平成30年度

卒業論文

ブロックチェーン技術を用いた三式薄記モデルの応用

統計ファイナンス研究室

北田　清人  
学籍番号：1G159024

**目次**

1. ブロックチェーンの構築  
   第1章　序論・目的　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.1  
   第2章　ブロックチェーン技術について　　・・・・・・・・・・・・・・・・・・・・　P.2  
   第3章　トークンに使う自作クラス　　・・・・・・・・・・・・・・・・・・・・・・　P.3  
   第4章　トランザクション同士の繋がり　　・・・・・・・・・・・・・・・・・・・・　P.6  
   第5章　実行結果画面，UIについて　 ・・・・・・・・・・・・・・・・・・・・・・　P.7  
   第6章　プログラムの説明　　・・・・・・・・・・・・・・・・・・・・・・・・・・　P.8  
   第7章　考察　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.11
2. 楕円曲線暗号の構築  
   第1章　序論　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.12  
   第2章　楕円曲線暗号について　　・・・・・・・・・・・・・・・・・・・・・・・・　P.12  
   第3章　楕円曲線による群　　・・・・・・・・・・・・・・・・・・・・・・・・・・　P.13   
   第4章　楕円曲線暗号の仕組みと理論　　・・・・・・・・・・・・・・・・・・・・・　P.15  
   第5章　プログラムの説明　　・・・・・・・・・・・・・・・・・・・・・・・・・・　P.18  
   第6章　考察　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.19
3. 付録・参考資料  
   第1章　付録について　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.21  
   第2章　既知の性質　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.21  
   第3章　定理・性質について　　・・・・・・・・・・・・・・・・・・・・・・・・・　P.22  
   第4章　記述したコード　　・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.30

謝辞　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.62

参考文献　　・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　P.62

1. ブロックチェーンについて
2. 序論・目的

ブロックチェーン技術は分散型台帳技術の一種で，取引のデータであるトランザクションをブロックと呼ばれるデータの塊にまとめ，ブロックをチェーン状に保存するというデータ構造を持つ．現在色々な場所で研究がなされているが，実装面では金融や仮想通貨が多い．

ブロックチェーン技術の中でもトランザクションに着目すると，以前に生成されたUTXOをいくつか用いて新しいトランザクションを生成する．それには暗号理論やハッシュ関数を用いてどのUTXOを用いたのかを明示しているが，なぜ選んだUTXOが必要なのか，なぜ別のUTXOではなかったのかが大事な点のひとつであると考えた．今までの研究では仮想通貨の金額や投票システムの投票数など数字を使うものや，トランザクションのメッセージをつける部分に文書を添付し，データを保存するものが多く見られた．どちらもトランザクション同士の関係性は入力に使うUTXOの数字の合計と出力されるアウトプットの数字の合計が等しくなるように選ばれている．このようなモデルを三式薄記モデルという．三式薄記モデルは「予算≧入力＝出力」の関係を保つモデルで，信頼できない人同士の取引も正常に行える仕組みである．現在この関係は数字同士の関係で成り立っているが，何か別の媒体も使えるのかもしれないと考えた．

本研究ではトランザクション同士の繋がりに数字ではなく自作クラスを使えないかと考えた．もしこの方法ができるのならば個々の事例に落とし込むにはそれぞれ工夫が必要ではあるが，数字以外でもブロックチェーン技術は成り立たせることが出来るといえる．これが可能なのであれば汎用性は高くなる可能性があり，他の多くの仕組みに応用できると考えられる．そこで本研究で作りたいシステムは  
というモデルである．このモデルを作ることが一番の目標であったが，最後の可逆な変換という条件を満たすものを考えることが難しかった．そのため，本研究ではその条件を緩め，非可逆な変換な変換をした．この式のモデルは本研究が初めてであるため，このモデルが実装可能であることを示したい．そのため，非可逆変換でも可能なことを示すことを一番とした．

本研究では星型の画像のクラスを作り，そのトークンの集合を考える．変換として，2つの星型から1つの星を合成する．また，それぞれの画像は圧縮されたデータとして保存する．そして今までに作ってきた画像を閲覧することも出来るような仕組みを目指すことにした．

全ての単語について説明することは出来ないが，ある程度の用語の説明はここで述べる．

・ノード

「結び目」「節」といった意味で，コンピュータネットワークの中の装置のこと．ブロックチェーン内ではクライアント，サーバ，ピアがそれに値する．

・UTXO

unspent transaction outputの略である．

トークンとは仮想通過の実装で言う通貨の部分のような三式簿記の比較対象となる部分である．詳しくは2.4で述べる．トランザクションはトークンの見方からすると，インプットのトークンの集合，アウトプットのトークンの集合，暗号や電子証明などのその他の部分の大きく3つに分けられる．このうちインプットは以前に登録された他のトランザクションのアウトプットを使用している．また，インプットからアウトプットへは「合計値が等しい」などの何らかの繋がり，そのインプットでなければいけない理由のようなものがある．アウトプットは他のトランザクションで1度のみ参照されることが出来る．使用されていないものはトークンが有効だと判断して参照されることが可能であるため，これをUTXOと呼ぶ．

今回の私の書いたコードではでトランザクションは無視している．

・マイニング

新しいブロックを作ることをマイニングという．仕事はトランザクションを取り込むこと，UTXOプールへ追加すること，ナンス値を求めること，今のチェーンにつなげて周りのノードに伝えることである．この操作は主にPoWでマシンパワーを多く使うため，マイニング報酬がもらえる．

1. ブロックチェーン技術について

2.1　ブロックチェーン技術についての概要

ブロックチェーン技術とは分散型台帳技術の1つで取引データ技術のことである．中央集中型のサーバに対して分散されて管理されるのが特徴で，全てのノードにデータが保存されていく構造を持つ．特定の管理機関がなくノード同士が対等な立場で通信するため，権限が一箇所に集中することはない．リーダーとなるノードがないため，どれが正しいかという判断基準が重要になる．この判断のことを合意といい，合意の基準を定めたアルゴリズムをコンセンサスアルゴリズムという．コンセンサスアルゴリズムにはPoSやPoI等の方法が研究，提案されているが，本研究ではPoWを採用した．コードではハッシュ関数に以下の数字が返ってくるsha-256関数を使っている．返り値は16進数64桁の文字列として変換したものを使用する．

を1つ前のブロックのハッシュ値，をブロックに取り込みたいトランザクション全てのハッシュ値，をシステムが定める定数とする．文字列として加算をしたをキーとしたハッシュ値が以下になるような新たな文字列を求める．このをナンス値，ナンス値を探すことをPoW（Proof of Work）という．トランザクションはツリー状に並べて下から順にハッシュ値を取っていくことで全てのトランザクションのハッシュ値としている．確認は上桁が0になっているかどうかで判断をし，は0から順に数字を試していく方法を取っている．また，はナンス値が見つかるまでが確率的に約10分かかるような値にしており，0が18桁分必要になっている．異なる長さのチェーンを比べると，長さの長いチェーンの方が多くのマイニングがされている．つまり，長いチェーンの方が多くのノードが確認したとみなす．そのため，破損していない異なるチェーンが存在したとき，一番長いチェーンが合意の取れたものだという判断をするという方法をとっている．この方法を取ることでマイニングが認められるのは最初に出来た人のみという制約などがつく．破損していないかどうかはハッシュチェーン構造とよばれる対改竄性のあるデータ構造をしている．そのため，改竄されたチェーンはすぐに破損していると認識される．

複数のトランザクションやナンス値等の証明を含めたデータの塊をブロックという．ブロックチェーンはこのブロックを1列に並べた構造を持ち，取引・ブロックが正しいという合意が覆る可能性が時間の経過とともに0へ収束する特徴を持っている．これはチェーンの合意方法からなる性質である．ほぼ同じ時間に違うマイナーがマイニングを成功してしまうと同じ長さの異なるチェーンが出来てしまう．このときどちらが合意の取れたチェーンかという判断が出来ないので，異なるチェーンがどちらも残ることになる．これをソフトフォークという．ソフトフォークが直るのはどちらかがもう一方を越える長さになるまでマイニングされたときである．このときに，その長い方が正しいチェーンと認識され，短くなってしまったほうは刈り取られてしまう．刈り取られた部分に覆ってしまったブロックを含むが，あるブロック自身の後に繋がれば繋がるほど，ソフトフォークによって覆る可能性が低くなるということである．

トランザクションやブロックは前のデータを参照しながら，繋がりを持って保存されている．1つを改竄するとそれ以降の参照されているデータを全て変更しなければならない．そのため，合意を取るためマイニングを複数回行い，その時点での正しいとされているチェーンよりも長さが長くなるまで続けなければならない．PoWには10分程かかることが，実質的に後から変更が出来ないという理由の1つである．そのため，ある程度チェーンが伸びると改竄が不可能だといえる．

データが保存されているのが一箇所ではないため一部が分断されても1台でも動けば停止しない．そのため可用性と分断耐性を保障している．また，あるトランザクションが登録されたとき，そのデータが他の全てのノードに渡るまでに誤差が生じるため，ユーザがアクセスしたときに必ずしも最新の情報が提供されるわけではなく，一貫性が完璧とはいえない．このような，可用性，分断耐性を持ち，一貫性を犠牲にしているシステムをAPシステムという．

トランザクションに使われるトークンはコピーされない作りになっているのではなく，コピーされることが前提で使う権利がコピーされないようなプログラムになっている．

2.2　暗号学的ハッシュ関数

ハッシュ関数とは要約関数とも呼ばれる関数で，引数に対してそれを代表するような数値を返す関数のことである．このときの引数をキー，戻り値をハッシュ値と呼ぶ．また，ハッシュ関数は単射とは限らないため異なる2つのキーについて同じハッシュ値が返ることがある．これを衝突といい，衝突が最小限に抑えるべきである．暗号学的ハッシュ関数とは，ハッシュ関数の中でも  
・キーがほとんど同じでも異なる場合にはハッシュ値がまったく異なる．  
・ハッシュ値を見て，それに対応するキーの1つを見つけることが難しい．（原像困難性）  
・同じハッシュ値となる，異なる2つのキーのペアを求めることが難しい．（衝突困難性）  
といった性質を持つものを言う．

私の書いたコードではトランザクション同士の参照やブロック同士のつながりに暗号的ハッシュ関数の1つであるSHA-256関数を使用している．以降単にハッシュ関数という時には暗号学的ハッシュ関数のことを指す．

2.3　三式簿記とトークンについて

三式簿記とは単式簿記や複式簿記の概念を拡張した考え方である．[1]では複式簿記を「財産＝資本」と見直すことによって時制式三式簿記と微分式三式簿記に拡張されることを示している．単式簿記は収入と支出に分けてその数字をまとめていく方式の帳簿である．複式簿記は貸借対照表と呼ばれる表を用いて，貸方と借方のバランスを取って帳簿を取る方法である．複式簿記では「資産=負債＋純資産」という関係式であるが，三式簿記では「財産＝資本＝予算」となる関係式である．ブロックチェーンではこの考えを各取引に取り込んだ「予算≧取引の入力＝取引の出力」となる関係を持つように取引を行う．ここで取引の入力が財産に，取引の出力が資本に対応する．

この関係式を満たすようにブロックチェーンは構成されている．この関係を持つための部分をトークンと呼ぶことにする．一般的な仮想通貨についてのブロックチェーンではコインやポイントと呼んでいる数字のことがトークンに値する．一般的な呼称ではないが本論文で述べたい点がこの点のため，名前をつけることにした．

トランザクション同士の繋がりに大事な部分であり，なぜそのUTXOを用いたのかという部分がこの部分に関係する．仮想通貨を例にすると，トランザクションの暗号部分や送信者・受信者も大事であるが，「何コイン分の価値があるトランザクションなのか」や「作りたい，送りたい価値を越すぐらいのUTXOを使わないといけない」といった部分になる．文書を保存するときにメッセージとして保存するならば，トークンは文書ではなく手数料のような繋がりを持つ部分であるとする．

1. トークンに使う自作クラス

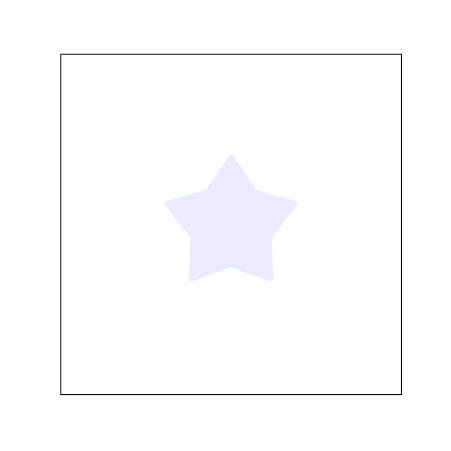
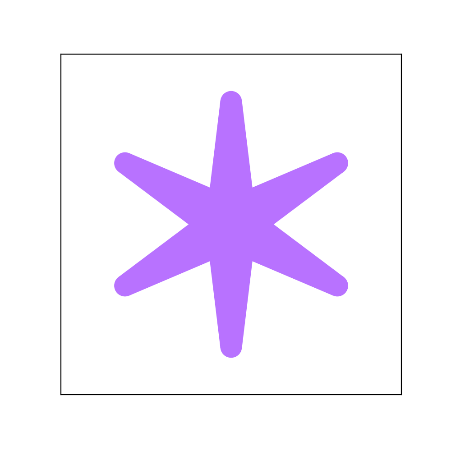
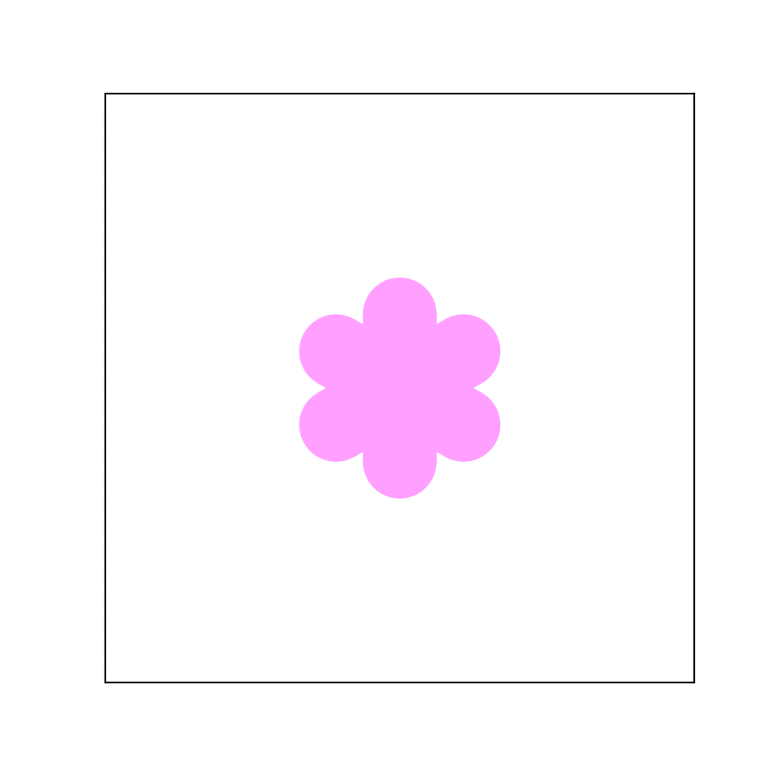
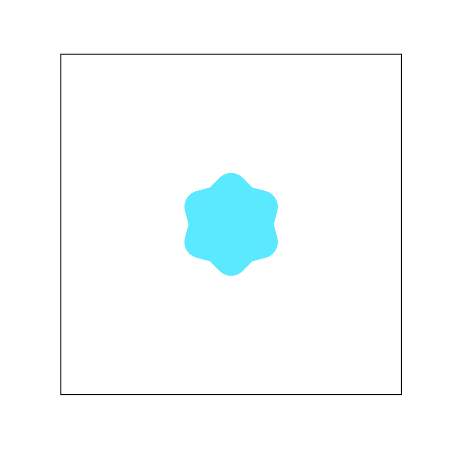
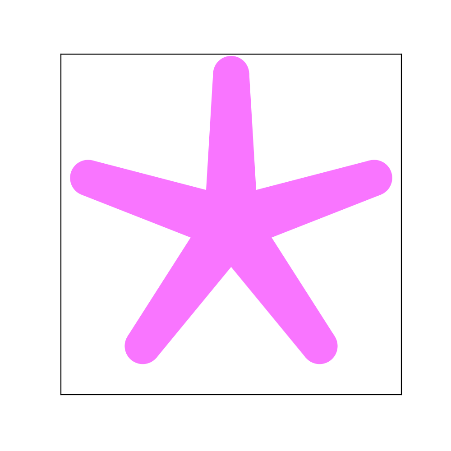
3.1　概要

今回のトランザクションに使うトークンとして画像を使用する．2つの画像のUTXOをインプットとして，その2つの画像の特徴を持った新しい画像をアウトプットにした．第1章で述べたとおり今回の目的は自作クラスのようなものでもトークンとして動くのか，トークンとして数字そのものではないものでも使えるのか，ということを調べたい．そのため，一般的な画像ではなく特徴がはっきりしているものにした．今回の実装では生成された画像は全て確認できるようになっていて，ブロック番号とトランザクションのハッシュ値さえ分かっていれば，誰の画像だろうが確認できる．しかし，画像に鍵をかけることも可能であるため，その場合はその秘密鍵を知るものしか見ることが出来ない．ここでの公開鍵，秘密鍵はトランザクションの送信に使う鍵ではなく，パスワードのような働きの1度きりの鍵でも構わない．今回の実装のような前者であれば論文公開等の公開したい場合のブロックチェーンに，後者のようなパスワード付きであれば，会議資料のような複数人で見たいが全員には見せたくないものにも使用できる．

画像は星型のみを使用し，特徴の要素として「頂点数」「とげの大きさ」「頂点の丸み」「色」の4つとした．第1章で述べた「」のとして2つの画像を引数として，各要素についてどちらか一方，もしくはそれらの中間に似るような画像を戻り値とする関数とした．例外として元の画像のどちらにも似ないときも低い確率で存在するようにしたが，その場合，最大値もしくは最小値に設定した値になりやすい．いずれの要素もランダムに引き継がれるようにしたため，同じ2つの画像を入力しても同じ出力がされるとは限らない．合成するときの方法であるが，頂点数はランダムでどちらかにし，他は重みつき平均を取る．考えうる最大値，最小値の近くになることもあるように考慮することで，合成をし続けることにより中間の値になってしまうことにならないようにした．を64%の確率で1に近い値，36%の確率で0に近い値をとるランダムな数，を0に近い値をとりやすい0から1の範囲にあるランダムな値，をほぼ0に近い値ばかりとる0から1の間にあるランダムな値とする．元となる2つの値を，2つの平均値を，最大値，最小値をとする．このときに合成される星の値は  
とした．この方法を取ることで23.0%の確率で，23.0%の確率で，41.3%の確率で，12.7%の確率で最大値か最小値に近い値になるようになっている．正確に表すと，  
とした．色についてはHSV色空間に変換し，その中で距離を定め2つの元となる色との距離の逆数の和が一定の値以上となる色をランダムに選んでいる．人の目に見て色の近い分布になっているため，形同様に中間の色というような混ざり方もするようになっている．

ブロックチェーンでは特にパブリックチェーンと呼ばれる仕組みではマイニング報酬という仕組みがよく使われる．これはマイニングにマシンパワーが多く必要になるため，報酬がないとマイニングが行われない可能性が生じるからである．

今回作成したブロックチェーンではマイニング報酬として新しい星を生成し，マイニングをしたアドレス宛にその星を与えるようにした．この仕組みにしたことで合成するたびに星が減っていくが，マイニングをするたびに星が追加され，データの枯渇がないようになった．

図1は上2つが合成元の画像で下4つがそれぞれ合成された星の例である．この例では同じ画像の組から4つの画像を例示したが，実際のコードでは1つしか合成されない

【図1】画像の生成の例

3.2　クラスについて

クラス名をStar\_Dataとした．実際のコードはP.56(第3部)を参照いただきたい．クラスでは上記の「頂点数」「とげの大きさ」「丸み」「色」の他に「作成時間」「名前」「親の名前」の変数を持ち，コンストラクタ，合成する関数，IDを求める関数，時間を文字列にする関数の4つの関数を持っている．IDとはハッシュ関数を用いてインスタンス自体をハッシュ化したものである．こうすることでハッシュ関数の性質により，異なるインスタンスならばIDが異なるようにできる．

ブロックチェーン上に登録するときはこのインスタンスを丸々トークンとして使用する．マイニング時の画像をランダムに生成するときを除いて，画像の生成には親となる2つのトークンが必要になる．この方法により，トークン同士の繋がりを持たせることが出来た．

3.3　Jsonについて

ハッシュ値のキーはJson形式にする必要がある．そのため，インスタンスを含むものをキーにするときに1度Json形式に変換する必要がある．自作クラスやdatetimeクラスなどの一部のパッケージのインスタンスはデフォルトではJson形式に対応していない．トランザクションを登録するときにハッシュ値をキーとした辞書で保存することや，ナンス値を求めるときにトランザクションのハッシュを取ることがあるため，必ずハッシュ値を取れなければならない．トークンに自作クラスのインスタンスを使うのならば，そのインスタンスを使うためにJson形式に対応させる必要がある．

json.JsonEncoderというクラスを継承した自作クラス（MyJsonEncoderという名前のクラスを作成し，その中でStar\_Dataのエンコード方法を指定する．そして，エンコードのときにcls要素としてMyJsonEncoderを指定する．

この操作をすることでStar\_Dataのハッシュ値を取ることができ，画像の出力の際は衝突を避けられるため，ハッシュ値を画像の名前として出力した．

1. トランザクション

4.1　トランザクション間の繋がり

トランザクション内のインプットからアウトプットへの繋がりは第4章で述べたとおりである．ここではUTXOとしてアウトプットされてから使われるまでの繋がりなどを述べる．

トランザクションやその中のトークン，UTXOは受信者以外も見る事が出来るが，それを使用できるのは受信者のみでなければならない．これは使う権利がコピーされない仕組みになっているからである．この仕組みは公開鍵暗号を元に作られている．公開鍵暗号とは暗号化と復号で違う鍵を使用し，暗号鍵からは復号鍵は導くことが出来ないものである．この導けないとは数学的に不可能な場合（情報理論的に解読不可能という）と計算量的に不可能な場合（計算量的に解読不可能という）がある．ここでは楕円曲線暗号を使用した．詳しくは第2部に述べたのでそちらを参考にしていただきたい．

トランザクションは大きくインプット，アウトプット，証明部分に分けられる．証明部分では受信者しか解けない暗号文と，送信者しか作れない電子署名が含まれる．この受信者しか解けない暗号文を復号できたということを主張できれば使う権利を持っているということに対応する．暗号化，復号は普通の公開鍵暗号の仕組みを用いて行う．復号された平文が正しいかの判断であるが，暗号文を作るときにそのハッシュ値も残しておく．こうすると平文のハッシュ値と比べることで合っているかの判断が出来る．インプットにはどのUTXOを使ったのかと証明部分の平文の組を記している．電子署名は秘密鍵を持っている人にしか施せないものであり，その人の公開鍵を用いて正しいか否かを判断できる．これによって誰かが成りすましているのでなく，確かに送信者本人が送っているのだと主張している．電子署名はインプット部分とアウトプット部分をまとめた部分に施す．

この証明部分の存在によってトランザクション同士のつながりが正しいものなのかということを示す．自分自身に送られてきたアウトプットを自分自身がインプットとして使ったという証明がこれで出来たためそのUTXOを使う権利があるといえる．このような権利の譲渡でトークンをコピーされても問題がない仕組みになっている．

4.2　暗号

トランザクションの繋がりの部分で権利を示すために暗号技術が使われている．ブロックチェーンを実装するために必要な暗号の性質は「公開鍵暗号であること」，「電子署名が施せる方式であること」の2つである．仮想通貨のようなトークンに数字を使うものには「準同型暗号であること」など，トークンの性質によって追加で存在すれば有用になる性質も存在する．しかし，ブロックチェーンを実装する上では上記の2つの性質さえあればどのような暗号方式でも構わない．RSA暗号もこの2つの性質を持っているためRSA暗号を実装しても良いが，暗号強度の面で楕円曲線暗号の方が優れている．そのため楕円曲線暗号を使用した．

現在量子コンピュータの研究が進んでいる．もし開発がされた場合，楕円曲線暗号は解読アルゴリズムが考えられているため，暗号としては不適切になる．上記の2つの性質を持った格子暗号等の耐量子暗号を使って実装することが好まれる．しかし，送信の際に暗号文や公開鍵を保存していくため，耐量子暗号を使用するためには平文に対して暗号文や公開鍵の鍵長が長くなってしまっていることが課題である．

1. 実行結果画面・UI

5.1　UIに必要なパッケージなど

実装はLinuxによるP2Pを実現するため，Apache上で動かす必要があった．また，操作を簡単にするため，HTML言語・CSS言語を用いたインターネットブラウザでの操作にすることにした．PythonからHTMLを動かすためにFlaskを使用したが，FlaskとApacheは直接つなげることが出来ないため，mod\_wsgiもダウンロードする必要があった．

コマンドプロンプトでも動かすことも出来るため，WindowsやMacでも試行運転は出来るが，その場合はApacheとmod\_wsgiは必要がない．ただ，WindowsやMacだとデフォルトのセキュリティでP2Pが上手くいかないこともあったため，Linuxでの起動をお勧めする．

5.2　Apach・mod\_wsgi

ApachとはWebサイトを公開するためのソフトウェアで，特定の場所にHTML文書などを置くことで他のサーバからそのWebサイトにアクセスすることが出来る．

LinuxでのApacheインストール後，以下のようなファイルツリーにする．  
/ ┳ etc ━ httpd ┳ conf ━ httpd.conf  
 ┃ ┣ conf.d  
 ┃ ┣ modules ━ モジュールファイル  
 ┃ ┗ conf.modules.d  
 ┗ var ┳ log  
 ┗ www ┳ cg-bin ━ cgiファイルを入れる　　★  
　 ┗ html ━ htmlファイルを入れる　 ★

基本的には星マークをつけたところのみを操作すればよい．ブラウザで「http://サーバ名/hogehoge.html」というURLを入れたとすると，「/var/www/html/hogehoge.html」が表示される仕組みである．

WSGIとはWeb Server Gateway Interfaceの略で，WebサーバとWebアプリケーションを接続するためのPythonのインターフェイスである．そのうちの1つである，Apache用のmod\_wsgiをダウンロードした．Apacheとmod\_wsgiがあればFlaskやDjangoなどを用いたPythonによるWebアプリケーションを公開できる．mod\_wsgiは最初の設定だけ出来れば，他は特に操作の必要はない．

ここでPythonはPython2系とPython3系の文法が少し異なり，新たに出来た関数も存在する．今回はPython3系の実装の方がしやすいため，Python3.6を使用している．ここで，Apacheをインストールしたときmod\_wsgi.soがデフォルトで一緒にダウンロードされている．しかしこれはPython2系の書き方をしているため，このままだとApacheが起動出来ない．そこで，mod\_wsgi-py36.cpyton-m-x86\_64-lnux-gun.soをダウンロードして/usr/lib64/python3.6/site-packages/mod\_wsgi/serverdディレクトリに保存しなければならない．また，/etc/httpd/conf.modules.d/10-wsgi.confで元のmod\_wsgiが自動的に読み込まれてしまうためここもコメントアウトしなければならない．

最初にPythonファイルをApache上で動かすにはPythonの起動が必要である．通常は先ほど述べたとおり/var/www/htmlディレクトリにHTMLファイルを保存するだけでよいが，Pythonの起動のために設定をする．/var/www/blockchainディレクトリを作成し，その中にblockchain\_Star.pyファイル，blockchain.wsgiファイル，staticディレクトリ，templatesディレクトリを入れる．そして，/etc/httpd/conf.dにwsgi.confファイルを保存する．実際のコードやファイルツリーは第3部P.32に載せているが，このとおりに配置しないと起動されない．この設定を行うと，Linuxを起動すればサーバが起動する．

5.3　Flask

Pythonのパッケージの1つで，サーバサイドWebアプリケーションの構築に使うモジュール群である．構築に最小限の機能だけが用意されているため，小さく軽量なツールである．戻り値にHTML文書を使用することが出来るが，HTML文書やHTML文書内で参照するCSSファイルや画像ファイルは戻り値を返す前に別の特定のファイルに入っている必要がある．HTML文書はPythonファイルと同じ階層のtempletesディレクトリに，その他の参照するファイルはPythonファイルと同じ階層のstaticディレクトリに保存する．

HTML文書は通常の静的な作り方でも良いが，Pythonのコードから変数を渡してその値を表示させることも出来る．HTML文書内で{{ hoge }}とすると変数hogeに代入された値で変換されたものが表示される．他にも条件文，繰り返し文と，拡張が出来る．条件文は{% if 条件 %}{% endif %}，繰り返し文は{% for 要素 in リスト等 %}{% endfor %}のようなPythonに似た文法を持つが，HTMLではTabによる判断が出来ないため，終了タグが必要である．また，辞書やリストは「辞書名.キー」のようにドットでつなぐことで使用でき，クラスの場合クラス内の関数もドットで繋げば使用可能である．ここで注意が必要なのが，変数を使った場合，変数に代入がなされていないとエラーになることと，HTMLの意味でコメントアウトされていても二重の波カッコがあれば変数として認識されているということである．

拡張とは，各HTMLで共通なヘッダー部分などのみを記載したファイルbase.htmlを用意しておき，各ファイルがbase.htmlを参照して固有な部分のみ変えられる仕組みである．名前はbase.htmlでなくても良いが，説明のためこの名前を使う．base.htmlの固有な部分があるところを{% block body %}{% endblock %}で括っておき，各HTML文書で{% extends ‘base.html’ %}{% block body %}内容{% endblock %}としておく．すると，base.htmlが内容の部分で置き換わったHTML文書が返される．

URLからの入力でPythonを動かすためには関数デコレータを使用する．appをFlaskインスタンスとし，「@app.route(‘URL’, methods=[‘HTTPメソッド’])」というデコレータをつければURLを打ったときにその関数を動かせる．HTTPメソッドはコンマで複数指定可能で，基本的にはGETとPOSTしか使わない．また，methods要素は省略化でその場合はGET通信になる．HTML文書を使用しているため，formタグを使えば容易にPOST通信も出来るため，UIは作成しやすい．

5.4　Matplotlib

Pythonのパッケージの1つでグラフを描画するためのモジュール群である．今回の画像は円や直線で書けるつくりになっているため，このパッケージを用いて描画した．今回は陰関数でなく円や多角形の描画を用いた．Patchを円や多角形を表すデータとし，matplotlib.puyplot.axis().add\_patch(Patch)で複数指定した後に画像として保存すれば星のマークを保存することが出来る．

FlaskとMatplotlibはLinuxには初期設定では入っていない．なのでyumコマンド等でインストールする必要がある．

5.5　HTML・CSS

よく使われる言語なので言語としての説明は省略する．UIの見せ方として一番前に出てくる文書で，使用者はこのファイルで作成された画面を見て操作をする．実際のファイルは第3部P.67に載せている．GET通信では長すぎるメッセージが送れないこと，パスワード等もURLから見えてしまうことが弱点であるため，それが問題となる部分はformタグでPOST通信にしなければならない．それ以外はaタグで十分である．

1. プログラムの説明

6.1　blockchain\_Star.pyのblockchainクラス

このコードの中ではブロックチェーンをブロックチェーンクラスのインスタンスとして持っている．まずはその中の変数と関数についての説明をする．

6.1.1　変数

・chain <list>  
ブロックをリスト型として保存している．ブロックには  
index <int>：ブロック番号．最初から数えて何番目かで，最初のブロック番号を1とする．  
timestamp <float>：このブロックが作成された時間のタイムスタンプ．  
transaction <dict>：このブロックに含まれるトランザクション．キーは取引部分のハッシュ値．  
proof <str>：PoWによって見つけたナンス値．  
previous\_hash <str>：一つ前のブロックのハッシュ値．  
が辞書型で保存されている．また，transactionは  
trans <dict>：取引部分で，ユーザの指定する部分．この部分のハッシュ値をキーとする．この中身は，  
　　sender <str>：送信者の公開鍵．  
　　recipient <str>：受信者のアドレス．  
　　star <Star\_Data>：合成された星のデータ．  
　　parents <tuple>：(trans部分のハッシュ値，含まれるブロック番号)をtuple型で2つ指定する．  
が辞書型で保存されている．  
parents <tuple>：(trans部分のハッシュ値，codeの答え)をtuple型で2つ指定する．  
proof <dict>：自分自身に送られたものを自分自身が送ったと証明する部分．この中身は，  
　　code <tuple>：受信者の公開鍵で暗号化した暗号文．  
　　answer <str>：平文のハッシュ値  
　　proof <tuple>：trans部分とparents部分を電子署名する．  
が辞書型で保存されている．UTXOにはここにどのブロックに含まれるのかのindex <int>も持つ．

・current\_transactions <dict>

まだチェーンに含まれていない，これから取り込まれる予定のトランザクションプール．  
・nodes <set>

P2Pをする相手のURLを覚えている．

・UTXOs <dict>

UTXOプール．この中にあるもののみ使用できる権利がある．

6.1.2　関数

・new\_block(ナンス値，前のブロックのハッシュ値)

新しいブロックを作ってチェーンに繋げる．そのときに取り込んだトランザクションをUTXOプールに入れる．ブロックを作成した後に他のノードにマイニングをしたことを伝える．

・new\_transaction(送信者の秘密鍵，受信者の公開鍵，UTXOの情報，合成された星)

使用するUTXOはそのアドレスに送られたものなのか等の確認をした後に電子署名や暗号文などを含めトランザクションを発行する．その後にトランザクションプールに追加し，他のノードにも伝える．

・transaction\_set(トランザクション)

他のノードで登録されたものを合っているかの確認をして，妥当ならば登録する．

・last\_block()

チェーンの中で最新のブロックを返す．

・valid\_proof(前のブロックのハッシュ値，ナンス値)

文字列として「前のブロックのハッシュ値＋ナンス値」をする．これのハッシュ値が求めたい条件に見合うか否かを返す．この関数がマイニングの肝で，難易度設定はここで行う．本当はここにトランザクションのハッシュ値も必要．「前のブロックのハッシュ値＋ナンス値＋トランザクションのハッシュ値」にしなければならない．

・proof\_of\_work(前のブロックのハッシュ値)

条件に合うナンス値を総当りで探す．本当はこの引数にもトランザクションのハッシュ値が必要．

・register\_node(URL，チェックするか否か)

他のノードの登録をする．登録されたノードにのみデータの分配をする．チェックとはそのURLに繋がるか否かのチェックをするかどうかで，デフォルト値はTrueである．

・change\_chain(チェーン部分)

そのチェーンが妥当かの判断をし，間違いでなく，自分自身より長ければ交換する．

・resolve\_conflicts()

コンセンサスアルゴリズムのことである．はじめにも書いたが，一番長いチェーンが正しいものと判断される．他のノードのチェーンを確認して，そのチェーンに間違いがなくそちらのほうが妥当なのならばそのチェーンに交換する．

6.2　blockchain\_Star.pyの関数について

・get\_hash(キー)

キーに対してSHA-256関数にかけたハッシュ値をだす．

・new\_transaction() /transactions/new，GET，POST通信

新しい星の合成した後のページを表示する関数．

・Star\_ID() /star．POST通信

ブロック番号とIDを代入したらそのデータの詳しいデータを表示する関数．

・transacton\_from() /transactions/neighbor，POST通信

他のノードで登録されたトランザクションを確認して登録する．通信用の関数．

・register\_nodes() /nodes/register，GET，POST通信

他のノードの登録する関数．

・check() /chain/check，GET，POST通信

持っている星や特定の星

6.3　blockchain\_Star.pyのコード実行部分

app=Flask(\_\_name\_\_)  
でFlaskを起動し，関数を定義した後に  
if \_\_name\_\_=='\_\_main\_\_':

from argparse import ArgumentParser

parser=ArgumentParser()

parser.add\_argument('-p', '--port', default=5000, type=int, help='port to listen on')

args=parser.parse\_args()

port=args.port

node\_identifer=my\_ECC.privkeyToPubkey(str(port))

app.run(host='0.0.0.0', port=port, debug=True)  
とすることでローカルホストで起動される．Apacheで動かす場合はこの行は無くても構わない．

1. 考察

実行した結果としてはブロックチェーンの起動やP2Pの動作，インスタンス同士の繋がりや保存は上手く作動した．よってここまでに述べたような方法は実行可能なものであるといえる．HTMLで表示をするためにstatic/Starディレクトリに画像ファイルを保存したが，出力をする方法や，他のインターフェイスを使う方法などを使うのならばサーバ上に画像ファイルを保存しておく必要はない．トークンは受信者のみではなく，ブロック番号とトランザクションのハッシュ値を知っている人は誰でも見ることが出来る仕組みになった．このような仕組みが出来たことから，特許等の公開するための保存技術にも応用できる仕組みにも十分使えることも示せたのではないかと思う．

今回の研究の目的は  
1．トークン同士のトランザクション内の繋がりに直接的な数字ではないものを使うこと．  
2．トークンに自作クラスのインスタンスを使うこと．  
の2点をみたすシステムを構築することであった．画像の特徴をとして数字に変換し，インプットとアウトプットの関係を作った．この方法は数字を使ってはいるが，以前に構築した仮想通貨の受け渡しでは「インプットの合計と，手数料を含めたアウトプットの合計が等しい」という関係であったため，これと比べると，1の条件を満たしているのではないかと思っている．

今回構築したシステムは直接用途があるものではなく，先行研究として構築している．そのため，個々の案件についてはまた別の比較のための工夫や構築方法が必要である．ブロックチェーン技術によるシステムの構築の際に，数字を使わなければならない等の制約に縛られなくても良いということを示すことが出来た．

まとめると，以前までの三式簿記モデルの適用方法は  
であったが，本研究では  
という関係を保つモデルとなった．研究結果からを上手く取ればUTXOモデルは上手く作動することが分かった．本研究ではこのモデルが作動することを示すための研究であったため，は非可逆かつランダム値であった．しかし，を可逆な変換，もしくはハッシュ関数のような可逆でなくとも一意な変換となるものを採用すると，入力から出力までの動作が妥当かどうかを数学的に評価できる．このような性質を持つ関数を用いて新しい構築方法を取ればこのモデルを用いた新しいブロックチェーンモデルが完成することが期待される．

同じ研究室でのメッセージ欄ではなくトークンとして文書の保存が出来ないか，という研究もしていた．トークンとしてさまざまなものを採用できることがこの2つの研究から分かった．また，自作クラスのインスタンスをトークンとして採用できたため，もっと大きなインスタンスでもトークンに使うなどして保存できるのではと考えられる．このようなことから，ブロックチェーン技術を用いたシステムの構築は他にもまだ可能性が考えられるのではないかと期待できる．

今回の研究では時間の関係上マイニングのナンス値を探す作業で，トランザクションのハッシュ値を含めない状態で探した．本来であればトランザクションも含めなければセキュリティ面で危ない可能性がある．このブロックチェーンを構築したPythonファイルを元に修正，研究を進めていくのならばここを修正して構築する必要がある．

1. 楕円曲線暗号について
2. 序論

暗号技術は通信の秘密保護だけでなく，著作権保護や偽造防止など情報社会ではさまざまな部分で使用されている．この情報社会では暗号技術はもはやなくてはならないものになっている．

さて，暗号は共通鍵暗号方式と公開鍵暗号方式，ハイブリッド方式に分けられる．今回のテーマである楕円曲線暗号は公開鍵暗号方式に分類される．公開鍵暗号方式にはほかにもRSA暗号やエルガマル暗号が存在するが，同じ安全性に達するための暗号鍵の桁数を比べると指数的に楕円曲線暗号の方が短くなると考えられている．

今回の研究は楕円曲線暗号をpythonによって構築し，どのようなアルゴリズムで暗号化・復号をするのかを調べるとともに，ブロックチェーンの構築に対してこのパッケージを使えるようにするためのものである．主な目的は楕円曲線暗号をブロックチェーンのコード内で使用するために構築するだけである．また，第3部には楕円曲線についての性質もまとめた．そこでは素体上の楕円曲線の格子点の集合に定義される群について，位数が素数となる場合がある．この場合はどのような場合なのかということを研究した結果をのせている．

また，次の研究のため，長さがどのようになるのかが不明な平文を暗号化する必要がある．また，このときに平文に関係する数を出力する必要があった．この条件を満たすアルゴリズムを新しく提案する．このアルゴリズムは4.4節で紹介する．このアルゴリズムは楕円曲線ElGamal暗号と強度や鍵長は同程度であるが，平文の長さが無制限になること，出力の一部に平文の長さに対応した自然数が含まれることが新しい点である．

プログラムの説明を書くときの表記について，点といったときにはは2次元座標なので成分と成分を持つ．点の成分を成分をと略記することにする．

1. 楕円曲線暗号について

2.1　楕円曲線とは

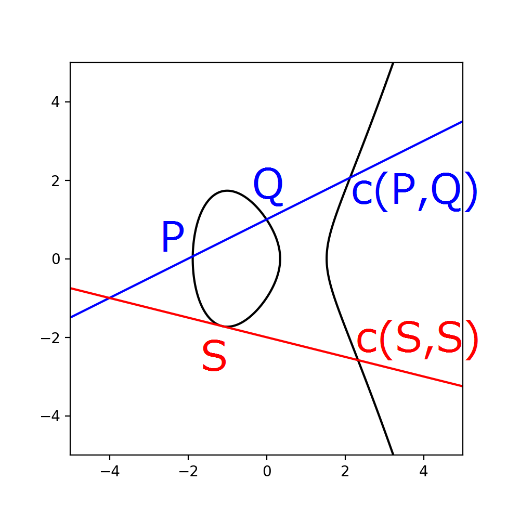
歴史的には楕円積分の研究で発見されたものである．楕円の弧長を求める際に出てくるものであるが，これは説明したいものではないためここでは割愛する．楕円曲線とは次数が3の非特異な代数曲線のことであり，  
と表される曲線である．ここで適切な変数変換をするとWeierstrassの標準形  
の形に変換できることが知られている．以降楕円曲線はこの形であらわされているものを扱うことにする．この楕円曲線の有理点上に演算を定義することで群をなすことが出来る．

により有理点を素数の剰余群に写すと，これは準同型写像となり楕円曲線上の有理点の集合に入る群と同様に有限群になる．

2.2　楕円曲線暗号とは

素数の剰余群上で定義された群の性質を用いた暗号が楕円曲線暗号である．暗号の種類は公開鍵暗号方式に分類され，運用側が使用する素数と基準とする有理点を選ぶが，その選び方を上手く取ると電子署名も行うことが出来る．群において有理点についてスカラー倍が分かっているとき，それが何倍されているのかを求めるのが難しい．この性質を暗号に使ったものが楕円曲線暗号である．現在ビットコインの暗号にも使われ，他のシステムでも楕円曲線暗号が推奨されているところである．[2]では楕円曲線ElGamal暗号と電子署名について示されているが，他にも楕円曲線ペアリング暗号等も考案されている．

1. 楕円曲線による群

3.1　楕円曲線上の集合とに定義される演算について

【図2】関数の定義

を定数とし，を楕円曲線の方程式で  
と定める．ここでは軸と同じ方向に存在するとみなす無限遠点である．についてを通る直線をとする．（どちらかがならば軸と平行な直線，なら接線とする．）するとととのもうひとつの交点は有理点となるため，その交点をとするとである．をひとつ取り，に対してを通る直線ととのもうひとつの交点を図2のようにと表すことにする．に演算＋をに対し，と定める．すると，は群をなす．(3.2で証明をする．)

素数について，を定数とし，を  
と定める．ここでは軸と同じ方向に存在するとみなす無限遠点であり，以下と表す．上記のとしてを選び，同様な操作で演算＋を入れると群となる．での計算式を具体的に書くと，次のようにについて演算＋が定義される．

という演算とする．

3.2　がアーベル群になることの証明

3.1で定義した関数はを満たす．これは3次曲線と直線の交点の個数は無限遠点と重解を含めるとちょうど3つ存在することを確認すれば証明できるため証明は略する．[9]で単位元をとしたときの証明が載っているので，それを参考にする．それではアーベル群であることの証明をする．

【補題】はアーベル群になる．

Proof

(a)単位元はである．  
に対し，

(b)に対する逆元はである．

(c)について，  
とすると，は共に3次曲線で＋の定義から，交点にをもち，はこの8つ全てを通る．Cayley-Bacharachの定理より，もう1つの交点も通るが，の交点はであり，の交点はであるため，

(d)について，

以上よりはアーベル群になる．

□

同様にもアーベル群になる．単位元の点をとするのはに対し，を求める操作，つまりを求める操作が楽になり，＋をするときの計算量が減るからである．

3.3　プログラミングについて

実際のプログラムしたコードは第7章に記載しているが，書くときに注意したところを次に示す．ここで取り上げる関数の名前はEC\_Add，EC\_double，EC\_multである．以降も必要なときは第7章を参照してほしい．

プログラムでは素数を法とした空間で考え，単位元を無限遠点にする．座標はタプル型でのように表すが，無限遠点はこの方法でかけない．この制限の元では，という点はどのような素数を法としたとしてもの点にはならないため，無限遠点をと表すことにした．つまり，演算を行う際には例外な点であるため外しておかなければならない点として注意する必要がある．

では逆数は必ず存在し，任意の元の逆数はユークリッドの互除法のアルゴリズムを用いて求められる．このアルゴリズムは1回あたりの計算量であるため出来るだけ1回で済ませるようにする．つまり，その計算をした値を覚えておいてから次に進めたらよい．

まず，足し合わせたい2点に対し，が含まれている場合はでないほうを返し，となる場合，つまり軸対称な位置に存在する場合は(0, 0)を返す．この後から一般の計算のプログラムを書く．を通る直線(同じ点の場合は接線)の傾きをとし，，(接線の場合は  
としてとして，という計算になる．ここで傾きは平均変化率または微分したものである．+の定義からとすると，が成り立つ．3.1に記したの定義よりが成り立つための係数を比較してよりが求まり，よりが求まる．このようにを先に求めることで分数を扱う回数を減らしている．

3.4　Double and Add法

演算の定義された集合においてある要素を整数回作用させた要素を求めるときに，一度に全て足すことや，順次足していくことが難しいときに使う手法である．に対し，を求めるとする．

2進数として桁が1つ上がるということは2倍されるということであるため，を2進数表記した状態でどう操作するのかを考える．から始めて1から2進数表記のでただの桁上がりなら2倍，1が出てくるなら桁上がりしてからを足す．計算の見方としてはの2進数表記の左から2つ目から考えて，に対し0なら2倍する，1なら2倍して1を足すという操作をして1の位まで操作した要素がの計算結果となる．

1. 楕円曲線暗号の仕組みと理論

4.1　エルガマル暗号について

楕円曲線暗号はエルガマル暗号の理論を応用させた仕組みとなっている．そこで先にエルガマル暗号について述べる．

まずシステムが素数とにおける原始根を定める．以下有限群での乗算を考える．秘密鍵はとなるにし，公開鍵はとする．の取り方によりとはなる．また，平文は以下の数字としてのみ使えるため，を大きく取り，また，平文もそれ以上になる場合は小さくちぎらなければならない．以下平文を表す数をとおく．

暗号化はとなるランダムな数を1つ取り，  
とし，この組を暗号とする．

復号は受け取ったに対して，  
とすると，となり，が復号した平文である．

この暗号はのとき，からを導出するのは難しいという対数離散問題に基づいており，この暗号が正しく復号されるための性質としてフェルマーの小定理を使っている．

4.2　楕円曲線ElGamal暗号の暗号化と復号

まずシステムが素数，楕円曲線と基準点を定める．このときが成り立つ．この性質を持つ自然数のうち最小のものとしてを取っておく．（このをの位数という．）秘密鍵はとなるにし，公開鍵はとする．の取り方によりとなる．

この方式では平文は以下の数字が暗号化できないため，を大きくするためにを大きく取る．また，平文もそれ以上になる場合は小さく分解することが奨められる．以下平文を表す数をとおく．

暗号化はとなるランダムな数を1つ取り，  
とし，このの組を暗号とする．

復号は受け取ったに対して，  
とすると，となり，が復号した平文である．

この暗号ものとき，からを導出するのは難しいという対数離散問題に基づいており，この暗号が正しく復号されるための性質としてを使っている．

コードの中ではというようにコーディングをした．チェックサムつきの文字列を入れるため4文字分に対応する数字はチェックサム用の桁となり，が小さいときのことも考えると，確実に隠せる文字列の長さとしてはせいぜい4桁が限界である．それ以上の長さのものを暗号化した場合，の下何桁が変わるだけで他は変わっていない，ということになってしまう．それゆえ，長い文章を暗号化したい場合には小さなブロックに分けてそれぞれ暗号化するなどのアルゴリズムが必要であった．

4.3　楕円曲線ElGamal暗号の妥当性

任意の平文に対して作った暗号文から復号が出来ればこの暗号が妥当であることを証明できる．

よってであるため復号できたといえる．

4.4　新しい楕円曲線暗号の暗号化と復号

今までの暗号方式だと限界の長さが存在し，その実質的な限界の長さLに対して暗号長が2Lだけ必要になってしまうことが問題にあげられる．ブロックに分けると暗号化できない限界の長さは無くなるが，平文の長さに対し暗号はと長くなるか，隠れないこともあるが少ないと仮定してよりも少し大きくなるかのどちらかになる．今回私が考えた方法では最大でもで，だいたいぐらいとなる．が大きいときには既存のアルゴリズムよりも良いアルゴリズムであるといえる．

まずシステムが素数，楕円曲線とを定める．の位数をとする．秘密鍵はとなるにし，公開鍵はとする．の取り方によりとなる．以下平文を表す数をとおき，となるランダムな数を1つ取り，とする．

ここまでは同じだが，あらかじめシステムが定数を定めている．数列を  
と定める．ここでとはをで割った余りをさす．であるためは単調増加な数列である．となる最小のをとおく．の組を暗号とする．

復号は受け取ったに対して，とし，  
とすると，となり，が復号した平文である．

数列が漸化式で一般項を求めるのは難しいが単調増加性からは必ず存在する．この性質から確実に平文を隠すことが出来る．また，引き算をするためたまたまのときにはとなることもある．この差によっては平文よりも暗号文が短いこともありうる．また，は平文によって対数的に増加する．今回書いたコードではとした．

4.5　新しい楕円曲線暗号の妥当性

任意の平文に対して作った暗号文から復号が出来ればこの暗号が妥当であることを証明できる．まずであることを言うが，であるため，．よって定義式が同じであるためである．

よってであるため復号できたといえる．

4.6　電子署名について

電子署名とはある情報に関して，その作成者がその内容で書いたのだということを証明するための仕組みである.作成者のみが持っているはずの秘密鍵でしか作成されず，作成者とされる人の公開鍵を使って有効な署名かどうかの判断をする．また，電子署名や電子署名される文書が少しでも改竄されていれば無効と判断される．楕円曲線暗号による電子署名では証明したい内容のハッシュ値に対して署名し，電子署名を公開鍵でどんな変換をしたとしてもハッシュ値に戻るわけではない．また，電子署名に使うためにはシステムの選ぶの位数が素数でなければならない．

署名は証明する文書のハッシュ値ととなるランダムな数を1つ取り，  
とし，このの組を電子署名とする

電子署名の確認は  
として，*U*の座標とが等しければこの電子署名が有効であるとする．

4.7　電子署名の妥当性

任意のハッシュ値に対して作った電子署名についてになればこの署名が妥当であることを証明できる．なぜならばのときはの座標とが等しくなり，また座標が同じになる点は2点のみであるため，の要素数が十分に大きいとき2点というのは誤差と考えても良い．

よってであるため電子署名の妥当性が言えたといえる．

4.8　電子署名の注意点

電子署名をするときには必ず乱数は同じものを使ってはいけない．なぜならば，違う文書に同じ乱数と秘密鍵で署名した場合，秘密鍵を割り出せるからである．これからそれを証明する．

仮定は上のとおりであるため，異なる2つの文書のハッシュをとし，乱数をとしたときの電子署名をそれぞれとおく．すると，が秘密鍵である．

なぜならば，よりである．

よって秘密鍵が求められてしまうため電子署名には注意が必要である．

4.9　楕円曲線暗号の性質

今回の実装では準同型暗号にはなっていないが，関数を適切な変換にすると楕円曲線暗号は準同型暗号になる．準同型とは2つの平文とから暗号化した暗号とが存在した時に，のみを用いてを暗号化したものを得られるという性質である．まずはこれを証明する．

を+が定義された群とし，を準同型な写像とする．(恒等写像はこの性質を持つため少なくとも1つはこの条件を満たすの組は存在する．)2つの平文について，それぞれランダムな値を用いて，  
とする．ここで新たな暗号文を  
とすると，これがを暗号化したものになっている．なぜならば，  
よって楕円曲線暗号は適切な変換を採用すると，準同型暗号になる．

今回のプログラムのコードでは準同型でない写像を使っていないため準同型暗号ではない．また，4.4で述べた新しいアルゴリズムの暗号も準同型暗号にはなっていない．

1. プログラムの説明

5.1　60進数と関係するもの

60進数は'0123456789ABLOCKCHAINDEFGHIJKLMNPQRSTUVWXYZablockchaindefghijkmnopqrstuvwxyz'の大文字のオーと小文字のエルを除いた計60文字を数字として扱っている．データはstr型として保存している．

・int\_b60(数字，0埋め桁)，b60\_int(60進数)

10進数と60進数の交換を行う．

・b60endode(文字)，b60decode(60進数)

文字をアスキーコードにした状態で60進数に変換する．

・EncodeBase60Check(文字)，DecodeBase60Check(60進数)

チェックサムをつけた状態で60進数に変換する．チェックサムとは打ち間違い防止のための文字．

・rand60(桁数，最小桁数)

60進数のランダムな値を返す．

5.2　楕円曲線上の群に関係するもの

・inv\_mod(数，素数)

を返す関数．逆数を求めるのにの計算量が必要になる．

・EC\_add(P，Q)，EC\_double(P)，EC\_mult(n，P)

和と整数倍を求める．

5.3　公開鍵，秘密鍵，アドレスについて

・pointToPubkey(P)，pubkeyToPoint(公開鍵)，privkeyToPubkey(秘密鍵)，pubkeyToAddress(公開鍵)

それぞれ秘密鍵や公開鍵から必要なものを算出する．それぞれ秘密鍵から公開鍵，公開鍵からアドレスは導くことは出来るが，逆向きは難しい．計算量的に不可能な難易度である．

5.4　暗号化，復号に関するもの

・EC\_Encode(平文，公開鍵)，EC\_Decode(暗号文，秘密鍵)

一番基本的な楕円曲線暗号方式．鍵をかけられる最大の大きさが存在する．そのため長い平文ならばいくつかのブロックに分けなければならない．

・EC\_EncodeLong(平文，公開鍵)，EC\_DecodeLong(暗号文，反復数，秘密鍵)

4.4で説明した新しいアルゴリズムによる暗号である．反復数は暗号化したときに暗号文と共に出てきて，平文の長さと正の相関を持つ数字が出てくる．どれだけ長くても隠し切ることが出来る．

・EC\_Ensign(平文，秘密鍵)，EC\_Design(平文，電子署名，公開鍵)

平文に対して電子署名を施す．

1. 考察

素数を法とした素体上の楕円曲線の有理点による群による演算自体は容易に出来て，整数倍もある程度容易に出来る．しかし，離散対数問題が難しく，ある元が外の元の何倍であるのかというこのを求めるのは簡単ではない．この性質を使ったのが楕円曲線暗号であるが，このアルゴリズムは楕円曲線上の群でなくても離散対数問題が難しい群であればよいのだといえる．

楕円曲線上の点の位数に関して，が合成数ならば任意のの約数について位数がである点をから構成できる．なぜならば，約数であるためとなる整数が存在するが，とするとの位数はになる．公開鍵暗号方式の暗号化・復号のみの使用であれば位数が大きいほうが安全というだけで特に制約はないが，電子署名の利用にも使うのならば位数を素数にしなければならない．そこで，最初にとる点の選び方として，  
　(i)位数が素数になる点が見つかるまで探す．  
　(ii)の位数を求め，その約数の中で最大の素数が位数となるようなを構成する．  
のいずれかの方法を取らなければならない．

今回私が使った方法は(i)の位数が素数となる点を探す方法で，さらに位数が素体を構成する素数よりも大きくなるような点を探した．なぜこのような点を探せたかの理論は第3部第3章p.23で述べるが，楕円曲線がの形のとき，が小さければ妥当な時間内に探索するアルゴリズムを発見したからである．

ハッセの定理などよりの位数はとなる自然数が存在する．このとき位数の定義より，が成り立つ．このはで求められる．またこのが素数であるとき，は1かでなければならないため，ならばの位数は素数である．素数か否かの判定をするため以下の素数を求めておく．これはエラトステネスのふるいのアルゴリズムを用いてで求められる．理由となる性質は第3部P.31に書いているが，素体として取る素数についてのときは必ず6の倍数となり，のときとして選ばれる数字は6つに抑えられる.

以上を用いて探索をする．の値は後で逆算するとして，考えたい数字以下のとなる素数に対して適当な座標を考え，となるを探す．このが素数なら探索終了，素数でなければ次の素数を探す操作をする．6つの数字を選べば十分であるため，として1~5の計25通りを調べる操作を次々に探す．そして見つかった点について適当な整数による整数倍をシステムの決める点とすればよい．

この点を求めるアルゴリズムのオーダーは以下の素数を求めるとすると

電子署名の衝突耐性を維持するために電子署名に使用するハッシュ関数の値域よりもの位数が大きいことが期待される．しかし今回作成したコードでは，マシンの素数探索性能を考慮してハッシュ関数の値域と比べて小さな値までのみ扱っている．使ったコードでは億として

を見つけたため，これを使うこととした．

4.4で述べた新しい暗号化アルゴリズムであるが，自作インスタンスの保存の研究では使用していない．任意のデータの保存を行うとして研究も行っていたが，実装されているものを見ると仮想通貨のメッセージ部分で保存していることが多い．トークンとして任意のデータを保存できないかというときに，そのデータをこのアルゴリズムで暗号化する．このときに出てくる数字を利用して保存をしようというものである． 他にも暗号化できる平文の長さに制限がないことも1つの利点であるため，今後長さに制限がかけられない状況があったとき，この暗号アルゴリズムを使用できる．

1. 付録・参考資料
2. 付録について

今回私は楕円曲線暗号を作成するために楕円曲線について調べた．なぜ調べる必要があったかは第2部第5章(P.20)で述べたとおりだが，電子署名を施すためには素体上の楕円曲線の基準となる点の位数が素数でなければならない．素体上の楕円曲線の格子点の集合に定義される群について，位数が素数となる場合がある．このときの楕円曲線を使うことでより多くの点を持つ巡回群を構成できるため，位数が素数となる場合にはどのような条件があるのか，ということを研究した．有限群の中で考えるため位数は必ず有限であるが，それが素数となるようなものは何か，その素数と座標を見つけるアルゴリズムはどうすればよいのか，ということを主に考えながら進む．

今回は一般の楕円曲線についての理論ではなく，少し制限された形の楕円曲線の理論である．これは楕円曲線暗号を構築するために考えたものであり，全ての場合を考えなくても良かったからである．ここでどのような楕円曲線を扱うのかを述べる．

まず考える範囲がを素数として素体上の格子点で考え，方程式もを定数として，  
という式で考える．また，単位元として無限遠点を選ぶ．

1. 既知の定理や理論

これから述べるいくつかの定理や性質は既に証明された命題である．なのでページの都合により紹介と命題部分のみ記述し，証明は省略させていただく．

【定理】　有限生成アーベル群の基本定理

群が有限生成アーベル群であれば，に対し，を満たす自然数列と非負整数が一意に存在して，  
が成り立つ．

これはアーベル群に対する定理であるが，楕円曲線の有理点上に定義される群もアーベル群であるためこの定理が成り立つ．さらに，上の格子点で考えるとEの位数が有限であるためと考えてよい．

【定理】　原始根の存在定理

任意の素数に対し，を法とする原始根は存在する．

原始根とは，の点で乗して始めて1となる元のことである．原始根を使うことで全ての元を表しやすくなるため，証明に使用した．

【定理】　ルジャンドル記号・平方剰余の相互法則

ルジャンドル記号はと定義される．

を相異なる任意の奇素数として，次が成り立つ．

【定理】　オイラーの規準

素数と互いに素な任意の整数に対してが成り立つ．

証明ではまず補題2でオイラーの規準の拡張されたものを証明し，それを使うこともでてくる．しかし，オイラーの規準では両辺がのいずれかのみに限定されているため，補題とこの定理の両方を使う．

【定理】　ハッセの定理

素数を法とした楕円曲線を考える．この楕円曲線上の格子点の数をとすると，が成り立つ．

これから位数の話になるが，有限生成アーベル群の基本定理と合わせて考えると，この楕円曲線上の格子点の数，つまりこの群の位数が大事になってくる．ここで位数の存在しうる幅を示しているため，性質を考えるだけで，大体個ぐらい見つかると考えられる．また，この幅があるおかげでがより大きい素数となる点も見つかるのではないか，という期待もされるわけである．

1. 定理・性質について

以下ではを素数として，素体上の楕円曲線を考える．以下でというときはこの定義であることを省略している．

まず上の点の分布はを6で割った余りが1か5のいずれなのかで変わる．この分布の仕方の違いのせいで上の点の位数の取り方が性質の違うものになる．まずはその違いの元となっている補題と，他の多くの場所で使用されるオイラーの規準を拡張した補題を証明する．

【補題１】となる素数について以下のいずれかが成り立つ．  
 と表せるとき，  
 と表せるとき，

Proof

型のとき

と表すとする．

をルジャンドル記号とする．平方剰余の相互法則より

また，より，なので

よって，

である．オイラーの規準より，

となるので，

型のとき

と表すとする．

平方剰余の相互法則より

また，より，なので

よって，

である．オイラーの規準より，

となるので，

□

【補題2】素数について，がを割り切るときの任意の定数について以下のいずれかが成り立つ．  
が成り立つ．  
が成り立つ．

Proof

が成り立つとき

のとき

原子根の存在定理より，について原子根が存在し，が成り立つ．  
より

原子根の性質よりのときはの倍数になるので，と表せる．よって

とすると

が成り立つ．対偶を取っても成り立つ．

□

この補題以降は基本的に6で割った余りで2通りに分かれていく．まずはの場合について考える．有限生成アーベル群の基本定理よりの点の個数を考えることによりの性質が見えてくる．なので，これから上の点の個数を数えることにする．

【定理1】素数 と表せるとき，任意についてである．

Proof

についてのとき，

ここで．また， 型なので補題1より．よってオイラーの基準より，は存在しないので，

よって，

□

【定理2】素数 と表せるとき，任意の定数についてを法とした立方根がただ1つ存在する．

Proof

とおく．定理1よりは単射である．また定義域と写像先の集合が共に有限集合でその元の個数が等しいのは明らか．よって，は全単射写像である．

よって，任意の定数についてあるがただ1つ存在しが成り立つ．

□

これは素体の性質であるが，ここからへ話題をつなげることで位数を考えることが出来る．

【定理3】素数 と表せるとき，任意の定数についてを法とした楕円曲線を考える．の整数点の個数は無限遠点も含めると，個存在する．

Proof

任意のに対し，定理2よりとなるが唯一存在する．また，の取り方は通りあるので，無限遠点も含めると，個の整数点が存在する．

□

ここまででのときの個数が個だと分かった．これが1つ目に言いたかったことであった．ここから次の定理が言える．

【定理4】素数 と表せるとき，任意の定数についてを法とした楕円曲線を考える．任意の上の点の位数はの約数である．

Proof

定理3より格子点の個数は個存在しに入る群はアーベル群である．よって有限生成アーベル群の基本定理より明らか．

□

楕円曲線暗号で電子署名も使えるようにするためには，上の点のうちその位数が素数になるものを選ばなければならない．定理4より素数がと表されるとき，上の点のうちその位数が素数になる可能性のあるものの最大がである．よって選んだ素数と比べて考えられる数がとても少なくなってしまうので，もし見つかるのならばの場合のほうがよいと考えられる．

ここでの場合の楕円曲線は楕円曲線暗号には使わないほうがよいとしたが，電子署名に使わないのであれば全く問題はなく，むしろほとんど制限がなく定義できるので多くのシステムで違う暗号にしたければよい方法になると言える．次にそれを示す．

【補題5】素数 と表せるとき，任意の定数についてはの範囲で既約である．

Proof

型なので補題1より．よってオイラーの基準より，は存在しないので，に対しで，

よって，の中にの解は存在しないので，はの範囲で既約である．

□

【定理5】素数 と表せるとき，任意の定数についてを法とした式を考える．は特異点を持たない．

Proof

のとき重根を持たないことを示せばよい．任意のに対し，定理2よりとなるが唯一存在する．であるが，補題5よりには解がない．よっての解はのみで重解もないので特異点を持たない．

□

定理5より定数部分が0でない任意の式は特異点を持たないので楕円曲線として定義できている．また，について上の格子点の個数は等しい．なので，判別式を考えなくとも良いし，としてで同じ性質を持つ点を複数探すことが簡単である．

次にの場合を考える．補題1よりの数字が変わるだけで定理5までと違った結果が出てくる．

【定理6】素数 と表せるとき，で位数6の点が存在する．  
またこの数をとすると，を満たす．

Proof

補題1より 型なので，

よってオイラーの規準より，が存在する．また，である．

とおくと，

よっての位数は6で2つの条件式を満たす．

□

【系6】素数 と表せるとき，を法とした1でない1の立法根が存在する．  
またこの数をとすると，を満たす．

Proof

定理6と同じをとり，とすると上記を満たす．

□

先ほどと同様にここまではの性質であるが，ここから楕円曲線に絡めていく．

【定理7】素数 と表せるとき，任意の定数についてを法とした楕円曲線を考える．格子点が上にあるとすると，あるが存在し，相異なる6つの格子点も上にある．

Proof

系6より 型のとき1の立方根が存在する．とする

に対してとすると，

よって，も上にあること示された．

となるので，である．

□

この定理から1つ点を見つけると他の点も見つかることを示している．このほかにも点の位数の特徴を表すものも存在する．そのために新しくの部分空間を作るが，この部分空間がの性質を分割する．

【補題8】素数 と表せるとき，とするとであり，の部分空間となる．

Proof

原子根の存在定理より，について原子根が存在する．また，6はを割り切る．

についてとする．(原始根の性質より)

ここでであるので，補題2よりのとき，のときである．

また，についてとすると原始根の性質よりなので，である．

の単位元1はより

とするとが成り立つ．  
　についてより，

とするとが成り立つ．の逆元をとする．  
とすると，である．

の逆元はなので，である．

よりは部分空間である．

□

【定理8】素数 と表せるとき，とし，の同値類を作る．  
と定義すると以下は同値である．

Proof

補題8よりは定義でき，は6つの商集合に分けられる．

は同値として定めた演算の通りなので自明である．

の両辺を乗すると，

のとき，

以上よりすべて同値であると証明された．

□

【定理9】素数 と表せるとき，とし，の同値類を作る．のときの格子点の個数は等しい．

Proof

補題8よりは定義でき，は6つの商集合に分けられる．であり，6乗してとなる点は定理6より位数6の点を用いて，の6つ存在する．

格子点が存在するとき，つまりが成り立つときを考える．

のとき

定理7よりとすると6つの格子点が存在する．

とおくと

よってはの格子点となる．

同様にとしても格子点となるが，

となるが存在するので，の場合のみで十分である．

よって6点の格子点から6点の格子点が見つかる．

のとき

と同様に考えると，としての2点が格子点として現れるが，それに対応したの2点が見つかる．

のとき

と同様に考えると，としての3点が格子点として現れるが，それに対応したの3点が見つかる．

のとき

これはであることを示す．これは楕円曲線としてふさわしくないので考えない．

よりの格子点の数をと表すとすると，が成り立つ．

同様の議論によりも成り立つのでである．

つまり，の格子点の数との格子点の数は等しい．

□

以上よりの場合は定数部分に対して6通りの位数の種類があることが分かる．

【定理10】素数 と表せるとき，を定理6と同じ1の6乗根とするとである．

Proof

原始根の存在定理より，は巡回群とみなせる．となる数の個数は6つ．また定理6よりはすべて異なり，．よって，が成り立つ．

なのでが成り立つ．

□

定理9の時点で6通りしかないことを述べたが，定理8，定理10より，その性質はがのどの元かのみを考えればよいことが分かる．

【定理11】素数 と表せるとき，任意の定数についてを法とした楕円曲線を考える．上の格子点の個数について考えると6通り存在し，それらをとすると，を6で割った余りはの組み合わせとなる．また，は相異なる3つの解が存在するか，解が存在しないかのいずれかである．

Proof

定理9より格子点の個数の種類は6通り以下である．定理9，定理10より，について，としてについての場合分けをすればよい．

のとき，つまり，のとき

なので，定理7よりが成り立つ．

よって，が上に存在し，座標に0を含む点はこれらのみである．また，他の点があったとき，定理3より6つずつの組として存在する．この6点ずつの組数をとする．

も含めて格子点の個数はとなるので，個数を6で割った余りは0である．

また，このときは相異なるので重解を持たない．

のとき

なので，定理7よりが成り立つ．

よって，座標のいずれかが0の点は上に存在しない．また，他の点があったとき，定理3より6つずつの組として存在する．この6点ずつの組数をとする．

も含めて格子点の個数はとなるので，個数を6で割った余りは1である．

また，このときは解を持たない．

のとき

なので，定理7よりが成り立つ．

よって，が上に存在し，座標が0の点はこれらのみであり，座標が0の点は存在しない．また，他の点があったとき，定理3より6つずつの組として存在する．この6点ずつの組数をとする．

も含めて格子点の個数はとなるので，個数を6で割った余りは3である．

また，このときは解を持たない．

のとき

なので，定理7よりが成り立つ．

よって，が上に存在し，座標が0の点はこれらのみであり，座標が0の点は存在しない．また他の点があったとき，定理3より6つずつの組として存在する．この6点ずつの組数をとする．

も含めて格子点の個数はとなるので，個数を6で割った余りは4である．

また，このときは相異なるので重解を持たない．

のとき

なので，定理7よりが成り立つ．

よって，が上に存在し，座標が0の点はこれらのみであり，座標が0の点は存在しない．また，他の点があったとき，定理3より6つずつの組として存在する．この6点ずつの組数をとする．

も含めて格子点の個数はとなるので，個数を6で割った余りは3である．

また，このときは解を持たない．

のとき

なので，定理7よりが成り立つ．

よって，座標のいずれかが0の点は上に存在しない．また，他の点があったとき，定理3より6つずつの組として存在する．この6点ずつの組数をとする．

も含めて格子点の個数はとなるので，個数を6で割った余りは1である．

また，このときは解を持たない．

□

定理11よりのときの位数の性質が分かった．5以上の素数は全て6で割った時の余りが1または5なので，各素数のうち，となる2つの定数のみ考えれば位数が素数の点で，大きい数になるものが見つかるかもしれないといえる．さらに，ハッセの定理よりそのような位数の点が見つかるのであれば，に近い大きな素数を見つけることが出来る．

1. 記述したコード

HTML文書やインポートするパッケージなどを多く含むプロジェクトになっている．そのファイルツリーを以下に示す．この通りに配置しないと動かないので注意が必要である．hogeディレクトリはコマンドプロンプトで実行するなら任意のディレクリトリでかまわない．  
hoge ┳ blockchain\_Star.py  
 ┣ my\_ECC.py  
 ┣ my\_Graph.py  
 ┣ templates ━ 8つのHTMLファイル  
 ┗ static ┳ style.css  
 ┣ Logo.png  
 ┣ favicon.ico  
 ┗ back\_img.jpg

Apacheで実行するならば次の/etc/httpd/conf.d/wsgi.confと/var/www/blockchainを変更すればよい．  
/ ┳ etc ━ httpd ┳ conf ━ httpd.conf  
 ┃ ┣ conf.d ━ wsgi.conf  
 ┃ ┣ modules ━ モジュールファイル  
 ┃ ┗ conf.modules.d  
 ┗ var ┳ log  
 ┗ www ━ blockchain ┳ blockchain\_Star.py  
 ┣ my\_ECC.py  
 ┣ my\_Graph.py  
 ┣ blockchain.wsgi  
 ┣ templates ━ 8つのHTMLファイル  
 ┗ static ┳ style.css  
 ┣ Logo.png  
 ┣ favicon.ico  
 ┗ back\_img.jpg

/etc/httpd/conf.d/wsgi.confのコード

<VirtualHost \*:80>

ServerName 172.21.21.171

LoadModule wsgi\_module /usr/lib64/python3.6/site-packages/mod\_wsgi/server/mod\_wsgi-py36.cpython-36m-x86\_64-linux-gnu.so

#LoadModule wsgi\_module /etc/httpd/modules/mod\_wsgi.so

WSGIDaemonProcess blockchain user=apache group=apache

WSGIProcessGroup blockchain

WSGIScriptAlias / /var/www/blockchain/blockchain.wsgi

WSGIScriptReloading On

DocumentRoot /var/www/blockchain

<Directory /var/www/blockchain>

#Allowoverride All

#WSGIProcessGroup blockchain

#Options +ExecCGI

Options ExecCGI MultiViews Indexes

MultiViewsMatch Handlers

WSGIApplicationGroup %{GLOBAL}

AddHandler wsgi-script .py

AddHandler wsgi-script .wsgi

#Order allow, deny

#Allow from all

Require all granted

</Directory>

</VirtualHost>

/var/www/blockchain/blockchain.confのコード

#coding: utf-8

import sys, os

sys.path.append('/var/www/blockchain/')

from blockchain import app as application

楕円曲線暗号に使用する素数と点の探索

素数と点を探索するコードが以下である．

from math import sqrt

def primes(limit:int)->list:

"""

エラトステネスの篩のアルゴリズムでlimit以下の素数をリストアップする．

"""

array=list(range(3, limit, 2))

ans=[2]

l=sqrt(limit)

while ans[-1]<l:

ans.append(array[0])

array=[i for i in array if i%ans[-1]!=0]

return ans+array

def Order(G:tuple, mod:int)->int:

"""

Gのmodを法とした楕円曲線でのnG=Oとなる数nを求める．nは位数であることが多いが位数の倍数になっているときもある．ただし，|n-(mod-1)|<2√(mod)を満たす．

"""

plus=G

ans=mult(mod, G, mod)

if G==ans:

return mod-1

if ans==(0, 0):

return mod

lim=int(sqrt(mod)\*2+5)

for i in range(2, lim):

G=add(G, plus, mod)

if G==ans:

return mod-i

G=ans

for i in range(1, lim):

G=add(G, plus, mod)

if G==(0, 0):

return mod+i

次のコードを一度回すことで以下の素数を法としてが素数となる組を5つ得る．

p=60\*\*8-5

p\_list=primes(int(sqrt(p+1+2\*sqrt(p))))

k=0

while k<5:

while not is\_prime(p, p\_list): p-=6

m=0

n\_list=[35, 35, 35, 35, 35, 35, 35, 35, 35, 35]

for i in range(1, 5):

for j in range(1, 5):

if (j\*\*2-i\*\*3)%p!=0:

m\_list=[mult(n, (i, j), p) for n in n\_list]

if not (0, 0) in m\_list:

n\_list[m]=Order((i, j), p)

m+=1

for n in n\_list:

if is\_prime(n, p\_list):

　print(p, n)

k+=1

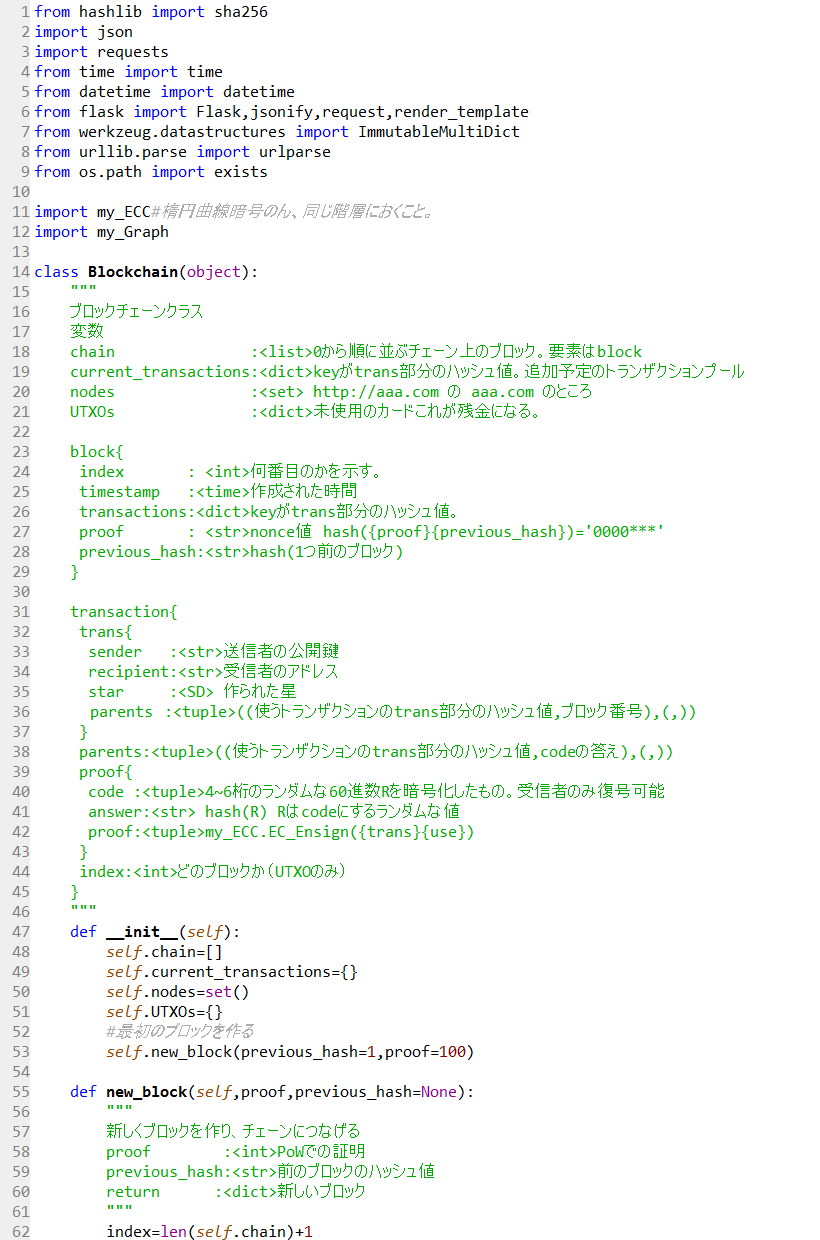
p-=6

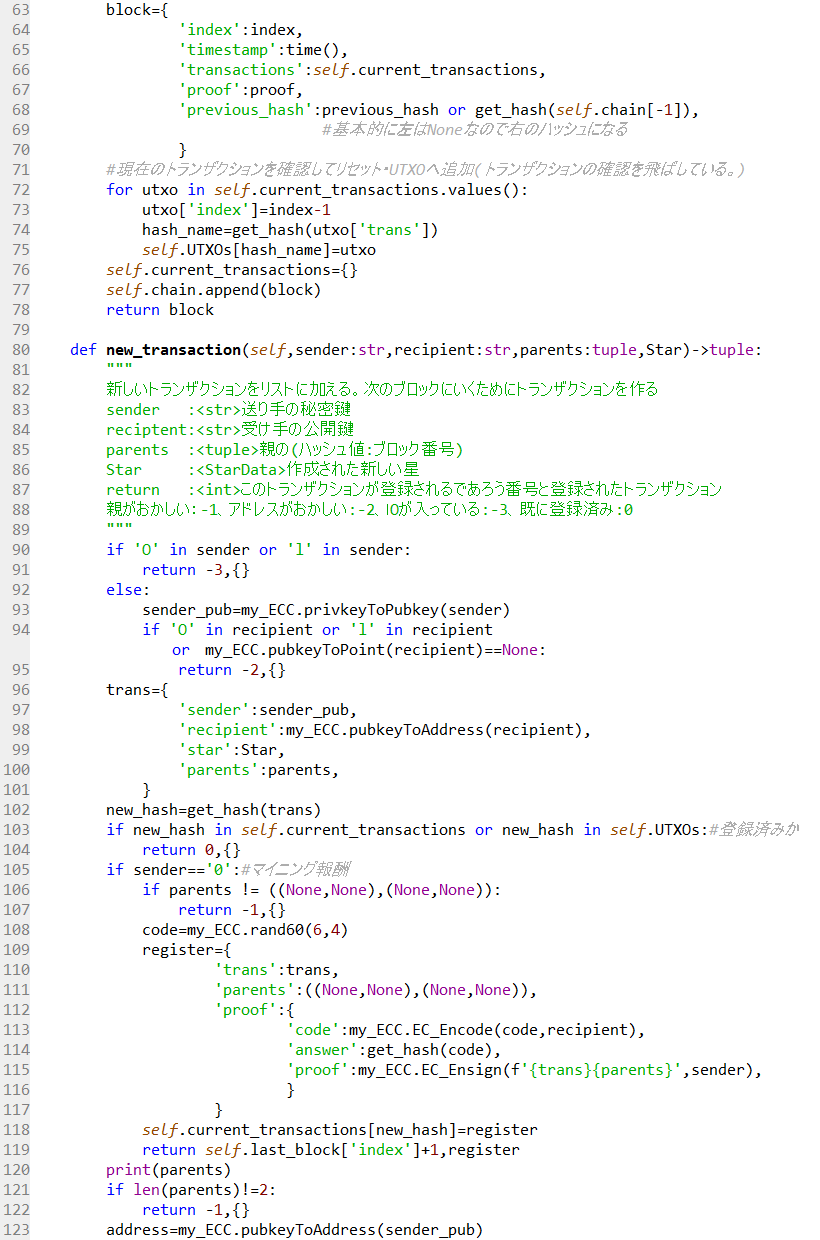
5つの候補のうちとなる  
から

を構成し，これらを使うことにした．

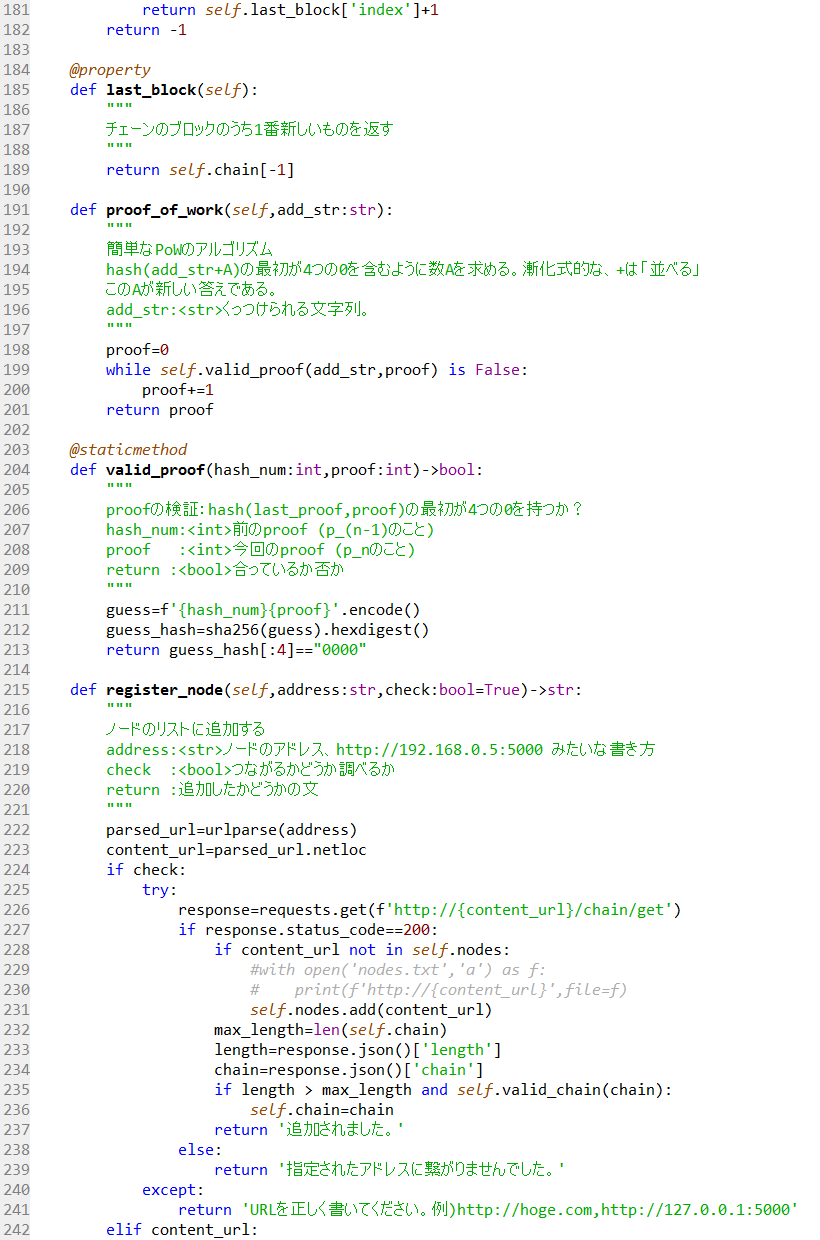
blockchain\_Star.pyのコード

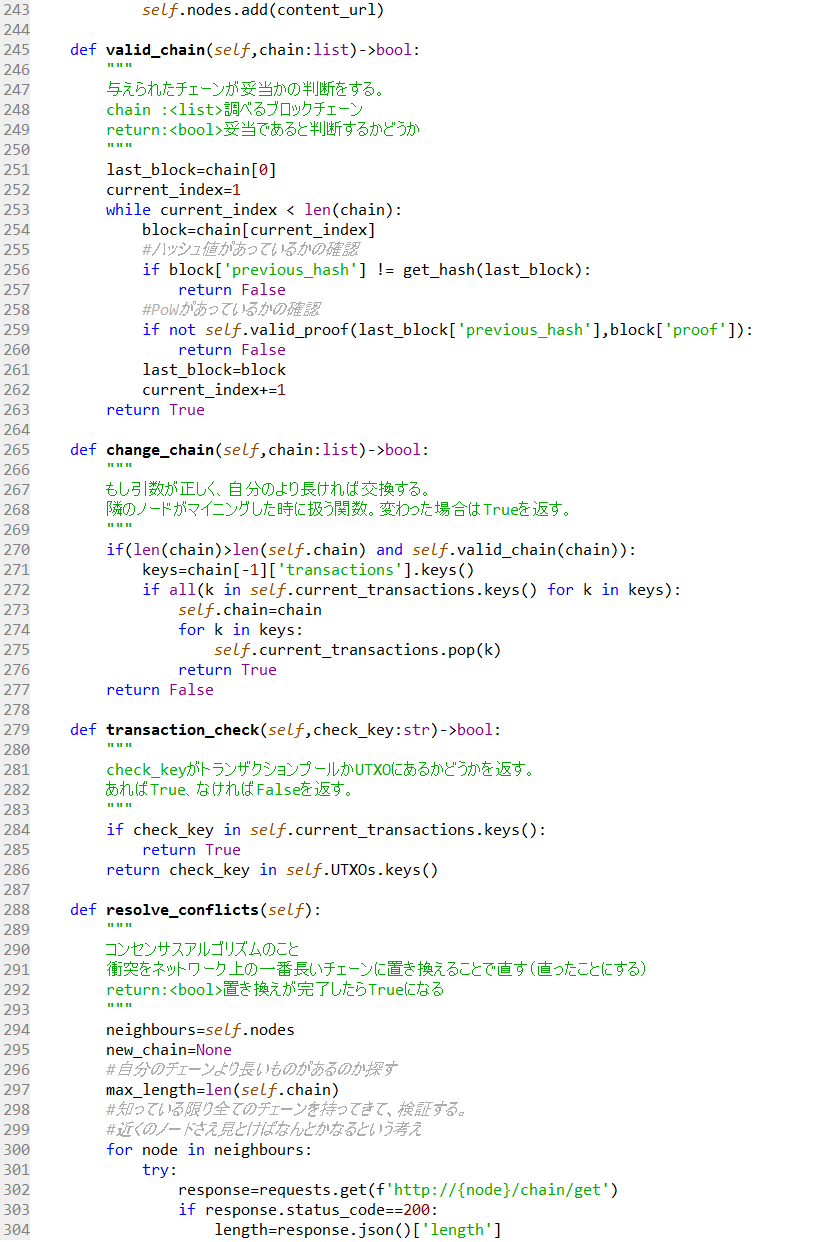
[10]が元のコードである．本研究のための実装にあたって大きく変更した部分が多いが，参考にしている箇所が多い．

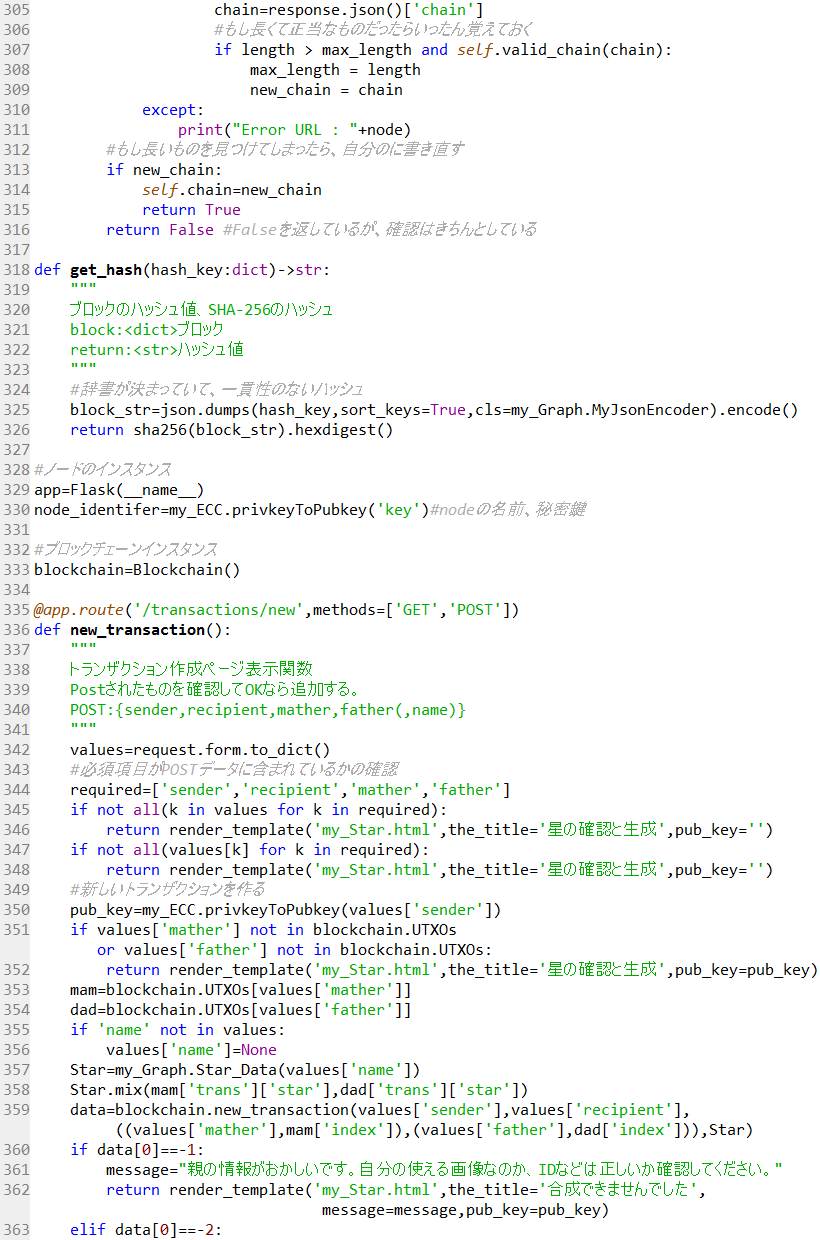








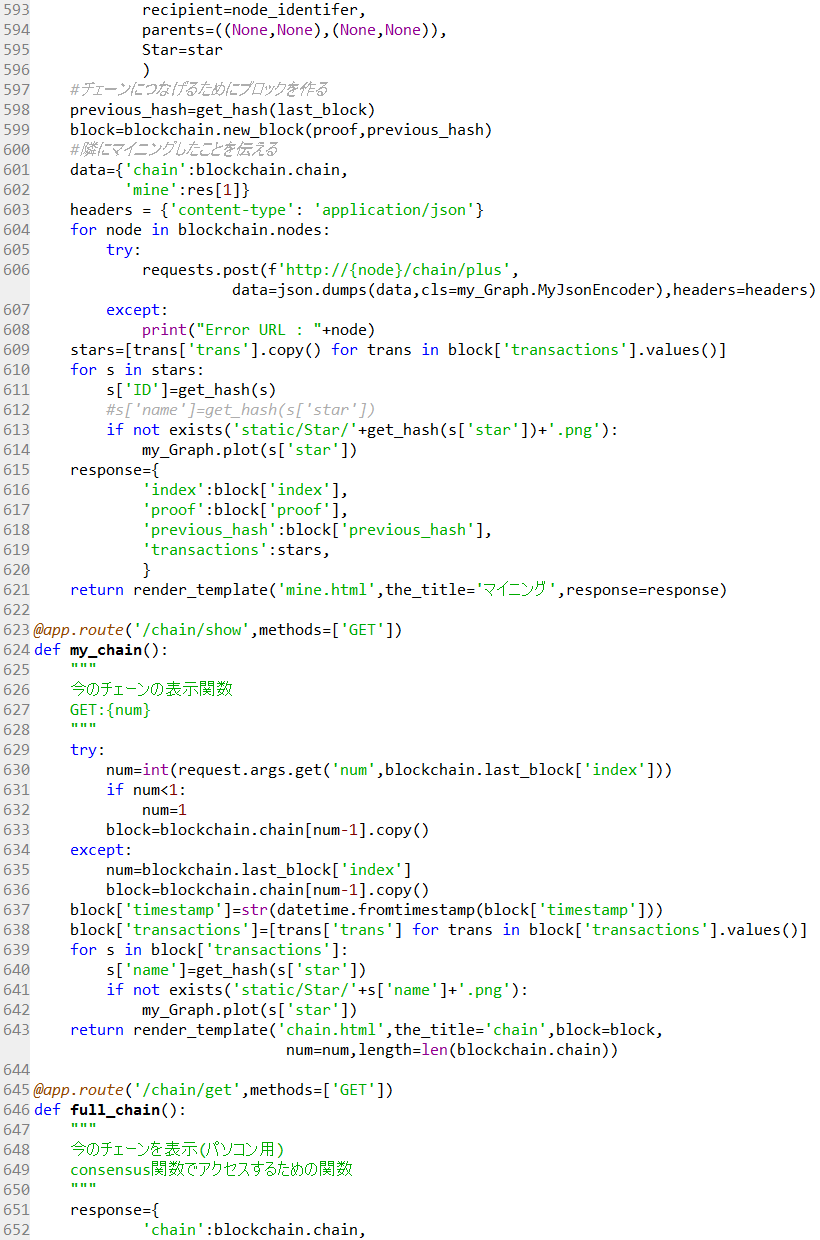






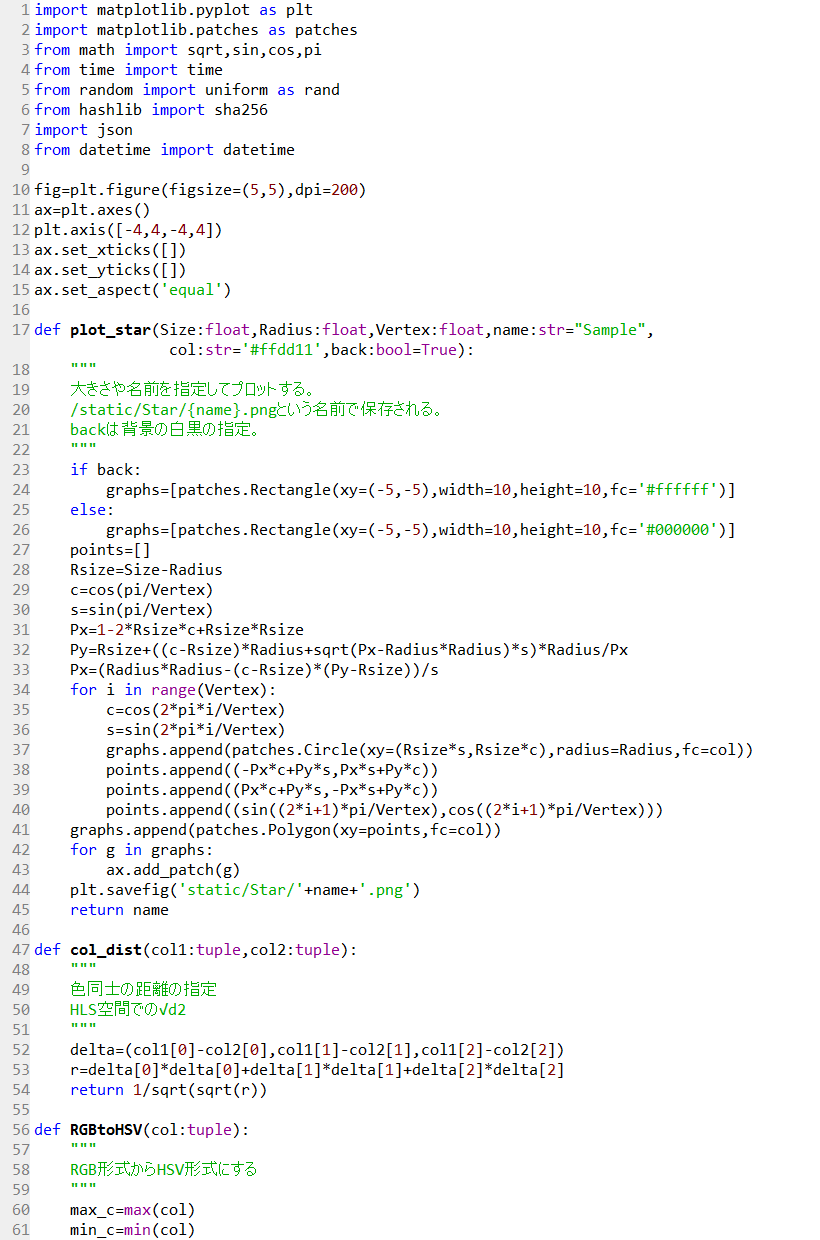




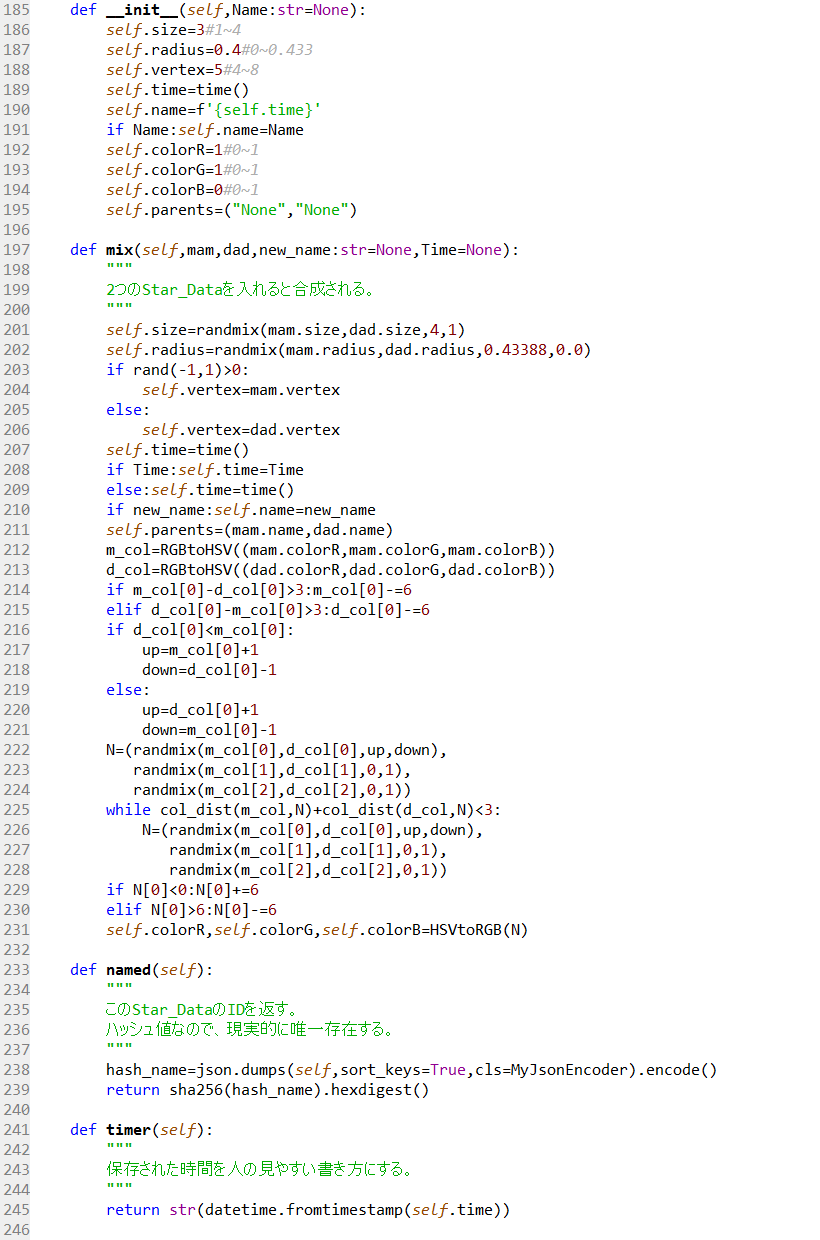
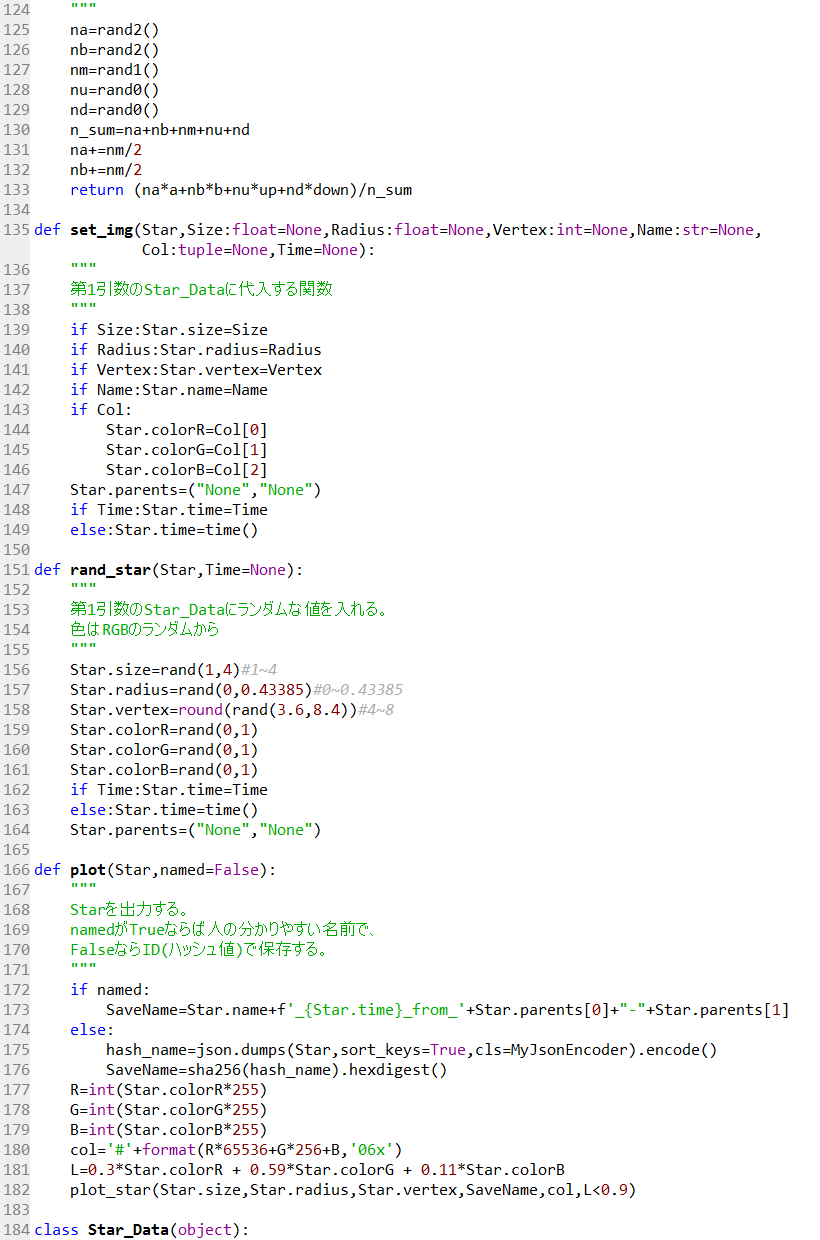




my\_Graph.pyのコード



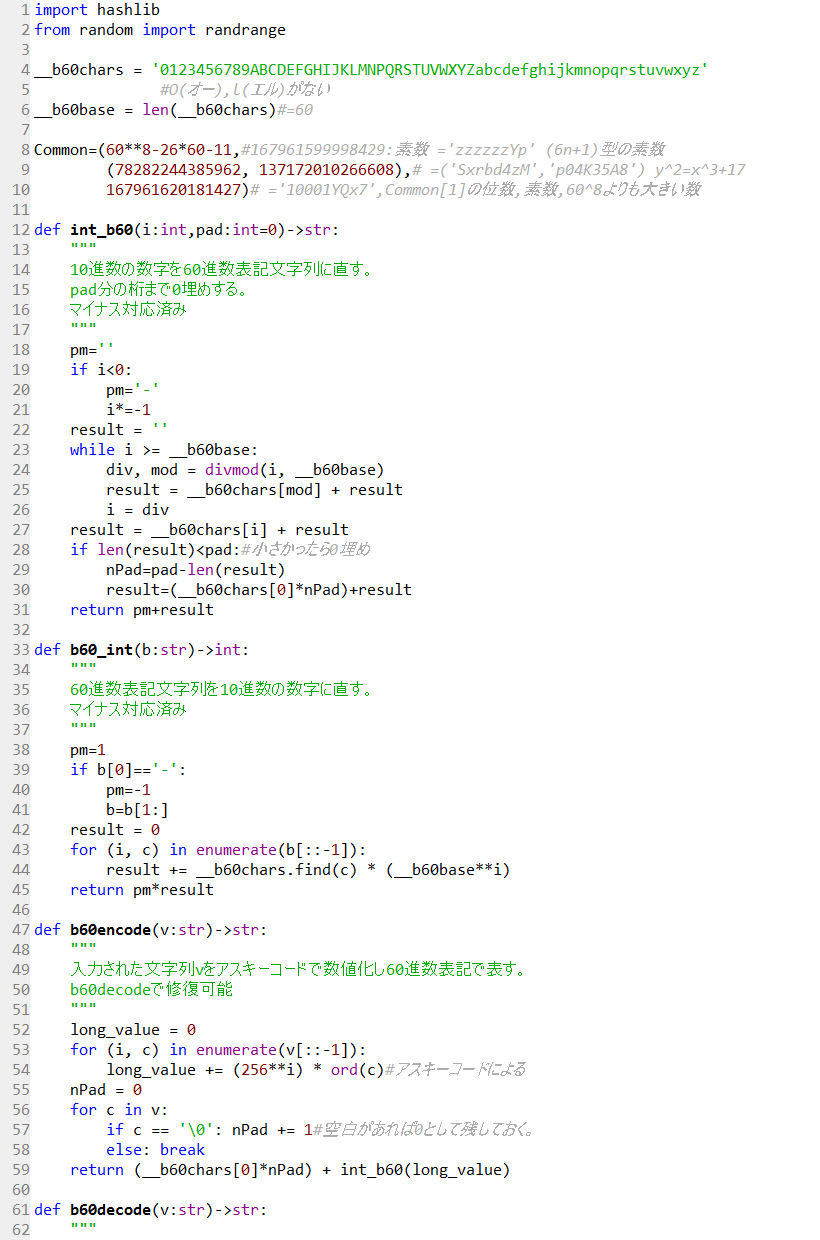




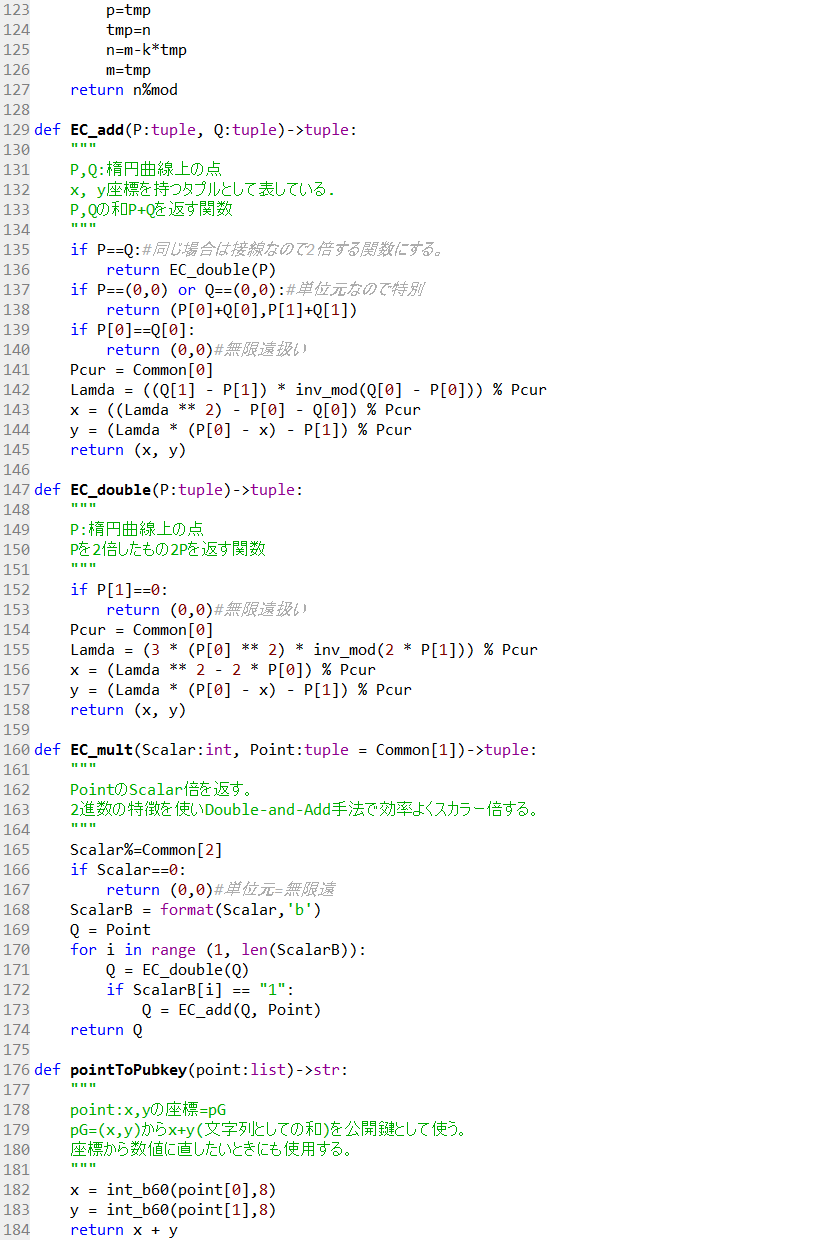


my\_ECC.pyのコード

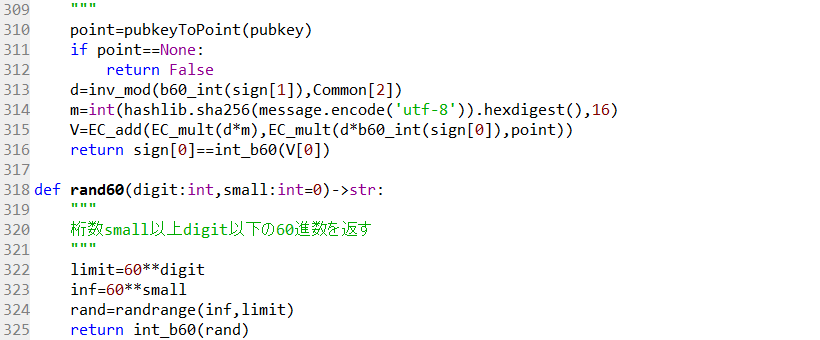
[11]を参考にしている．[11]では無限遠点を考慮していない等の箇所があるので，その修正なども行っている．







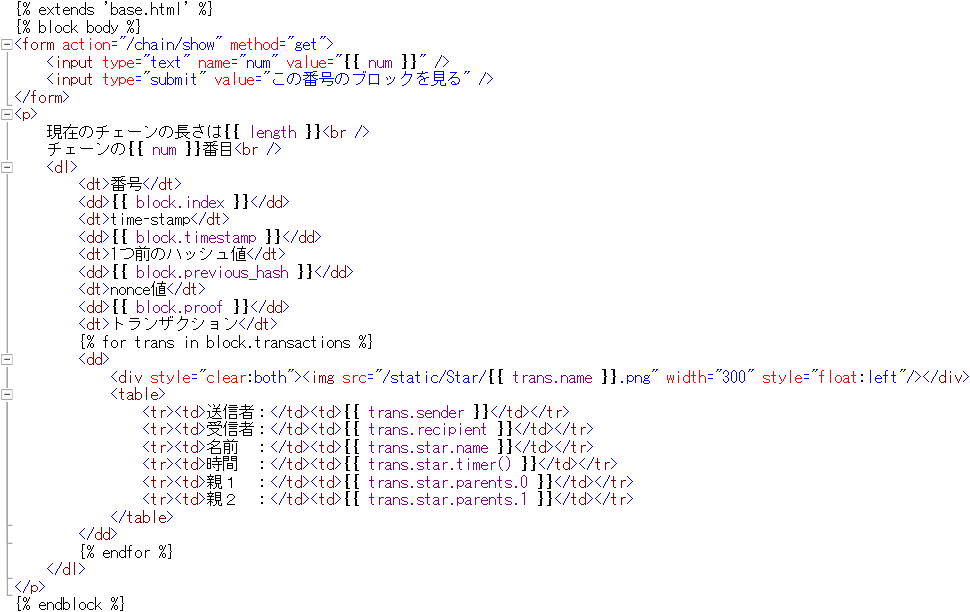




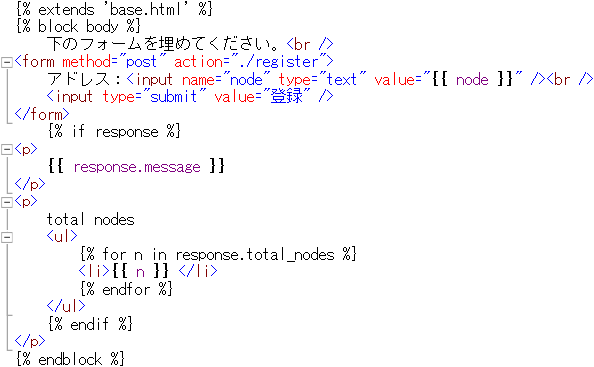
base.htmlのコード



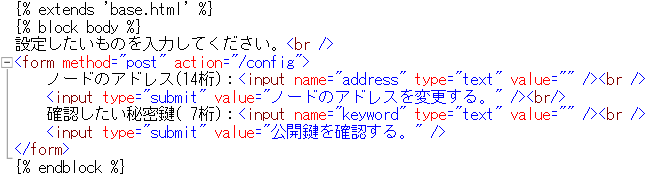
chain.htmlのコード



nodes\_set.htmlのコード



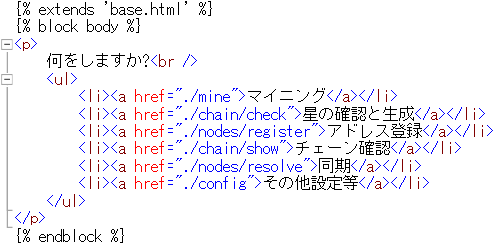
config.htmlのコード



my\_Star.htmlのコード



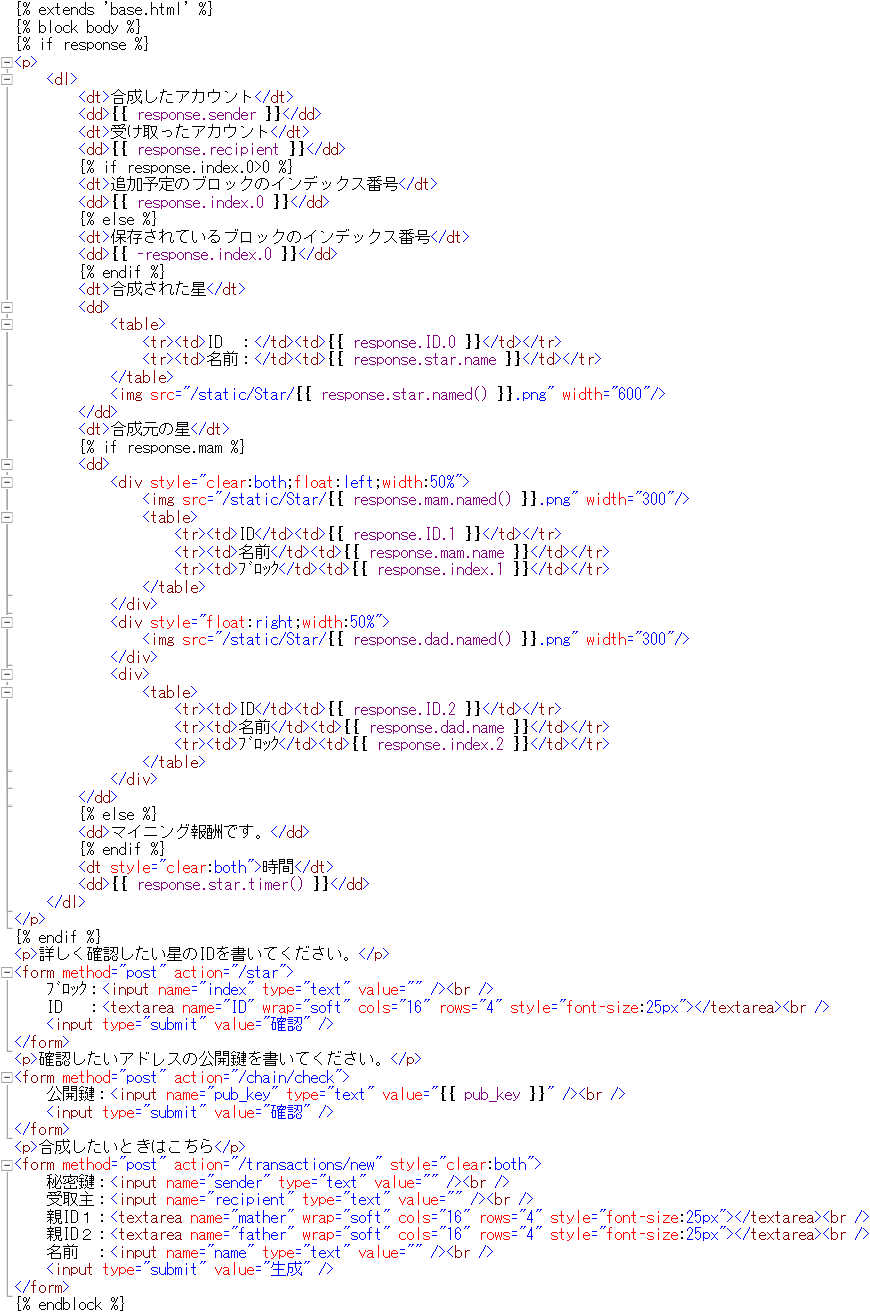
menu.htmlのコード



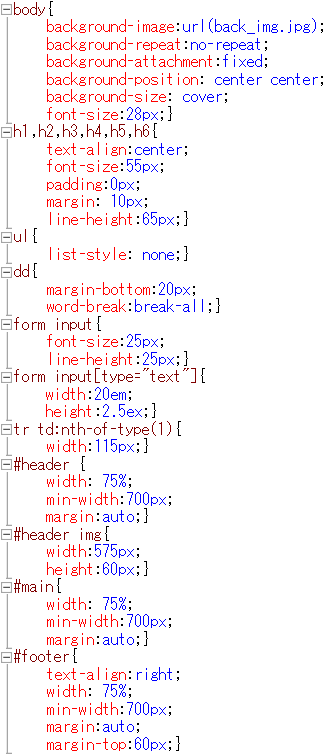
mine.htmlのコード



new\_Star.htmlのコード



style.cssのコード



謝辞

指導教官の溝畑教授，津田教授には卒業論文のみならず，大学生活全体を通して力強いご指導ご鞭撻を，研究室配属後にはより他分野の知識のご教授を賜った．この場を借りて厚く感謝の意を表したい．

参考文献

[1] 大藪俊哉(1984) 「<研究ノート>複式簿記から三式簿記へ：井尻雄二著「三式簿記の研究」を中心にして」，横浜経営研究5(3)，p.296-301，横浜国立大学．

[2] 永田清(1998)「暗号と楕円曲線：鍵交換，文書の暗号化，電子署名」，パソコンリテラシ23(6)，p.15-20．

[3] 帳睿暎(2018)「著作権登録およびコンテンツ利用におけるブロックチェーン技術の活用可能性と課題」，獨協法学105，p.231-256，獨協大学法学会．

[4] 小川健(2018)「サーベイ論文：非技術／情報系の経済系に仮想通貨・ビットコイン・ブロックチェーンをいかに教えるか」，専修経済学論集52(3)，p.167-182，専修大学経済学会．

[5] 堀内啓次，布田裕一，境隆一，金子昌信，衣笠正雄(1999)「素数位数を有する楕円曲線の構成とその計算量評価」，電子情報通信学会論文誌J82-A(8)，p.1269-1277，電子情報通信学会．

[6] 大竹龍史，市木秀男，山本道子，山崎佳子(2018)『標準テキストCentOS7構築・運用・管理パーフェクトガイド』ナレッジデザイン．

[7] 杉井靖典(2017)『いちばんやさしいブロックチェーンの教本―人気講師が教えるビットコインを支える仕組み』インブレス．

[8] バリー，ポール(2018)『Head First Python 第2版 ―頭とからだで覚えるPythonの基本』嶋田健志監修，木下哲也訳，オライリー・ジャパン．

[9] 光成滋生(2016)「楕円曲線入門　トーラスと楕円曲線のつながり」https://www.slideshare.net/herumi/ss-58815597　2018年6月アクセス

[10] ダニエル　バン　フライマン(2017)　「ブロックチェーンを構築しながら学ぶ」https://postd.cc/learn-blockchains-by-building-one/ 2018年4月アクセス

[11] Jonathan Underwood(2015)「技術者向けビットコイン講座 第1回 楕円曲線を自分の手で扱う」https://btcnews.jp/elliptc-curve-diy/　2018年6月アクセス

[12] Qiita@hatai「Python Windowsでmod\_wsgiをApacheに組み込む方法」https://qiita.com/hatai/items/279ec1dfd28cb9693a1f 2018年5月アクセス

[13] Hatena Blog id:segafreder (2017)「【Python】jsonで自作クラスを含んだデータをシリアライズ/デシリアライズする」http://segafreder.hatenablog.com/entry/2017/10/01/140125 2018年11月 アクセス

[14] Qiita @sh-miyoshi(2016)「楕円曲線暗号の超簡単な理論の紹介」https://qiita.com/sh-miyoshi/items/7479f6e647a324638b9a　2018年6月アクセス

[15] 光成滋生(2015)「鍵が漏れることも想定せよ――クラウド時代における「楕円曲線暗号」の必然性」http://www.atmarkit.co.jp/ait/articles/1507/28/news002.html　2018年6月アクセス

[16] Aram Mine(2017)「楕円曲線暗号　秘密鍵から公開鍵を生成するアルゴリズム」https://gaiax-blockchain.com/elliptic-curve-cryptography　2018年6月アクセス