特集《生成 AI・ブロックチェーンと知財》

ブロックチェーンの現状と今後の展望について

情報セキュリティ大学院大学教授　**大塚　玲**

要　約

スマートコントラクトを利用したブロックチェーン技術は各分野での活用が期待されており、知財分野においてもブロックチェーン技術は注目されている。ブロックチェーン（Nakamoto［2008］）の第一の革新は、Proof of Work と呼ばれる公平かつ信頼性の高い抽選システムにより、貨幣発行に必要な強力なコンセンサスを達成し、新しい形のデジタル通貨を実現したことにある。これは、ブロックチェーンは、全ての取引が公開されているにも関わらず過去の記録を改ざんできないという性質で実現される。古代メソポタミアでは粘土板に貸借記録を刻んでいたと聞く。ブロックチェーンは現代的な技術で粘土板をインターネット上に実現したものとみなすこともできる。但し、古代の環境に優しい台帳システムとは異なり、現代版の台帳は原発 10 基に匹敵すると言われる電力消費の削減が課題である。Proof of Work に代わるコンセンサスメカニズムの発明が期待されている。ブロックチェーンの第二の革新は、Turing 完全（Turing 機械と等価な能力を持つ計算機のクラス）なスマートコントラクト（Wood［2014］）の発明にある。非常に多数の計算ノードによってコンセンサスが得られた計算結果のみがブロックチェーンに刻まれることにより、処理プロセスが完全に透明化されている。分散型自律組織（DAO）、ステーブルコイン、非代替性トークン（NFT）、分散金融（DeFi）、自動マーケットメーカー（AMM）など、新しい技術が活発に提案されている。本稿では、これらについて最新動向の解説を試みる。

目次

1　はじめに

ブロックチェーン（Nakamoto［2008］）は、Bitcoin や Ethereum（Wood［2014］）の基幹技術である。公開鍵暗号技術（Diffie and Hellman［1976］, Rivest, Shamir, and Adleman［1978］）の発明により、デジタル署名でメッセージの真正性を、信頼できる第三者に頼らずに手元で検証できるようになった。この発明を契機に、ゼロ知識証明やマルチパーティ計算などのさまざまな暗号・認証技術が開発され、匿名性の高いデジタル通貨や、投票の匿名性と開票結果の検証可能性を保証する電子投票などの暗号プロトコルが提案されてきた。ビザンチン障害耐性をもつ分散合意アルゴリズムである PBFT[(1)]（Castro and Liskov［1999］, Castro［2001］）もこの延長にあると考えて良い。PBFT は許可型ブロックチェーンあるいはコンソーシアム型ブロックチェーンの基幹技術になっている。

一方、本稿が主題とする（自由参加型）ブロックチェーンは、ビットコイン（Nakamoto［2008］）を起源とする新しい合意形成アルゴリズムである。ブロックチェーンのブロックは計算パズルの解（Proof of Work：PoW）になっており、解（PoW）を見つけるにはブロック内の変数であるノンス（Nonce）をランダムに変化させて、当該ブロックのハッシュ値が条件を満たすまで偶然に頼って探すしかない。この作業を採掘（マイニング）という。ビットコインでは平均 10 分でマイニングに成功するように計算パズルの難度が設定されており、マイニングに成功すれば高額の報酬（執筆時点では 3.125BTC ≈ 3400 万円）が得られる。マイニング報酬の獲得を目指して、多くの投資家が最新のマイニング機器をフル稼働させてブロックの生成に力を入れている。このため、新しいブロックが予測通りの頻度で次々に公開され、原始ブロックから最新ブロックに至る一直線のブロックチェーンが形成される[(2)]。ブロックチェーンは、Proof of Work による合意形成システムの画期的な発明により、膨大な通貨発行益[(3)]をマイニング報酬の形で全ての利用者に公平に配布することで半永久的にシステムを維持するエコシステムを築き上げた。このシステムにより、貨幣発行に必要な強力なコンセンサスを達成し、新しい形のデジタル通貨を実現したことにある。ブロックチェーンは、PoW の条件を満たす強固なブロックの鎖で構成されているため、全ての取引が公開されているにも関わらず過去の記録を改ざんできない。この性質の上に、スマートコントラクトや Web3 などの新しい公共インフラが築かれ、新しい産業が生まれている。

すなわち、ブロックチェーンの第二の革新は、通常の計算機サーバーと同等の計算能力を持つ Turing 完全な（万能な）スマートコントラクト（Wood［2014］）の発明にある。ここで言う計算能力は理論的な意味での万能性であり、計算性能は技術的なスケーラビリティの限界と計算手数料の 2 つの要因で決定される。スマートコントラクトでは、マイニングの際にマイニング対象となる次のブロックに、通常の資金移動取引に加えて、スマートコントラクトを実行し、その実行結果を記録しておく必要がある。この計算は、全ての計算ノードで一斉に同じ計算を実行し、全ての計算ノードで値が一致することをもって不正なブロックでないことを検証する[(4)]。

非常に多数の計算ノードによってコンセンサスが得られた計算結果のみがブロックチェーンに刻まれることにより、処理プロセスが完全に透明化されている。分散型自律組織（DAO）、ステーブルコイン、非代替性トークン（NFT）、分散金融（DeFi）、自動マーケットメーカー（AMM）など、新しい技術が活発に提案されている。

以下、2 章でブロックチェーンのコンセンサス機構についてより具体的に解説し、3 章でスマートコントラクトを中心とした最新の動向を解説する。最後に、4 章で今後の見通しを述べる。

2　コンセンサス機構

（1）　Proof of Work（PoW）

PoW は、先行するブロックのハッシュ値 B_{i-1} と、マークル・ルート H_i が与えられたときに、その時点での難度（D）に応じて、

$$\mathrm{H}(B_i) = \mathrm{H}(\langle \mathrm{H}(B_{i-1}), H_i, N_i \rangle) < \frac{2^{256}}{D} \tag{1}$$

を満たす N_i を見つける仕事である。ここで、H は任意長のビット列を入力とし、区間［0, $2^{256}-1$］のランダムな整数値を出力するハッシュ関数を表す。

暗号資産で用いられる SHA-256 などの一方向性ハッシュ関数と呼ばれる関数で、出力値が（1）式のような特

定の性質を満たすような入力値を求めること、すなわち、逆関数を求めることが暗号学的に困難なので、順にさまざまな入力を試してみて出力が（1）式を満たすか検査するしか方法がない。一方向性ハッシュ関数の出力分布は一様ランダム分布に非常に近いため、（1）式を満たす N_i が見つかる確率は $\frac{1}{D}$ である。従って、平均で D 回の試行を繰り返すまで見つからないことになる。この事実を用いて、ビットコインでは約 10 分間隔でマイニングに成功するように難度 D が制御されており、現在の難度はおよそ $D \approx 10^{22.9}$ に設定されている。したがって、1 つのノンスを用いた試行で成功する確率 p は

$$p = \frac{1}{D} \approx \frac{1}{10^{22.9}}$$

で表される。

他方、現在の世界のハッシュレート[5]は 1 垓回（10^{20} 回）／秒と観測されており、10 分間に平均して 600×10^{20} 回のハッシュ関数が実行されていることになる。現在、全世界で原発 20 基分に相当する 19.6GW[6]の電力が定常的に計算に投入されている。それほどの電力を投入しなければブロックを採掘できないことから、採掘された新しいブロックが（1）式を満たしていれば、大きな仕事量（と幸運）の証明として利用でき、競合する別のブロックを採掘して対抗しようという競争心を喪失させる効果がある。

（2） マイニング

ビットコインにおけるブロックは、下式に示す 3 項組である。

$$B_i = \langle \mathrm{H}(B_{i-1}), H_i, N_i \rangle$$

i 番目のブロック B_i は、先行ブロックのハッシュ値 $\mathrm{H}(B_{i-1})$、登録する取引の順序付リストのハッシュ値 H_i、ノンス N_i の 3 項組で定義される。実装上は、さらにタイムスタンプなどの補助情報が追加されるが、本節の範囲では不要であるため省略する。各ブロックは図 1 に示すように、前ブロックのハッシュ値を次のブロックに含めることで、鎖（チェーン）構造をとる。ここで、H は暗号学的ハッシュ関数を表し、256 ビットのビット列を出力する SHA-256 もしくは SHA-3 を想定している[7]。

H_i は、一定数の取引の集合を（実用上）一意に表すハッシュ値であり、ビットコインではマークル木（Merkle Tree, Merkle［1982］）が用いられている。マークル木は図 1 に示すようにトーナメント方式でハッシュ値をとる

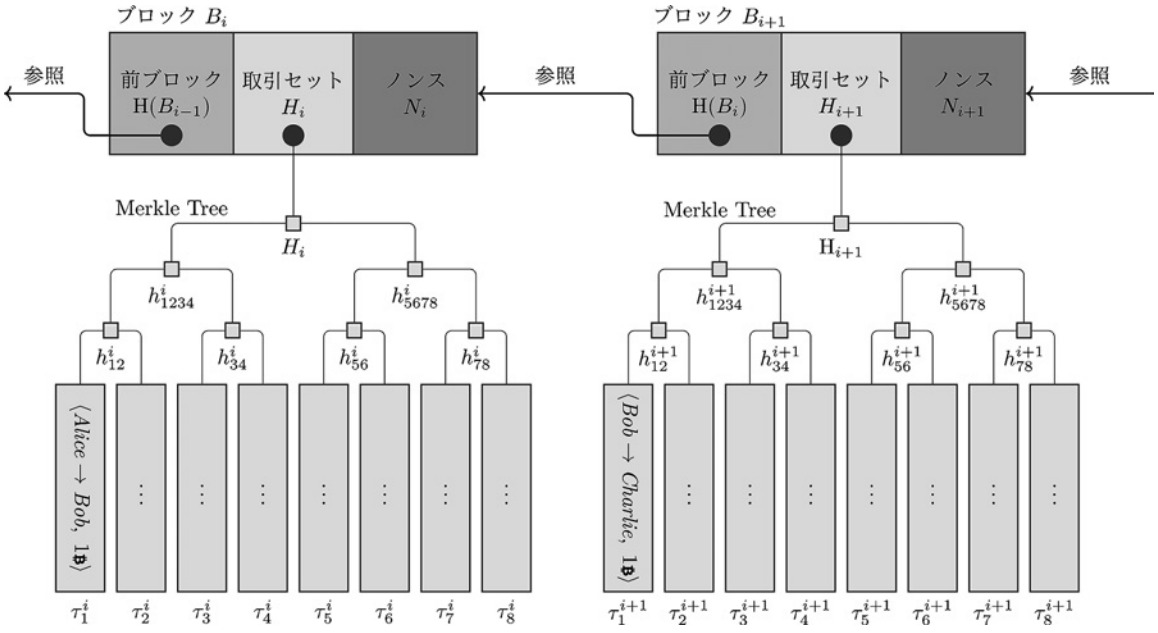

図 1　ブロックチェーンの模式図。各ブロックは $B_i = \langle \mathrm{H}(B_{i-1}), H_i, N_i \rangle$ の形をしている。各ブロックのハッシュ値は PoW により一定値以下のものだけが有効とされる。ブロックには取引のマークルルートが含められ、大量の取引の集合を 1 つのハッシュ値で指示している。

ことにより、複数の取引の順序付リストについてのハッシュ値を計算するアルゴリズムである。

図1の例では、まず取引 $\tau_1,\cdots,\tau_8$ の各組のハッシュ値 $h_{12}=\mathrm{H}(\tau_1,\tau_2)$、$h_{34}=\mathrm{H}(\tau_3,\tau_4)$、$h_{56}=\mathrm{H}(\tau_5,\tau_6)$、$h_{78}=\mathrm{H}(\tau_7,\tau_8)$ を用いて、マークル木は以下の式で計算される。

$$H_i=\mathrm{H}(\mathrm{H}(h_{12},h_{34}),\mathrm{H}(h_{56},h_{78}))$$

マークル木の根にあたるハッシュ値をマークル・ルート（根）と呼ぶ。この手続きを再帰的に繰り返すことにより、任意の数の順序付リストに対するマークル・ルートを計算することができる。

正しい採掘者は矛盾[8]を含まない最長のブロックチェーンを独立に見つけて、正しいブロックチェーンとみなす。このルールにより、常に最長のブロックチェーンに後続するブロックの採掘にマイニングパワーが集中するため、たとえ複数の同じ長さのブロックチェーン（フォーク）が存在しても、長さに違いが生じた時点で長い方のブロックチェーン1本に収斂する。

（3） 共通プレフィックス定理

ブロックチェーンに記録された取引の改ざんに関する脅威は、以下の3点にまとめられる。

- 取引の偽造耐性（暗号資産の不正利用耐性）

取引を偽造するためには、(1) デジタル署名を偽造する、(2) 秘密鍵を盗取するのいずれかが必要である。ビットコインやイーサリアムといった一般的な暗号資産では、暗号学的に安全なデジタル署名（ビットコインではECDSAなど）が用いられることから、(1) の攻撃は現在のところ困難と考えられる。他方、(2) についてはコールド・ウォレットなどの鍵管理技術で利用者自身で自衛策を講じる必要がある。

- ブロックチェーンの改ざん耐性

ブロックチェーンに含まれるブロックや取引セットを改ざんするには、少なくとも実装されているSHA-256やSHA-3などの暗号学的ハッシュ関数の一方向性や衝突困難性を破る必要があり、現在のところ困難と考えられる。

- ブロックチェーン合意形成

フォーク状態が続くと、ブロックチェーンの合意形成を阻害され、不正取引を受け入れる余地が生じる。取引の完了性（ファイナリティ）が特に求められる取引については、当該取引が登録されたブロックが数ブロック以上の深さに到達するまで待つことで、合意が覆される確率を十分小さくできる。

3点のうち上の2点は、暗号技術を組み込む上での一般的な課題と共通する部分が多いので、ここではブロックチェーンに固有の「ブロックチェーン合意形成」に対する脅威を中心に扱う。

ブロックチェーンがフォークしている状態は、最長ブロックチェーンについての合意が達成されていない不安定な状態である。フォーク状態では、一方のブロックチェーンでは確定している取引が他方のブロックチェーンでは確定していないといった状況が生じ、この状況が悪用されれば、二重支払い[9]などの不正取引が横行するリスクがある。安全性が確認されている、ブロックチェーン[10]に対する最も有効な攻撃は、利己的マイニングなどを活用してフォーク状態をできる限り長く維持し、不正取引の成功確率を高めることである。

そこで、正しい採掘者のマイニング能力が不正な採掘者のマイニング能力を一定以上の割合で上回る場合に、取引が事実上確定したと考えてよい範囲を明らかにするために、フォーク長の上限を見積もることが重要となる。これにより、2人の正しい採掘者が保持するブロックチェーンは、末尾（葉）の k ブロックを取り除くと、互いに他方の保持するブロックチェーンの共通プレフィックスになっている。この k が、事実上取引が覆らないと考えてよいブロックの深さを表す。以下では、k と確率の関係を具体的に見積もる方法を共通プレフィックス定理（Garay, Kiayias, and Leonardos［2015］）が示されている。ここで、フォークしているブロックチェーンのうち、合意が達成されている部分（共通プレフィックス）を除いた末尾部分の長さ k の最小値をフォーク長と呼ぶ。

今、採掘者の総数を n 人とし、このうち t 人が不正であるとする。大多数は正しい（善良である）と仮定する（Honest Majority Assumption）。より厳密には、

$$t < (1-\delta)(n-t)$$

を仮定する。ここで、δは正しい採掘者の優位度を表すパラメータである。例えば、$\delta = 0.5$のとき、不正な採掘者の数は全体の1/3未満であることを示す。最悪時のフォーク長の上限を求めるために、不正な採掘者はすべて結託して一体の攻撃者として振る舞い、フォーク長をできるだけ長く維持することを目標として共謀していると考える。

定理 2.1（共通プレフィックス定理（Garay, Kiayias, and Leonardos［2015］））。2つの正直なノードがブロックチェーンC_1, C_2をそれぞれ保持しているとき、典型実行の条件が成立しているならば、期間$S > 0$に対してある$k > 0$が存在して、

$$C_1^{\lceil k} \prec C_2 \text{ かつ } C_2^{\lceil k} \prec C_1 \tag{2}$$

が成り立つ。ここで、$C^{\lceil k}$はブロックチェーンCの末尾からkブロックを取り除いたプレフィックス部分を示し、二項関係$C_1 \prec C_2$は、C_1がC_2の前部分と一致している（プレフィックスになっている）関係を示す。

典型実行とは、さまざまな期間で確率的試行を観測した時に、期間内で観測された事象が外れ値になっていない、すなわち、期待値からの許容乖離率の内側に収まっている状態を指す。以下、ブロックチェーンにおける典型実行について説明する。

1時間や1日など、ある長さを持った観測期間をSで表し、期間Sの中で正しい採掘者のいずれかがマイニングに成功する回数を$X(S)$で表し、正しい採掘者が1人だけマイニングに成功する回数を$Y(S)$で表す。さらに、不正な採掘者がマイニングに成功する回数を$Z(S)$で表す。

正しい採掘者がマイニングに成功するブロックの数$X(S)$は、その期待値$E[X(S)] = E[\Sigma_{i \in S} X_i]$を中心に揺らぐが、$S$を十分に長くとれば、中心極限定理により期待値のごく近くの範囲に収まり、適当な乖離許容率$\epsilon > 0$をとることで、高確率で以下の不等式が成立するようにできる。この状況を典型実行（typical execution）であると定義する。

$$\begin{aligned}
&(1-\epsilon)E[X(S)] < X(S) < (1+\epsilon)E[X(S)] \\
&(1-\epsilon)E[Y(S)] < Y(S) \\
&\qquad\qquad Z(S) < (1+\epsilon)E[Z(S)]
\end{aligned}$$

fを正直な採掘ノードが1単位時間でブロックの採掘に成功する確率とすると、乖離許容率$\epsilon > 0$を許す典型実行において、不正な採掘ノードが採掘ブロック数が正直な採掘ノードのブロック数を越えない条件である分離条件：

$$(1+\epsilon)E[Z(S)] < (1-\epsilon)E[Y(S)] \tag{3}$$

が成り立つ条件は、$3\epsilon + 3f < \delta$として求められる（大塚［2022a］）。

図2に、正直な採掘ノードと不正な採掘ノードのマイニング成功数の分布を示す。Honest Majority仮定の下では、正直なノードのマイニング成功数が不正なノードのマイニング成功数よりも平均的に大きくなるため、十分長い期間Sを取れば、乖離許容率を大きく取っても分離条件（3）を満たすことができる。共通プレフィックス定理のkは、Sに比例するため、kを小さくする（ブロックが確定したと見做すまでの時間を短くする）ためにはSは短い方が良い。

図3に、現実的なパラメータの下で長いフォークが生じる確率の試算を示す。Honest Majority仮定に関連する不正な採掘者の割合δが少なければ、比較的小さいkでフォークが生じる確率を抑えられることが分かる。ブロックが確定したと見做すまでのブロック数kと、自身の取引を含むブロックが別のフォークによって覆される確率の関係を示している。

このように、ブロックチェーンの先端部分（末尾部分）はフォークにより不安定な状況にあることを認識する必

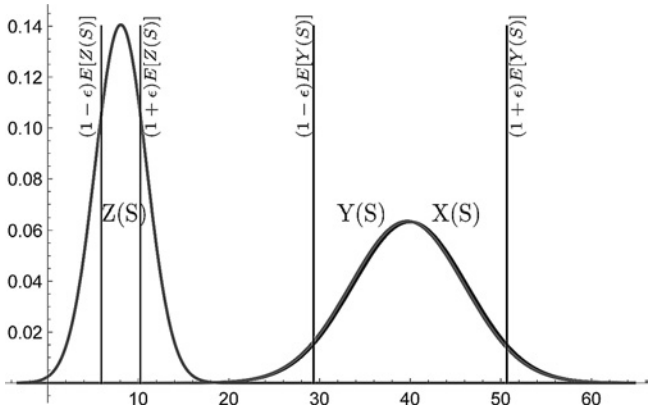

図２　正直な採掘ノードと不正な採掘ノードのマイニング成功数の分布。Honest Majority 仮定の下で、正直なノードのマイニング成功数が不正なノードのマイニング成功数よりも平均的に大きくなる。さらに観測期間を長く取れば期待値の差が広がるため、期待値からの乖離許容率 ε を大きくしても、マイニング成功数が逆転する確率を小さく抑えられる。

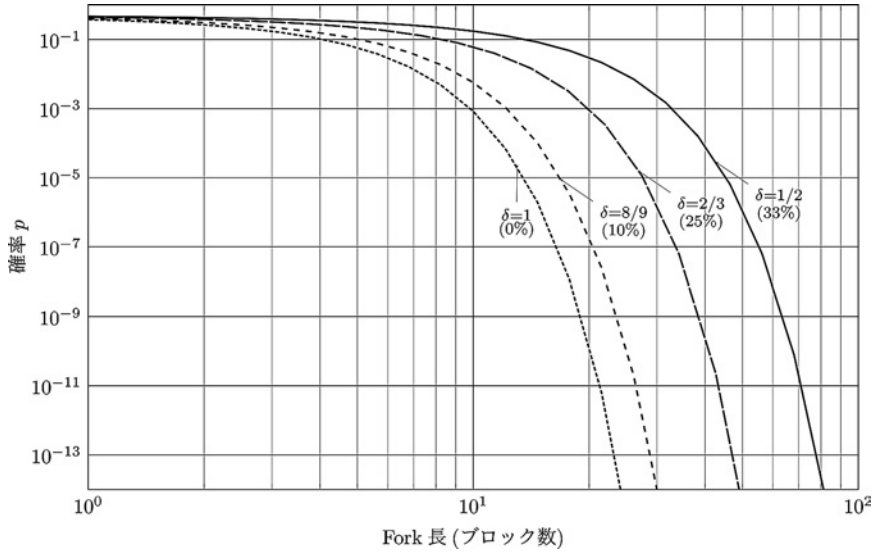

図３　長いフォークが生じる確率。不正な採掘者が利己的マイニングなどにより長いフォークを起こすことができる確率を、不正な採掘者の割合を δ ＝ 1/2（33%）、δ ＝ 2/3（25%）、δ ＝ 8/9（10%）、δ ＝ 1（0%）としてプロットしたもの。横軸はフォーク長、縦軸は確率で表示している。

要があるため、スピードが求められるサービスを設計する際には注意が必要である。他方、一定の時間が経過したブロックは改ざんが極めて難しく、安定していることも事実である。次章に紹介するスマートコントラクトも、ブロックチェーンの合意形成に関して本節で述べたリスクを考慮する必要がある。

３　スマートコントラクト

以上の議論から、ブロックチェーンの利用を前提にすれば、高度な暗号プロトコルを用いなくても、状態機械の動作について全参加者が合意できる形で安全にプログラムを実行することができる。実際、Ethereum の仮想機械（EVM[11]）は Turing 完全[12]であるため、自由度の高いプログラムを実行可能である。

ただし、ブロックチェーンは公開されたブロックデータを元に計算が進められるので、一般的なコンピュータプログラムと異なり、以下の制限があることに注意する必要がある。

1. 秘密を保持する機構を持たない。
2. 決定的アルゴリズムのみが実行できる。

秘密を保持する機構がないため、プライバシーに関わるデータの処理やブロックチェーンに秘密鍵を保持させる

形の暗号プロトコルの実行はできない。また、スマートコントラクトで確率的なアルゴリズムを実行することは合意形成と矛盾するため不可能である。秘密の保持や確率的アルゴリズムの実行には、ブロックチェーン外部に格納された秘密鍵の利用や、外部ノードから入力された乱数の値を利用するなどの工夫が必要になる。

（1） Bitcoin スクリプト

Bitcoin 仮想機械は図 4 に示すようなスクリプトを実行する。図 4 では、$TX1$ の out（太枠、UTXO）には、Bitcoin 受領者から取得したアドレス（公開鍵 p_k のハッシュ値）を含む Output スクリプト（下方太枠）が埋め込まれている。この UTXO を使用する $TX2$ の in（太枠）には、受領者の公開鍵 p_k と p_k で検証可能なデジタル署名 Sig を含む Input スクリプト（下方太枠）を与える。Bitcoin VM はスタック機械[13]として構成されており、Input スクリプトと Output スクリプトを連結して Bitcoin VM の入力として与えると、図 5 に示すように、先頭から 1 命令づつ読み込んで処理を進める。図 5 左には、スクリプト処理途中のスタックメモリの状態を示している。処理

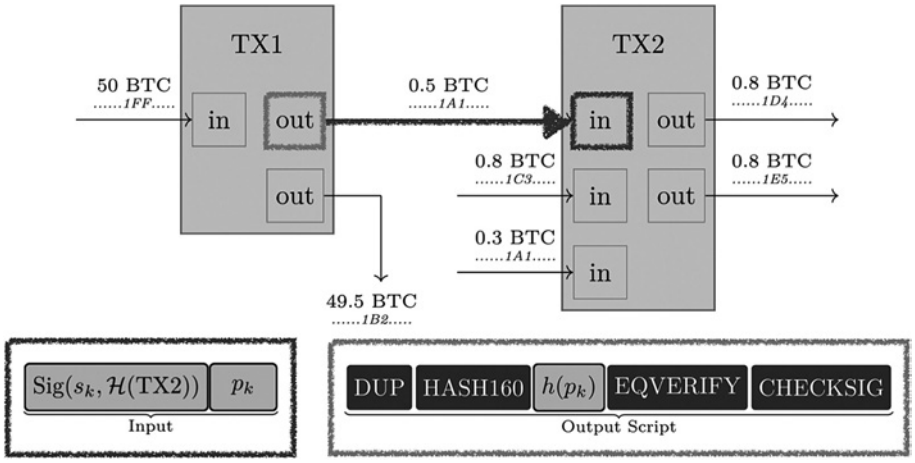

図 4　Bitcoin スクリプトの例（Pay2PKH）。

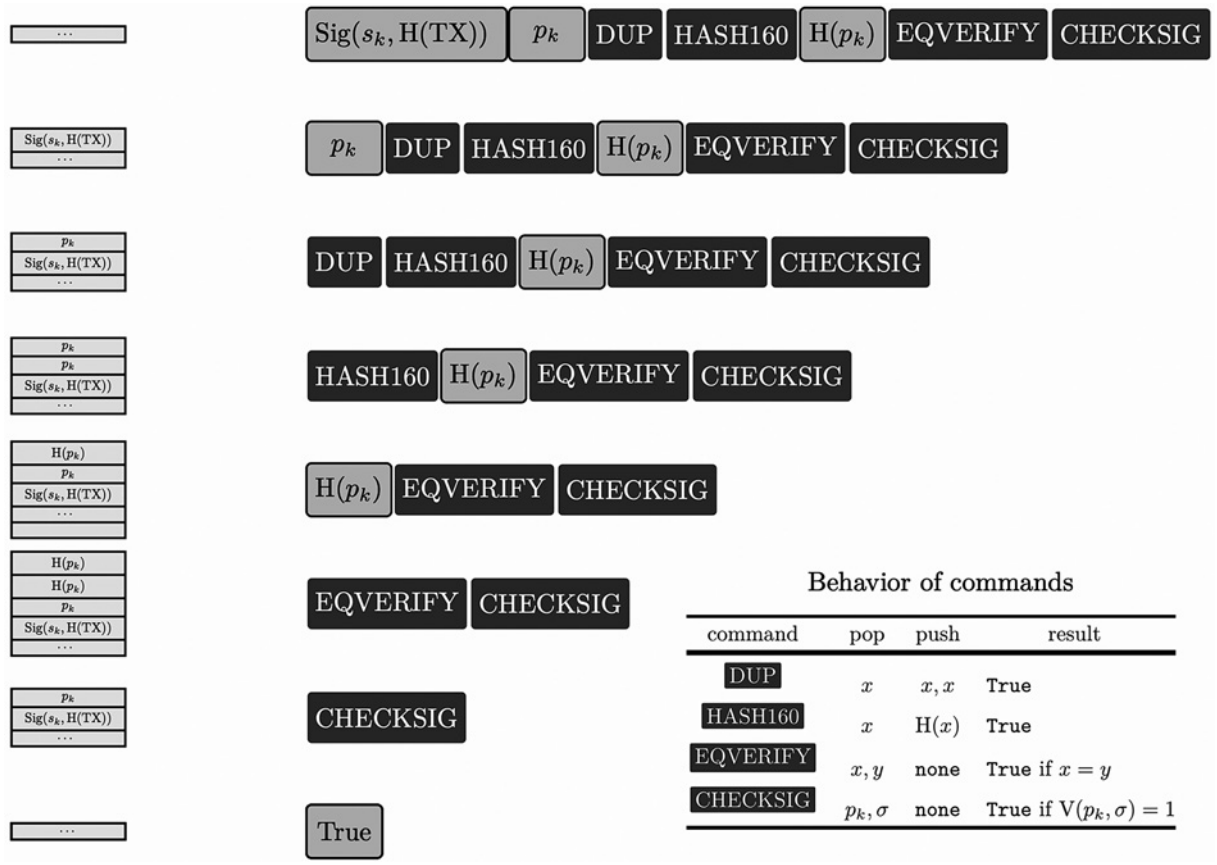

図 5　Bitcoin VM によるスクリプトの処理過程。

の途中で署名の検証に失敗するなどしてFalseを得ると、Bitcoin VMは直ちに処理を中断する。スクリプトの命令を全ての処理を終えた時点でTrueが残ればBitcoinスクリプトは受理され、図4の例では有効な送金として認識される。

Bitcoinには複数の送金者の署名MultiSigや、ブロックの高さが指定した値以降に有効になるタイムロック条件式を組み合わせることで、マイクロペイメントチャネルなどの多様なレイヤー2プロトコル（オフチェーンプロトコルを絡めたスマートコントラクト）が利用されている。Ethereum VMに比較すると、Bitcoin VMの機能は大幅に制限されているが、圧倒的なシェアを誇るBitcoinでスマートコントラクトを利用できることから活発に利用されている。

（2） Ethereum Solidity

Ethereumトランザクションには、表1に示すように（1）通常のETHの送金（Payments)、(2）スマートコントラクトへの送金と実行（Contract call)、(3）スマートコントラクトの初期資産の送金とEthrerum VM用バイトコードの登録（Contract creation）の3種類がある。

Ethereum VMはストレージやアドレス毎の残高などを状態として保持しており、新しいブロックが追加される毎に新しい状態に更新される。例えば、(3）Contract creationは、スマートコントラクトのインスタンスを生成し、ストレージ上に領域を確保してバイトコードのハッシュ値を保存する。

Solidity等でスマートコントラクトのソースコードをコンパイルすると、バイトコードが出力される。このバイトコードを（3）Contract Creationのデータとしてトランザクションを構成し、署名してブロックチェーンに送信すると、そのトランザクションがブロックにconfirmされた時点で、スマートコントラクトがブロックチェーン上に常駐し、contract addressが発行される。このcontract addressを参照して（2）Contract callをデータとして含むトランザクションを構成し、ブロックチェーンにconfirmさせることで、ブロックチェーン上のスマートコン

表1　Ethereumトランザクションの種類

	Payments	Contract call	Contract creation
From	sender	sender	sender
To	recipient	contract address	0
Start gas	21,000	21,000＋	53,000＋
Value	送金ETH	送金ETH	開始資産
Data	(opt) Memo	*f*( ), args	code
Nonce	乱数	乱数	乱数

表2　各命令のgasコスト

	命令	gas費用
基本命令	ADD, MUL, PUSH, JUMP等	2－10
ストレージread	SLOAD	200
ストレージwrite	SSTORE	5000
ストレージwrite（領域確保）	SSTORE	20000
ストレージ消去	SSTORE	－10000
Contract call	CALL等	700
Contract消去	SELFDESTRUCT	－19000

トラクト関数が実行される。

（3） 分散型自律組織（DAO）

分散型自律組織（DAO）は、Ethereum のホワイトペーパー（Buterin［2014］）で主な応用の一つに挙げられ、法域を越えた新しい組織形態として注目されている。DAO を実装するスマートコントラクトには様々なものがあるが、基本的に（1）DAO スマートコントラクトに提出された議案（proposal）に対して、（2）ガバナンストークンの所有割合に比例した重みで投票（token-weighted voting）し、（3）期限内に議決に必要な賛成票が確保できれば、資金を拠出し、議案に従って契約を締結する。（4）契約に従って収益の配分を受け、内部留保資金に組み入れるといった形で運営される。株主と株式会社に似たガバナンス構造が実現されているが、議案の提出や投票、議決、契約などの一連のプロセスは全てブロックチェーン取引として記録・公開されるため、透明性の高い組織運営が可能である。

図 6 に示すように、議案（proposal）は一般的な事業計画を記したものである。紙面の都合で省略したが、加えて判断に必要なスケジュールなどの詳細を追加資料で定義する必要がある。

投票の仕組み　図 7 に Solidity で記述した投票用のコードの例を示す。DAO では出資額の証明として交付されるガバナンストークンのシェアに比例した票数の投票権が与えられる[16]。プログラム中の sender.weight は、投票者（sender）の投票可能票数を表している。3 行目の require（sender.weight!＝0, ”投票権なし”); は、ガバナンストークンを未保有者の投票を禁じるコードであり、4 行目の require（!sender.voted, ”投票済”); は二重投票を禁じる

DAO 提案書: 「CryptoArt Marketplace」

提案の要旨: この提案は、新しいマーケットプレイス「CryptoArt Marketplace」を立ち上げることを目的としています。このマーケットプレイスは、アーティストが自分のデジタルアート作品を NFT（非代替性トークン）として販売できるプラットフォームです。

背景と目的: 現在、デジタルアートの需要は急速に拡大しており、アーティストはより多くのプラットフォームを必要としています。CryptoArt Marketplace は、ブロックチェーン技術を活用して、アーティストが作品を公正かつ安全に販売できる環境を提供します。

提案内容:

プロジェクト計画:

- プロジェクト開始から完了までの詳細なタイムラインを作成します。
- プロジェクトのフェーズ毎に必要なリソースと予算を見積もります。

技術仕様:

- NFT の発行と取引をサポートするために、Ethereum ブロックチェーンを使用します。
- ユーザーインターフェースは使いやすさを重視し、直感的なデザインを採用します。

予算:

- 開発費用: $100,000
- マーケティング費用: $50,000
- 運営費用: $30,000
- 合計: $180,000

リスク管理:

- プロジェクトのリスクを評価し、リスク緩和策を計画します。
- セキュリティ対策を強化し、ハッキングや詐欺のリスクを最小限に抑えます。

収益化計画:

- トランザクション手数料として、販売価格の 2%をプラットフォーム手数料として徴収します。
- 特定のアーティストや作品をプロモートするためのスポンサーシッププランを導入します。

提案の利点:

- アーティストにとって新しい収益源となります。
- コミュニティメンバーがデジタルアートを購入することで、DAO のエコシステムが活性化します。
- ブロックチェーン技術の普及と信頼性向上に貢献します。

投票: この提案に対してコミュニティメンバーは賛成または反対の投票を行います。投票期間は 2 週間です。

結論: 提案が承認された場合、プロジェクトは直ちに開始され、上記のタイムラインに従って進行します。

図 6　DAO の投票対象となる議案の例[14]

```
function vote(uint256 proposal) external {
  Voter storage sender = voters[msg.sender];
  require(sender.weight != 0, "投票権なし");
  require(!sender.voted, "投票済");
  sender.voted = true;
  sender.vote = proposal;
  // proposalの値が不正の場合は無効
  proposals[proposal].voteCount+=sender.weight;
}
```

図7　Solidity で記述した投票コード(15)

```
function winningProposal() public returns (
    uint256 winningProposal_) {
 uint256 winningVoteCount = 0;
 for (uint256 p=0; p<proposals.length; p++) {
  if (proposals[p].voteCount>winningVoteCount) {
   winningVoteCount=proposals[p].voteCount;
   winningProposal_=p;
  }
 }
}
```

図8　Solidity で記述した集計コード(17)

コードである。8行目の

proposals[proposal].voteCount＋＝sender.weight;

は、proposal で指定される提案にトークンのシェアに応じた票（sender.weight）を投じる主要部分である。

集計の仕組み　図8は、Solidity で記述した開票コードである。この例は、複数の議案（proposal）の中で投票数の最も多い proposal の番号を

winningProposal_

として出力する。議案（proposal）が1つのみであり、DAO コントラクトが発行した全てのガバナンストークン（total_balance とする）の3分の2の得票をもって議決する重要議案の場合は、

winningVoteCount＞2/3*total_balance

が可決を判定する条件の Solidity コード表現になる。

（4）　ステーブルコインと非代替性トークン（NFT）

スマートコントラクトで発行されるトークンは、トークンの価値を量で測る ERC-20 と、個々のトークンに識別子で区別し、特定の識別子を持つトークンの価値を質で測る ERC-721（非代替性トークン、NFT：Non-Fungible Token）の大きく2つに大別される。最近では、権利の分割保有を可能にする ERC-400 や ERC-1155 といった中間的な性質を持つトークンも標準化されているが、本稿では違いを明確にするために、ERC-20 と ERC-721 を中心に解説する。

A　ステーブルコイン

トークンの価値を量で測る ERC-20 は、ステーブルコインや DAO のガバナンストークンなどに用いられる。ステーブルコインは、政府発行通貨との交換レートを1：1に保つように価値の上昇や下落をコントロールする仕組みを備えたトークンである。

ステーブルコインが交換レートをコントロールする仕組みの代表的な例は、発行済みトークンの全量の換金に備えるドルや金の準備資産を保持し、準備残高を公表することにより信頼を獲得する方法（Fiat-Collateralized Stablecoins）が知られている。Tether（USDT）や Circle（USDC）がこの方式のステーブルコインの代表例である。他にも準備資産をイーサリアム（ETH）などの主要暗号資産で準備する（Crypto-Collateralized Stablecoins）も知られている。この種の暗号資産準備型ステーブルコインは、暗号資産の激しい価格変動に備えて準備資産を多めに保有することが多い。代表例の Dai は、1ドル＝1Dai の交換レートを維持するために150％の暗号資産を準備資産として維持する政策を採っている(18)。

表3に、流通量上位のステーブルコインを示す。Tether の USDT は $112B 以上の流通量があり、ステーブルコインが社会に定着している様子が分かる。図9は USDT の価格推移を示している。取引量が急増した2020年半ば

表３　流通量上位のステーブルコイン

トークン	価格	流通量
USDT	$0.9995	$112.39 B
USDC	$1.00	$32.19 B
DAI	$0.9992	$5.34 B
FDUSD	$1.00	$3.28 B
USDD	$0.995	$727.93 M
FRAX	$0.9978	$647.97 M
TUSD	$0.9982	$496.34 M
PYUSD	$0.9993	$399.27 M
RSR	$0.006708	$339.44 M
USDe	$1.00	$3.11 B

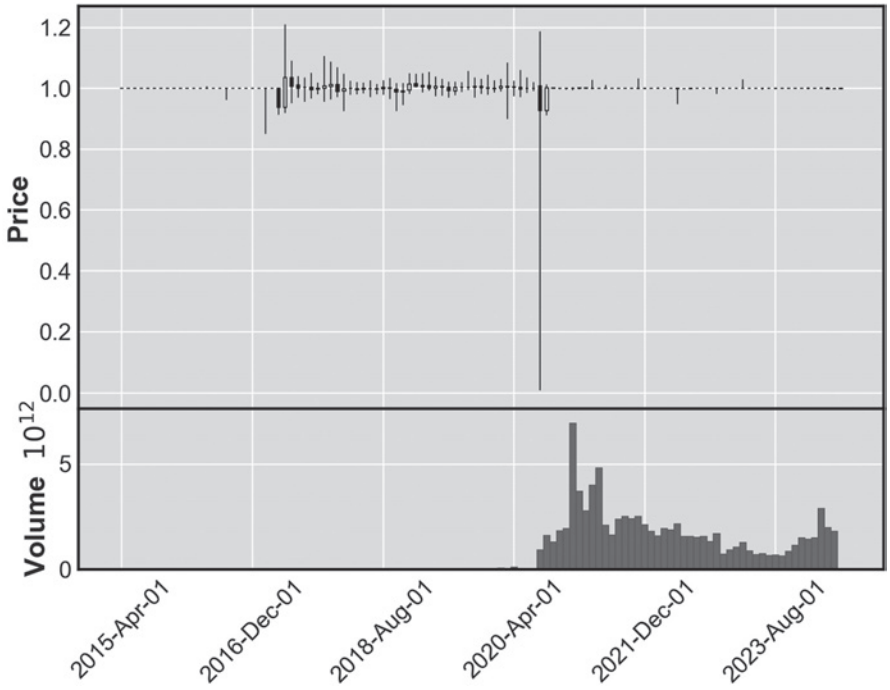

図９　ステーブルコイン（USDT）の交換レート推移

から価格が 1USDT＝1USD±1％で非常に安定している。

B　ERC-20

ステーブルコインに代表される仮想通貨あるいはトークンの多くは、ERC-20 をベースに実現されている。本稿では Bitcoin や Ethereum 等のブロックチェーンと不可分な暗号資産をベースコインと呼び、ERC-20 等のスマートコントラクトで実現されたトークンを仮想通貨あるいはトークンと呼んで区別している。以下では、ERC-20 スマートコントラクトについて、スマートコントラクトの内部状態を定義するステート変数（state variables）とステート変数を操作する関数（function）群を解説する。

ステート変数

○string name;/*トークンの名称*/

○string symbol;/*トークンのシンボル*/

○uint256 totalSupply;/*総発行額*/

○mapping（address＝＞uint256）balanceOf;

利用者の address に割り当てられた ERC-20 トークンの残高を記録する変数

○mapping（address＝＞mapping（address＝＞uint256））_allowances;
利用者が、自身の持ち分の範囲で別の利用者に許可した代理支払い可能なトークンの上限額を記録するステート変数

関数

○constructor（uint256 initialSupply, string memory tokenName, string memory tokenSymbol）
コンストラクタ（constructor）は、スマートコントラクトがブロックチェーン上に展開された時に実行される初期化関数である。

```
1 totalSupply = initialSupply * 10 ** 18;
2 balanceOf[msg.sender] = totalSupply;
3 name = tokenName;
4 symbol = tokenSymbol;
```

総トークン量 totalSupply＝initialSupply×10^{18} でコンストラクタ実行時に発行される。2 行目の msg.sender は、ERC-20 スマートコントラクトの Contract Creation を実行した利用者のアドレスを指している。発行した前トークンは msg.sender の所有になる。3 行目、4 行目は ERC-20 トークンの名前とシンボルを設定している。

○function transfer（address_to, uint256_value）public returns（bool success）

```
1 address _from = msg.sender;
2 require(_to != address(0x0));
3 require(balanceOf[_from] >= _value);
4 require(balanceOf[_to] + _value >=
      balanceOf[_to]);
5 uint previousBalances = balanceOf[_from]
      + balanceOf[_to];
6 balanceOf[_from] -= _value;
7 balanceOf[_to] += _value;
8 assert(balanceOf[_from] + balanceOf[_to]
      == previousBalances);
9 return true;
```

msg.sender のアドレスから _to アドレスへトークンを送金する。この際、送金先アドレス（_to）は address（0x0）（トークンの焼却、burn）でない（2 行目）、送金額は msg.sender の保有額以下である（3 行目）、送金額が正の値（_value＞0）である（4 行目）、送金の前後で残高の総和が変化しない（5、8 行目）などを検証した上で、送金を実行する（6、7 行目）。

○function burn（uint256_value）public returns（bool success）

```
1 require(balanceOf[msg.sender] >= _value);
2 balanceOf[msg.sender] -= _value;
3 totalSupply -= _value;
4 return true;
```

burn は、ERC-20 トークンの焼却を実行する。焼却の実行を指示した msg.sender に十分な残高があれば、msg.sender の残高を減額し、totalSupply も減額して総トークン量に焼却を反映する。

以上の 3 つが基本的な ERC-20 トークンの関数である。他に、利用者 msg.sender が第三者 spender に一定額（value）の管理を委任する

function approve（address spender, uint256 value）

や、委任された spender が委任者の資金移動を行う

function transferFrom（address from, address to, uint256 value）

委任された spender が委任者の資金の焼却を行う

function burnFrom（address from, uint256 value）

などがある。これらは他のスマートコントラクトを spender に設定した自動取引などで重要な役割を果たす関数群である。

C　非代替性トークン（NFT）

トークンの価値を質で測る ERC-721 は、非代替性トークン（NFT）などに用いられる。NFT は個々のトークンに識別子（id）が与えられ、利用者がどのトークンを保有しているのかを保有トークンのリストで管理する。非代替性トークン（NFT）は 2017 年に開始した CryptoKitties を契機に注目を浴びたと言われている。CryptoKitties は、一つ一つ異なる遺伝子を持つ猫の絵の所有権を NFT トークンで表し、交配させることで新しい遺伝子を持つ猫を産むことができる。所有する NFT は譲渡可能で、希少な遺伝子を持つ猫を手にした場合は高額の収入を得ることができる。

D　ERC-721

ERC-721 は個々のトークンに固有の id を与え、個々のトークンの所有者を管理するスマートコントラクトである。NFT は個々のトークンと 1 対 1 に対応する所有権などの権利を対応させ、トークンの譲渡に権利譲渡を対応させることで、権利取引をブロックチェーン上で実現する技術である。以下では、ERC-721 スマートコントラクトのステート変数と、ステート変数を操作する関数群を解説する。

ステート変数

○string_name;/*名称*/

○string_symbol;/*シンボル*/

○mapping（uint256＝＞address）_ownerOf;

トークン ID から所有者のアドレスを検索するステート変数。

○mapping（address＝＞uint256）_balanceOf;

所有者が保有するトークンの数を管理するステート変数。

○mapping（uint256＝＞address）_approvals;

トークン ID の管理を委任された利用者のアドレスを検索するステート変数。

○mapping（address＝＞mapping（address＝＞bool））isApprovedForAll;

所有者と委任先のアドレスの組に対して、所有者が所有する全てのトークンの管理を委任先に任せている場合は true、そうでない場合は false を記憶するステート変数。

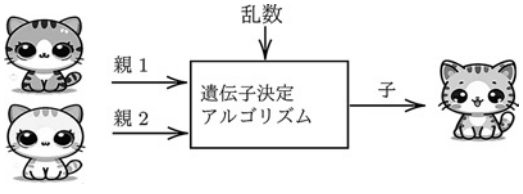

図 10　CryptoKitties の概念図（生成 AI を使用）

関数

○constructor（string memory name_, string memory symbol_）

```
1 _name = name_;
2 _symbol = symbol_;
```

ERC-721 はスマートコントラクトの展開時には NFT が存在しないため、コンストラクタは単に名称とシンボルを設定するのみで良い。

○function mint（address to, uint256 tokenId）

```
1  address from = _ownerOf(tokenId);
2  if (from != address(0)) {
3      approve(address(0), tokenId, address(0),
           false);
4      _balances[from] -= 1;
5  }
6  if (to != address(0)) {
7      _balances[to] += 1;
8  }
9  _owners[tokenId] = to;
10 emit Transfer(from, to, tokenId);
```

mint 関数は、tokenId に対応する新しいトークンを生成して to で指定されるアドレスを新たな所有者とする関数（7、9 行目）である。この際、指定した tokenId に既に所有者がいた場合（from!＝address（0））は 4、前所有者の所有トークン数を一つ減らし（4 行目）、全所有者の委任設定（approval）を解除した上で、新しい所有者 to に tokenId の所有権を譲渡する。最後にトークンの譲渡イベントをログに記録する（10 行目）。

○function transferFrom（address from, address to, uint256 tokenId）

```
1  address owner = _ownerOf(tokenId);
2  address spender = msg.sender;
3  require( from == owner );
4  require( spender == owner ||
       isApprovedForAll(owner, spender) ||
       _approvals(tokenId) == spender);
5  if (from != address(0)) {
6    approve(address(0), tokenId, address(0),
         false);
7    _balances[from] -= 1;
8  }
9  if (to != address(0)) {
10   _balances[to] += 1;
11 }
12 _owners[tokenId] = to;
13 emit Transfer(from, to, tokenId);
14 }
```

transferFrom 関数は、from アドレスから to アドレスに tokenId の所有権を譲渡する関数である。この際、from は、トークンの所有者（owner）であり、かつ transferFrom 関数の実行者（spender）はトークンの所有者であるか（spender＝＝owner）、所有者から全トークンの管理を委任されているか、

isApprovedForAll（owner, spender）

譲渡対象のトークン（tokenId）の管理を委任されているか

_approvals（tokenId）= =spender

のいずれかを満たさなければならない。残りの処理は mint 関数と同じく、前所有者の所有トークン数を一つ減らし（7 行目）、全所有者の委任設定（approval）を解除した上で、新しい所有者 to に tokenId の所有権を譲渡する（10、12 行目）。最後にトークンの譲渡イベントをログに記録する（13 行目）。

○function burn（uint256 tokenId）

```
1 address from = _ownerOf(tokenId);
2 if (from != address(0)) {
3       approve(address(0), tokenId, address(0),
              false);
4       _balances[from] -= 1;
5 }
6 _owners[tokenId] = address(0);
7 emit Transfer(from, address(0), tokenId);
```

トークンの所有権の抹消（burn）を実施する。tokenId の所有者がいた場合（from!＝address（0））は、所有トークン数を一つ減らし（4 行目）、所有者のアドレスを address（0）に設定して（6 行目）、全所有者が tokenId の所有権を失ったイベントをログに記録する（7 行目）。

以上の 2 つが基本的な ERC-721 トークンの関数である。加えて、tokenId にインターネット上の URI（Universal Resource Identifier）を対応づける関数群や、ERC-20 と同様に NFT の所有者（_ownerOf（tokenId））が第三者にトークンの管理を委任する関数群が定義されている。

E　まとめ

これまで見てきたように、ステーブルコインに代表される ERC-20 と非代替性トークンに代表される ERC-721 では、同じトークンという言葉が用いられているが管理方法が全く異なることが分かる。

ERC-20 では、スマートコントラクト生成時にトークンの総量が決定され、総量がスマートコントラクトの設置者に与えられる。トークンは量で管理され、自由に分割して他の利用者に送金できる。他方、ERC-721 では、スマートコントラクト生成時にはトークンは存在せず、mint 関数によって定められた権利に対応するトークンが生成され、利用者に譲渡される。トークンは分割できず、全てのトークンに（tokenId）が割り当てられ、一つ一つのトークンの所有者がスマートコントラクトで管理される。

（5） 分散金融（DeFi）と自動マーケットメーカー（AMM）

複数のデジタル通貨間の為替取引技術として、自動マーケットメーカー（AMM：Automated Market Maker）に基づく分散取引所（DEX：Decentralized EXchange）が台頭している。代表例である UniSwap V3（Adams et al.［2021］）は、既に 2 億取引を約定し、新たなイノベーションを巻き起こしている。本節では、UniSwap の基本的なメカニズムについて解説する。

A　CPMM アルゴリズム

UniSwap は CPMM[19] と呼ばれるアルゴリズムで、市場に提供された各通貨プールの残高（これを流動性プールという）の積を常に一定に保つように価格を決定するマーケットメーカーである。参加者には流動性供給者（li-

quidity provider）と利用者（trader）がおり、既に通貨 X と通貨 Y の流動性プールの残高 X、Y が投入されている状況を想定する。ここで、ある利用者が通貨 X の ΔX を通貨 Y に交換（swap）する際の交換レートを求めよう。CPMM では、取引の前後で総積が一致するように価格を決定する。すなわち、

$$XY = (X + \Delta X)(Y - \Delta Y) \tag{4}$$

が成り立つように ΔY を算出する。この ΔY は利用者が投入した通貨 X 建ての価値 ΔX の対価として受け取る通貨 Y 建ての価値である。

具体的には以下で求められる。

$$\Delta Y = Y - \frac{XY}{X + \Delta X} = \frac{Y \Delta X}{X + \Delta X} \tag{5}$$

同様に、通貨 X、Y 間の交換レートは、

$$R_{XY} = \frac{\Delta Y}{\Delta X} = \frac{Y}{X + \Delta X} \tag{6}$$

で求められる。

以下に、具体的な例を示す（図 11 参照）。今、通貨 ETH の流動性プールに 10ETH の残高があり、通貨 OMG の流動性プールに 500OMG の残高があるとする。この状態で、流動性プールの残高の積（定数）は 5000 である。ここで、利用者が 1ETH を OMG に交換する為替取引を申請すると、ETH の流動性残高が状態 1 の 10 から状態 2 では 11 に増加する。流動性プールの残高の積を 5000 で一定のに保つためには、OMG の流動性残高は 5000/11 = 455.545…に減額する必要があるため、利用者には 1ETH に対して差額の 45.45OMG が UniSwap スマートコントラクトによって自動的に支払われる[20]。

図 12 は、より複雑な CPMM の例を示す。今、状態 a = (8, 500) は、ETH と OMG のそれぞれの流動性プールに 8ETH と 500OMG が入っている状態を示す。

①状態 a で 2ETH を支払い、UniSwap スマートコントラクトに OMG への交換を指示すると、100OMG が返金され、UniSwap は状態 a = (8, 500) から b = (10, 400) に遷移する。

②状態 b で流動性プールに 100OMG を追加すると、UniSwap は状態 b = (10, 400) から c = (10, 500) に遷移する[21]。

③状態 c で 125OMG を支払い、UniSwap スマートコントラクトに ETH への交換を指示すると、2ETH が返金され、UniSwap は状態 c = (10, 500) から d = (8, 625) に遷移する。

④状態 d で流動性プールから 125ETH を引き出すと、UniSwap は状態 d = (8, 625) から a = (8, 500) に遷移する。

このように、交換（swap）は流動性プール残高の積を一定とする曲線上を状態が遷移する。一方、流動性プールにトークンを追加または引出を指示すると、曲線を決定する定数が増加（減少）する。いずれの取引でも、交換レートが変化し、例では $1_{ETH} = 40.00 \sim 78.13_{OMG}$ の間で交換レートが変化していることが分かる。さらに、例では取引を重ねることにより開始状態 a に戻っているが、実際には取引毎（状態遷移毎）に手数料が課されるため、各遷移で手数料を追加で支払わなければ開始状態には戻れないことに注意が必要である。

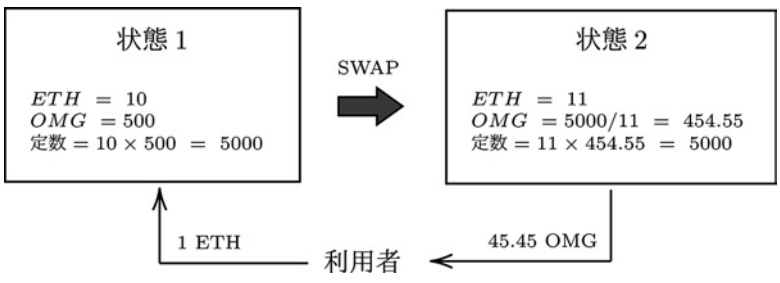

図 11　UniSwap の実行例

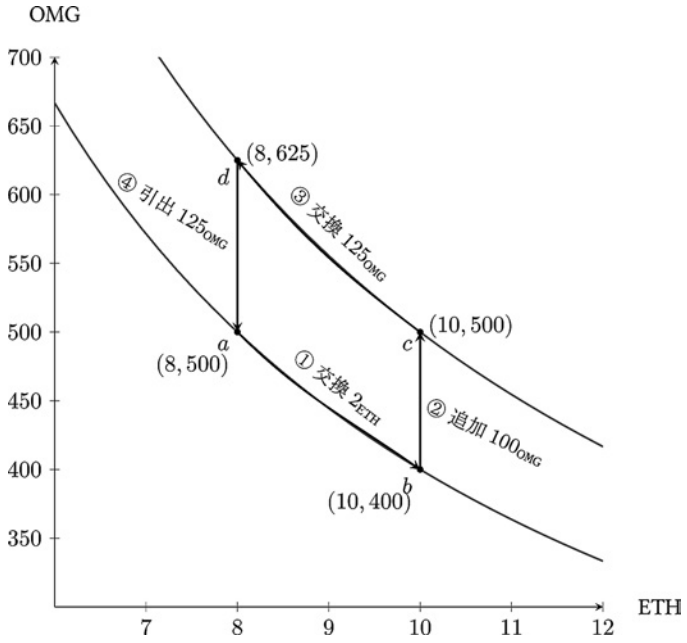

図 12　CPMM[22] の概念

このように、CPMM に基づく AMM アルゴリズムは、流動性プールの残高に応じて交換レートを決定し、自動的に異なるデジタル通貨間の売買を成立させる仕組みである。AMM アルゴリズムには、CPMM の他に、市場に提供された各通貨プールの残高（これを流動性プールという）の和を常に一定に保つように価格を決定するマーケットメーカーである CSMM（Constant-Sum Market Maker）や、それらのハイブリッドアルゴリズムが提案されている。CSMM は、交換レートが一定に保たれるため、Slippage リスク[23]が低い利点があるが、流動性の確保に課題がある。また、UniSwapV3 では流動性プールに資金を投入する際に指値取引に近い、交換価格の区間を指定する機能が追加され、単純な CPMM よりも流動性供給者の資金運用効率を高められるとしている。

B　裁定取引（arbitrage）

裁定取引とは、価格変動において、同一の金融資産の間に市場間で価格差が生じている場合に、同資産を割安な市場で買い、割高な市場で売ることにより、理論上リスクなしに収益を確定させる取引のことを指す。DeFi では、裁定取引により AMM 内の流動性プールの資金残高（すなわち交換レート）を実勢価格に合わせる働きが期待される。裁定取引に関連するスマートコントラクトを理解する上で重要な概念である原子性（atomicity）とフラッシュローン（flash loan）を紹介する。

原子性（atomicity）　ブロックチェーンは、状態レプリケーションを達成する状態遷移機械である。状態レプリケーションは、スマートコントラクや資金移動などのトランザクションが、全てのマイニングノード（EVM）[24]で、完全に同一順序で実行される（合意される）こととして定義されている。EVM の状態は、トランザクションが正常に実行された場合のみ変更され、そうでない場合は、EVM の状態は以前の変更されていない状態に戻される。このような、二者択一の性質をスマートコントラクトの実行に関する原子性と言う。

フラッシュローン（flash loan）　フラッシュローン[25]は、スマートコントラクトの実行に関する原子性を活用して、単一スマートコントラクトの実行中に、融資と返済を完了させる無担保融資の仕組みである。フラッシュローンの流動性プールの残高を上限に、手数料（数ドル程度）の支払により無担保で融資を受けられる。返済に失敗すれば、スマートコントラクトの実行が異常終了し、融資自体も無効化されるため、脆弱性による異常状態に陥るリスクを無視すれば、貸し手は債務不履行のリスクにさらされることはない。

フラッシュローンを用いれば、資金を所持しない利用者でも高額の裁定取引が実行可能になる。図 13 に、典型

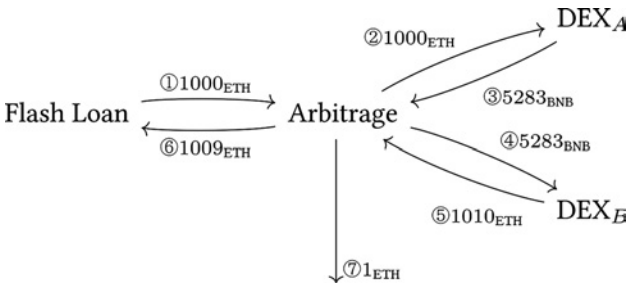

図 13　Flash Loan を用いた裁定取引（arbitrage）

的な裁定取引スマートコントラクト[26]の概要を示す。

今、取引所 A（DEX_A）と取引所 B（DEX_B）で ETH と BNB の 2 つのトークンの交換レートが異なっていた仮定する。フラッシュローンを用いた裁定取引は以下の手順で実行される。

①フラッシュローンの借入れ

Arbitrage はフラッシュローンプロバイダー（Flash Loan）から 1000_{ETH} を無担保で借りる。この時点で、借りた資金は Arbitrage コントラクト内に一時的に保持される。

②③購入

Arbitrage は、取引所 A（DEX_A）で 1000ETH でトークン BNB を安く購入（5283_{BNB}）する。

④⑤売却

Arbitrage は、取引所 B（DEX_B）で 5283_{BNB} を 1010_{ETH} で高く売却する。

⑥⑦返済

裁定取引で得た利益を利用して、フラッシュローンの元本（1000_{ETH}）と発生する手数料（9_{ETH}）を返済します。残額を利益として Arbitrage コントラクトの実行者に送金して利益を確定する。

以上の取引は、一連のスマートコントラクトで実行されるため、返済に失敗した場合（例えば裁定取引が見込み違いだった場合）はフラッシュローンの借り入れから、購入、売却、返済の全取引が無効化され破棄される。

Aave[27]や dYdX[28]は有力なフラッシュローンプロバイダとして知られている。Aave は執筆時点で 200 億米ドルを貸し出し可能な流動性プールに蓄積しており、いずれもフラッシュローン額の 0.09％を手数料として徴収し、日々巨大化している。このように非常に高度な DeFi 技術が構築されているが、意図的に交換レートを操作する等の攻撃も報告されている（Qin et al. [2021]）。安全な DeFi 技術を目指して文字通り日進月歩で技術革新が続いている。

4　今後の展望

ブロックチェーンは、Proof of Work による合意形成システムの画期的な発明により、膨大な通貨発行益をマイニング報酬の形で全ての利用者に公平に配布することで半永久的にシステムを維持するエコシステムを築き上げた。スマートコントラクトは、ブロックチェーンの延伸に伴ってメッセージ駆動型のプログラムが進行する巨大な仮想機械である。非常に多数の計算ノードによってコンセンサスが得られた計算結果のみがブロックチェーンに刻まれることにより、処理プロセスが完全に透明化されており、スマートコントラクトの実行結果は特別な暗号プロトコルを用いなくても、（プログラムに脆弱性がない限り）悪意のある攻撃に晒されることなく複雑な計算を実行できる。

本稿では、これらの画期的な発明の上に開花した新しいサービスとして、ステーブルコイン、非代替性トークン（NFT）、分散金融（DeFi）と自動マーケットメーカー（AMM）を取り上げて解説した。今後もさまざまなサービスが創造され、さらに市場が拡大していくと見込まれている。敢えて、今後の課題として以下を掲げたい。

A. Proof of Work に伴う膨大な電力消費の削減

B. スマートコントラクトの安全性検証技術の確立

C. 生成 AI や大規模言語モデルなどの AI 技術との融合

D. 匿名性と透明性の両立

第一に、Proof of Work による電力消費の増大は、半導体技術の進展による電力削減効果を打ち消すように手数料収入やマイニング報酬の価値が上昇することで生じている。Proof of Work に代わる合意形成技術として、PBFT（Castro and Liskov［1999］, Castro［2001］）とベースコインの所有高比率に応じた確率的要素を加えた Proof of Stake（Kiayias et al.［2017］, Daian, Pass, and Shi［2019］）などの研究があり、Ethereum などの主要ブロックチェーンでも実用化に向けて着手されている。

第二に、2016 年の DAO が Reentrancy 攻撃（Zhou, Milani Fard, and Makanju［2022］）で 360 万 ETH が盗取されたのをはじめ、スマートコントラクトの脆弱性が原因で多くの暗号資産が盗取されており、スマートコントラクトの安全性検証技術の確立が急がれている。スマートコントラクトは、バイトコードが公開され、常にアクセス可能な状態にあることから、複雑な計算状況が生じ得る可能性があり検証が難しい。既に、一般のソフトウェアの安全性検証に用いられるモデル検査技術を応用した手法（Kalra et al.［2018］, Mavridou and Laszka［2018］）やドメイン固有言語（Domain Specific Language）を用いてスマートコントラクト固有の攻撃モデルを記述し、動的な挙動を分析する脆弱性検査技術（Ferreira Torreset al.［2020］）なども提案されている。また、最近では大規模言語モデルを用いて脆弱性の検出や修復コードを提案させる研究も現れ始めている。

第三に、最近の大規模言語モデルや拡散モデルに代表される生成 AI の発展は目覚ましいものがある。本文中でも述べたように、スマートコントラクトは全てブロックチェーン上で公開で実行され、かつ全体で同一の実行結果になるよう実行結果のコンセンサスが達成される必要があるため、秘密情報の扱いや確率的アルゴリズムの導入が単純ではない。他方、生成 AI は膨大なパラメータを備えた確率的アルゴリズムで構成されており、これがスマートコントラクトに生成 AI を導入する際の壁になると予想される。新しい発想で壁を乗り越え、スマートコントラクトに生成 AI を融合させた新しいサービスの登場に期待したい。

最後に、ブロックチェーンの課題として匿名性と透明性の両立（大塚［2022a］、大塚［2022b］）の問題について触れたい。ブロックチェーンは鍵ペアを生成すれば誰でも口座を開設でき、匿名で利用できる極めて開放的なシステムである。近年は、暗号資産取引所に厳格な本人確認を求めることで、法定通貨への換金口座を押さえることで追跡可能性を法的に担保している。この方法は一定の効果があるものの、取引所における鍵の集中管理に起因するサイバー攻撃等による暗号資産の盗難リスクや、過大なプライバシー露出の問題を避けられない。近年、スマートフォンをベースにした本人確認手法の国際標準化が普及進み、社会実装に展開する動きが活発化している。この流れが進展し、CL02 署名（Camenisch and Lysyanskaya［2003］）や BBS＋署名（Boneh, Boyen, and Shacham［2004］）などをベースにした本格的な自己主権型アイデンティティ（Moriyama and Otsuka［2022］）技術が実現すれば、ブロックチェーンにおける匿名性と透明性の両立も解決に向けて大きく前進すると期待している。

（注）

(1) Practical Byzantine Fault Tolerance

(2) ブロックチェーンは一般には木（ツリー）構造をとり、最長のパスが採用される。しかし、最長のパスが複数存在する状況も想定され、この状況はフォーク（Fork）と呼ばれる。フォークは、後で述べる木構造上の最長パスを選択するアルゴリズムを適用することにより、一定時間後に解消され鎖（チェーン）構造になることが証明されている（Garay, Kiayias, and Leonardos［2015］）。

(3) 通貨供給量を増加させた時に信用創造から得られる発行体の収益

(4) 具体的には PoW の条件を満たすブロックが公開された際に、提案された新しいブロックに不正な取引や不正なスマートコントラクトの実行結果が含まれていないことを検証する。これは、さらにその次のブロックのマイニングを開始する際の無駄なマイニングを避けるために必要な検証作業である。

(5) 全採掘者が 1 秒間に試行するハッシュ計算の回数。

(6) ビットコイン・エネルギー消費指標：https://digiconomist.net/bitcoin-energy-consumption/

(7) 暗号学的ハッシュ関数については補論 3 を参照されたい。

(8) ブロックチェーンが矛盾を含まないとは、各ブロックに含まれるすべての取引のデジタル署名などの出力条件が満たされ、かつ取引を先頭から順に実行した際に残高が負の値をとるアドレスを 1 つも含まないこと、さらに各ブロックのハッシュ値が PoW になっていることである。

(9) 二重支払い（double spending）：同一資産を複数の取引に使用し、保有資産額以上の利益を得る攻撃。
(10) より正確には、ビットコインなどの PoW 型のブロックチェーンを対象としている。Proof of Stake（QuantumMechanic［2011］）などの合意形成アルゴリズムに基づくブロックチェーンについては、個別の検討が必要である。
(11) Ethereum Virtual Machine
(12) EVM と Turing 機械は互い他方の動作をシミュレートできることから、等価な計算能力を持つことが証明されている。
(13) スタック型メモリのみを持つ Turing 機械
(14) https://medium.com/tomipioneers/how-to-dao-part-1-proposal-basics-613a9be1f495
(15) https://github.com/samnang/solidity-examples/blob/main/contracts/sample_apps/09_ballot/Ballot.sol
(16) ガバナンストークンの保有数に比例した票数の投票権が与えられる投票方式を token-weighted voting あるいは coin voting と呼ばれる。
(18) https://developer.makerdao.com/dai/1/
(19) Constant Product Market Maker
(20) 実際には、1ETH の 0.3%が交換手数料として差し引かれるが、説明を簡単にするために割愛した。
(21) 状態 c は、a で流動性プールに 2ETH を追加した結果の状態と等価である。
(23) 為替取引の際に交換レートが変動するリスク
(24) マイニングノードは、EVM（Ethereum Virtual Machine）を内部で実行し、実行結果をトライ木に反映し、トライ木のハッシュ値をブロックに組み込む。EVM は各マイニングノード内部で自律分散的に実行されるが、状態レプリケーションが達成されていれば、各マイングノード内部の EVM は結果的に同一の状態で統一される。
(25) Flash lending smart contracts:
(26) https://github.com/marbleprotocol/flash-lending/
(27) https://github.com/aave/aave-protocol
(28) https://dydx.exchange

（参考文献）

Adams, Hayden et al.（2021）. “Uniswap v3 core”. In: URL: https://api.semanticscholar.org/CorpusID:232416764.

Boneh, Dan, Xavier Boyen, and Hovav Shacham（2004）. “Short Group Signatures”. en. In: *Advances in Cryptology–CRYPTO 2004*. Ed. by David Hutchison et al. Vol.3152. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.41–55. ISBN: 978-3-540-22668-0 978-3-540-28628-8. DOI: 10.1007/978-3-540-28628-8_3. URL: http://link.springer.com/10.1007/978-3-540-28628-8_3（visited on 02/26/2023）.

Buterin, Vitalik（2014）. “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.” en. In: URL: https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf.

Camenisch, Jan and Anna Lysyanskaya（2003）. “A Signature Scheme with Efficient Protocols”. en. In: *Security in Communication Networks*. Ed. by Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi. Vol.2576. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.268–289. ISBN: 978-3-540-00420-2 978-3-540-36413-9. DOI: 10.1007/3-540-36413-7_20. URL: http://link.springer.com/10.1007/3-540-36413-7_20（visited on 03/02/2023）.

Castro, Miguel（Jan. 2001）. “Practical Byzantine Fault Tolerance”. Ph.D. MIT.

Castro, Miguel and Barbara Liskov（1999）. “Practical byzantine fault tolerance”. In: *OSDI: Symposium on Operating Systems Design and Implementation* 99, pp.173–186.

Daian, Phil, Rafael Pass, and Elaine Shi（2019）. “Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake”. In: *Financial cryptography and data security*. Ed. by Ian Goldberg and Tyler Moore. Cham: Springer International Publishing, pp.23–41. ISBN: 978-3-030-32101-7. DOI: 10.1007/978-3-030-32101-7_2.

Diffie, Whitfield and Martin E. Hellman（1976）. “New directions in cryptography”. In: *IEEE Trans. Information Theory* 22.6, pp.644–654.

Ferreira Torres, Christof et al.（2020）. “ÆGIS: Shielding vulnerable smart contracts against attacks”. In: *Proceedings of the 15th ACM asia conference on computer and communications security*. Asiaccs ’20. New York, NY, USA: Association for Computing Machinery, pp.584–597. ISBN: 978-1-4503-6750-9. DOI: 10.1145/3320269.3384756. URL: https://doi.org/10.1145/3320269.3384756.

Garay, Juan, Aggelos Kiayias, and Nikos Leonardos（2015）. “The Bitcoin Backbone Protocol: Analysis and Applications”. en. In: *Advances in Cryptology–EUROCRYPT 2015*. Vol.LNCS 9057. Springer, pp.281–310.

Kalra, Sukrit et al.（2018）. “ZEUS: Analyzing safety of smart contracts”. In: *Network and distributed system security symposium*. DOI: 10.14722/ndss.2018.23082. URL: https://api.semanticscholar.org/CorpusID:3481056.

Kiayias, Aggelos et al.（2017）. “Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol”. en. In: *Advances in Cryptolo-*

gy–CRYPTO 2017. Vol.10401. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, pp.357–388. ISBN: 978–3–319–63687–0 978–3–319–63688–7. DOI: 10.1007/978–3–319–63688–7_12. URL: http://link.springer.com/10.1007/978–3–319–63688–7_12（visited on 01/14/2021）.

Mavridou, Anastasia and Aron Laszka（2018）. "Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach". en. In: *Financial Cryptography and Data Security*. Ed. by Sarah Meiklejohn and Kazue Sako. Vol.10957. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.523–540. ISBN: 978–3–662–58386–9 978–3–662–58387–6. DOI: 10.1007/978–3–662–58387–6_28. URL: http://link.springer.com/10.1007/978–3–662–58387–6_28（visited on 06/10/2024）.

Merkle, R.C.（Jan. 1982）. *Method of providing digital signatures*. Google Patents. URL: https://www.google.com/patents/US4309569.

Moriyama, Koichi and Akira Otsuka（Aug. 2022）. "Permissionless Blockchain–Based Sybil–Resistant Self–Sovereign Identity Utilizing Attested Execution Secure Processors". en. In: *2022 IEEE International Conference on Blockchain（Blockchain）*. Espoo, Finland: IEEE, pp.1–10. ISBN: 978–1–66546–104–7. DOI: 10.1109/Blockchain55522.2022.00012. URL: https://ieeexplore.ieee.org/document/9881847/（visited on 03/02/2023）.

Nakamoto, Satoshi（2008）. *Bitcoin: A peer–to–peer electronic cash system*. URL: https://bitcoin.org/bitcoin.pdf.

Qin, Kaihua et al.（2021）. "Attacking the DeFi ecosystem with flash loans for fun and profit". In: *Financial cryptography and data security*. Ed. by Nikita Borisov and Claudia Diaz. Berlin, Heidelberg: Springer Berlin Heidelberg, pp.3–32. ISBN: 978–3–662–64322–8.

QuantumMechanic（July 2011）. *Proof of stake instead of proof of work*. URL: https://bitcointalk.org/index.php?topic=27787.20.

Rivest, R. L., A. Shamir, and L. Adleman（Feb. 1978）. "A method for obtaining digital signatures and public–key cryptosystems". In: *Communications of The Acm* 21.2, pp.120–126. ISSN: 0001–0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

Wood, G.（2014）. *Ethereum: A secure decentralised generalised transaction ledger*. Tech. rep. Ethereum Project Yellow Paper.

Zhou, Haozhe, Amin Milani Fard, and Adetokunbo Makanju（May 2022）. "The State of Ethereum Smart Contracts Security: Vulnerabilities, Countermeasures, and Tool Support". en. In: *Journal of Cybersecurity and Privacy* 2.2, pp.358–378. ISSN: 2624–800X. DOI: 10.3390/jcp2020019. URL: https://www.mdpi.com/2624–800X/2/2/19（visited on 06/10/2024）.

大塚　玲（Jan. 2022a）。"ブロックチェーンを利用した暗号資産の安全性と匿名性：原理と限界"。ja. In：**日本銀行金融研究所** 44.1、pp.1–56。

－（Oct. 2022b）。"耐タンパ一性に基づくデジタル通貨ウオレットの研究動向：匿名性と透明性の両立に向けて"。ja. In：**日本銀行金融研究所** 2022–J–9、p.35。

（原稿受領 2024.6.11）