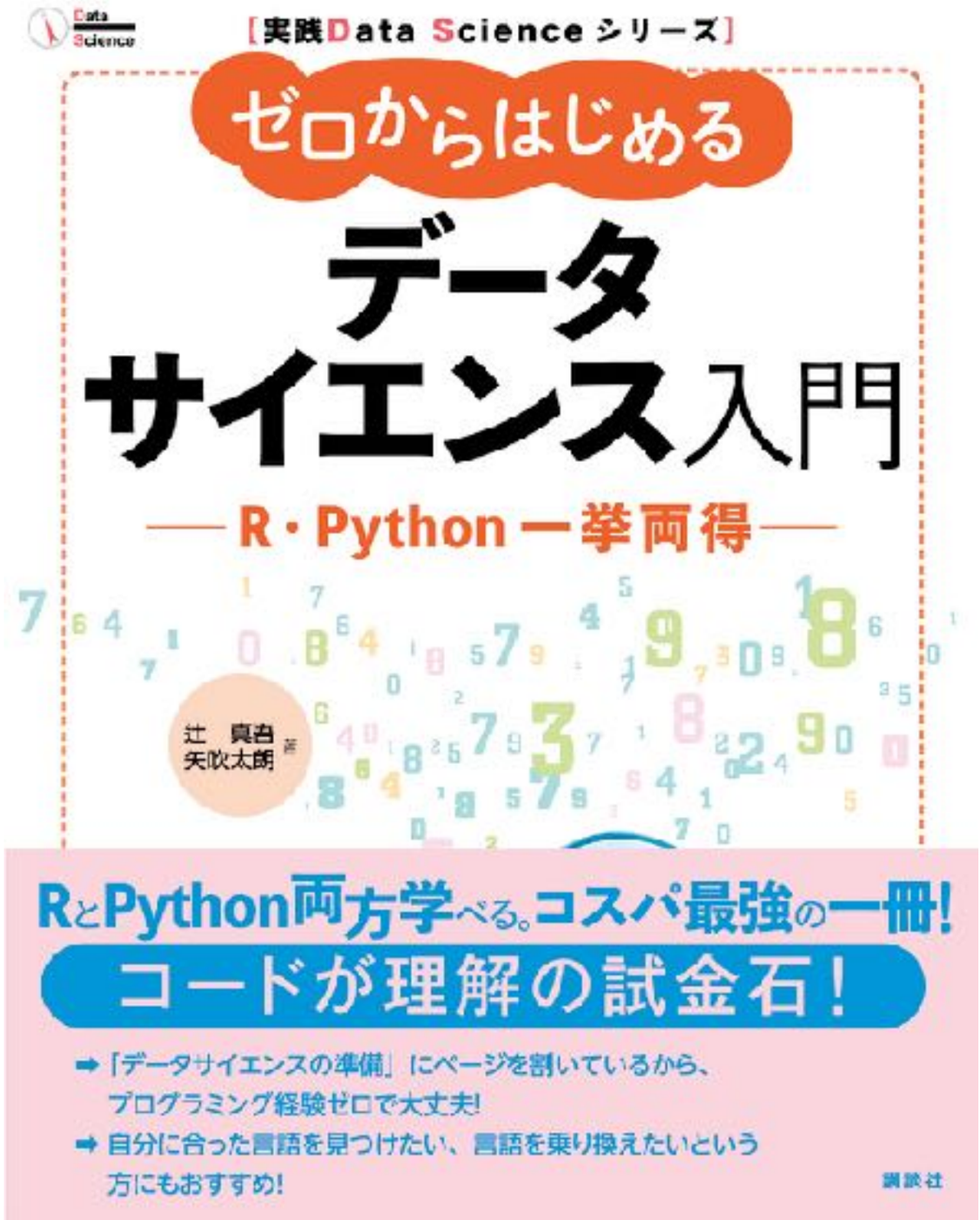
いまさら聞けないブロックチェーンの仕組み

みんなのPython勉強会 #78

辻真吾（@tsjshg）

お前、誰よ？（自己紹介）

- 大学の研究所に勤めています
    - エネルギーシステムとバイオインフォマティクス
- データサイエンスを中心に、コンピュータ関連の仕事で生計を立てています
- 2005年ごろJavaからPythonに乗り換えました
    - 2010年に「Pythonスタートブック」初版をだしてから技術書を何冊か執筆しています
- ゼロからはじめるデータサイエンス入門（講談社）好評発売中です
- www.tsjshg.info



あの有名ブログで紹介いただきました！

https://tjo.hatenablog.com/entry/2022/02/09/170000

ブロックチェーンの基本的な仕組みが知りたい

まずはこちらをご覧ください

Python 3.10.2のダウンロードサイト

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		67c92270be6701f4a6fed57c4530139b	25067363	SIG
XZ compressed source tarball	Source release		14e8c22458ed7779a1957b26cde01db9	18780936	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	edced8c45edc72768f03f66cf4b4fa27	39805121	SIG
Windows embeddable package (32-bit)	Windows		44875e70945bf45f655f61bb82dba211	7541211	SIG
Windows embeddable package (64-bit)	Windows		f98f8d7dfa952224fca313ed8e9923d8	8509629	SIG
Windows help file	Windows		342cabb615e5672e38c9906a3816d727	9575352	SIG
Windows installer (32-bit)	Windows		ef91f4e873280d37eb5bc26e7b18d3d1	27072760	SIG
Windows installer (64-bit)	Windows	Recommended	2b4fd1ed5e736f0e65572da64c17e020	28239176	SIG

これ、なんだかご存じですか？

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		67c92270be6701f4a6fed57c4530139b	25067363	SIG
XZ compressed source tarball	Source release		14e8c22458ed7779a1957b26cde01db9	18780936	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	edced8c45edc72768f03f66cf4b4fa27	39805121	SIG
Windows embeddable package (32-bit)	Windows		44875e70945bf45f655f61bb82dba211	7541211	SIG
Windows embeddable package (64-bit)	Windows		f98f8d7dfa952224fca313ed8e9923d8	8509629	SIG
Windows help file	Windows		342cabb615e5672e38c9906a3816d727	9575352	SIG
Windows installer (32-bit)	Windows		ef91f4e873280d37eb5bc26e7b18d3d1	27072760	SIG
Windows installer (64-bit)	Windows	Recommended	2b4fd1ed5e736f0e65572da64c17e020	28239176	SIG

Gzipped source tarball（Python-3.10.2.tgz）をダウンロード

Hashlibモジュールを使って、md5アルゴリズムでハッシュ値を計算

```
import hashlib

with open('Python-3.10.2.tgz', 'br') as f:
    file = f.read()

hashlib.md5(file).hexdigest()
```

```
'67c92270be6701f4a6fed57c4530139b'
```

MD5のハッシュ値は128ビット（16進数で32桁）

（ちょっと横道）Pythonの組み込み関数hash

```
hash('Cat')
```

```
-8397438943415445470
```

```
hash('cat')
```

```
-2195553092007495 23
```

```
hash(23)
```

```
23
```

```
hash(128)
```

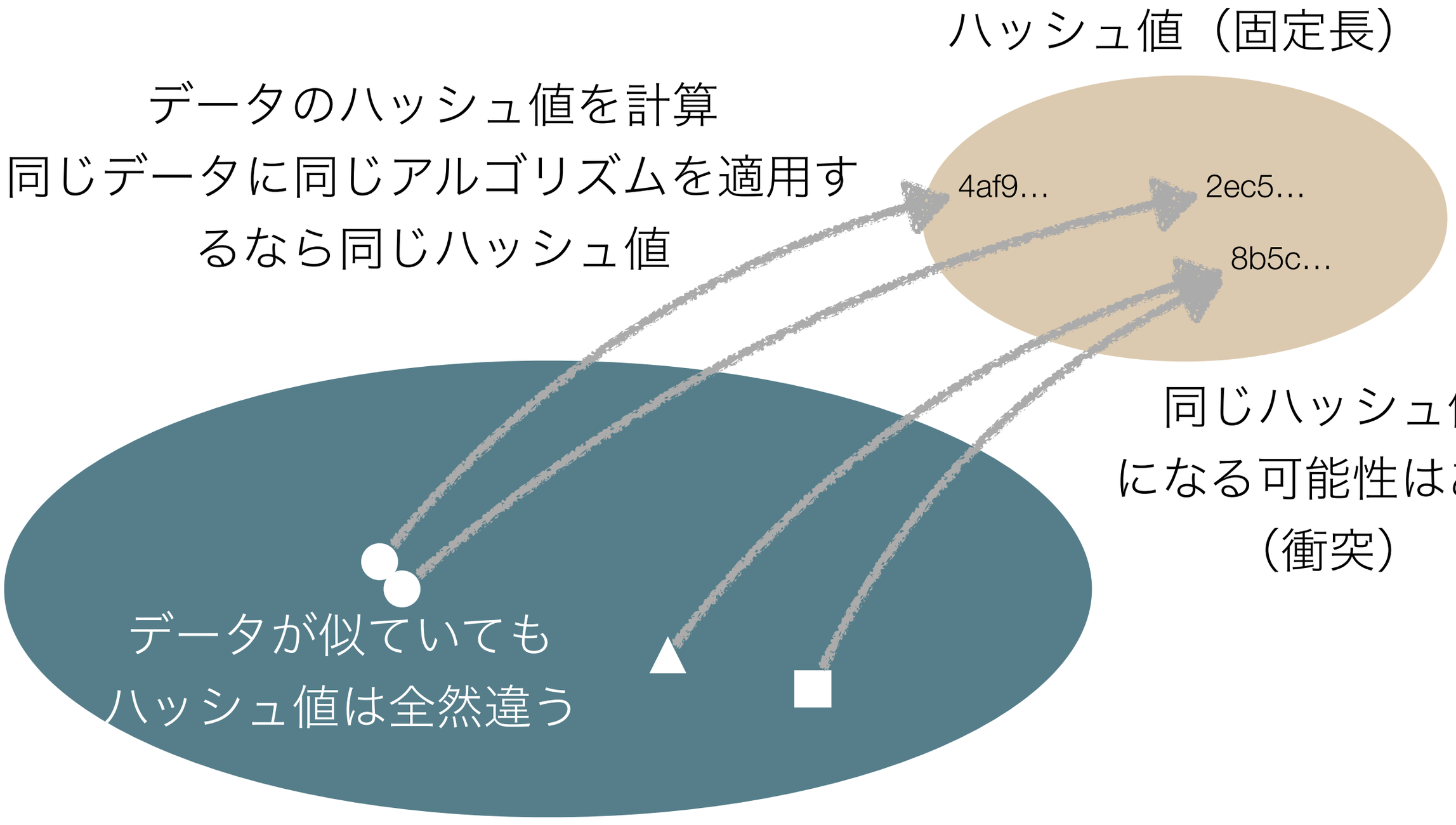
```
128
```

```
hash([1, 2, 3])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-84d65be9aa35> in <module>
----> 1 hash([1, 2, 3])

TypeError: unhashable type: 'list'
```

ハッシュ関数



なんらかのデータ（1つの整数や文字からPDFなどのファイルまで）

MD5の出力の長さは128ビット

入力が128ビットより長くても短くても出力は128ビット

全部で2の128乗＝340282366920938463463374607431768211456種類

128ビットは長いので、ハッシュ値を16進数で表現するが一般的

（128ビットを32桁の16進数で表現できる）

```
hashlib.md5('Cat'.encode('utf-8')).hexdigest()
```

```
'fa3ebd6742c360b2d9652b7f78d9bd7d'
```

```
hashlib.md5('cat'.encode('utf-8')).hexdigest()
```

```
'd077f244def8a70e5ea758bd8352fcd8'
```

似ても似つかないハッシュ値が生成される

暗号学的ハッシュ関数（一方向ハッシュ関数）

（ひらたく言うと）

入力と出力の対応がまったくもって意味不明な値を出力する関数

組み込み関数hashは暗号学的ハッシュ関数ではない

もうすこし暗号学的ハッシュ関数について

- 出力（fa3ebd6742c360b2d9652b7f78d9bd7d）を見て入力（Cat）が簡単にわからない

- 出力がfa3ebd6742c360b2d9652b7f78d9bd7dになるCat以外の入力を簡単に探せない

- というか、出力が同じになる2つ以上の別の入力を簡単に探せない

何に使われているの？

- パスワードのサーバ側での保存
    - Webサイトのパスワードをデータベースに保存するとき、暗号学的ハッシュ関数でハッシュ値にしておけば、万が一流出しても元のパスワードは漏れない
- passwordとか12345678とかにしているとバレる可能性がある
    - これには対応策があります（今日は割愛）

暗号学的ハッシュ関数の種類

- MD5は安全ではない
- SHA: Secure Hash Algorithm（シャー）がある
    - SHA0〜SHA3に大別
        - SHA-1は攻撃方法が知られている
            - https://shattered.io/
- SHA-2、SHA-3は今のところまだ安全
- ビットコインはSHA2に属するSHA-256を利用

ブロックチェーンの仕組み

- ブロックを数珠繋ぎにする
- あるブロックには1つ前のブロックのハッシュ値が含まれる
- どこか1つのブロックを改竄するとそのブロックのハッシュ値が変わる
- 次のブロックも変わる
- 以下同文
- というわけで改竄はほぼ無理

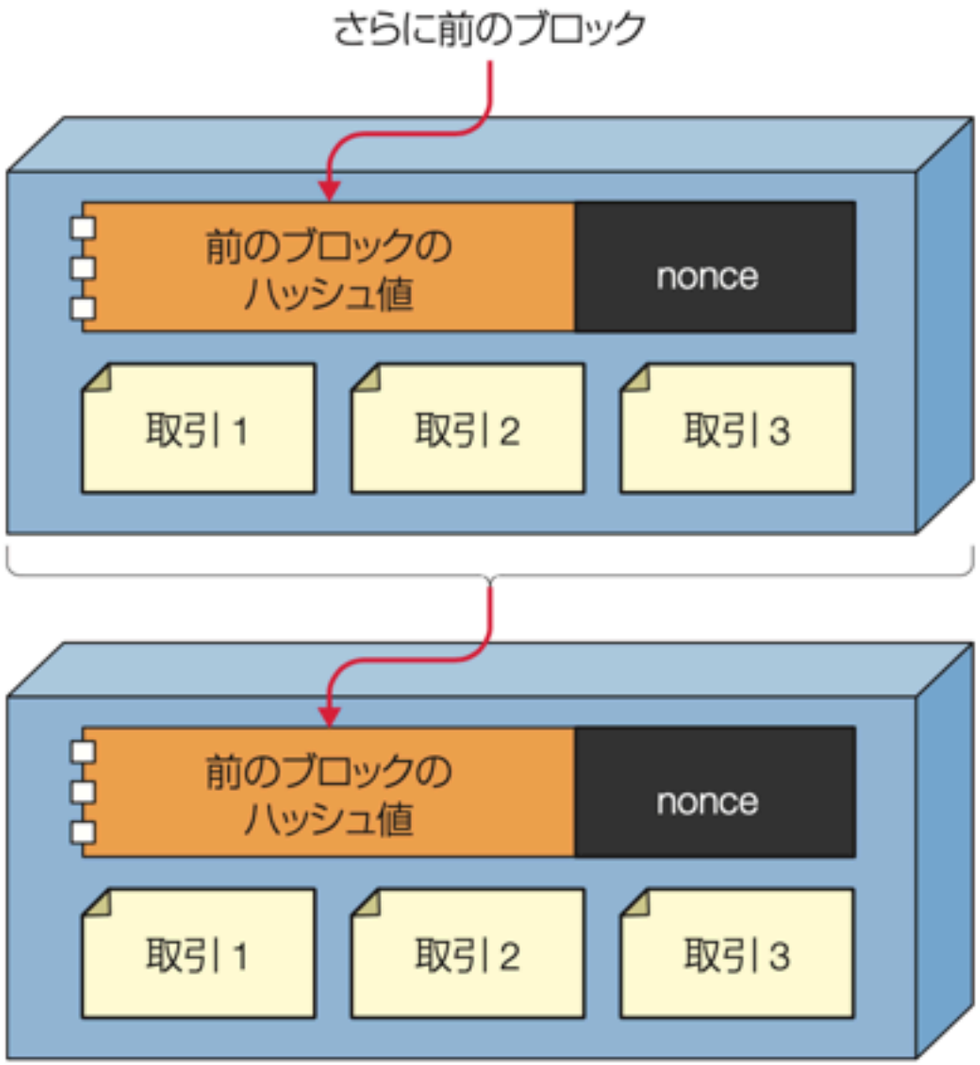


図 10.2　ハッシュ値を使ったブロックの接続

辻真吾著・下平英寿編「Pythonで学ぶアルゴリズムとデータ構造」P167

ビットコインマイニングの意味

ビットコインの場合、ブロックを繋ぐためのハッシュ値は、
先頭の何桁かは連続した0である必要がある

これを実現するために、ブロックの中のnonce（ナンス）という小さなデータを変化させてハッシュ値の計算を繰り返す

0の桁数が増すとマイニングがどんどん難しくなる

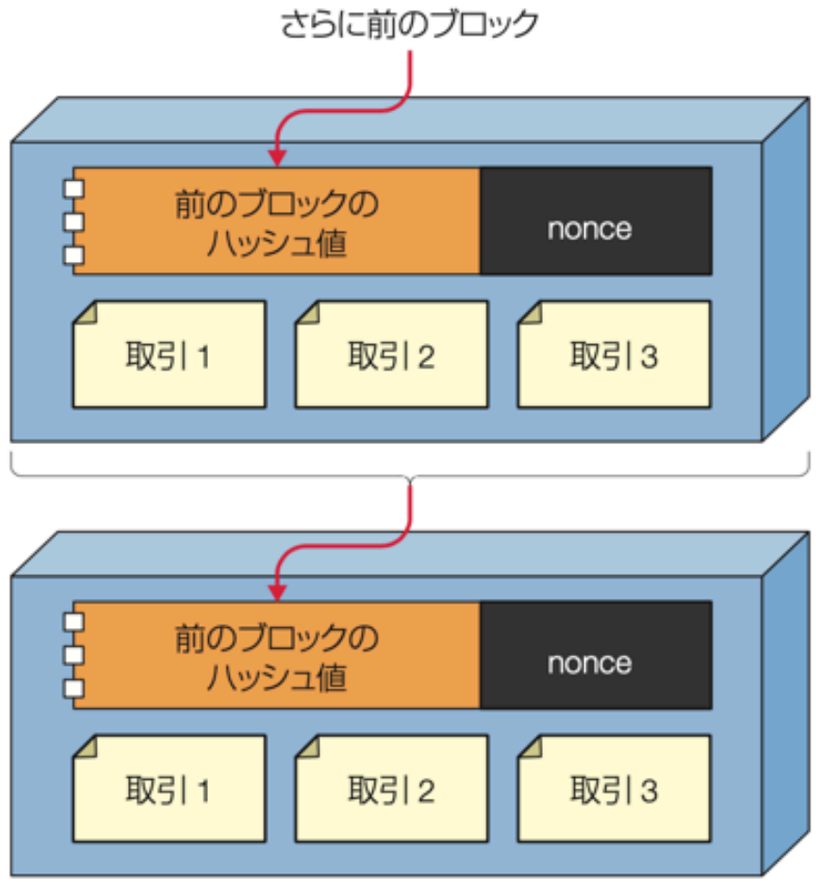


図 10.2　ハッシュ値を使ったブロックの接続

ビットコインのマイニングを体験

```python
def block_to_hash(block_data, nonce):
    # 関数の引数にnonceを連結して、sha256を使ってハッシュ値を計算
    input_str = str(block_data) + str(nonce)
    h = hashlib.sha256(input_str.encode('UTF-8')).hexdigest()
    # 先頭から0がいくつ並んでいるかを数える。
    cnt = 0
    for v in h:
        if v == '0':
            cnt += 1
        else:
            break
    return h, cnt
```

```python
# 先頭に0が5つ並んだハッシュ値を探す
my_block = 'prev_block_tx0_tx1_tx2'
c = 1
while True:
    # 試行回数cをそのままnonceにする
    hash_val, cnt = block_to_hash(my_block, c)
    if cnt == 5:
        print('{}回目の計算で成功しました'.format(c))
        print(hash_val)
        break
    c += 1
```

```
799585回目の計算で成功しました
00000472cdbfc8538016d49f00d7c5ad01cb9e6fc7c733ddbe70ed1e90deb773
```

まとめ

- 暗号学的ハッシュ関数は、入力データから固定長の予想もできない出力を生成する

- ブロックチェーンは、1つ前のブロックのハッシュ値を次のブロックに含めることでブロックを繋げていく

- 途中の一部を変更するとハッシュ値がドミノ倒しで変わるため、後からの改竄は事実上不可能

ご静聴ありがとうございました

10章「現代社会を支えるアルゴリズム」で公開鍵暗号などと一緒にブロックチェーンの仕組みを解説しています

https://bookclub.kodansha.co.jp/product?item=0000276050