

# ブロックチェーン公開講座 第2回

## ビットコインその1

---

芝野恭平

東京大学大学院工学系研究科技術経営戦略学専攻

ブロックチェーンイノベーション寄付講座

特任研究員

shibano@tmi.t.u-tokyo.ac.jp

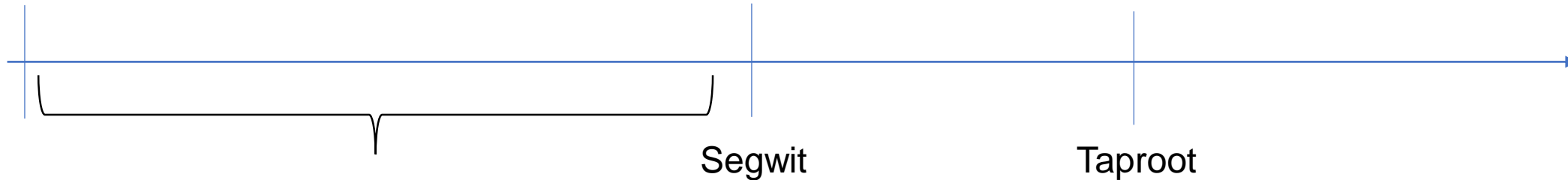


# ビットコインプロトコルの主なアップグレード

2009年1月

2017年8月

2021年11月



Taproot

- ・ マークル化抽象構文木 (MAST)
- ・ シュノア署名



講義ではこの初期のビットコインプロトコルを取り扱います.

→ より基本的なブロックチェーンの原理を学ぶことが目的.



# 教材

- Mastering Bitcoin (second edition)
  - [https://github.com/bitcoinbook/bitcoinbook/tree/second\\_edition\\_print3](https://github.com/bitcoinbook/bitcoinbook/tree/second_edition_print3)
- ビットコインとブロックチェーン:暗号通貨を支える技術 (Mastering Bitcoin日本語訳)
  - <https://www.amazon.co.jp/dp/B072JL66R>

 README  License

## Mastering Bitcoin

Mastering Bitcoin is a technical book that explains what Bitcoin is and how it works.

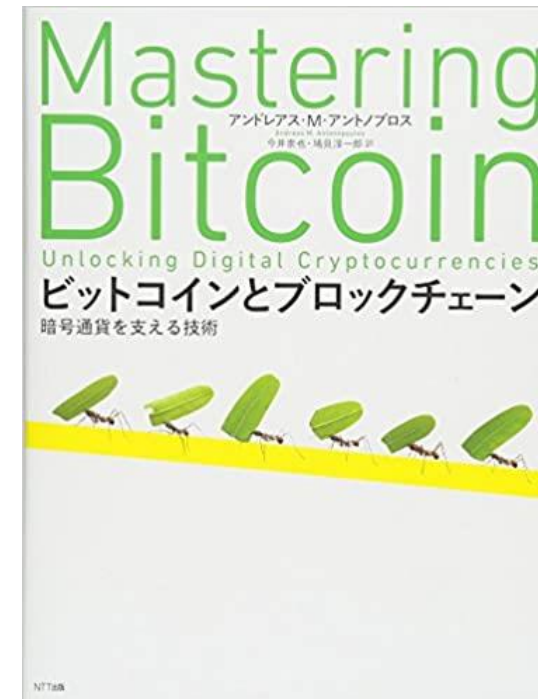
This repository contains the complete text of three editions of the book Mastering Bitcoin as published in paperback and ebook formats by O'Reilly Media. Different editions of this book are covered by different op licenses (see [LICENSE](#)). The three editions are:

- The [first edition, second print](#), published in December 2014
- The [second edition, third print](#), published in March 2018
- The [third edition, first print](#), published in December 2023

## Reading This Book

To read this book *for free*, see [BOOK.md](#). Click on each of the chapters to read in your browser.

*Please note that some of the links in each chapter do not work when reading the book on Github. This is because those links are intended for the print and ebook editions of the book and only work when all the chapters are rendered together. Unfortunately, Github does not have the ability to render the complete book at once.*



# ビットコインの仕組み

---



## 概要

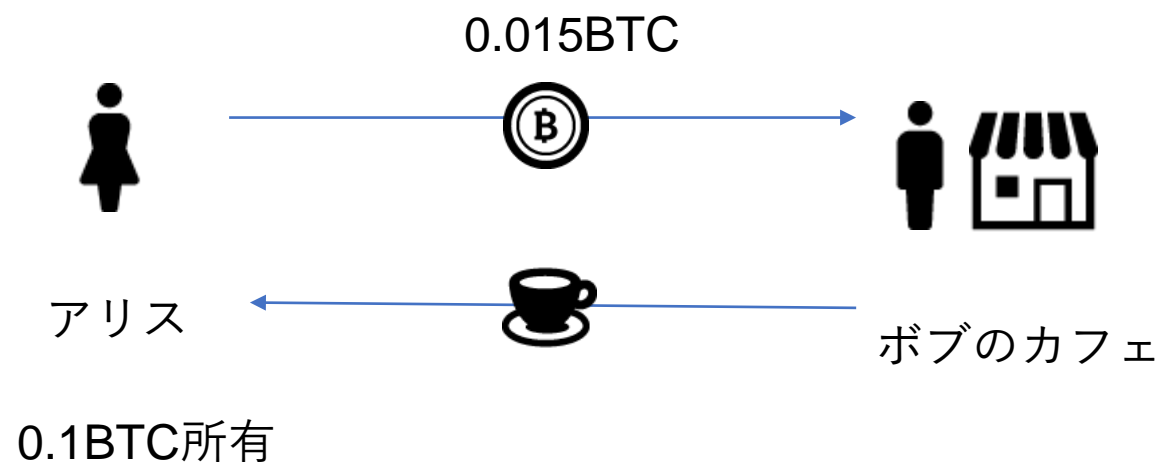
---

- アリスがボブへ送金する例を考える.
- このときビットコインはシステムの的にどのように動作して送金を実現しているのか概観する.
  - ウォレットと鍵
  - トランザクション作成
  - ビットコインネットワークにブロードキャスト
  - マイニングによるブロック生成, 伝搬



## コーヒー代金の支払いをビットコインで行う

- アリスはボブのカフェでコーヒーを注文した.
- アリスはもともと0.1BTC持っていた.
- コーヒーの代金として0.015BTCをボブに支払った.



## ハッシュ関数とは

- 一方向関数
- 順方向は一瞬で値が出力される
- 逆方向は求めることが困難.
- ハッシュ関数はいくつも形式がある
  - MD5, SHA1, SHA256, . . .
- SHA256がよく使用される.
  - 暗号的ハッシュ関数
    - 現像計算困難性 . . . ハッシュ値から入力値を求めるのが困難
    - 第二現像計算困難性 . . . ある入力値が与えられたとき, 同じハッシュ値を持つ異なる入力値を求めるのが困難
    - 強衝突耐性 . . . 同じハッシュ値となる2つの異なる入力値を求めるのが困難

```
shibano@user-pc:~/git/doc_blockchain$ time echo -n 'test' | shasum -a 256
9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08 -
real    0m0.048s
user    0m0.031s
sys     0m0.016s
```

あいいうえお



fdb481ea956fdb654afcc327cff9b626966b2abdabc3f3e6dbcb1667a888ed9a

あいいうえこ



59d9b0d8d3e1262bd220ebc58989569c064b48df22c28dfcc3fd2186b484af34

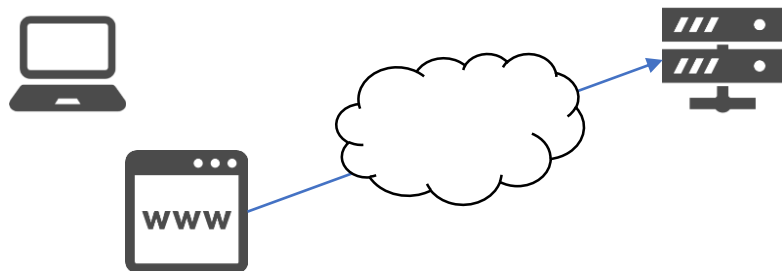
あいいうおえ



a762518a8b8eff0175e1e0e9493594db95b23aba9f04dae7f52621c1dafddea9

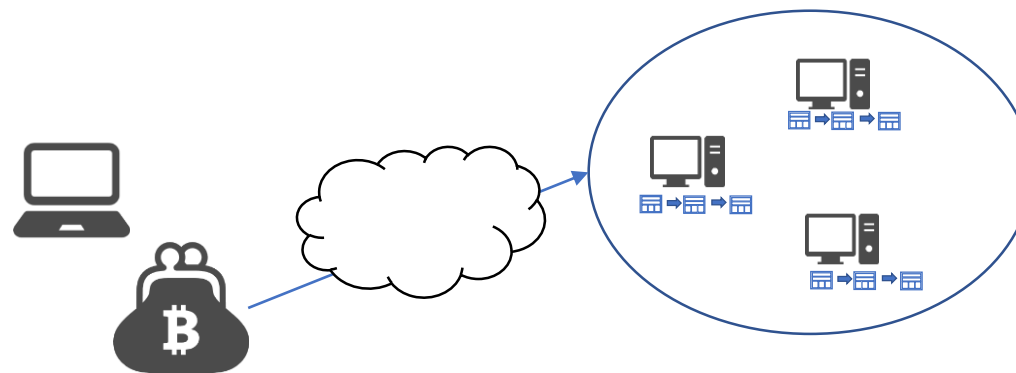
# 一般的なWebシステムとビットコインシステム

## Webのシステム



ブラウザを使って、HTTPリクエストを送信するとWebサーバーはレスポンスを返す。

## ビットコインのシステム



ウォレットソフトウェアを使ってトランザクションを生成して、ビットコインネットワークにブロードキャストする。しばらく待つと、ブロックチェーンに取り込まれる。



# ビットコイン概観

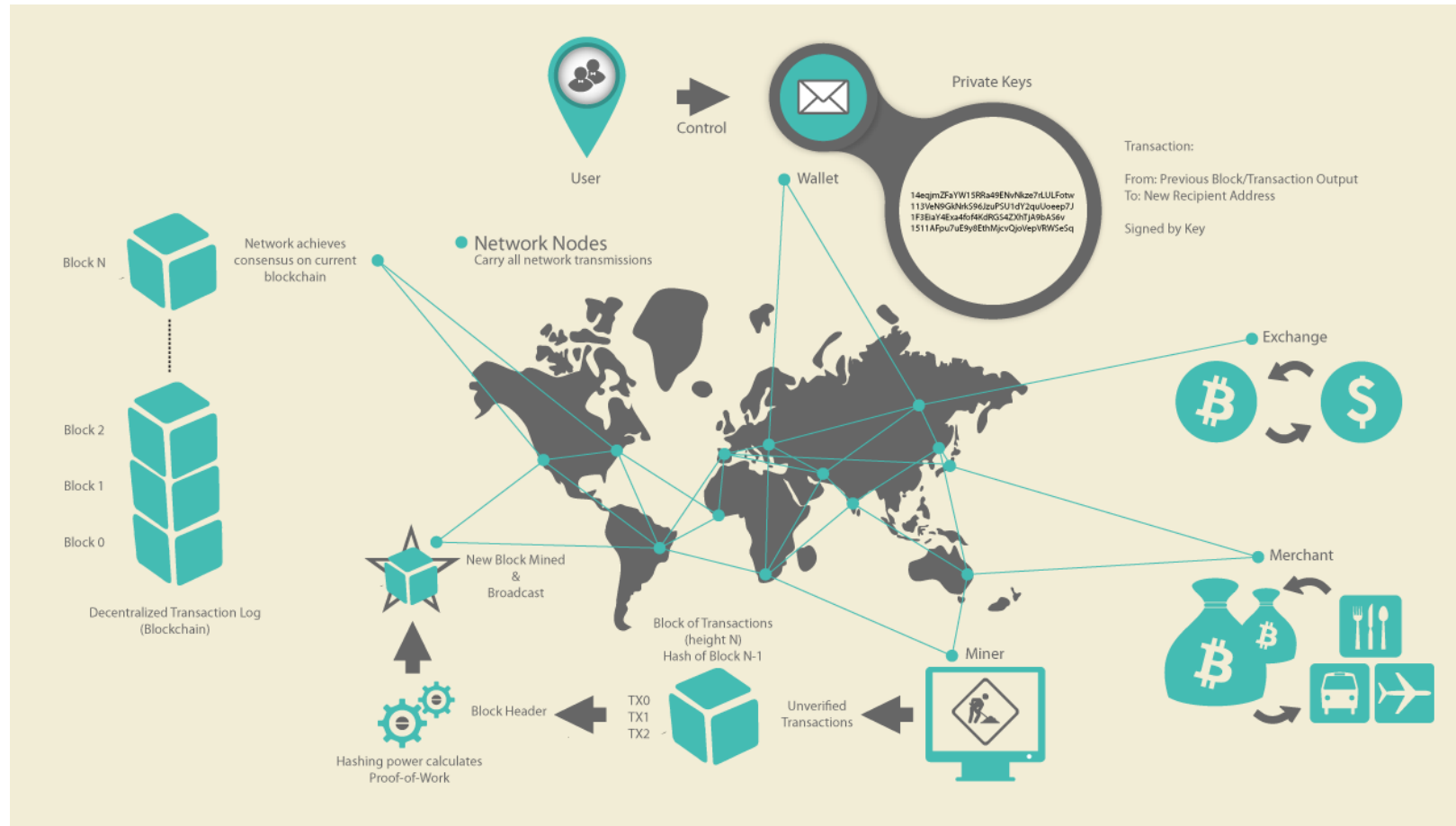
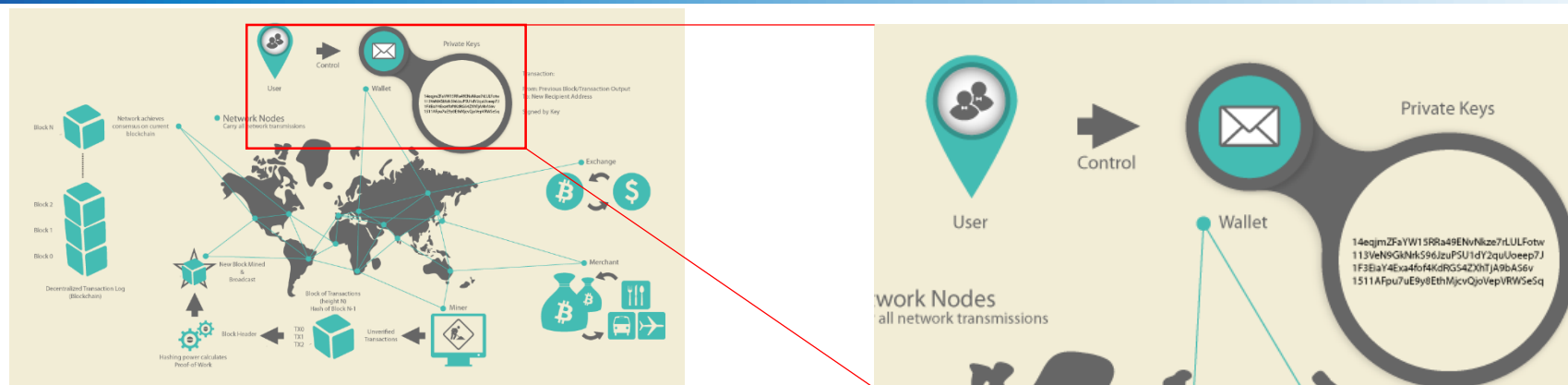


Figure 1. Bitcoin overview

[https://github.com/bitcoinbook/bitcoinbook/blob/second\\_edition\\_print3/ch02.asciidoc](https://github.com/bitcoinbook/bitcoinbook/blob/second_edition_print3/ch02.asciidoc)

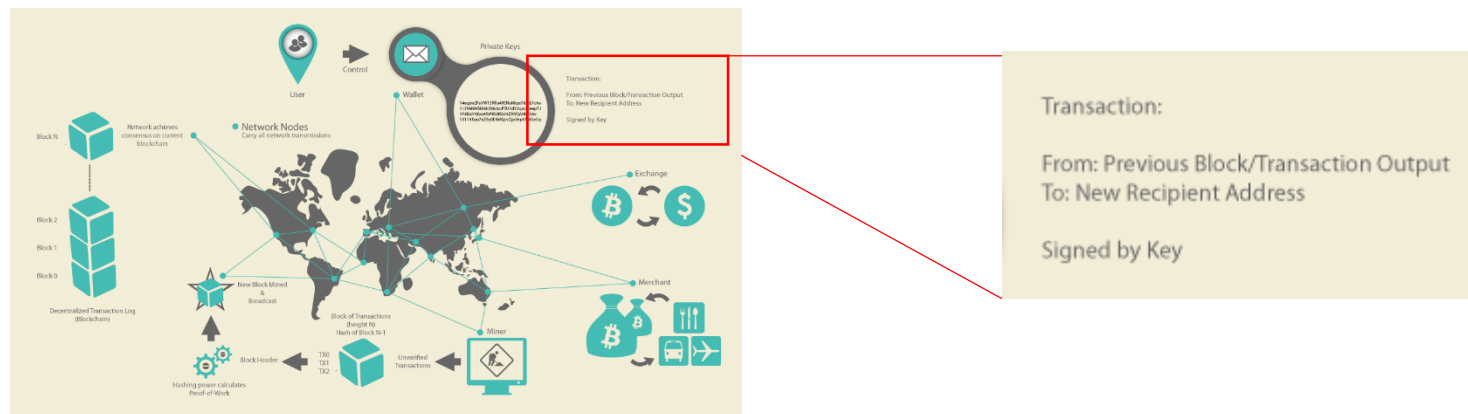
## ユーザーとウォレット，秘密鍵



- ウォレットの利用
  - Bitcoinの保有者は，自身のアドレスと，それに紐づく秘密鍵，それらを管理するウォレットを使用します。
- アドレス
  - ・・・ ビットコインを利用するための口座番号のようなもの．ユーザーはEメールをだれかのメールアドレスに送信するのと同様に，ほかの人が持っているアドレスに送金します。
- 秘密鍵
  - ・・・ 自身のアドレスから送金を行うために必要。
- ウォレット
  - ・・・ 自身のアドレスの残高を確認したり，秘密鍵を管理したり，送金を行うことができるソフトウェア。

「アリスは，自身の秘密鍵が管理されているウォレットを使ってボブのアドレス宛に0.015BTC送金するトランザクションを生成する」

# トランザクション



ウォレットにて作られたトランザクションは、ビットコインネットワークに送信される。

トランザクションには

- どのトランザクションから
- どのアドレスに
- いくら

送金するかという情報を含み、送金元の電子署名も含む。

## アリスからボブへの送金トランザクション

ビットコインはUTXO (Unspent Transaction Output)モデルを採用しており、未使用のトランザクションから新しいトランザクションを作る。

- アドレスごとに残高が記録されているわけではない
- アドレスからではなく未使用のトランザクションから作る

アリスへ送金されたトランザクション

Transaction 7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18	
INPUTS From	OUTPUTS To
From (previous transactions Joe has received): Joe 0.1000 BTC	Output #0 Alice's Address 0.1000 BTC (spent) Transaction Fees: 0.0000 BTC

作成するトランザクション

Transaction 0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2	
INPUTS From	OUTPUTS To
7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18 : 0 Alice 0.1000 BTC	Output #0 Bob's Address 0.0150 BTC (spent) Output #1 Alice's Address (change) 0.0845 BTC (unspent) Transaction Fees: 0.0005 BTC

Transaction 2bbac8bb3a57a2363407ac8c16a67015ed2e88a4388af58cf90299e0744d3de4	
INPUTS From	OUTPUTS To
0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2 : 0 Bob 0.0150 BTC	Output #0 Gopesh's Address 0.0100 BTC (unspent) Output #1 Bob's Address (change) 0.0045 BTC (unspent) Transaction Fees: 0.0005 BTC

Figure 4. A chain of transactions, where the output of one transaction is the input of the next transaction

## トランザクションの形式

- 一般的なトランザクション
  - 1つのアドレスから1つのアドレスに送金
  - 元のアドレスへのおつり送金が含まれる
- 集約型トランザクション
  - 複数のインプットを集めて1つのアウトプットにまとめる
  - おつりとして受け取った小額トランザクションをまとめるときに使われる
- 分配型トランザクション
  - 1つのインプットから複数のアウトプットに分ける
  - 給与の支払いなどに使われる

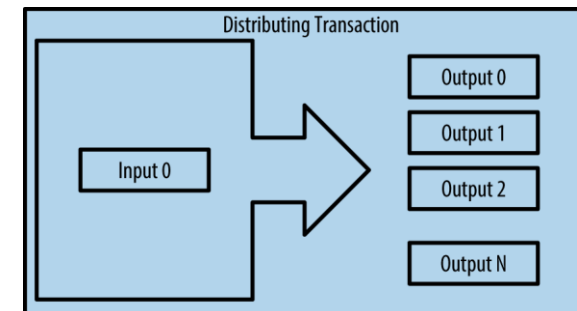
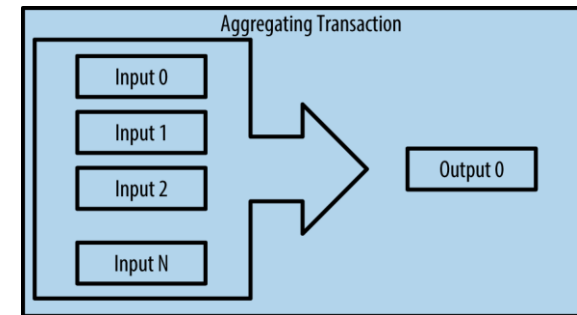
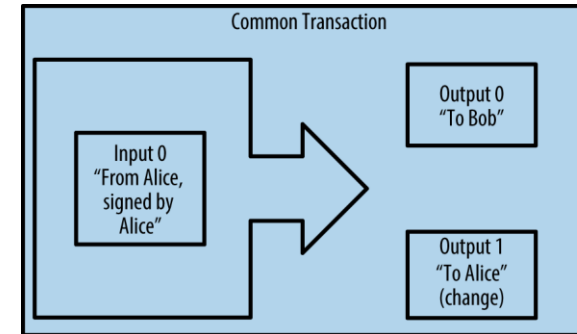


Figure 5. Most common transaction  
Figure 6. Transaction aggregating funds  
Figure 7. Transaction distributing funds

## 正しいインプットをどのように得るか

- ほとんどのウォレットはブロックチェーンデータとは別に、UTXO（Unspent Transaction Output, 未使用トランザクションアウトプット）を保持するデータベースを内部に持っている
  - ブロックチェーンのデータをもとにDBを生成
  - そうしないと最初から最後までブロック(80万超え, 2024-04時点)に含まれるTxすべてをチェックしないといけなくなる.
- フルインデックスウォレットの場合：
  - フルノード（全ブロックチェーンデータを持っている）
  - ブロックチェーンデータより全アドレスに対するUTXOデータベースを生成可能
  - フルノードなので、データ量が非常に大きくなる.
    - データ量は600GB強.
- 軽量ウォレットの場合：
  - 持ち主のUTXOデータベースのみ保持
  - ビットコインネットワークに問い合わせで生成する
    - フルノードに問い合わせる

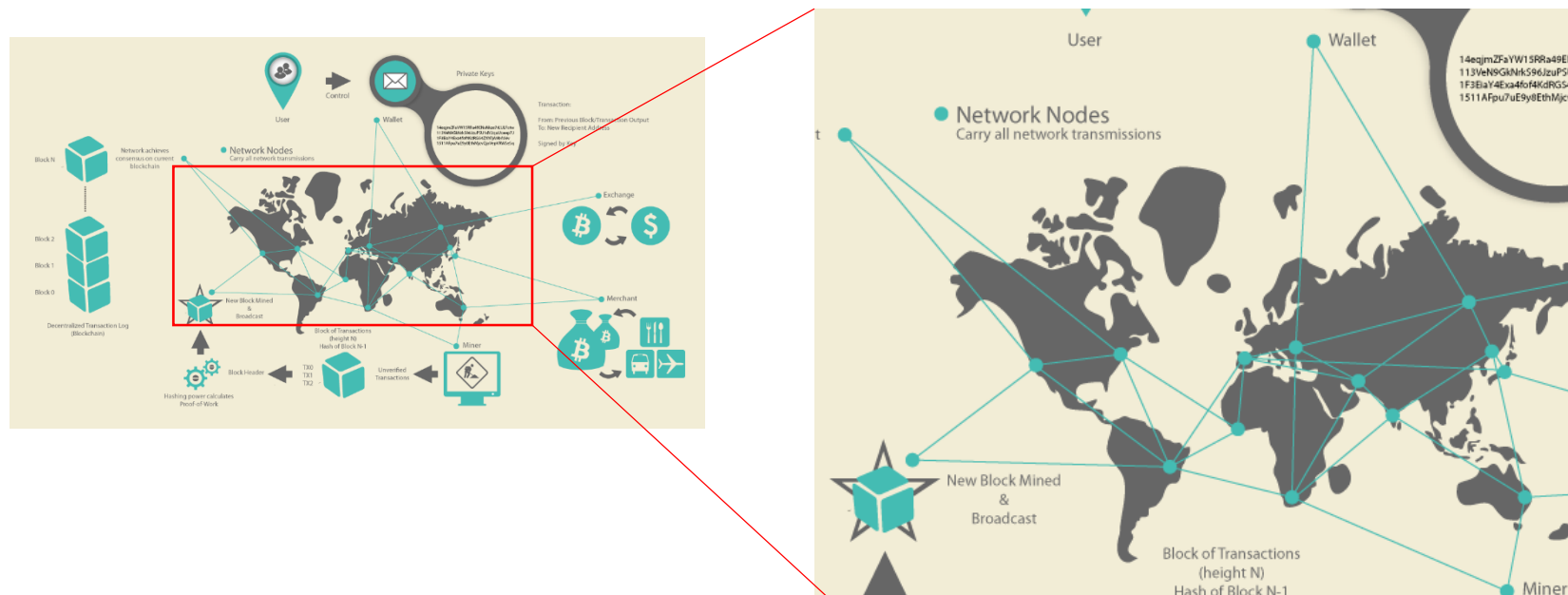


## アウトプットの作成

- アウトプットは2つ
  - ボブのアドレスへ0.015BTC
  - アリスのアドレスへ0.0845BTC
  - (手数料0.005BTC)
- トランザクションアウトプットはスクリプトの形で作成されている。
  - スクリプトはその資金を使用するための解除条件
    - 誰でも任意のアドレスに関するスクリプトを生成できる.
  - ボブへの送金アウトプット
    - ボブのアドレスに対応する秘密鍵からの署名を作れる人であればだれでもこのアウトプットから送金可能.
    - 実際は, ボブだけが秘密鍵を持っているので, アウトプットに対する署名を作成できる.
  - アリスへの送金アウトプット
    - 同様にアリスのみが自分の秘密鍵を用いてそこから送金するトランザクションを作成できる



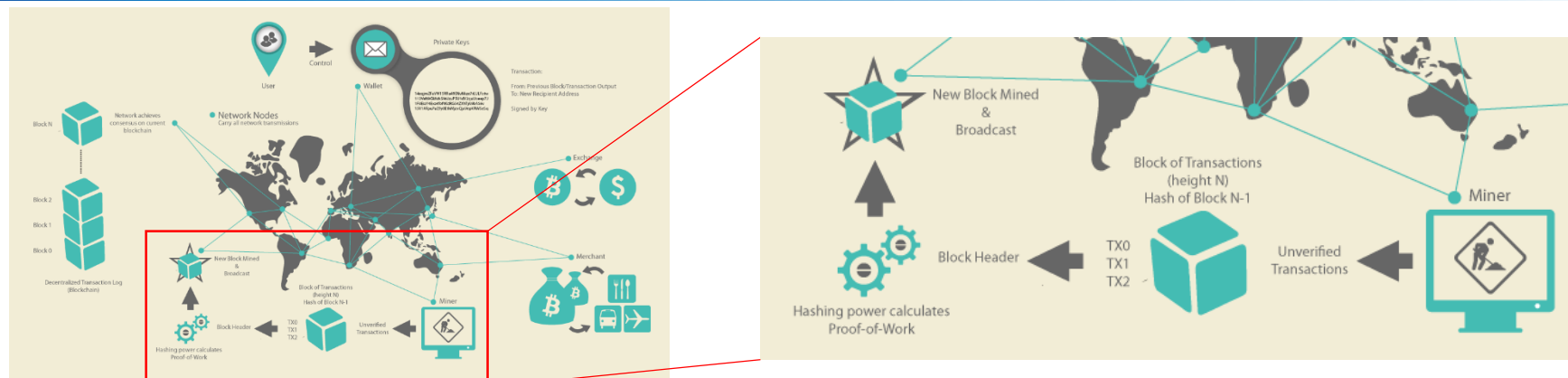
## トランザクションをビットコインネットワークへ送信, 伝搬



- 世界中にあるビットコインノードの一部に送信する
- そのノードはインターネットを通じて別のノードにそれを転送する
- 数秒以内に大半のノードに到達する
- このときはまだブロックには入っていない。
  - トランザクションプール内に存在



# ブロックの生成・マイニング



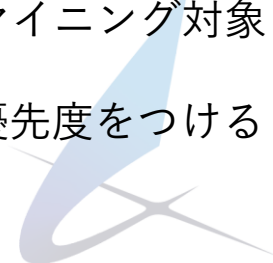
- トランザクションがブロックに記録されるにはマイニングが必要
  - ブロックにはトランザクションが複数記録される。
  - 世界にあるどこかのノードでマイニングに成功し、ブロック内に該当のトランザクションが取り込まれる必要がある。
- PoW (Proof of Work)
  - トランザクションを含むブロックに十分な計算量がつぎ込まれた場合のみそのトランザクションが承認される。
  - 「計算」とは（正確ではないが）乱数を生成して、ある条件の値が出るまで繰り返し乱数を生成し続けるというもの。
    - すごく目の大きいサイコロを振り続けるイメージ
    - 具体的にはある条件に合致するハッシュ値が出る元のインプットを探す
  - マイニングに成功すると報酬がもらえる
    - 2024/4現在マイニング報酬は6.25BTC（そろそろ半減期が来て、3.125 BTCに変更される）
    - 4年に1回報酬は半額になる
  - 手持ちのパソコンで誰でもマイニングはできる
  - マイニングには膨大な計算が必要だが、検証は一瞬で可能（ハッシュを利用）

## マイニングの目的

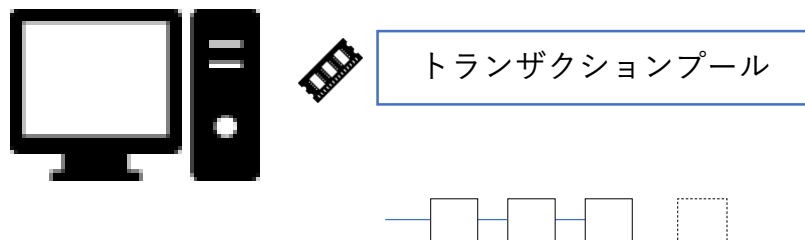
- マイニングは、それぞれのブロックの中に新しいビットコインを作り出します。これは、あたかも中央銀行が新しいお金を印刷するようなものです。ブロックごとに作り出されるビットコインの量は決められており、時間とともに減少していきます。
- マイニングは、信用を作り出します。マイニングは、「トランザクションを含むブロックに十分な計算量がつぎ込まれた場合にのみ、このトランザクションが承認される」ことを保証することによって、信用を作り出します。より多くのブロックがあるということは、より多くの計算量を要したことを意味し、したがって、より多くの信用を得ていることを意味するのです。

ビットコインとブロックチェーン:暗号通貨を支える技術P.27より引用

- いくつも正しいTxがある中で、一定の計算量をつぎ込んでまで次のブロックに含めたい、というマイナーの意思を表現。
- Txを作る側から考えると、手数料を多めに設定することで多くのマイナーにマイニング対象のブロック中に取り込んでもらえることが期待される。
- つまり、「正しいTx」すべてをブロックに取り込むのが無理だから取り込む優先度をつけるのがPoWとも言える。



## トランザクションとマイニング

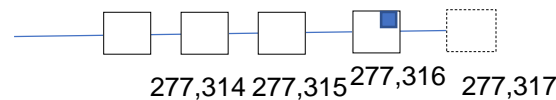
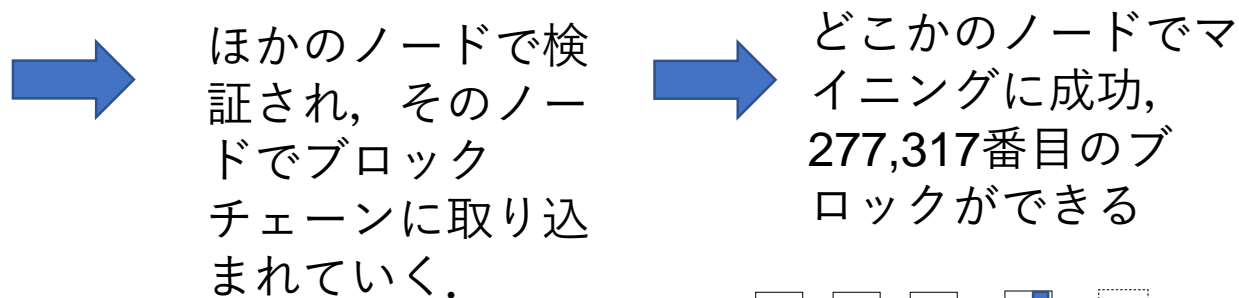
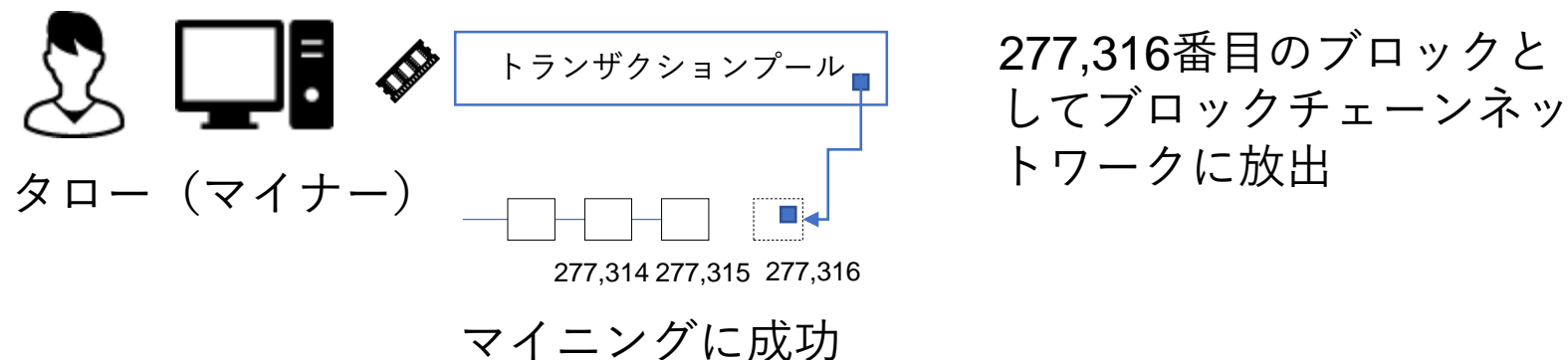


ウォレットから送信されたトランザクションはトランザクションプールに入る。(ノードのメモリ内のみで管理されており、まだブロックチェーンには記録されていない)

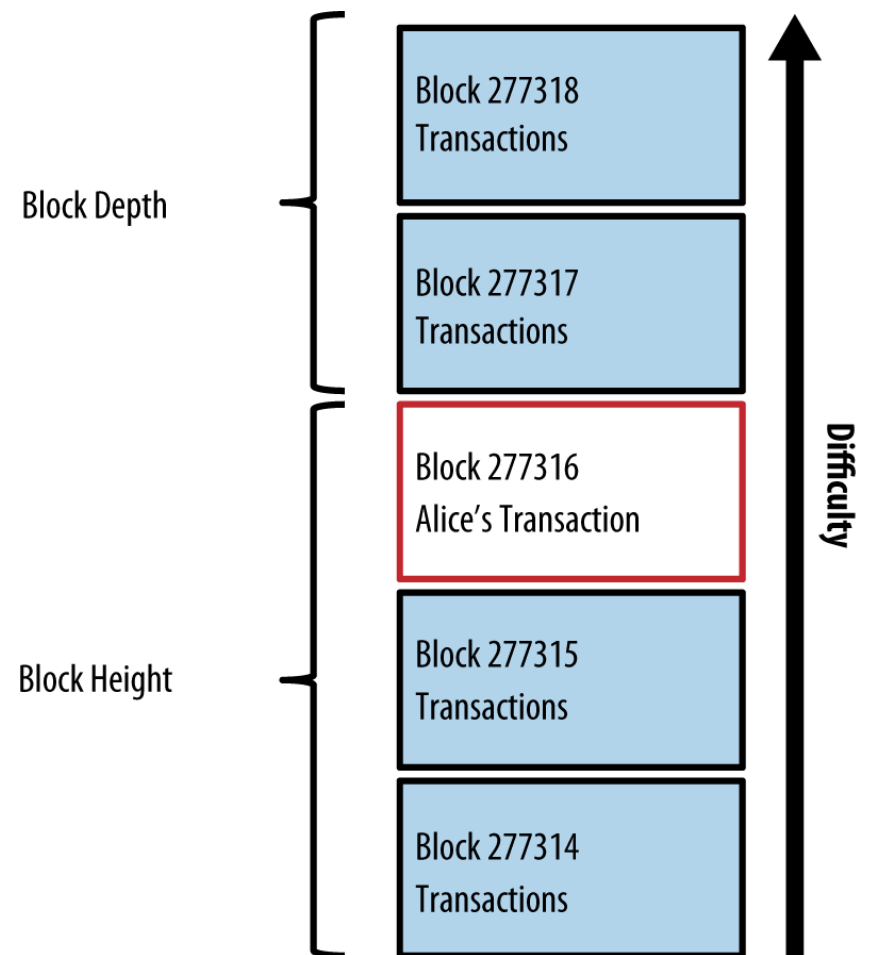
トランザクションプールの中から、手数料が高いトランザクションをいくつか選び、ブロックサイズ1MBに入るだけのトランザクションを詰め込み、マイニングを行う。

マイニングに成功したら、新しいブロックが作られる。作られたブロックは、即時にほかのノードに送信される。新しいブロックを受け取ったノードは、検証を行い、問題なければ新しいブロックとしてブロックチェーンに取り込む。

## アリスのトランザクションがブロックに取り込まれた



## ブロック高・深度・ファイナリティ



ブロック高

→ 初期ブロックからのブロック数

→ block #277316 : 277,316

ブロック深度

→ 最新ブロックからのブロック数

→ block #277316 : 3

深さが深ければ深いブロックほど翻る可能性が低くなる。

→ ブロック取り消しのためにはより多くの計算が必要

慣例的に6個のブロックがひるがえることはないと言われている

→ ファイナリティ

Figure 9. Alice's transaction included in block #277316

## トランザクションのその後

- アリスからボブに支払われた0.015BTCのトランザクション
  - タローのマイニングによりブロックに取り込まれブロックチェーンの一部になった
- ボブは晴れてその0.015BTCをほかの人への支払いに使用できる。
  - トランザクションが取り込まれたブロックが正しいことをボブはウォレットで検証できる。
    - ゼロ番目のブロックから順次正しさの検証ができる。

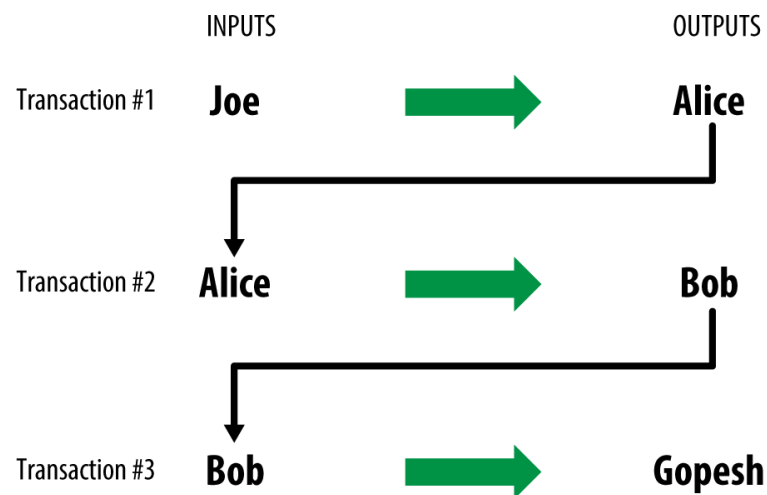


Figure 10. Alice's transaction as part of a transaction chain from Joe to Gopesh

# アリスからボブへの送金トランザクションをブロックチェーンで確認

## Transaction

View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)



1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA  
- (Unspent) 0.015 BTC  
1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK -  
(Unspent) 0.0845 BTC

97 Confirmations

0.0995 BTC

### Summary

Size 258 (bytes)

Received Time 2013-12-27 23:03:05

Included In 277316 (2013-12-27 23:11:54 +9  
Blocks minutes)

### Inputs and Outputs

Total Input 0.1 BTC

Total Output 0.0995 BTC

Fees 0.0005 BTC

Estimated BTC Transacted 0.015 BTC

<https://www.blockchain.com/btc/tx/0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2>

Figure 8. Alice's transaction to Bob's Cafe

ブロックチェーンに記録されているかどうかでその送金が完了していることを確認できる。  
全世界の人に公開され、半永久的に記録される。



## まとめ

---

- アリスがボブへ送金する例で、ビットコインのシステムの流れを概観した
  - ウォレットと鍵
  - トランザクション作成
  - ビットコインネットワークにブロードキャスト
  - マイニングによるブロック生成, 伝搬





## 補足) Bitcoin Core

---

- Bitcoin coreとはビットコインクライアントのリファレンス実装
  - <http://bitcoin.org/> からDLできる
  - <https://github.com/bitcoin/bitcoin> にソースコードが公開されている.
    - C++で実装されている
- Bitcoin Coreはビットコインシステムのリファレンス実装.
  - つまり, 各機能 (ウォレット, トランザクション, ブロック検証, P2Pネットワーク) の実装をどのようにするか, というお手本になっている



## 補足) bitcoin explorer

- Bitcoin explorer
  - ビットコイン用の様々なコマンドラインツール
    - bxコマンド
    - ビットコインで使われているハッシュ計算をしたり, 鍵のフォーマットをしたりできる
  - libbitcoinのプロジェクト中にある
    - ビットコインクライアント用の様々なライブラリ
- インストールしておくと便利
- <https://github.com/libbitcoin/libbitcoin-explorer>
- 以下は/home/shibano/bin/libbitcoin以下にインストールするコマンドです(Ubuntu 22.04). 自分の環境で適宜置き換えてください.

```
sudo apt-get install build-essential autoconf automake libtool pkg-config git
git clone https://github.com/libbitcoin/libbitcoin-explorer.git
cd libbitcoin-explorer/
git checkout -b version3 origin/version3
./install.sh --prefix=/home/shibano/bin/libbitcoin --build-boost --build-zmq --disable-shared
```



# 鍵，アドレス，ウォレット

---



# 概要

---

- 公開鍵暗号方式
- 秘密鍵, 公開鍵, アドレス
- 公開鍵, 秘密鍵
  - 公開鍵
  - 秘密鍵 → 銀行ATMのPINコードやハンコのようなイメージ
    - 自分以外誰にも知られてはいけない
  - ※ あくまでイメージです
- 秘密鍵の作り方
- 公開鍵の計算の仕方
- アドレスの公開鍵からの求め方
- 秘密鍵・公開鍵のフォーマット
- ウォレット
  - 秘密鍵を管理する
  - 鍵の収納, 生成など



## 公開鍵暗号方式について

- HTTPS通信（SSL/TLS）でも使用されている。
- 代表的なアルゴリズムには、ECDSAに加えてRSAがある。
- ECDSA（Elliptic Curve Digital Signature Algorithm：楕円曲線デジタル署名アルゴリズム）がBitcoinでは採用されている。
- 公開鍵暗号方式とは
  - 秘密鍵と公開鍵のペアを一意に生成できる。
  - 公開鍵は誰にでも渡して問題ない
  - 署名ができる
    - 秘密鍵を持っている人しか生成できない署名文字列を生成
    - その秘密鍵に対応する公開鍵で署名を検証できる。
- ビットコイン所有者はトランザクションに公開鍵と署名を記載する
  - ビットコインネットワークのすべての参加者はその公開鍵と署名をもって検証できる。

## 秘密鍵, 公開鍵, アドレスの関係

- Bitcoinでは各参加者の口座番号を「アドレス」と表現する.
- アドレスとは27から34文字の文字列からなる.
- 秘密鍵は乱数で生成する.
- 楕円曲線をもとに秘密鍵から公開鍵が生成され, その公開鍵からハッシュ関数を通じてアドレスが導出される.

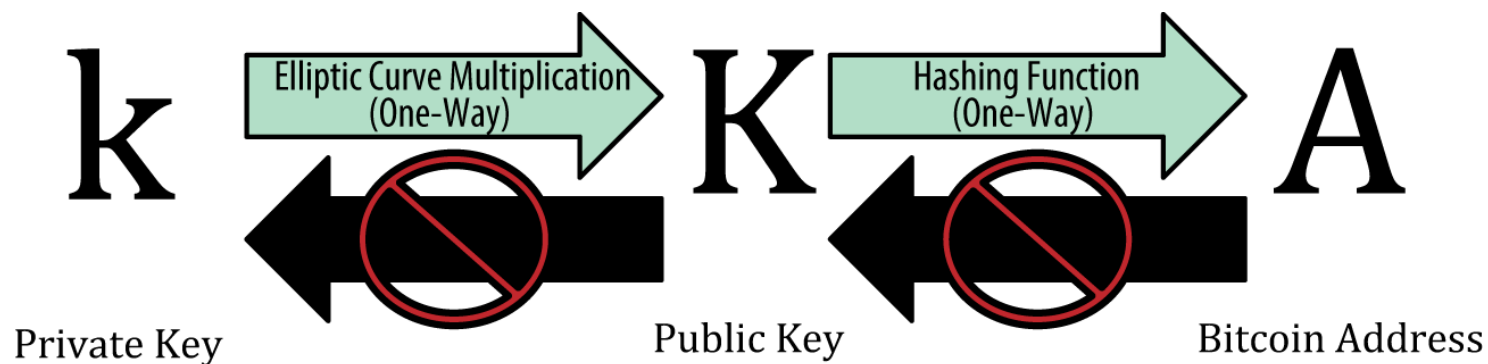


Figure 1. Private key, public key, and bitcoin address

[https://github.com/bitcoinbook/bitcoinbook/blob/second\\_edition\\_print3/ch04.asciidoc](https://github.com/bitcoinbook/bitcoinbook/blob/second_edition_print3/ch04.asciidoc)

## 秘密鍵の生成

- 秘密鍵は自分だけしかもってない（持ってはいけない）鍵
- これを持っている人だけが対応するアドレスから送金可能
- 乱数により生成される.
- バックアップが非常に重要になる一方, 流出しないように守る必要もある.
  - 秘密鍵は同じものを再生成することは不可能.
- 256bitの数
  - 正確には楕円曲線secp256k1の位数として定義される定数 $n$ に対して $n-1$ より小さい数
    - $n = 1.158 * 10^{77}$ で,  $2^{256}$ よりわずかに小さい
    - $n$ でmodを取る
- 乱数で生成すると誰かほかの人の秘密鍵とかぶるのでは?
  - 1から $2^{256}$ までの数字の中から一つを選ぶ
  - 参考)  $136 * 2^{256}$ 乗 → エディトン数 (宇宙に存在する全陽子数)
    - <https://qiita.com/zaburo/items/074995ae8cb81d506222>
- コンピュータで生成するときには注意
  - 疑似乱数を用いて生成すると結構簡単に類推されてしまう.
  - CSPRNG (cryptographically secure pseudorandom number generator) と呼ばれるような実装を用いることが望ましい



## 秘密鍵と公開鍵について

---

- 秘密鍵と公開鍵の関係
  - 秘密鍵はスカラー（正の整数）です.
  - 一方，公開鍵は楕円曲線の点 $(x, y)$ です.
    - $x$ も $y$ も正の整数です.
- ということをこれから話します.





## 公開鍵

- 秘密鍵から導出
- 楕円曲線暗号を使用して導出される.
- NIST策定のsecp256k1という標準で定義された楕円曲線と定数を使っている.
  - NIST: National Institute of Standards and Technology アメリカ国立標準技術研究所

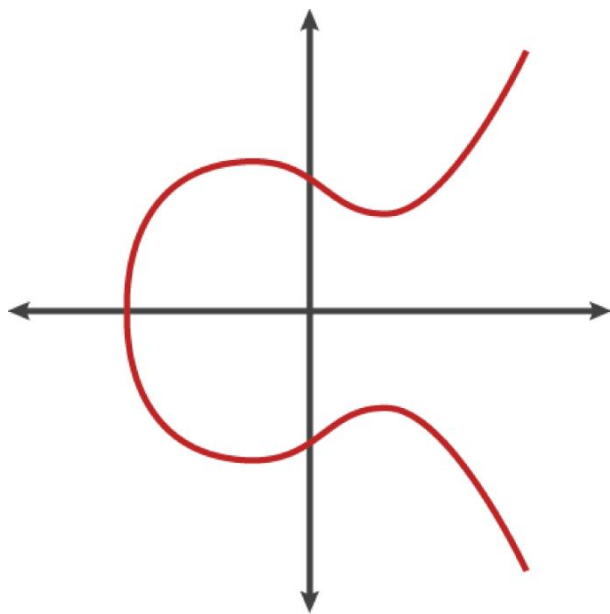


Figure 2. An elliptic curve

$$y^2 \bmod p = (x^3 + 7) \bmod p$$

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

: 位数と呼ばれる. 素数.  $2^{256}$  よりわずかに小さい数

または,

$$y = (x^3 + 7) \text{ over } (\mathbf{F}_p)$$

有限体  $\mathbf{F}_p$  上で定義されている曲線.  
と表現される.

※ この曲線は, 実数ではなく素数位数の有限体上で定義されているため, 2次元にちりばめられたドットパターンのように見える.

## 例：位数が素数17である楕円曲線

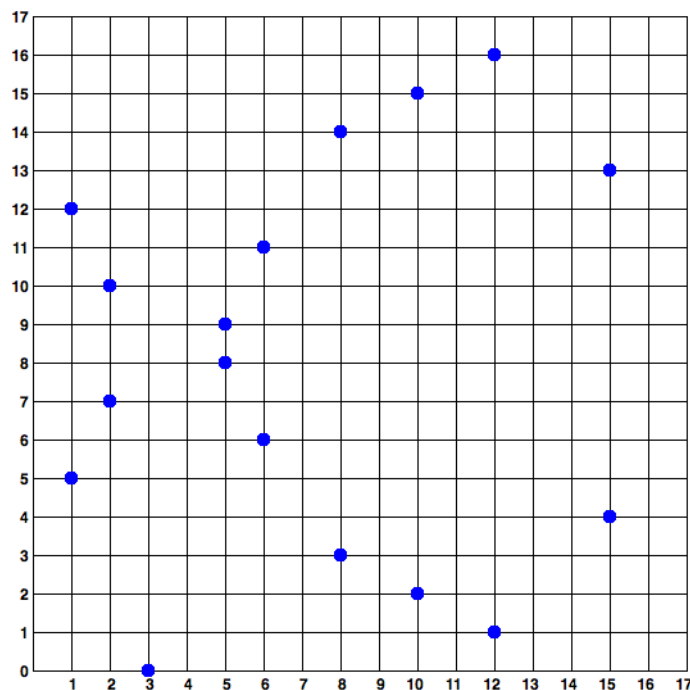


Figure 3. Elliptic curve cryptography: visualizing an elliptic curve over  $F(p)$ , with  $p=17$

$$y^2 \bmod 17 = (x^3 + 7) \bmod 17$$

$x=1$ のとき,  
 右辺 =  $(1+7) \bmod 17$   
 = 8

$$y^2 \bmod 17 = 8$$

となる $y$ は

$$y = 5, 12$$

$$\text{※) } 25 \bmod 17 = 8$$

$$144 \bmod 17 = 8, (17 \times 8 = 136)$$

ビットコインに使われるsecp256k1楕円曲線は、これよりもっと大きいグリッド上にもっと複雑に書かれたドットパターン

## 例：secp256k1曲線上の点の例

どんな点がsecp256k1の点か？

以下の点Pはsecp256k1上の点.

P = (55066263022277343669578718895168534326250603453777594175500187360389116729240,  
32670510020758816978083085130507043184471273380659243275938904335757337482424)

$$y^2 \bmod p = (x^3 + 7) \bmod p$$

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

Pythonで計算して確認してみる.

```
shibano@DESKTOP-940THE0:~/git/bitcoin$ /usr/bin/python3.7
Python 3.7.5 (default, Nov 7 2019, 10:50:52)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.

>>> x = 55066263022277343669578718895168534326250603453777594175500187360389116729240
>>> y = 32670510020758816978083085130507043184471273380659243275938904335757337482424
>>> p = 2**256 - 2**32 - 2**9 - 2**8 - 2**7 - 2**6 - 2**4 - 1
>>> (x**3 + 7 - y**2)%p
0
>>> p
115792089237316195423570985008687907853269984665640564039457584007908834671663
```

## 楕円曲線上での演算

- 楕円曲線上の演算として、ゼロ元、加法、正整数倍が定義されている。

- 0

- 無限遠点と呼ばれる

- +

- 加法

- $P3 = P1 + P2$ の求め方：

- $P1$ と $P2$ を通る直線が $P1$ と $P2$ とは別に交わる点 $P3'$
    - $P3'$ を $x$ 軸に対して対象な点が $P3$

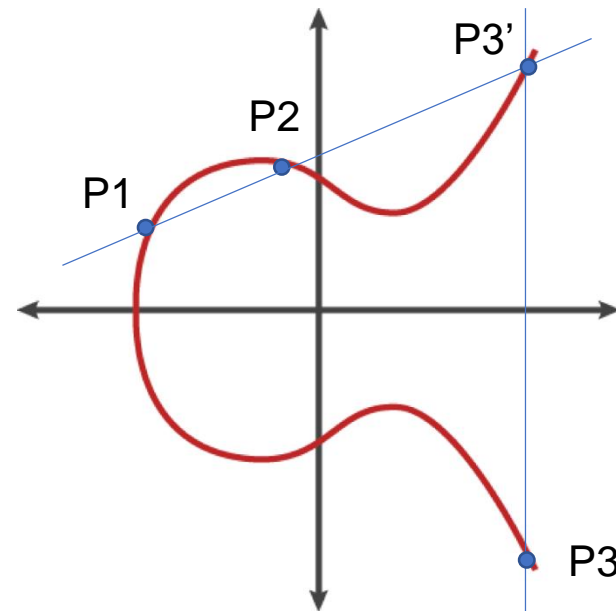
- 同じ値の加法

- $P1 + P1$ は？

- $P1$ での楕円曲線の接線を直線として、それと交わる点が $P3'$ とする。

- 正の整数倍

- その数だけ加法を繰り返すこと



## 公開鍵の生成

- 楕円曲線上で、あらかじめ決められた生成元  $G$  (generator point) を秘密鍵  $k$  倍することで、公開鍵  $K$  を得られる。

$$K = k * G$$

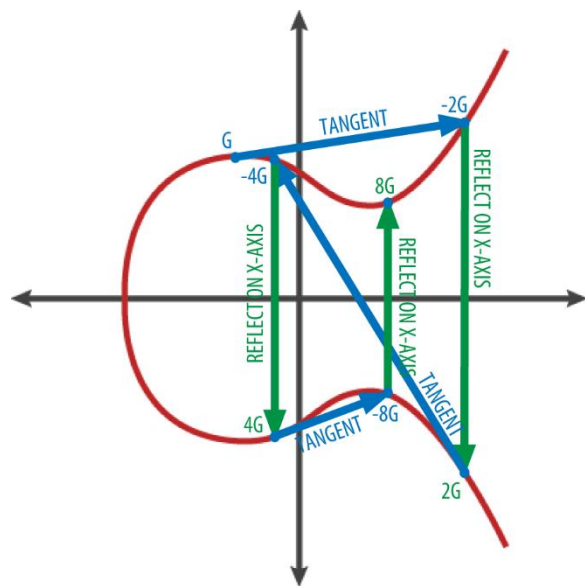
$K$  : 公開鍵

$k$  : 秘密鍵, スカラー

$G$  : 生成元

- 公開鍵  $K$  は  $(x,y)$  座標である。

$G=(0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798, 0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8)$



秘密鍵から公開鍵を計算することは可能。  
大きい数であるが正のスカラー倍。

逆に公開鍵をもとに秘密鍵を求めることは現実的には不可能（楕円曲線上の離散対数問題，ECDLP）。

Figure 4. Elliptic curve cryptography: visualizing the multiplication of a point  $G$  by an integer  $k$  on an elliptic curve

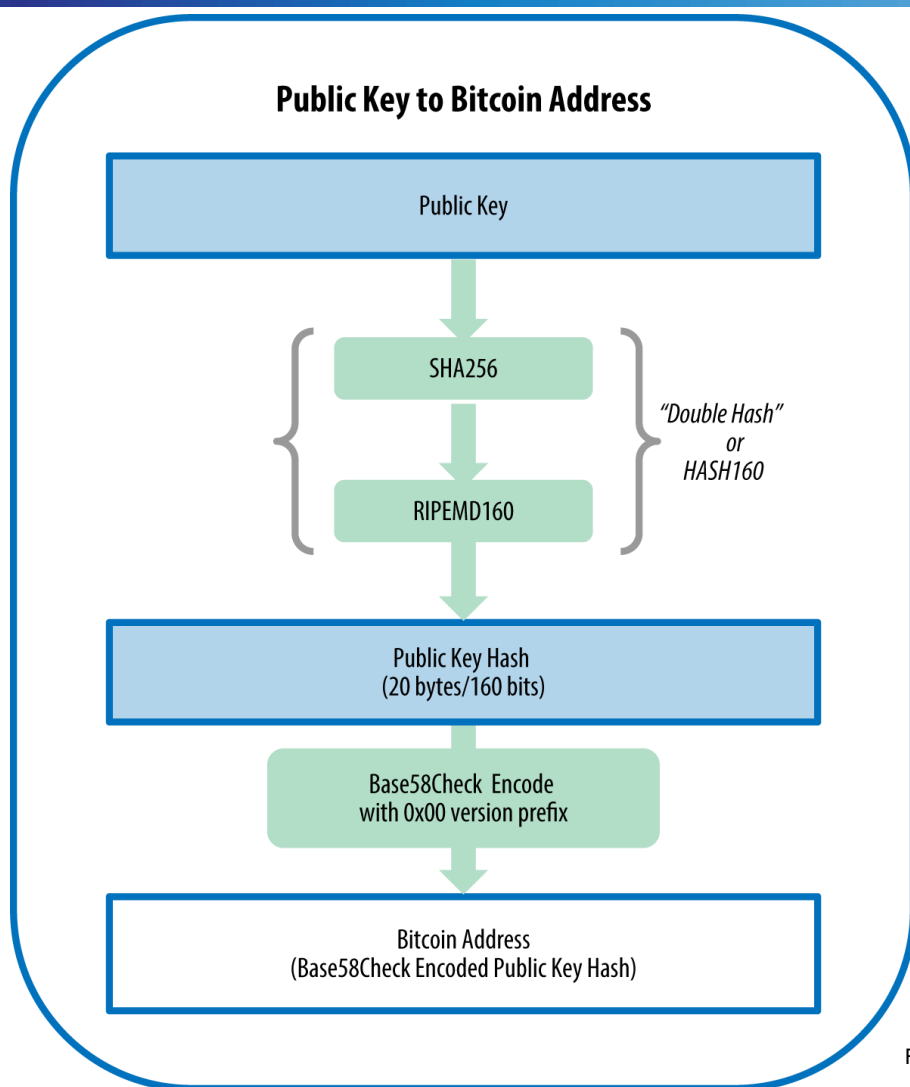
## ビットコインアドレス

---

アドレスは公開鍵から作られる。  
アドレスの例：

1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy

## 公開鍵からアドレスを生成する流れ



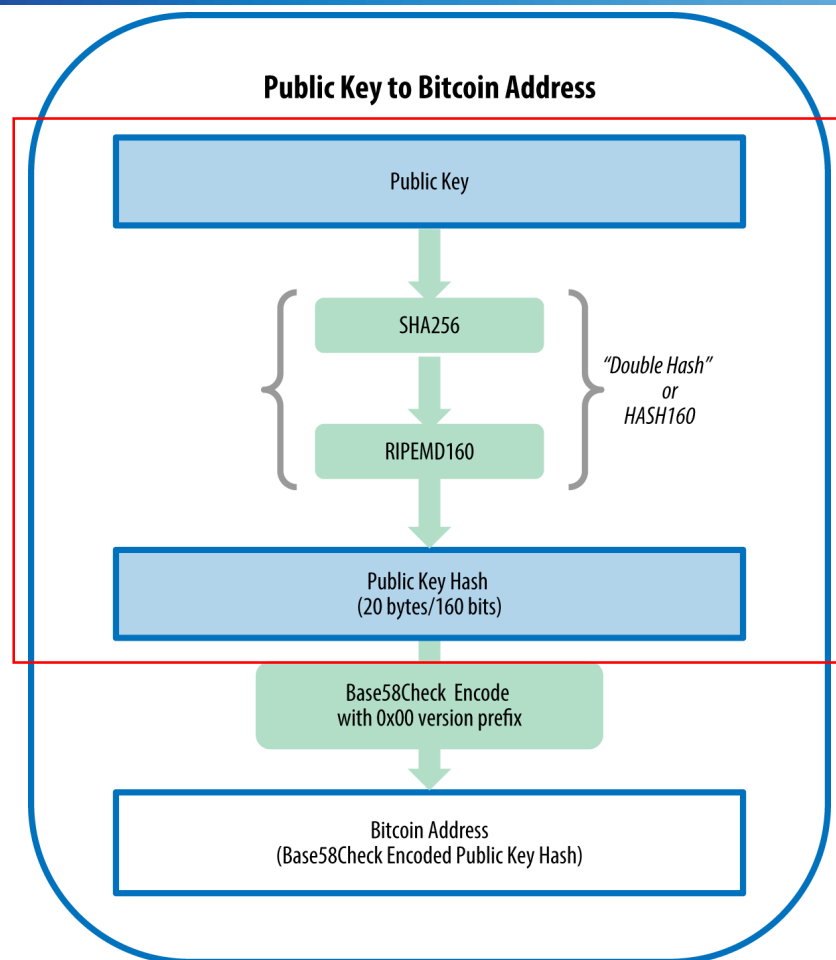
公開鍵からアドレスのもととなるデータを作成し、文字列での表現（エンコード）をします。

文字列で表現できないとあらゆる情報システムで取り扱うことが面倒なためです。  
→ メールでの記述など。

ライプエムディー160

Figure 5. Public key to bitcoin address: conversion of a public key into a bitcoin address

## 公開鍵からアドレスを生成する流れ



公開鍵をSHA256とRIPEMD160という2つのハッシュ関数でハッシュ化する。  
ここで入力する公開鍵は後述するフォーマットで表現されたもの。

$H1 = \text{SHA256 (Public Key)}$   
:256bit (32Bytes)

$H2 = \text{RIPEMD160}(H1)$   
: 160bit (20Bytes)

Figure 5. Public key to bitcoin address: conversion of a public key into a bitcoin address



## 公開鍵からアドレスを生成する流れ

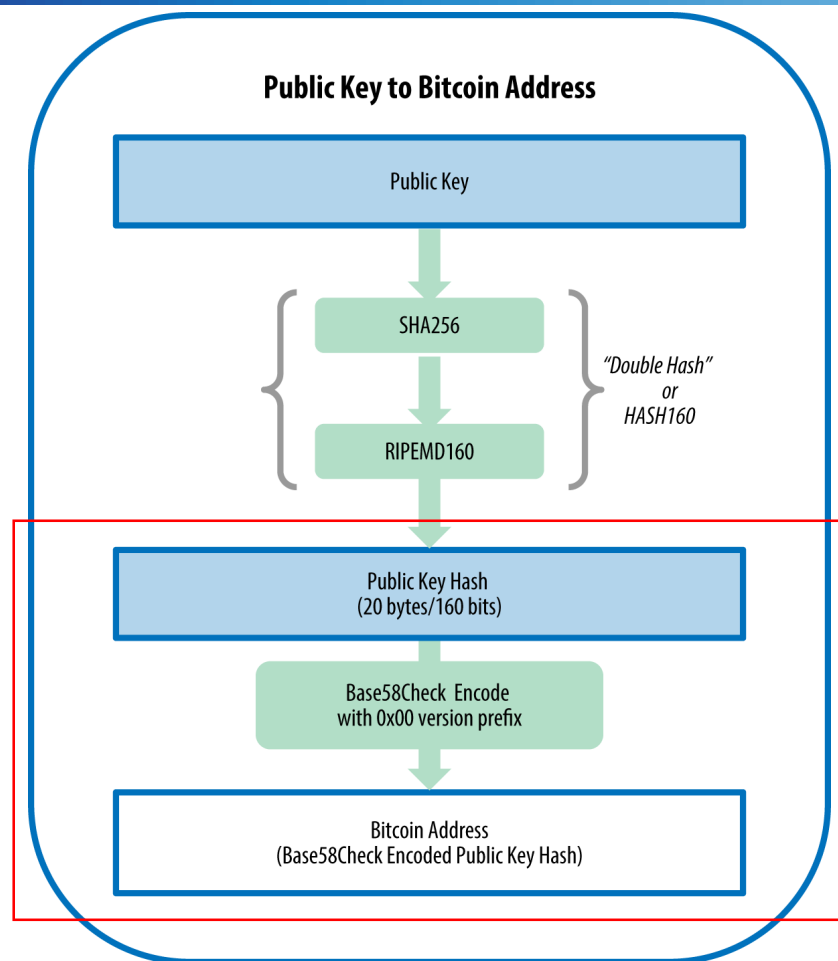


Figure 5. Public key to bitcoin address: conversion of a public key into a bitcoin address

### Base58Checkエンコード

エンコード方式とは

・・・データをアルファベットや数字を組み合わせた文字列で表現する。  
可逆的。

・ 参考：Base64

Eメールで添付データを表現するのに使用されている。

26個のアルファベット小文字, 26個の大文字, 10個の数字, 「+」 「/」 の2個の記号

表現できる文字の種類が大きければ多いほど短く表記できる。


## エンコード方式：Base64の例

- 例えば画像をBase64形式で表現してみる.

### Base64エンコーダー

公開日: 2015/08/21 | 更新日: 2017/07/26

画像をbase64エンコードするツールです。ローカル環境で処理するため、画像を当サーバーにアップロードする必要がありません。

画像	Data URI
 <a href="#">取り消し</a>	<pre>KsoOpTnNemUHcCjS4hloLHVJ12GmhquwS1f3dNKy7CL7IRPnhsj7eoJaNfr9BsuFT6Kg21vJ2VV DabZ4kJrDR74TZpbVJaaUoL+r1VG0FrAcy89BCsUlftmVh3xBC9ImO53T87Og5EYhMHTP4rdF3 7vdfFW3lnYoXQSk7W+cwQA3luJ5deePY9P/prf/UnNxzQAPCx Dz7xzxDi+73DEZhPex9CJRpyyl fAfFRRIss7R3TkfXgDag3w7pmcchdjYCasialjrjnEEOc5Va5MXPl6KPvu9CPU/5E7MLb05yveE9g YCbQfy559oHSOKYejK9WVDhHxLne6YJ3KZU7AHw6dvFe792XN8ebTRpn54J/Ydjb+2ECJ3D9 yriZTvr1cD+Dv5Bzqc7Rmio+6zv/8Lyda57nOXThnAsxOuee854OasWjoYsn43b8K8fIDINuq4yr 3rsrAO11fX+xlOxKym8E+Bk4d1+e80fj0D3snGNyfq/M837wdFSYDnwMnwvePZjG+ZO+CyvO 9bAwf51r3SPg9JxTchSOus4PaU57jqifUznpAu2TDzWXMscYuFZ+N53z67nUGvvuxRj9XXOqV2 P0p1LJz3bO35Nr3bjoX0QqrysVQxizhm5es51oBD3S8lbqvUE5L6LwZfSOG/J0UTOHcXg7siZX Yzhak5zgneXHfk7K5cQvR8z01xz5n41nEXFX1cufcn1jn6lqMyn51Q+7oB7GZyosgOXnl04lvA95 Pwq53QxOJ/l+dkRHxauEyo6F3x13n19PN6cGVbDAPDVObMPwSU46nPKxQFXQhcP0jw/985 f+MOff7W4fe3x2uO1x2uP1x6vPV57vPZ47fHa47XHjT/+P3I98UJs+NpCAAAAAEIFTKSuQmCC</pre>

<https://lab.syncer.jp/Tool/Base64-encode/>

[https://www.irasutoya.com/2020/12/blog-post\\_0.html](https://www.irasutoya.com/2020/12/blog-post_0.html)

- 本スライドの著作権は、東京大学ブロックチェーンイノベーション寄付講座に帰属しています。 自己の学習用途以外の使用、無断転載・改変等は禁止します。
- ただし、
  - Mastering Bitcoin 2nd edition  
[https://github.com/bitcoinbook/bitcoinbook/tree/second\\_edition\\_print3](https://github.com/bitcoinbook/bitcoinbook/tree/second_edition_print3)
  - ビットコインとブロックチェーン --- 暗号通貨を支える技術 （著：Andreas M. Antonopoulos  
訳：今井崇也，鳩貝 淳一郎）
- を使用した部分の使用のみ，CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0/deed.ja> のライセンスを適用するものとします。