


図 17.4 Pappus の定理と結合法則

Pappus の定理は 2 本の直線 L_1, L_3 上に点 A, B, C, D, E, F をとると、線分 AE と BD , AF と CD , BF と CE のそれぞれの交点が同一直線 L_2 上にあるというものです。中学生あたりで習う（かもしれない）Menelaus（メネラウス）の定理を使うと示せます [青空]。 L_1, L_2, L_3 のセットを 3 次曲線とみなすとその曲線上の結合法則の成立は Pappus の定理を意味します [山田]。面白いですね。

17.7 楕円曲線の加法公式

単位元 O は楕円曲線 E 上のどこでもよかったのですが、特に無限遠点にとると計算が容易になります。 $y^2 = x^3 + ax + b$ は x 軸に対して対称なので、点 P と無限遠点 O を通る直線は y 軸に平行な直線となり、 P の逆元は y 座標の符号を反転したものになります。

図 17.5 O が無限遠点にあるときの楕円曲線の結合法則

一般の点 $P = (x_1, y_1)$ と $Q = (x_2, y_2)$ に対して点 $P + Q$ を図形的な定義にしたがって計算しましょう。点 P と Q を通る直線 l_2 の式は

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1 = \lambda(x - x_1) + y_1, \quad \text{ただし } \lambda := \frac{y_2 - y_1}{x_2 - x_1}$$

です。直線 l_2 が点 P, Q 以外に交わる点を求めるために橋円曲線の定義式に代入します。

$$(\lambda(x - x_1) + y_1)^2 = x^3 + ax + b.$$

展開すると

$$x^3 - \lambda^2 x^2 + (a + 2\lambda(\lambda x_1 - y_1))x + b - (\lambda x_1 - y_1)^2 = 0.$$

これが解 $x = x_1, x_2, x_3$ を持つので解と係数の関係で x^2 の係数を見ると

$$x_1 + x_2 + x_3 = \lambda^2.$$

つまり

$$x_3 = \lambda^2 - (x_1 + x_2) = \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - (x_1 + x_2). \quad (17.3)$$

この値を直線の式 l_2 の x に代入し、符号を反転させたものが y 座標になります。

P と Q が同じ点になると、直線 l_2 は P における接線となります。 $y^2 = x^3 + ax + b$ を x で微分すると

$$2yy' = 3x^2 + a.$$

よって $x = x_1$ における傾きは

$$y'|_{x=x_1} = \frac{3x_1^2 + a}{2y_1}.$$

よって

$$\lambda := \frac{3x_1^2 + a}{2y_1}$$

とおくと直線 l_2 の式は $y = \lambda(x - x_1) + y_1$ となり、後は P, Q が異なる 2 点のときと同様にできます。全てまとめると次のようになります。

1. 任意の $P \in E$ に対して $P + O = O + P = P$.
2. $-O := O$. O でない任意の $P := (x, y) \in E$ に対して $-P := (x, -y)$.
3. O でない任意の $P := (x_1, y_1), Q := (x_2, y_2)$ に対して

$$\begin{aligned} x_3 &:= \lambda^2 - (x_1 + x_2), \\ y_3 &:= -\lambda(x_3 - x_1) - y_1. \end{aligned}$$

とすると $P + Q = (x_3, y_3)$. ただし

$$\lambda := \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & (x_1 \neq x_2), \\ \frac{3x_1^2 + a}{2y_1} & (x_1 = x_2). \end{cases}$$

3番目の条件で $x_1 = x_2$ かつ $y_1 = 0$ となるのは $P = Q = (x_1, 0)$ のときです。これは2番目の条件に含まれています。

演算が計算式で与えられたので初等的な方法でも結合法則を示せそうに思えます。ところがこれがとてもやっかいなのです。単に計算すれば求められるはずなのですが、場合分けが多く式も複雑なため一筋縄ではいきません。私も何度か挑戦してみましたが挫折しました。

初等的な方法でがんばって示したものとしては [Fri98] がありましたので興味ある方はごらんください。実はこのテキストも一番ややこしい証明の部分は CoCoA^{*1} という多項式を計算するための数式処理ソフトを使っていました。

17.8 トーラスと橙円曲線の同型対応

前節で導入した橙円曲線の加法公式とトーラス上の足し算を見直します。トーラス上の2点 z_1, z_2 の足し算は z を複素数（あるいは2次元実数ベクトル）とみて素直に足したものでした。

$$z_3 := z_1 + z_2.$$

17.3節の式 (17.2) によるトーラス T_Λ と橙円曲線 E_Λ を繋ぐ写像 ϕ を思い出します。

$$\phi : T_\Lambda \ni z \mapsto (\wp(z), \wp'(z)) \in E_\Lambda \quad (17.4)$$

$P = (x_1, y_1) := \phi(z_1) = (\wp(z_1), \wp'(z_1)), Q = (x_2, y_2) := \phi(z_2) = (\wp(z_2), \wp'(z_2))$ とすると

$$y_i^2 = 4x_i^3 - 20c_2x_i - 28c_4, \quad i = 1, 2$$

を満たします。 x_i^3 の項に 4 がついているのを消去するために $u_i := 2x_i, v_i := \sqrt{2}y_i$ とすると,

$$1/2v_i^2 = 1/2u_i^3 - 10c_2u_i - 28c_4,$$

つまり

$$v_i^2 = u_i^3 - 20c_2u_i - 56c_4.$$

点 P と Q の橙円曲線上での和 $R = (x_3, y_3) := P + Q$ を考えると、加法公式 (17.3) より

$$u_3 = \lambda^2 - (u_1 + u_2) = \left(\frac{v_2 - v_1}{u_2 - u_1} \right)^2 - (u_1 + u_2).$$

x_i, y_i の式に直すと

$$2x_3 = \frac{1}{2} \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - 2(x_1 + x_2).$$

もし z_3 の ϕ による行き先が点 R になるなら $x_3 := \wp(z_3) = \wp(z_1 + z_2)$ なので、

$$\wp(z_1 + z_2) = \frac{1}{4} \left(\frac{\wp'(z_2) - \wp'(z_1)}{\wp(z_2) - \wp(z_1)} \right)^2 - (\wp(z_1) + \wp(z_2)).$$

*1 <http://cocoa.dima.unige.it/WhatIsCoCoA/WhatIsCoCoA-Japanese.html>

これは実際に成り立ち、 ϕ 関数の加法定理と呼ばれます。加法定理は ϕ が

$$\phi(z_1 + z_2) = \phi(z_1) + \phi(z_2)$$

を満たすことを示しています。左側の $+$ は複素数の加算、右側の $+$ は楕円曲線上の点の加算です。つまりトーラス T_Λ と E_Λ は点が対応しているだけでなく、点の加算の構造も対応しているのです。すごいですね。点の足し算がそれぞれの写像の先の足し算に対応しているとき準同型写像というのでした。 ϕ は更に1対1の対応をしているので同型写像といいます。

17.9 射影空間

楕円曲線の定義には無限遠点 O がでてきました。これは直感的には原点からとても遠いところにある点です。無限遠点を定義するために射影空間というものを導入します。射影空間を使うと無限遠点の付近での挙動を扱いやすくなります。

まず実数の射影空間を紹介しましょう。任意の $t \in \mathbb{R}$ に対して原点 $(0, 0)$ を通り、傾き t の直線 $L_t : y = tx$ を考えます。異なる t に対しては異なる L_t が対応するので \mathbb{R} と $\{L_t : t \in \mathbb{R}\}$ は1対1の対応をしています。 t が大きくなるとどんどん y 軸に近寄りますが、同じにはなりません。 y 軸を表す直線 $x = 0$ は L_t の形では表せないからです。直線 $x = 0$ は t が無限大の L_t に相当するといえるでしょう。原点を通る直線の集合を $\mathbb{P}^1(\mathbb{R})$ と書き、実1次元射影空間といいます。

直線 $x = 0$ (傾き無限大)

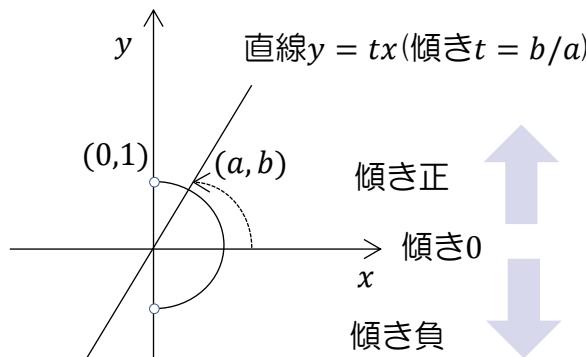


図 17.6 原点を通る直線

原点を通る任意の直線は原点と原点以外の点 (a, b) を通る直線

$$L_{(a,b)} : bx - ay = 0$$

と表せます。 $a \neq 0$ のときは $t := b/a$ とおくと $L_{(a,b)} = L_t$ です。 $a = 0$ のときは $b \neq 0$ なので $L_{(0,b)}$ は直線 $x = 0$ になります。 $L_{(a,b)}$ を使うと L_t と y 軸と同じ形で表わせて都合がよいです。

ただ $L_{(a,b)}$ の表し方は一意ではありません。 $L_{(2a,2b)}$ も $L_{(3a,3b)}$ も同じ直線を表します。傾きが一定であれば同じ直線を表すからです。2本の直線 $L_{(a,b)}$, $L_{(c,d)}$ が同じ直線を表す必要十分条件はある $\alpha \neq 0$ があって

$$\begin{aligned} a &= \alpha c, \\ b &= \alpha d \end{aligned}$$

です。同じ直線を表す (a, b) の組の集合を $(a : b)$ と書くことにすると

$$\mathbb{P}^1(\mathbb{R}) = \{ (a : b) \mid a, b \in \mathbb{R}^2 \setminus \{ (0, 0) \} \}$$

となります。 $(a : b)$ を射影座標とか齊次座標（せいじぎひょう）といいます。 $\mathbb{P}^1(\mathbb{R})$ の元のうち、傾きが有限の直線の集合を $U_1 := \{ (a : b) \mid b \neq 0 \}$ とします。 $\mathbb{P}^1(\mathbb{R}) = U_1 \cup \{ (1 : 0) \}$ です。

$$\begin{array}{ccc} \varphi_1 : & \mathbb{R} & \longrightarrow & U_1 \\ & \Downarrow & & \Downarrow \\ & t & \longmapsto & (t : 1) \end{array}$$

は1対1の対応をしています。 φ_1 を使うと無限遠点以外の値を通常の \mathbb{R} のままで扱えます。

また $U_0 := \{ (a : b) \mid a \neq 0 \}$ とすると $\mathbb{P}^1(\mathbb{R}) = U_0 \cup \{ (0 : 1) \}$ となります。このときは

$$\begin{array}{ccc} \varphi_0 : & \mathbb{R} & \longrightarrow & U_0 \\ & \Downarrow & & \Downarrow \\ & t & \longmapsto & (1 : t) \end{array}$$

が1対1の対応をしていて $t = 0$ は $(1 : 0)$ に対応するので無限遠点の付近を扱えます。

この考え方を2次元空間に適用してみましょう。 k を体として、2次元射影空間を

$$\mathbb{P}^2(k) := \{ (X : Y : Z) \mid (X, Y, Z) \in k^3 \setminus \{ (0, 0, 0) \} \}$$

と定義します。ここで $(X : Y : Z) := \{ (\alpha X, \alpha Y, \alpha Z) \mid \alpha \in k \setminus \{ 0 \} \}$ です。 $Z \neq 0$ となる $\mathbb{P}^2(k)$ の部分集合 $U_2 = \{ (X : Y : Z) \mid Z \neq 0 \}$ を考えます。 k^2 の元 (x, y) に対して $\varphi_2(x, y) := (x : y : 1) \in U_2$ とするとこれは k^2 と U_2 の1対1の対応を与えます。

$$\begin{array}{ccccc} \varphi_2 : & k^2 & \longrightarrow & U_2 & \subset \mathbb{P}^2(k) \\ & \Downarrow & & \Downarrow & \\ & (x, y) & \longmapsto & (x : y : 1) & \\ \varphi_2^{-1} : & (X/Z, Y/Z) & \longleftrightarrow & (X : Y : Z). & \end{array}$$

逆写像は $\varphi_2^{-1}(X : Y : Z) = (X/Z, Y/Z)$ です。

楕円曲線の定義方程式 $y^2 = x^3 + ax + b$ を $\mathbb{P}^2(k)$ の中で考えると、上記 φ_2^{-1} により $(Y/Z)^2 = (X/Z)^3 + a(X/Z) + b$ 、つまり

$$Y^2Z = X^3 + aXZ^2 + bZ^3$$

となります。この式で $Z = 0$ とすると $X^3 = 0$, つまり $X = 0$ 。したがって $(X : Y : Z) = (0 : Y : 0) = (0 : 1 : 0)$ はこの定義方程式をみたします。これが無限遠点 O の正体です。つまり, $\{(x, y) \in k^2 \mid y^2 = x^3 + ax + b\}$ に無限遠点 O を追加した集合は

$$\{(X : Y : Z) \in \mathbb{P}^2(k) \mid Y^2Z = X^3 + aXZ^2 + bZ^3\}$$

という一つの式で与えられることが分かりました。

括弧の中の $Y^2Z = X^3 + aXZ^2 + bZ^3$ という式はどの項も 3 次です。 Y^2Z は Y について 2 次, Z について 1 次, 合計 3 次。 XZ^2, Z^3 も 3 次。このように, この項も同じ次数である多項式を同次多項式, あるいは齊次多項式といいます。 $(X : Y : Z)$ を齊次座標というのはこれが由来です。

17.10 射影座標での加法公式

射影空間のメリットは椭円曲線の無限遠点をきちんと定義するだけではありません。通常のアフィン座標での加法に比べて射影座標での加法は効率がよくなります。コンピュータで椭円曲線暗号を扱うときはたくさんの加法を行うので効率のよい加法公式は重要です。

有限体上の四則計算のうち一番処理が大変なのは除算です。加減算の演算コストは一番軽いです。パラメータや処理するコンピュータに強く依存しますが乗算は足し算の 5 倍から数十倍, 除算は乗算の数十倍から 100 倍以上のコストがかかります。

したがって除算ができるだけ回避できる公式が望ましいのです。アフィン座標で定義された加法公式 (17.3) では直線の傾きを求めるところに除算があります。これを射影座標を使って傾きを比べて表し, 除算をしないように書き直してみましょう。

$(x_1, y_1) := (X_1/Z_1, Y_1/Z_1), (x_2, y_2) := (X_2/Z_2, Y_2/Z_2)$ とします。まず 2 点を通る直線の傾き λ を求めます。2 点が異なるときは $\lambda := (y_2 - y_1)/(x_2 - x_1)$ だったので

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{Y_2/Z_2 - Y_1/Z_1}{X_2/Z_2 - X_1/Z_1} = \frac{Y_2Z_1 - Y_1Z_2}{X_2Z_1 - X_1Z_2}.$$

この分子と分母を $u := Y_2Z_1 - Y_1Z_2, v := X_2Z_1 - X_1Z_2$ とします。2 点 (x_1, y_1) と (x_2, y_2) が通る直線が椭円曲線と通る残りの点を (x_3, y_3) とすると

$$x_3 := \lambda^2 - (x_1 + x_2) = u^2/v^2 - (X_1/Z_1 + X_2/Z_2) = (u/v)^2 - \frac{X_1Z_2 + X_2Z_1}{Z_1Z_2}.$$

$X_1Z_2 + X_2Z_1 = v + 2X_1Z_2$ なので両辺に $v^2Z_1Z_2$ を掛けると

$$x_3(v^2Z_1Z_2) = u^2Z_1Z_2 - v^2(v + 2X_1Z_2).$$

右辺を w とおくと

$$x_3 = \frac{w}{v^2Z_1Z_2}.$$

次に y_3 を考えます.

$$y_3 := -\lambda(x_3 - x_1) - y_1 = -(u/v) \left(\frac{w}{v^2 Z_1 Z_2} - x_1 \right) - y_1.$$

両辺に $v^3 Z_1 Z_2$ を掛けると

$$y_3(v^3 Z_1 Z_2) = -u(w - v^2 Z_1 Z_2(X_1/Z_1)) - v^3 Z_1 Z_2(Y_1/Z_1) = u(v^2 X_1 Z_2 - w) - v^3 Y_1 Z_2.$$

$(x_3, y_3) = (X_3/Z_3, Y_3/Z_3)$ が成り立つように, かつ除算が入らないように X_3, Y_3, Z_3 を決めるとき

$$\begin{aligned} u &:= Y_2 Z_1 - Y_1 Z_2, & v &:= X_2 Z_1 - X_1 Z_2, \\ w &:= u^2 Z_1 Z_2 - v^3 - 2v^2 X_1 Z_2, \\ X_3 &:= v w, \\ Y_3 &:= u(v^2 X_1 Z_2 - w) - v^3 Y_1 Z_2, \\ Z_3 &:= v^3 Z_1 Z_2 \end{aligned}$$

で求められます. 足し算や引き算, 小さな値の定数倍は無視して同じ乗算をしないように, 乗算の回数を数えてみると 14 回必要なことが分かります.

2 倍公式も同様にします. $(x, y) = (X/Z, Y/Z)$ として

$$\lambda = \frac{3x^2 + a}{2y} = \frac{3(X/Z)^2 + a}{(Y/Z)} = \frac{3X^2 + aZ^2}{2YZ}.$$

$u := 3X^2 + aZ^2, v := YZ$ とします. $\lambda = u/(2v)$.

$$x_3 := \lambda^2 - 2x_1 = \frac{u^2}{4Y^2 Z^2} - \frac{2X}{Z} = \frac{u^2 - 8XY^2 Z}{4Y^2 Z^2} = \frac{u^2 - 8XYv}{4v^2}.$$

$w := u^2 - 8XYv$ とおくと $x_3 = w/(4v^2)$. $y_3 = -(u/2v)(w/(4v^2) - X/Z) - Y/Z$ より

$$8v^3 y_3 = -u(w - 4v^2 X/Z) - 8v^3 Y/Z = u(4XY^2 Z - w) - 8Y^4 Z^2 = u(4XYv - w) - 8Y^2 v^2.$$

よって

$$\begin{aligned} u &:= 3X^2 + aZ^2, & v &:= YZ, & w &:= u^2 - 8XYv, \\ X_3 &:= 2vw, \\ Y_3 &:= u(4XYv - w) - 8(Yv)^2, \\ Z_3 &:= 8v^3 \end{aligned}$$

となります. 有限体の乗算を M , 逆元操作を I として表にまとめます.

表 17.1 楕円曲線上の演算のコスト比較

演算の種類	加算 ($P + Q$)	2 倍算 ($2P$)
アフィン座標	$3M + I$	$4M + I$
射影座標	$14M$	$12M$

アフィン座標ではなく射影座標を使ってコストが下がるのは $3M + I \geq 14M$, つまり $I \geq 11M$ のときです. 通常逆元操作は乗算操作より数十倍重たいです. したがって大抵の場合は射影座標を使うとよいことが分かりました.

射影座標を少し変形した $x := X/Z^2$, $y := Y/Z^3$ という座標系を使うこともあります. これは Jacobi 座標と呼ばれ, 足し算は $16M$, 2 倍算は $10M$ で計算できることが知られています. 2 倍算のコストが射影座標より小さいので 2 倍算を多くする場面で利用されることがあります.

17.11 Edwards 曲線

より効率のよい加法公式を持つ楕円曲線の表現がいろいろ研究されています. [LB] では様々な種類の楕円曲線の公式が紹介されています. その中からここでは 2007 年に Edwards により提案された Edwards 曲線を紹介します [BL07]. Edwards 曲線は効率のよい楕円曲線の加法が可能なため注目されています.

k を標数が 2 でない体とし, $c, d \in k$ で $c \neq 0, d \neq 0, dc^4 \neq 1$ となるものをとります.

$$x^2 + y^2 = c^2(1 + dx^2y^2)$$

で定義される曲線を Edwards 曲線といいます. Edwards 曲線は楕円曲線のある特別なグループです.

たとえば $y^2 = x^3 + 4x$ という Weierstrass の方程式で定義された楕円曲線を考えます.

$$x = \frac{2(1+t)}{1-t}, \quad y = \frac{2x}{s} = \frac{4(1+t)}{(1-t)s} \tag{17.5}$$

という変換を考えます. この変換は逆写像を作れます. 一つ目の式を t について解き, 二つ目の式に代入すると

$$t = \frac{x-2}{x+2}, \quad s = \frac{2x}{y}$$

とできるからです. さて $y^2 = x^3 + 4x$ に式 (17.5) を代入して整理すると

$$s^2 + t^2 = 1 - s^2t^2$$

となりました. これは $c = 1, d = -1$ の Edwards 曲線です. $(x, y) \leftrightarrow (s, t)$ は互いに逆写像があるので $y^2 = x^3 + 4x$ は Edwards 曲線の形にできることが分かりました.

17.12 Edwards 曲線の加法公式

Edwards 曲線 $x^2 + y^2 = c^2(1 + dx^2y^2)$ の 2 点 $P = (x_1, y_1)$, $Q = (x_2, y_2)$ が与えられたとき, 点 $R := P + Q = (x_3, y_3)$ は

$$(x_3, y_3) := \left(\frac{x_1y_2 + y_1x_2}{c(1 + dx_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{c(1 - dx_1x_2y_1y_2)} \right)$$

で与えられます。単位元は $O := (0, c)$ で点 $P := (x_1, y_1)$ の逆元は $-P := (-x_1, y_1)$ です。Weierstrass の方程式上で提案された様々な公式と一番異なるのは加法と 2 倍算の場合分けが不要なことです。これにはいくつかのメリットがあります。

まず当たり前ですが $P + Q$ で $P = Q$ かどうかを気にせずにプログラムのコードを記述できます。従来の楕円曲線では加法と 2 倍算の公式が異なるため、通常その実行回路や演算速度が異なります。するとたとえば楕円曲線暗号が組み込まれた IC カードに電極を挿して消費電力やタイミングを測定することで加法と 2 倍算の違いから秘密鍵を推測できることがありました。これはサイドチャネル攻撃 (side-channel attack) と呼ばれるものの一つです。Edwards 曲線はそれに対して耐性があります。

次に 3 倍点の公式を作ることができます。そうするとより効率のよいスカラーベ倍算を構築できることが知られています。加法と 2 倍算の公式が異なると 3 倍点の公式は場合分けが複雑になり、そのメリットがでません。

またそもそもその目的の一つですが射影座標で計算すると加法は $12M$ ($c = 1$ なら $11M$)、2 倍算は $7M$ でできることが知られています。

17.13 この章のまとめ

浮輪の形で定義したトーラスは二重周期関数である \wp 関数を利用すると $y^2 = x^3 + ax + b$ で定義される楕円曲線と対応がつくことが分かりました。楕円曲線上の点の加法は楕円曲線と直線の交点で表現できます。これにより暗号で利用できる計算手順を書き下せます。

楕円曲線に登場する無限遠点を統一的に扱うために射影空間を導入しました。そして楕円曲線上の点の加法は、射影座標を用いるとより高速に計算できることが分かりました。楕円曲線の演算性能は非常に重要なことで様々な手法が提案されています。

第 18 章

ペアリング

この章ではペアリングの定義と具体的な計算式について説明します。少々準備が多いのですが一つずつ確認していきましょう。

18.1 関数の零点と極

複素数体 \mathbb{C} 上の 0 でない関数 $f(x)$ を考えます。 $f(x) = 0$ となる $x = a$ を f の零点 (zero) といいます。 $a_i \in \mathbb{C}$ ($i = 0, 1, \dots$) を定数として

$$f(x) = \sum_{i=0}^{\infty} a_i(x - a)^i = a_0 + a_1(x - a) + a_2(x - a)^2 + \dots$$

と書けるとします (f の $x = a$ における Laurent (ローラン) 級数といいます)。 $f(a) = 0$ なので $a_0 = 0$ です。 $a_i \neq 0$ となる最小の i を f の $x = a$ における位数といい

$$\text{ord}_a(f) := i$$

と書きます。たとえば $f(x) = x^3$ のとき、零点は $x = 0$ で $\text{ord}_0(x^3) = 3$ です。

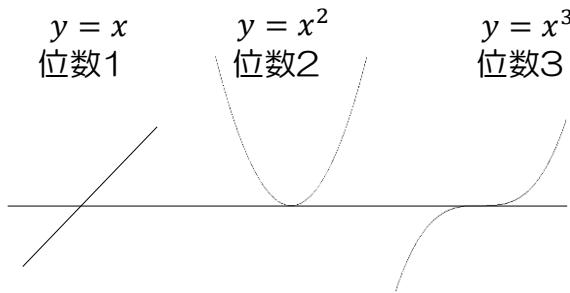


図 18.1 関数の零点とその位数

位数はその点で関数 f が x 軸とどれぐらいで接しているかを表しています。直線 $y = x$ と直線

$y = 0$ は原点 $(0, 0)$ で交差しています。関数 $y = x^n$ は x 軸と $(0, 0)$ で n 重に接しているといいます。 $f(x) = x(x+1)(x-1)^2$ だと零点は $x = 0, -1, 1$ の 3 個です。 $x = 0$ での位数は

$$f(x) = x - x^2 - x^3 + x^4$$

なので $\text{ord}_0(f) = 1$ 。 $x = 1$ での位数は

$$f(x) = 2(x-1)^2 + 3(x-1)^3 + (x-1)^4$$

と変形できるので $\text{ord}_1(f) = 2$ 。 $x = -1$ での位数は

$$f(x) = -4(x+1) + 8(x+1)^2 - 5(x+1)^3 + (x+1)^4$$

と変形できるので $\text{ord}_{-1}(f) = 1$ です。いちいち式変形しなくても、たとえば $x = 1$ に十分近いところでは $x(x+1)$ は 2 にとても近いので

$$f(x) \approx 2(x-1)^2$$

と考えると $\text{ord}_1(f) = 2$ だと分かります。一般に $x = a$ の付近で $f(x) \approx c(x-a)^n$ (c は定数) の形になるなら $\text{ord}_a(f) = n$ となります。

もし $f(z) \approx c(z-a)^n$ の n が負ならどうなっているのでしょうか。 $z \rightarrow a$ で $f(z)$ は無限大になります。このとき $z = a$ は f の極 (pole) といい、その位数を零点と同じく $\text{ord}_a(f) := n$ とします。より正確には、関数 $f(z)$ が $z = a$ で $n (> 0)$ 位の極を持つのは $a_i \in \mathbb{C}$ ($i = -n, -n+1, \dots$) を定数、ただし $a_{-n} \neq 0$ として

$$f(z) = \sum_{i=-n}^{\infty} a_i (z-a)^i = \frac{a_{-n}}{(z-a)^n} + \frac{a_{-n+1}}{(z-a)^{n-1}} + \dots$$

と書けるときをいいます。こうすると零点と極の位数の定義が同じ式で書けます。

関数 f が $x = a$ で零点でも極でもない普通の値 $a_0 \neq 0$ をとるときは $\text{ord}_a(f) := 0$ と定義しましょう。するとこのときも同じ式で表せます。

関数の中にはたとえば $f(z) = e^{1/z} = \sum_n 1/n!(1/z)^n$ のような $z = 0$ での位数がマイナス無限大になってしまふようなものもあります。そのようなものはここでは考えません。関数 $e^{1/z}$ における $z = 0$ は真性特異点といわれます。

18.2 位数の性質

関数の零点や極の位数の性質を調べましょう。まず関数 f が $z = a$ で位数 n の零点を持つなら関数 $1/f$ は $z = a$ で位数 n の極を持ちます。

$$\text{ord}_a(f) = -\text{ord}_a(1/f).$$

また2個の関数 f, g の点 a における位数を n, m とします。つまり

$$f(z) := \sum_{i=n} f_i(z-a)^i, \quad g(z) := \sum_{i=m} g_i(z-a)^i.$$

すると2個の関数の積 fg は $f_n g_m (z-a)^{n+m} \neq 0$ から始まります。つまり

$$\text{ord}_a(fg) = \text{ord}_a(f) + \text{ord}_a(g)$$

が成り立ちます。

2個の関数の和の位数はどうなるでしょう。 n と m が異なれば小さい方の位数になります。たとえば $f(x) = x^3, g(x) = x^5$ なら $(f+g)(x) = x^3 + x^5$ なので $\text{ord}_0(f+g) = 3$ 。

位数が同じときは係数がキャンセルしてもっと大きい値の位数になることがあります。たとえば $f(x) = x^3 + x^4, g(x) = -x^3 + x^5$ なら $\text{ord}_0(f+g) = \text{ord}_0(x^4 + x^5) = 4$ 。つまり一般に

$$\text{ord}_a(f+g) \geq \min(\text{ord}_a(f), \text{ord}_a(g))$$

が成り立ちます。

よってたとえば点 $z = a$ で位数 n 以上の零点（あるいは極）を持つ関数の集合 V を考えると、 V の2個の元を足したり引いたりしても V の元になることが分かります。ここで定数関数 $f = 0$ はどの点でも位数無限大と解釈することにして、 V の元とみなします。すると V は $f = 0$ を単位元、関数の足し算を演算とする加法群になります。関数を定数 ($\neq 0$) 倍しても、その位数は変わらないので V は \mathbb{C} ベクトル空間にもなります。

18.3 楕円曲線上の関数の零点と位数の例

前節で定義した関数の零点や極の位数は楕円曲線上の関数に対しても考えることができます。定義方程式 $y^2 = x^3 + ax + b = (x - e_1)(x - e_2)(x - e_3)$ (e_i は互いに異なる) の楕円曲線を E とします。

$$E \ni P = (x, y) \mapsto y$$

という関数を考えます。 E 上の点 $P = (x, y)$ に対してその y 座標を返す関数です。本来なら f_1 といった名前をつけるところですが、 y という名前にします。座標と関数を同じ記号で書くのは分かりにくいかもしれません、慣れると自然に扱えます。

関数 y の値が 0 になるのは $P_i := (e_i, 0)$ ($i = 1, 2, 3$) のときのみです。点 P_1 の付近での関数 y の挙動を観察するために、 $s := x - e_1$ としましょう。すると

$$y^2 = ((s + e_1) - e_1)((s + e_1) - e_2)((s + e_1) - e_3) = s(s + e_1 - e_2)(s + e_1 - e_3)$$

です。 $e_1 - e_2, e_1 - e_3 \neq 0$ で s が 0 にとても近いときは $y^2 = cs$ とみなせます。ここで c は $c := (e_1 - e_2)(e_1 - e_3) \neq 0$ という定数です。つまり点 P_1 の付近で楕円曲線 E の方程式は

$y^2 = c(x - e_1)$ と近似でき、非常に小さいパラメータ t を用いて $P_t := (t^2/c + e_1, t) \in E$ と表現できます。このとき y 座標の値を返す関数 y は t そのものになります。

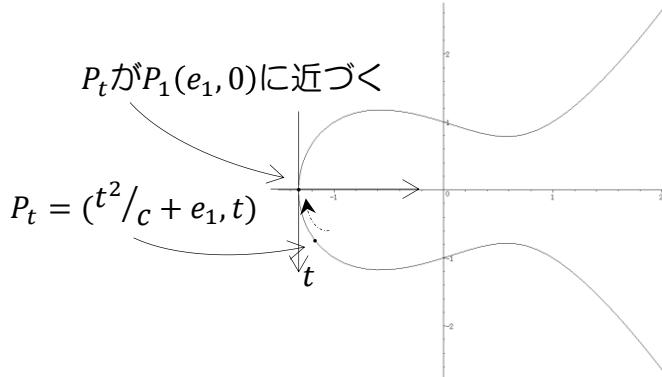


図 18.2 点 P_1 付近でのパラメータ t による表示

P_1 の付近では $t = 0$ なのでその位数は 1 です。 P_2, P_3 についてもそれぞれの点の付近でパラメータをとって考えると同様に位数は 1 です。よって $\text{ord}_{P_i}(y) = 1$ 。

関数 y に極はあるのでしょうか。橿円曲線 E は無限遠点 O を持っているのでした。 O は y が無限大になったときとみなせるので O が y の極になります。その位数はいくつでしょう。 y が大きくなると x も多くなり、

$$y^2 = x^3 + ax + b = x^3 \left(1 + \frac{a}{x^2} + \frac{b}{x^3}\right) \approx x^3$$

と近似できます。つまり E は O の付近でパラメータ t （先程の t とは別物）を用いて (t^2, t^3) と表現できます。関数 y は $y = t^3$ となるので $\text{ord}_O(y) = -3$ です。

もう一例あげておきましょう。

$$E \ni P = (x, y) \mapsto x - e_i$$

を考えます。関数 y と同じく $x - e_i$ を関数表記でも使います。 $x - e_i$ が 0 になるのは $P = P_i$ のときです。点 P_1 の付近で E 上の点を $(t^2/c + e_1, t)$ と表現すると、 $x - e_1 = t^2/c$ となります。よって $t = 0$ の付近を考えると $\text{ord}_{P_i}(x - e_i) = 2$ 。極は関数 y と同じく無限遠点 O です。 O の付近では大きな t を用いて (t^2, t^3) と表現できたので $\text{ord}_O(x - e_i) = -2$ となります。

18.4 因子

これから関数とその零点と極の関係を見るために因子（divisor）という概念を導入します。

楕円曲線 E 上の因子 D とは E 上の有限個の点 $P_1, P_2, \dots \in E$ の整数個 ($n_1, n_2, \dots \in \mathbb{Z}$) の形式的な有限和

$$D := \sum_i n_i(P_i)$$

のことです。たとえば $2(P)$ とか $3(Q) - 2(O)$ などが因子です。 $2P$ や $3Q - 2O$ と書いてしまうと、楕円曲線上の点の加法と間違えてしまうので点には括弧をつけています。

2 個の因子 $D_1 = \sum_i n_i(P_i)$ と $D_2 := \sum_i m_i(Q_i)$ があったとき、因子の和 $D_1 + D_2$ を

$$D_1 + D_2 := \sum_i n_i(P_i) + \sum_i m_i(Q_i)$$

と定義します。単に形式的に 2 個の因子の要素を並べただけです。 P_i や Q_i に同じ点がある場合はその係数を足すことにします。たとえば $2(P) + 3(P) = 5(P)$. $(3(Q) - 2(O)) + (4(O) - (P)) = -(P) + 3(Q) + 2(O)$ となります。

和が空っぽのときの因子を $D = 0$ と書くと、因子全体の集合は 0 を単位元、上記の因子の和を演算として加法群になります。因子 $D = \sum_i n_i(P_i)$ の逆元 $-D$ は $\sum_i -n_i(P_i)$ ですね。

ある関数 f が点 P_i で位数 n_i の零点、点 Q_j で位数 m_j の極を持っているとき、 f に対応する因子 $\text{div}(f)$ を

$$\text{div}(f) := \sum_i n_i(P_i) - \sum_j m_j(Q_j)$$

と定義します。 $\text{div}(f)$ を主因子 (principal divisor) といいます。たとえば前節の楕円曲線 E 上の関数 y や $x - e_i$ に対応する因子は

$$\begin{aligned} \text{div}(y) &= (P_1) + (P_2) + (P_3) - 3(O), \\ \text{div}(x - e_i) &= 2(P_i) - 2(O) \end{aligned}$$

です。

18.5 楕円曲線上の関数の主因子の性質

さて、楕円曲線上の関数の主因子には著しい性質があります。

定理 6 f を楕円曲線上の 0 でない関数、その主因子を $\text{div}(f) = \sum_i n_i(P_i)$ とすると、

$$\begin{aligned} \sum_i n_i &= 0, \\ \sum_i n_i P_i &= O \end{aligned}$$

が成り立つ。逆に、ある因子 $D = \sum_i n_i(P_i)$ がこの 2 個の等式を満たすとき、 $\text{div}(f) = D$ となる f が定数倍を除いてただ一つ存在する。特に $D = 0$ という因子を考えると零点も極も持たない関数は 0 でない定数関数に限る。

ここで一つ目の式の和は f の零点や極の位数の和、つまり整数の足し算です。一般に因子 $D = \sum_i n_i(P_i)$ に対してその次数を $\deg(D) := \sum_i n_i$ と定義します。一つ目の式は関数 f に対して

$$\deg(\text{div}(f)) = 0$$

を意味します。

二つ目の和は楕円曲線上の点の加法による和です。これは関数の零点や極が n 個あるとき、 $n - 1$ 個の点の位置を決めると、残りの点の位置が決まってしまうことを意味しています。零点や極という局所的な性質が、関数の全体の形を決めてしまうのは興味深いです。残念ながらここでは準備が多すぎてこの定理の証明はできません。

それでいくつかの例についてこの定理の前半が成り立っていることを確認するのに留めます。関数 y に対して $\text{div}(y) = (P_1) + (P_2) + (P_3) - 3(O)$ でした。よって

$$\deg(\text{div}(y)) = 1 + 1 + 1 - 3 = 0.$$

点 P_i は楕円曲線 E と直線 $y = 0$ との交点なので、楕円曲線の加法の定義から $P_1 + P_2 = -P_3$ 。よって

$$P_1 + P_2 + P_3 - 3O = O$$

となり二つ目の式も成り立っています。

また関数 $x - e_i$ についても $\text{div}(x - e_i) = 2(P_i) - 2(O)$ でした。よって

$$\deg(\text{div}(x - e_i)) = 2 - 2 = 0.$$

点 P_i は楕円曲線 E と直線 $x = e_i$ との接線なので $2P_i = 0$ 。よって

$$2P_i - 2O = O$$

より定理が成り立っていることを確認できます。

もう一つ、定理を紹介しましょう。まず記号の定義をします。関数 f と因子 $D = \sum_i n_i(P_i)$ に対して

$$f(D) := \prod_i f(P_i)^{n_i}$$

とします。たとえば $f(x) = x^2$, $D = (3) + 2(-1)$ なら $f(D) = f(3)f(-1)^2 = 9 \cdot 1^2 = 9$ です。

定理 7 (Weil 相互法則 (Weil reciprocity law)) 楕円曲線上の関数 f, g について

$$f(\text{div}(g)) = g(\text{div}(f))$$

が成り立つ。

ここで一般の証明はできませんが \mathbb{R} 上の单なる多項式のときに観察します。 $f(x) := \prod_i (x - a_i)^{n_i}$, $g(x) := \prod_j (x - b_j)^{m_j}$ とします。ただし $x \rightarrow \infty$ で発散しないように $\sum_i n_i = \sum_j m_j = 0$ とします。すると、

$$\operatorname{div}(f) = \sum_i n_i(a_i), \quad \operatorname{div}(g) = \sum_j m_j(b_j)$$

となります。よって

$$\begin{aligned} f(\operatorname{div}(g)) &= \prod_j f(b_j)^{m_j} = \prod_j \left(\prod_i (b_j - a_i)^{n_i} \right)^{m_j} = \prod_{i,j} (b_j - a_i)^{n_i m_j} \\ &= \left(\prod_{i,j} (a_i - b_j)^{n_i m_j} \right) \prod_{i,j} (-1)^{n_i m_j} \\ &= g(\operatorname{div}(f))(-1)^{\sum_{i,j} n_i m_j} = g(\operatorname{div}(f))(-1)^{(\sum_i n_i)(\sum_j m_j)} = g(\operatorname{div}(f)). \end{aligned}$$

18.6 ペアリング

ようやくペアリングの定義に関する準備が整いました。

定義 8 横円曲線 E と素数 m に対して $E[m] := \{ P \in E \mid mP = O \}$ とします。 $E[m]$ の点を m 等分点といいます。 m 等分点は m^2 個あります。

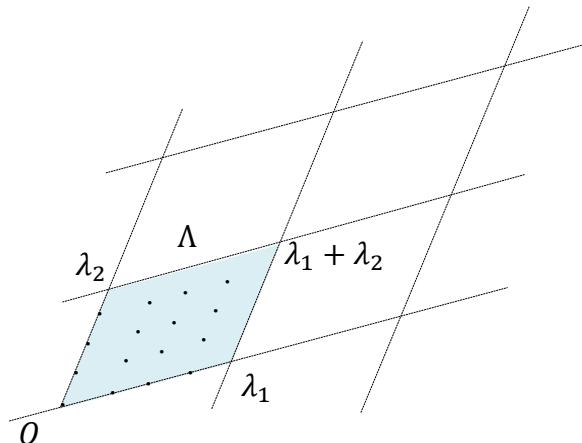


図 18.3 複素トーラスにおける 4 等分点の集合 $E[4]$

$E[m]$ の 2 点 P, Q をとり因子 $D_P := m(P) - m(O)$, $D_Q := m(Q) - m(O)$ を考えます。 $\deg(D_P) = \deg(D_Q) = 0$ かつ $mP = mQ = O$ なので定理 6 より、ある関数 f_P, f_Q で

$\text{div}(f_P) = D_P$, $\text{div}(f_Q) = D_Q$ となるものが存在します。適当な $S \in E[m]$ に対して,

$$e(P, Q) := \frac{f_P(Q + S)}{f_P(S)} \cdot \frac{f_Q(-S)}{f_Q(P - S)}$$

をペアリングと呼びます¹。

定理 9 $e(P, Q)$ は f_P, f_Q, S のとり方に依らずに定まり, P, Q に関して双線形写像となる。

特に f_P や f_Q を一番次数の大きい項の係数を 1 になるようにとる（これを正規化といいます）と, $S \rightarrow O$ とすると $f_Q(-S)/f_P(S) \rightarrow 1$ となり,

$$e(P, Q) = f_P(Q)/f_Q(P)$$

と書けます。また、この式の対称性から $e(Q, P) = 1/e(P, Q)$ も分かります。

まず f_P のとり方に依存しないことを確認しましょう。定理 6 より与えられた D_P に対して $\text{div}(f) = D_P$ となる f は定数倍の差しか違いません。 $f_P(Q + S)/f_P(S)$ と比をとっている部分で定数倍はキャンセルします。したがって f_P のとり方に依存しません。

次に S のとり方に依存しないことを確認します。点 P, Q と関数 f_P, f_Q を固定して関数 F を

$$F(S) := e(P, Q) = \frac{f_P(Q + S)}{f_P(S)} \cdot \frac{f_Q(-S)}{f_Q(P - S)}$$

と定義します。示したいことは $F(S)$ が S に関して定数関数であることです。

そのために $F(S)$ が零点も極も持たない関数であることを示します。そうすれば定理 6 より定数関数になるからです。

関数 f_P は $\text{div}(f_P) = m(P) - m(O)$ となるようにとりました。 S を動かすとき $f_P(Q + S)$ は f_P を Q だけ平行移動したものです。つまり $f_P(X)$ が $X = P$ で 0 となるなら $f_P(Q + S)$ は $S = P - Q$ で 0 になります。よって

$$\text{div}(f_P(Q + S)) = m(P - Q) - m(-Q).$$

同様に $\text{div}(f_Q(-S)) = m(-Q) - m(O)$, $\text{div}(f_Q(P - S)) = m(P - Q) - m(P)$ となります。したがって

$$\text{div}(F) = (m(P - Q) - m(-Q)) - (m(P) - m(O)) + (m(-Q) - m(O)) - (m(P - Q) - m(P)) = 0$$

となります。 F は零点も極も持たないので定数であることが分かりました。

¹ この定義と証明は [Aft11] を参考にしました。ただこのテキストで定義されている Weierstrass の σ 関数は間違っているのでご注意ください。正しくは

$$\sigma(z) = z \prod_{\omega \in \Lambda \setminus 0} \left(1 - \frac{z}{w}\right) \exp\left(\left(\frac{z}{w}\right) + \frac{1}{2} \left(\frac{z}{w}\right)^2\right).$$

次に双線形性を示しましょう。点 $P_1, P_2 \in E[m]$ をとり $e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)$ を示せば十分です。この示したい式を定義にしたがって展開すると

$$\frac{f_{P_1+P_2}(Q+S)}{f_{P_1+P_2}(S)} \frac{f_Q(-S)}{f_Q(P_1+P_2-S)} = \frac{f_{P_1}(Q+S)}{f_{P_1}(S)} \frac{f_Q(-S)}{f_Q(P_1-S)} \frac{f_{P_2}(Q+S)}{f_{P_2}(S)} \frac{f_Q(-S)}{f_Q(P_2-S)}.$$

$$f(X) := \frac{f_{P_1+P_2}(X)}{f_{P_1}(X)f_{P_2}(X)}$$

とおいて式変形すると、

$$\frac{f(Q+S)}{f(S)} = \frac{f_Q(P_1+P_2-S)f_Q(-S)}{f_Q(P_1-S)f_Q(P_2-S)}. \quad (18.1)$$

今から左辺が右辺に等しいことを示します。まず、

$$\begin{aligned} \text{div}(f) &= \text{div}(f_{P_1+P_2}) - \text{div}(f_{P_1}) - \text{div}(f_{P_2}) \\ &= m(P_1+P_2) - m(O) - m(P_1) + m(O) - m(P_2) + m(O) \\ &= m((P_1+P_2) - (P_1) - (P_2) + (O)). \end{aligned}$$

因子 $D := (P_1+P_2) - (P_1) - (P_2) + (O)$ は $\deg(D) = 1 - 1 - 1 + 1 = 0$, $P_1+P_2 - P_1 - P_2 + O = O$ と定理 6 の条件を満たすのである関数 g が存在して

$$\text{div}(g) = D$$

と書けます。すると

$$\text{div}(f) = m \text{div}(g) = \text{div}(g^m)$$

となり、 f と g^m の因子が同じなので定数倍を除いて $f = g^m$ と書けます。すると、式 (18.1) の左辺は

$$\begin{aligned} \frac{g^m(Q+S)}{g^m(S)} &= g(m(Q+S) - m(S)) \\ \text{ここで } h(X) &:= f_Q(X-S) \text{ とすると } \text{div}(h) = m(Q+S) - m(S) \text{ なので} \\ &= g(\text{div}(h)) \quad \text{ここで定理 7 を使うと} \\ &= h(\text{div}(g)) \\ &= h((P_1+P_2) - (P_1) - (P_2) + (O)) = \frac{h(P_1+P_2)h(O)}{h(P_1)h(P_2)}. \end{aligned}$$

h の定義に戻るとこれは式 (18.1) の右辺に等しく証明が終わりました。 \square

18.7 Miller のアルゴリズム

前節によりペアリングの定義はできました。この節では Miller (ミラー) のアルゴリズムと呼ばれる具体的なペアリングの計算方法を紹介します。

橭円曲線 E と素数 m に対して点 $P \in E[m]$ をとります。ペアリングの計算に必要な $\text{div}(f_P) = m(P) - m(O)$ となる関数 f_P を構成しましょう。いきなり構成するのは難しいので m が小さいところから作っていきます。そのために関数 $f_P^{(i)}$ で

$$\text{div}(f_P^{(i)}) = i(P) - (iP) - (i-1)(O)$$

となるものを構成します。記号が紛らわしいですが $i(P)$ は因子 (P) の i 倍、 (iP) は点 P の i 倍点からなる因子です。 $mP = O$ なら、

$$\text{div}(f_P^{(m)}) = m(P) - (mP) - (m-1)(O) = m(P) - (O) - (m-1)(O) = m(P) - m(O)$$

となり $f_P^{(m)}$ が求めたい関数 f_P です。

小さい i から順に $f_P^{(i)}$ を構成するために橭円曲線上の点の加算と直線の関係を見直します。

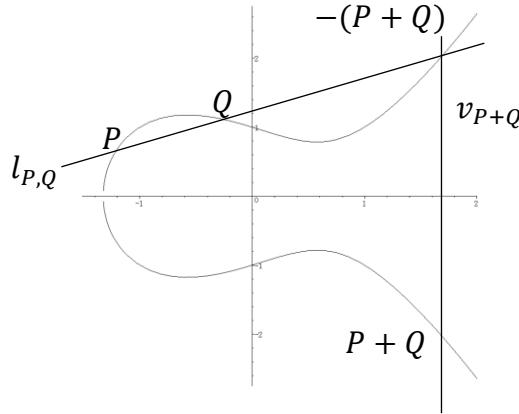


図 18.4 点 P と Q を通る直線および $P + Q$ と $-(P + Q)$ を通る直線

17.7 節で紹介したように点 $P = (x_1, y_1)$, $Q = (x_2, y_2)$ を通る直線は

$$y = \lambda(x - x_1) + y_1, \quad \text{ただし } \lambda := \frac{y_2 - y_1}{x_2 - x_1}$$

と書けました ($P = Q$ のときは接線でした)。 E 上の関数 $l_{P,Q}$ を

$$l_{P,Q}(x, y) := y - (\lambda(x - x_1) + y_1)$$

とします（関数 y の係数は 1 に正規化しています）。直線 $l_{P,Q}(x, y) = 0$ は点 $P, Q, -(P + Q)$ で E と交わるので $l_{P,Q}$ はそれらを零点を持ちます。無限遠点では y の効果がでて位数 3 の極になります。つまり

$$\text{div}(l_{P,Q}) = (P) + (Q) + (-P - Q) - 3(O).$$

同様に勝手な点 $R = (x_R, y_R)$ に対して関数 v_R を

$$v_R(x, y) := x - x_R$$

とすると直線 $v_R(x, y) = 0$ は点 $R, -R$ で交わり、無限遠点では位数 2 の極を持つので

$$\text{div}(v_R) = (R) + (-R) - 2(O).$$

関数 $l_{P,Q}$ も関数 v_R も点 P, Q, R が与えられれば具体的に計算できます。点 P, Q に対して

$$g_{P,Q} := l_{P,Q}/v_{P+Q}$$

とすると、

$$\begin{aligned} \text{div}(g_{P,Q}) &= \text{div}(l_{P,Q}) - \text{div}(v_{P+Q}) \\ &= (P) + (Q) + (-P - Q) - 3(O) - (P + Q) - (-P - Q) + 2(O) \\ &= (P) + (Q) - (P + Q) - (O). \end{aligned}$$

i, j を自然数として関数 $f_P^{(i)} f_P^{(j)} g_{iP,jP}$ を考えると

$$\begin{aligned} \text{div}(f_P^{(i)} f_P^{(j)} g_{iP,jP}) &= (i(P) - (iP) - (i-1)(O)) + (j(P) - (jP) - (j-1)(O)) \\ &\quad + ((iP) + (jP) - (iP + jP) - (O)) \\ &= (i+j)(P) - ((i+j)P) - (i+j-1)(O). \end{aligned}$$

これより

$$f_P^{(i+j)} := f_P^{(i)} f_P^{(j)} g_{iP,jP}$$

とできます。この関係式を使うと小さい i から順に $f_P^{(i)}$ を構成できます。

まず $\text{div}(f_P^{(1)}) = (P) - (P) - 0(O) = 0$ ですから $f_P^{(1)} := 1$ とします。すると

$$\begin{aligned} f_P^{(2)} &= f_P^{(1)} f_P^{(1)} g_{P,P} = g_{P,P}, \\ f_P^{(3)} &= f_P^{(1)} f_P^{(2)} g_{P,2P} = f_P^{(2)} g_{P,2P}, \\ f_P^{(4)} &= f_P^{(2)} f_P^{(2)} g_{2P,2P} = \left(f_P^{(2)}\right)^2 g_{2P,2P}. \\ f_P^{(5)} &= f_P^{(1)} f_P^{(4)} g_{P,4P} := f_P^{(4)} g_{P,4P}. \end{aligned}$$

このように小さい i から $2^i P$ と $f_P^{(i)}$ を順次求めることで $i = m$ まで持っていきます。2 倍しながら求めるので 3.8 節の巾乗を求めるバイナリ法と同様の演算量 $O(\log(m))$ です。この方法を Miller のアルゴリズムといいます。

楕円曲線全般の定番のテキストは『The Arithmetic of Elliptic Curves』(J. H. Silverman) です。国内の和書では『暗号理論と楕円曲線』(辻井重男, 笠原正雄編) が詳しいです。Miller のアルゴリズムも紹介されています。ペアリングを高速に計算することは新しい暗号方式にとっての非常に重要です。Tate ペアリング [岡本 11], optimal Ate ペアリングなど様々な高速手法が提案されています。私は 2014 年の時点で世界最速な実装の一つを <https://github.com/herumi/ate-pairing/> で公開しています。実装方法については、またの機会に紹介しましょう。

18.8 この章のまとめ

楕円曲線上の点の有限形式和という因子を定義しました。楕円曲線上の関数の零点と極の位数の関係は因子を用いて記述できます。ペアリングはある因子から定まる楕円曲線上の関数を用いて定義されます。ペアリングは Miller アルゴリズムを用いると比較的高速に計算でき、様々な高速化手法が提案されています。

近年、ペアリングを使った暗号の標準化が進んでいます。パラメータが整備され、使いやすいライブラリが出てくれれば少しずつ実験やアプリケーションも作りやすくなるでしょう。このテキストで紹介しきれなかった応用例も数多くあります。これからも普及が望まれます。

付録 A

安全性仮定に関する問題一覧

このテキストでは暗号の安全性仮定に使われる様々な問題を紹介しました。その一覧をまとめておきます。基本的にある性質を具体的に求める「計算〇〇問題」（頭に Computational の C がつく）と、ある値がある条件を満たしているかを判定する「判定〇〇問題」（頭に Decisional の D がつく）の 2 種類があります。C-や D-がついていないときは通常計算問題を意味します。

A.1 乗法群表記の問題

乗法に関する巡回群 G とその生成元 g をとる。

DL 問題 (Discrete Logarithm) : n を整数として g, g^n が与えられたときに n を求めよ。

CDH 問題 (Diffie-Hellman) : a, b を整数として g, g^a, g^b が与えられたときに g^{ab} を求めよ。

DDH 問題 : a, b, c を整数として g, g^a, g^b および g^{ab} か g^c のどちらかが与えられたときにそのどちらが与えられたかを判定せよ。

RSA 問題 : p, q を素数として $n := pq$ とする。

$$de \equiv 1 \pmod{(p-1)(q-1)}$$

となる整数 d, e を選ぶ。整数 m に対して $n, e, c := m^d \pmod{n}$ が与えられたときに m を求めよ。

A.2 加法群表記およびペアリングの問題

加法に関する巡回群 G とその生成元 P , 乗法に関する巡回群 G' , および対称ペアリング $e : G \times G \rightarrow G'$ をとする。

$$g := e(P, P) \in G'$$

は G' の生成元である。

表 A.1 このテキストに登場した問題一覧

問題の名前	秘密の値	公開される値	求める値	登場箇所
DL	a	P, aP	a	5.1 節
DH	a, b	P, aP, bP	abP	5.1 節
BDH (Bilinear DH)	a, b, c	P, aP, bP, cP	g^{abc}	7.7 節
DBDH (Decisional BDH)	a, b, c	P, aP, bP, cP, Q	$g^{abc} \stackrel{?}{=} Q$	11.6 節 9.3 節
DLIN	$a, b, c,$ x, y	$P, xP, yP, axP,$ byP, cP	$(a+b)P \stackrel{?}{=} cP$	12.4 節
n -DHI (DH Inversion)	a	$P, aP, \dots, a^n P$	$(1/a)P$	15.4 節
n -SDH (Strong DH)	a	$P, aP, \dots, a^n P$	$g^{1/(a+b)}$	15.5 節
n -BDHI (BDH Inversion)	a	$P, aP, \dots, a^n P$	$g^{1/a}$	15.5 節
n -BDHE (BDH Exponent)	a	$P, aP, \dots, a^n P,$ $a^{n+2}P, \dots, a^{2n}P$	$g^{a^{n+1}}$	10.6 節

参考文献

- [AB12] Jean-Philippe Aumasson and Daniel J. Bernstein. Siphash: a fast short-input prf. Cryptology ePrint Archive, Report 2012/351, 2012. <https://131002.net/siphash/>.
- [AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, Vol. 9, No. 1, pp. 1–30, February 2006. <https://eprint.iacr.org/2005/028/>.
- [Aft11] Alex Edward Aftuck. The weil pairing on elliptic curves and its cryptographic application. *UNF Theses and Dissertations Paper 139*, 2011. <http://digitalcommons.unf.edu/etd/139/>.
- [aKW11] Alexander “alech” Klink and Julian “Zeri” Wälde. Efficient denial of service attacks on web application platforms. In *28th Chaos Communication Congress*, 2011. http://events.ccc.de/congress/2011/Fahrplan/attachments/2007_28C3_Effective_DoS_on_web_application_platforms.pdf.
- [AMORH14] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodríguez-Henríquez. Weakness of $\mathbb{F}_3^6 \cdot 509$ for discrete logarithm cryptography. In *Pairing-Based Cryptography—Pairing 2013*, pp. 20–44. Springer International Publishing, 2014. [http://eprint.iacr.org/2013/446](https://eprint.iacr.org/2013/446).
- [AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, Vol. 1880 of *Lecture Notes in Computer Science*, pp. 271–286. Springer Berlin Heidelberg, 2000. <http://www.iacr.org/archive/crypto2000/18800272/18800272.pdf>.
- [BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *In EUROCRYPT*, pp. 127–144. Springer-Verlag, 1998. <http://www.semper.org/sirene/people/gerrit/papers/divertproxy.pdf>.
- [BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano.

- Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, Vol. 3027 of *Lecture Notes in Computer Science*, pp. 506–522. Springer, 2004. <http://www.iacr.org/cryptodb/archive/2004/EUROCRYPT/1954/1954.pdf>.
- [BF03] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, Vol. 32, No. 3, pp. 586–615, March 2003. <https://eprint.iacr.org/2001/090/>.
- [BFK⁺12] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In *CRYPTO*, pp. 608–625, 2012. <https://eprint.iacr.org/2012/417>.
- [BFK⁺13] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. 暗号ハードウェアに対する効率的なパディングオラクル攻撃, 2013. <http://www.cs.bham.ac.uk/~kawamoty/scis2013.pdf>.
- [BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, CRYPTO’05, pp. 258–275, Berlin, Heidelberg, 2005. Springer-Verlag. <http://crypto.stanford.edu/~dabo/abstracts/broadcast.html>.
- [BL07] Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In *Proceedings of the Advances in Cryptology 13th International Conference on Theory and Application of Cryptology and Information Security*, ASIACRYPT’07, pp. 29–50, Berlin, Heidelberg, 2007. Springer-Verlag. <https://eprint.iacr.org/2007/286>.
- [Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs1. In *Advances in Cryptology - CRYPTO ’98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, Vol. 1462 of *Lecture Notes in Computer Science*, pp. 1–12. Springer, 1998. <http://www.simovits.com/archive/pkcs.pdf>.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS ’93, pp. 62–73, New York, NY, USA, 1993. ACM. <http://www.iacr.org/archive/crypto2000/18800229/18800229.pdf>.

- [BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption padding – how to encrypt with rsa. In *Extended abstract in Advances in Cryptology - Eurocrypt 94 Proceedings*, EUROCRYPT'94, pp. 92–111. Springer-Verlag, 1994. <https://cseweb.ucsd.edu/~mihir/papers/oae.pdf>.
- [BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures—how to sign with rsa and rabin. In *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRIPT'96, pp. 399–416, Berlin, Heidelberg, 1996. Springer-Verlag. <http://www.cs.ucdavis.edu/~rogaway/papers/exact.pdf>.
- [BSCG⁺13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology - CRYPTO 2013*, Vol. 8043 of *Lecture Notes in Computer Science*, pp. 90–108. Springer Berlin Heidelberg, 2013. <http://eprint.iacr.org/2013/507>.
- [BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pp. 573–592, Berlin, Heidelberg, 2006. Springer-Verlag. <https://eprint.iacr.org/2006/045>.
- [CC05] Liqun Chen and Zhaohui Cheng. Security proof of sakai-kasahara’s identity-based encryption scheme. In *Proceedings of the 10th International Conference on Cryptography and Coding*, IMA'05, pp. 442–459, Berlin, Heidelberg, 2005. Springer-Verlag. <http://eprint.iacr.org/2005/226.pdf>.
- [Cha83] David Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology Proceedings of Crypto 82*, pp. 199–203, 1983. <http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF>.
- [Che06] Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pp. 1–11, Berlin, Heidelberg, 2006. Springer-Verlag. <https://www.iacr.org/archive/eurocrypt2006/40040001/40040001.pdf>.
- [CLT13] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and JuanA Garay, editors, *Advances in Cryptology - CRYPTO 2013*, Vol. 8042 of *Lecture Notes in Computer Science*, pp. 476–493. Springer Berlin Heidelberg, 2013. <https://eprint.iacr.org>.

- org/2013/183.
- [CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014*, Vol. 8383 of *Lecture Notes in Computer Science*, pp. 311–328. Springer Berlin Heidelberg, 2014. <https://eprint.iacr.org/2014/032>.
- [Coc73] C C Cocks. A note on 'non-secret encryption', 1973. http://www.cesg.gov.uk/publications/Documents/nonsecret_encryption.pdf.
- [CRYP08] CRYPTREC ID ベース暗号調査 WG. Id ベース暗号に関する調査報告書, 2008. http://www.cryptrec.go.jp/report/c08_idb2008.pdf.
- [Dwo05] Morris Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication, 2005. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.
- [FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. *J. Cryptol.*, Vol. 17, No. 2, pp. 81–104, March 2004. http://www.di.ens.fr/~pointche/Documents/Papers/2004_joc.pdf.
- [Fri98] Stefan Friedl. An elementary proof of the group law for elliptic curves, 1998. <http://math.rice.edu/~friedl/papers/AEELLIPTIC.PDF>.
- [FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pp. 368–387, London, UK, UK, 2001. Springer-Verlag. <http://www.iacr.org/archive/crypto2001/21390366.pdf>.
- [GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pp. 1–17, 2013. <https://eprint.iacr.org/2012/610/>.
- [GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure attribute based encryption from multilinear maps, 2014. <http://eprint.iacr.org/2014/622>.
- [GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, Vol. 18, No. 1, pp. 186–208, February 1989. <http://groups.csail.mit.edu/cis/pubs/shafi/1985-stoc.pdf>.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pp. 89–98, New York, NY, USA, 2006. ACM. <https://eprint.iacr.org/>

- 2006/309.
- [Gua13] Guardian. Lavabit founder refused fbi order to hand over email encryption keys, 2013. <http://www.theguardian.com/world/2013/oct/03/lavabit-ladar-levison-fbi-encryption-keys-snowden>.
- [Gui13] Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, Vol. 7954 of *Lecture Notes in Computer Science*, pp. 357–372. Springer Berlin Heidelberg, 2013. <https://eprint.iacr.org/2013/218>.
- [Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *Proceedings of the 4th International Symposium on Algorithmic Number Theory*, ANTS-IV, pp. 385–394, London, UK, UK, 2000. Springer-Verlag. <http://www-polsys.lip6.fr/~joux/pages/papers/TDH.pdf>.
- [KD98] Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *In Proceedings of Eurocrypt'98*, pp. 145–157, 1998. <http://link.springer.com/chapter/10.1007/BFb0054123>.
- [KSD13] Cameron F. Kerry, Acting Secretary, and Charles Romine Director. Fips pub 186-4 federal information processing standards publication digital signature standard (dss), 2013. <http://dx.doi.org/10.6028/NIST.FIPS.186-4>.
- [KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pp. 146–162, Berlin, Heidelberg, 2008. Springer-Verlag. <https://eprint.iacr.org/2007/404>.
- [LB] Tanja Lange and Daniel J. Bernstein. Explicit-formulas database. <https://www.hyperelliptic.org/EFD/>.
- [LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, Vol. 6110 of *Lecture Notes in Computer Science*, pp. 1–23. Springer Berlin Heidelberg, 2010. <https://eprint.iacr.org/2012/230>.
- [Mic12] Microsoft. セキュリティアドバイザリ 2718704: Flame の攻撃と wu の強化, 2012. <http://blogs.technet.com/b/jpsecurity/archive/2012/06/11/3503098.aspx>.
- [NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pp. 113–124, New York, NY, USA,

- [NSA09] NSA. Suite B cryptography, 2009. https://www.nsa.gov/ia/programs/suiteb_cryptography/.
- [OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pp. 195–203, New York, NY, USA, 2007. ACM. <https://eprint.iacr.org/2007/323>.
- [OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, pp. 191–208, Berlin, Heidelberg, 2010. Springer-Verlag. <http://eprint.iacr.org/2010/563>.
- [OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, Vol. 7237 of *Lecture Notes in Computer Science*, pp. 591–608. Springer Berlin Heidelberg, 2012. <https://eprint.iacr.org/2011/543>.
- [SEH⁺13] Yusuke SAKAI, Keita EMURA, Goichiro HANAOKA, Yutaka KAWAI, and Kazumasa OMOTE. Methods for restricting message space in public-key encryption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E96.A, No. 6, pp. 1156–1168, 2013. <http://dx.doi.org/10.1587/transfun.E96.A.1156>.
- [SHI⁺12] Y. Sakemi, G. Hanaoka, T. Izu, M. Takenaka, and M. Yasuda. Solving a discrete logarithm problem with auxiliary input on a 160-bit elliptic curve. In *PKC2012*, pp. 595–608, 2012. http://ecc2012.cs.cinvestav.mx/Presentaciones/Yumi_Sakemi.pdf.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, pp. 457–473, Berlin, Heidelberg, 2005. Springer-Verlag. <http://eprint.iacr.org/2004/086>.
- [TK03] M. Kojo T. Kivinen. More modular exponential (modp) diffie-hellman groups for internet key exchange (ike), 2003. <https://www.ipa.go.jp/security/rfc/RFC3526JA.html>.
- [ukk13] ukky3. Lavabit 事件とその余波、そして forward secrecy, 2013. <http://negi.hatenablog.com/entry/2013/11/05/093606>.
- [Vau02] Serge Vaudenay. Security flaws induced by cbc padding - applications to ssl,

- ipsec, wtls. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, Vol. 2332 of *Lecture Notes in Computer Science*, pp. 534–546. Springer, 2002. <http://www.iacr.org/cryptodb/archive/2002/EUROCRYPT/2850/2850.pdf>.
- [vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, Vol. 6110 of *Lecture Notes in Computer Science*, pp. 24–43. Springer Berlin Heidelberg, 2010. http://link.springer.com/chapter/10.1007%2F978-3-642-13190-5_2.
- [Wat14] Brent Waters. A punctured programming approach to adaptively secure functional encryption. *Cryptology ePrint Archive*, Report 2014/588, 2014. <http://eprint.iacr.org/>.
- [YSK⁺13] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiya. Secure pattern matching using somewhat homomorphic encryption. In *Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop*, CCSW ’13, pp. 65–76, New York, NY, USA, 2013. ACM. <http://research.icbnet.ntua.gr/DPM2013/resources/slides/05.pdf>.
- [伊豆 13] 伊豆哲也. 楕円曲線暗号入門（2013年度版）, 2013. http://researchmap.jp/mulzrkzae-42427/#_42427.
- [岡本 11] 岡本栄司, 岡本健, 金山直樹. ペアリングに関する最近の研究動向. 電子情報通信学会基礎・境界ソサイエティ Fundamentals Review, 2011. https://www.jstage.jst.go.jp/article/essfr/1/1/1_1_1_51/_article/-char/ja/.
- [花岡 14] 花岡悟一郎, 大畑幸矢, 松田隆宏. モジュール化に基づく高機能暗号の設計：実社会への高機能暗号の導入における障壁の低減に向けて. *Synthesiology*, Vol. 7, No. 2, pp. 93–104, may 2014. https://www.jstage.jst.go.jp/article/synth/7/2/7_93/_pdf.
- [岩田 06] 岩田哲. Omac: One-key cbc mac, 2006. <http://www.nuee.nagoya-u.ac.jp/labs/tiwata/omac/omac.html>.
- [結城 08] 結城浩. 新版暗号技術入門 秘密の国のアリス. ソフトバンククリエイティブ, 2008.
- [山田] 山田泰彦. 代数曲線と離散 painlevé 方程式. <http://www.math.kobe-u.ac.jp/~yamaday/kaisetsu.pdf>.
- [漆嶋 14] 漆嶋賢二. Bitcoin を技術的に理解する, 2014. <http://www.slideshare.net/kenjiurushima/20140602-bitcoin1-201406031222>.
- [森山 11] 森山大輔, 西巻陵, 岡本龍明. 公開鍵暗号の数理. 共立出版, 2011.
- [青空] 青空学園数学科. パップスの定理. <http://aozoragakuen.sakura.ne.jp/taiwa/>

- taiwaNch03/pascal2/node1.html.
- [土井 07] 土井洋. 情報セキュリティ大学院大学公開講座 ゼロ知識証明入門, 2007. <http://lab.iisec.ac.jp/~arita/lecture3.html>.
- [富士 10] 富士通株式会社, 株式会社富士通研究所. 楊円曲線暗号と RSA 暗号の安全性比較, 2010. <http://jp.fujitsu.com/group/labs/downloads/techinfo/technote/crypto/eccvsrsa-20100820.pdf>.
- [有田 14] 有田正剛. イデアル格子暗号入門, 2014. <https://www.iisec.ac.jp/proc/vol0006/arita14.pdf>.

索引

- AEAD, 42
- attribute-hiding, 109, 113, 114
- BDHE 問題, 96, 136, 140
- BDHI 問題, 139
- BDH 問題, 79
- bootstrap, 125
- CBC モード, 9–11, 15, 40, 42
- CBDH 問題, 79, 136
- CDH 問題, 63, 182
- CPA, 24, 28–31
- Cramer-Shoup 暗号, 29, 43
- CRYPTREC, 8, 14, 38, 40, 79
- CTR モード, 14, 15, 42
- DBDH 問題, 104
- DDH 問題, 135, 182
- DH 問題, 21, 27, 54, 78, 134, 136–140
- distortion 写像, 75, 76
- DLIN 假定, 114, 135, 136, 141
- DLP, 20, 21, 27, 28, 48, 54, 65, 66, 76–78, 134, 136, 139–141
- DPVS, 110–113
- DSA, 43, 55, 58
- ECDLP, 54, 58, 77, 78
- ECDSA, 55, 58
- ElGamal 暗号, 3, 17, 25–32, 35, 43, 50, 54, 59, 88, 116–118, 120, 129, 130, 132, 133, 135
- Euclid の互除法, 66, 145–147, 150, 153
- FDH, 43, 45, 46
- FE, 114
- GCM, 42
- ID ベース暗号, ix, 69, 70, 78–80, 82, 99, 102, 104, 106–108, 110, 139
- IND-CCA, 29, 31, 32, 35, 118, 192
- IND-CCA2, 29, 31, 32, 35
- (k, n) 閾値法, 99
- Lagrange 係数, 101
- Lagrange 補間, 106, 107
- LSSS, 105
- LWE, 122, 123, 126, 144
- MAC, 32, 39–42
- Miller のアルゴリズム, 178, 180
- MITM, 22
- NSA, 15, 55, 57
- payload-hiding, 109
- PFS, 3, 56, 57
- PKI, 3, 33–35, 79, 80
- POODLE, 3, 11, 42
- PRE, 87
- private-index, 109
- PSI, 46
- public-index, 109
- RSA-OAEP, 32
- RSA 暗号, 3, 8, 30–32, 42, 43, 55–57
- RSA 假定, 32, 43
- SDH 問題, 139, 140
- SHE, 125, 126
- SSL, 3, 9, 11, 42
- TLS, 42
- Vandermonde 行列, 101
- Vernam 暗号, 5–7, 11–14, 99
- Weierstrass の方程式, 144, 154, 158, 159, 168, 169
- アフィン, 158, 166–168
- 位数, 60, 63, 65, 66, 110, 112, 150, 170–175, 179–181
- 因子, 144, 173–176, 178, 179, 181
- 可換群, 61, 62
- 拡大体, 144, 150–153
- 閑数型暗号, 69, 70, 108, 114, 115, 122
- 危殆化, 7, 8
- 強秘匿性, 27, 28, 30, 31, 135
- 極, 46, 96, 109, 121, 125, 169–175, 177, 179–181
- 計算量の安全, 3, 7, 11, 14
- 結託攻撃, 89, 90, 94–96, 102, 104, 137
- 原始元, 62
- 原像計算, 37, 38
- 再暗号化, 87–90, 132
- サイドチャネル攻撃, 169
- 識別不可能性, 29
- 射影空間, 144, 164–166, 169

- 述語暗号, 108, 110, 114, 115
巡回群, 62–66, 75, 76, 88, 103, 112, 138, 140, 150, 182, 183
情報理論的安全, 3, 5–7, 14, 99
伸長攻撃, 41
正規化, 177, 179
齊次多項式, 166
生成元, 20, 43, 62, 63, 112, 141, 182, 183
ゼロ知識証明, 69, 122, 127–130, 132–134
選択暗号文攻撃, 28–31
選択平文攻撃, 24, 28, 30, 40
双線形写像, 75, 110, 177
属性ベース暗号, ix, 69, 70, 98, 102, 104, 106–110, 115
素体, 152, 153
体, 26
対称ペアリング, 75, 76, 89, 183
タイムリリース暗号, 80–82
代理暗号, 87
多重線形写像, 73, 111, 122
畳み込み, 124, 126
単位元, 60–63, 65, 144, 159–161, 169, 172, 174
中間者攻撃, 22, 32, 34
デジタル署名, 3, 32, 33, 36, 42, 48, 55, 82
トーラス, 52, 53, 144, 145, 154–158, 163, 164, 169, 176
内積暗号, 109, 110, 113
認証, 36
認証付き暗号, 42
バイナリ法, 23, 53, 180
ハッシュ関数, ix, 3, 29, 34, 36–41, 43, 46–48, 55, 65, 79, 85, 103, 129, 139
パディングオラクル攻撃, 10, 11, 30
非対称ペアリング, 75, 76
非展性, 28
秘密分散, 90, 94, 98–100, 102, 104–107
標数, 26, 152, 158, 159, 168
フィンガープリント, 34, 119
ブラインド署名, 3, 36, 44–47, 131
ペー関数, 156
ミックスネット, 131–134
乱数, 3, 5, 6, 11–16, 27, 32, 36, 45, 79, 81, 85, 99, 100, 105, 110, 113, 116, 129, 132, 133
ランダムウォーク関数, 64
離散対数問題, 20, 35, 47, 53, 54, 58, 59, 63, 64, 66, 77, 117, 127, 128, 140
 ρ 法, 63
ワンタイムパッド, 5