WIDE Technical-Report in 2014

ビットコインにおけるトランザクション、その展性と影響

wide-tr-ideon-bitcoin-transaction2014-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on WIDE documents, please contact to board@wide.ad.jp

Title: ビットコインにおけるトランザクション、その展性と影響
Author(s): 斉藤 賢爾 (ks91@wide.ad.jp)
Date: 2014-05-16

ビットコインにおけるトランザクション、その展性と影響

斉藤 賢爾
ks91@sfc.wide.ad.jp

2014 年 5 月 16 日

概 要

2014 年 2 月上旬、デジタル通貨ビットコイン **(Bitcoin)**[2, 5] のネットワークに対して「トランザクション展性 (transaction malleability)」を利用した攻撃が行われ、独自ソフトウェアを用いていた交換所の一部が機能を停止させた。そのような交換所のひとつだった Mt.Gox は、他の交換所が数日でソフトウェアの改修を終え再開したのに対し、2 月 25 日に交換取引を全面停止し、2 月 28 日には民事再生手続き開始の申立てを行った [15][1]。その記者会見の中で、Mt.Gox が経営破綻に至った理由として、トランザクション展性を用いた攻撃により大量のコイン (約 85 万 BTC[2]) が盗難に遭ったことが示唆された。しかし、2014 年 3 月に arXiv に投稿された論文 [3] によって、その時期の当該攻撃による損失が、全世界で 386BTC 程度であることが明らかにされた。

本稿は、ビットコインにおけるトランザクションのデータ構造と処理を説明し、[3] で示された結果を解説するとともに、トランザクション展性を利用した攻撃の、コインの損失に留まらない影響について論じる。そうした影響のひとつは、トランザクション処理を司るソフトウェアが展性を利用した攻撃に対して脆弱である場合、すなわち、コインが使用済みであるかどうかを独自に管理している場合に、そのことに起因して表れる現象であり、そうした場合、取引が正常に完了しないという一点において、秘密鍵が漏洩し送金がすでに行われてしまっている状態とも区別できない。このことは、当時の Mt.Gox における混乱をよく説明できるように思える。

1 はじめに

ビットコイン **(Bitcoin)** は、サトシ・ナカモトを名乗る人物により 2008 年に提唱 [5] され、2009 年に実運用が開始されたデジタル通貨 (デジタル通信技術を用いたオルタナティブな通貨) である。

ビットコインが社会的に話題になったことの背景には、円・ドル・ユーロなどの国民通貨によるコインの売買が盛んになり、特に 2013 年に、その価格が高騰したことがあげられる。そのような売買を行うサービスは「交換所 (exchange)」と呼ばれる。これは、各国の国民通貨間の交換になぞらえれば、外貨両替所に当たる。ビットコインを通常の商取引における支払いに用いる際は、交換所は不要であることを念のため特に述べておく。

2014 年 2 月上旬、ビットコインのネットワークに対して「トランザクション展性 (transaction malleability)」(第 3.1節) を利用した攻撃が行われ、独自ソフトウェアを用いていた交換所の一部が機能を停止させた。そのような交換所のひとつだった Mt.Gox は、他の交換所が数日でソフトウェアの改修を終え再開したのに対し、2 月 25 日に交換取引を全面停止し、2 月 28 日には民事再生手続き開始の申立てを行った [15] (同年 4 月 16 日、東京地裁は当該申立てを棄却した [17])。

2 月 28 日の記者会見の中で、Mt.Gox が経営破綻に至った理由として、トランザクション展性を利用した攻撃により、大量のコイン (額面にして約 85 万 BTC — 後に約 20 万 BTC が残存していたことが判明し、損失は約

[1] 2014 年 4 月 16 日、東京地裁は当該申立てを棄却した [17]。
[2] 後に約 20 万 BTC が残存していたことが判明し、損失は約 65 万 BTC と見込まれている [16]。

65 万 BTC と見込まれている [16]) が盗難に遭ったことが示唆された。しかし、2014 年 3 月に arXiv に投稿された論文 [3] によって、その時期の当該攻撃による損失が、全世界で 386BTC 程度であることが明らかにされた。

本稿は、ビットコインにおけるトランザクションのデータ構造と処理を説明し、[3] で示された結果を解説するとともに、トランザクション展性を利用した攻撃の、コインの損失に留まらない影響について論じる。

筆者による、ビットコインシステム全体の問題の分析および意見については、[13, 14] を参照されたい。

用語　交換所は「取引所」と呼ばれることも多いが、本稿ではビットコインにおける通常の支払い・送金の動作を「取引」と呼称するため、概念の明確化のために、コインの売買のサービスは「交換所」と呼ぶ。交換所でコインが売買される際に行われる送金については、特に「交換取引」と呼ぶ。

本稿では「トランザクション」および「取引」を同じ意味で用いる。ここで、トランザクションは日常語としての売買・取引の意味に近く、データベース分野におけるトランザクションとは異なる概念である。例えば、後述するように、ビットコインにおけるトランザクションは、確率的な意味でしか永続性を持たない。

本稿では、ビットコインのシステムを「ビットコイン」、取引される個々のコインを「コイン」、コインの額面を「BTC」と呼ぶ。本稿の性質上、特にコインと額面を分離して表記することは重要である。

対象とする読者　本稿の内容は、主として計算機科学 (computer science) の範囲に留まるが、話題の性質上、金融関係、報道関係、あるいは法曹関係で、必ずしも計算機科学の知識を持たない読者が読むことを想定し、計算機科学の基礎知識に当たる内容も解説している部分がある。

2　ビットコインにおけるトランザクション

この節では、ビットコインにおけるトランザクションのデータ構造と処理を解説する。技術仕様の詳細については [1] を参照されたい。

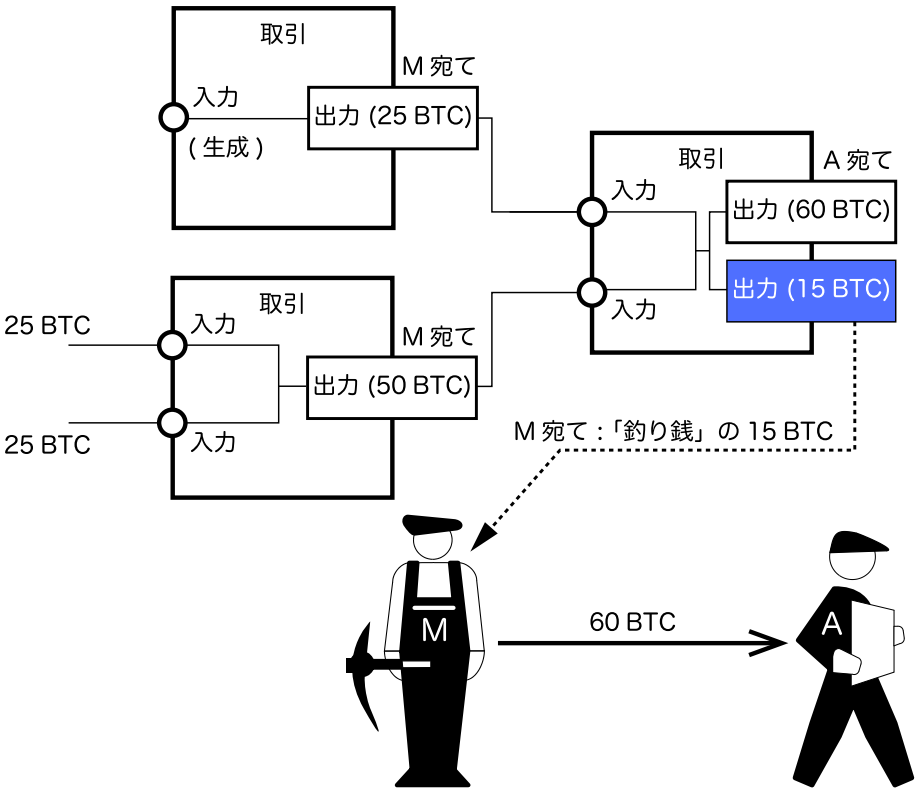
2.1　トランザクションの概要

ビットコインにおけるトランザクションとは、コインによる送金の表現である。取引データ自体を指してトランザクションと呼ぶことも多い。

図 1に、ビットコインにおけるトランザクションの概要を示した。これは M から A に 60BTC を送金している例である。トランザクションのデータ構造は、「出力 (出金するコイン)」と「入力 (入金に用いるコインへの参照)」を持つ。送金には、相手に渡すコインと、その額面、および宛先が必要であるが、それらを表現したものが出力である。自分に宛てられたコインを使う (=誰かに送金する) ためには、過去のトランザクションの出力を参照しなければならないが、それが入力である。

出力と入力は、それぞれ複数個指定できる。任意の額の送金を行うためには、図 1が示すように、過去の自分宛のトランザクションの出力を集め、これから行う取引の入力として指定する。入力に指定した BTC の総量が、送金したい額面を超える場合は、「釣り銭」用の出力を自分に宛てる。したがって、出力は、典型的には、相手に宛てるものと、「釣り銭」のふたつになる。

「釣り銭」を含む出力の総額が、入力の総額に満たない場合、差額は取引の手数料となる。手数料は、当該取引の承認に成功した「マイナー (miner)」のものとなる。



- 自分に宛てられた出力を入力として参照し、自分が渡したい相手への出力につなげて取引データを作る。
- 一度参照された出力は消費済みとなり、二度と使えない。
- この例では取引手数料は省いている。

図 1: トランザクションの出力と入力

2.2 トランザクションの承認

マイナーは、ビットコインのネットワークにブロードキャスト (=広く転送) された取引データを収集し、データのブロックに格納する。そして、ブロックの内容から得られる暗号学的ハッシュ値 (第 2.4節) を利用した確率的操作 (くじ引きに当たる) に成功すると、当該ブロックを、唯一の元帳である「ブロックチェイン (block chain)」(=承認されたブロックの連鎖) の末尾に追加するプロポーザルを行えることになる。

これが「マイニング」の操作であり、複数のマイナーによる競争的プロセスである。マイニングに成功したマイナーは、報酬 (2014 年現在で 25BTC) を得られる。この報酬および取引手数料を自分に宛てるため、マイナーは、コインを新たに生成するトランザクション (生成取引) をブロックに組み込んでからマイニングの操作を行う。生成取引は、無から (あるいはブロックの位置[3]と内容から) 新たにコインを生み出す操作であり、参照する入力を持たない (図 1の左上の取引)。

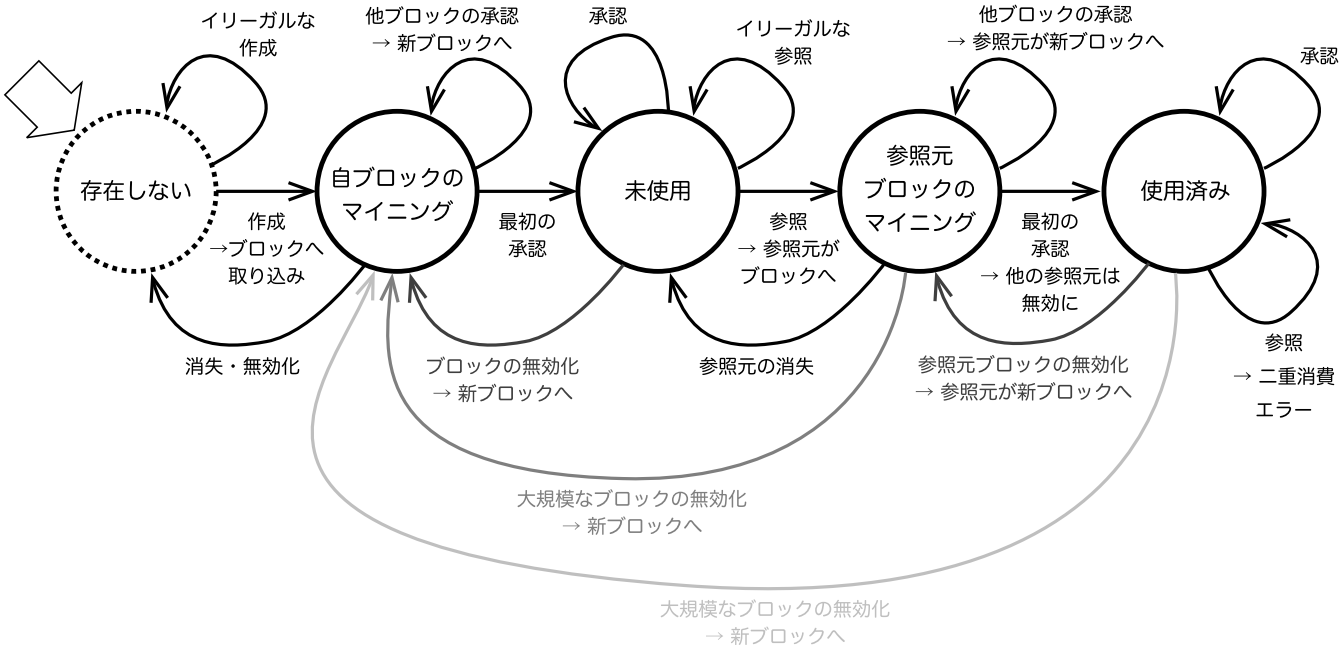
マイニングされたブロックの内容が正しいことが、他のマイナーたちから追認されると、当該ブロックはブロックチェインの新たな末尾となる。すなわち、その後ろにブロックが連なっていくことになる。ブロックの内容は、

[3] マイニングの報酬は、ブロックチェインの先頭で 50BTC から始まり、210,000 ブロック毎 (マイニングが平均 10 分間に 1 回、成功するとして、約 4 年毎) に半減する。

直前のブロックの内容から得られるハッシュ値に依存しているため、過去のトランザクションを改ざんするためには、当該ブロック以降のマイニングをやり直さなければならない。マイニングの難易度は、マイナー全体が持つ計算力に対して、平均して 10 分間に 1 回、成功するように調整されており、悪意をもつ一部のマイナーが、正規のマイニングのペースを追い越して自らによる改ざんを正当化させることは、極めて困難とされる。

2.3 コインのライフサイクル

ビットコインにおけるコインの実体は、トランザクションの出力であると言える。コインはトランザクションの出力として生まれ、当該トランザクションを格納するブロックがブロックチェインにつながる (=取引がネットワークにより承認される) ことで使用可能となり、実際に使用されることでその役割を終える。国が発行する硬貨に、1 円、5 円、10 円など、さまざまな額面があるように、ビットコインにおけるコインはそれぞれが額面をもつ。



- 薄い矢印ほど、実際に遷移する可能性が低い

図 2: コイン (トランザクションの出力) の状態遷移

図 2に、コインの状態遷移図を示した。状態遷移図は、対象がとりうる状態を円で示し、その間の遷移を矢印で示したものである。

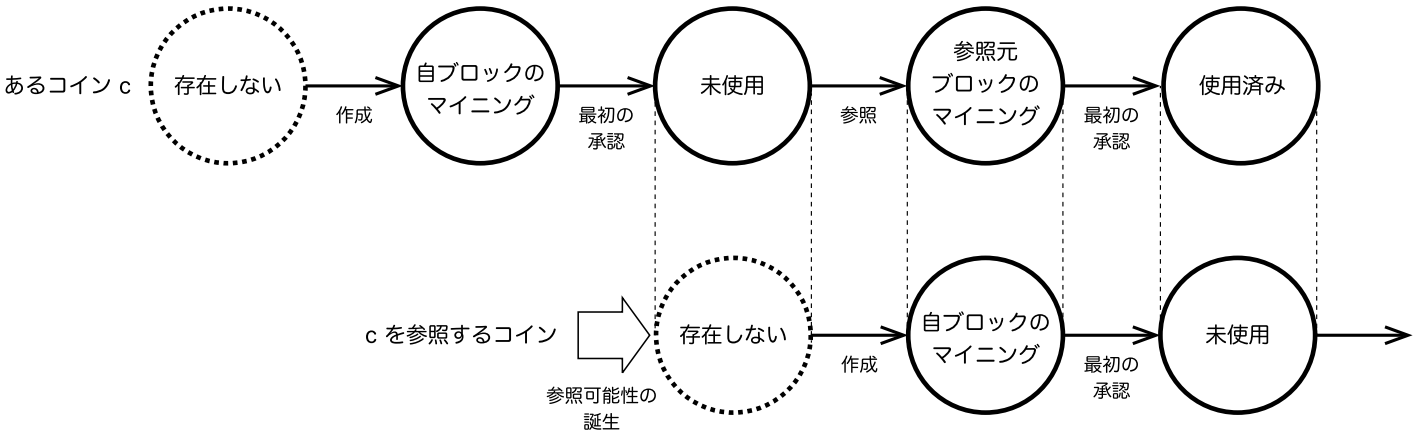
コインの状態の変化を大まかに捉えると、「存在しない」→「未使用」→「使用済み」となるが、実際に未使用あるいは使用済みのコインとしてネットワークに認められるためには、マイニングの操作を経なければならない。そこで、図ではマイニング中の状態を中間状態として置いている。

コインは、トランザクションの出力として作成され、当該トランザクションがブロードキャストされてブロックの中に取り込まれ、マイニングを経て承認されると、未使用の状態になる。しかし、マイニングは複数のマイ

ナーにより並行して行われるため、ネットワークの別々の箇所でほぼ同時に新たなブロックがそれぞれ承認され、追認されてブロックチェインが分岐して伸張していく可能性がある。その場合、より確率的に困難な方のチェイン (≒より長いチェイン) が採用され、そうでない方のチェインに含まれるブロックからはトランザクションが取り出され、マイニングがやり直される。そのため、希ではあるが、ブロックが無効化される遷移を考慮しなければならない。

後述するように、取引の入力は、参照するトランザクションをそのハッシュ値により識別する。そのため基本的に、過去のトランザクションを参照する取引については、正しく構成されているのであれば、ブロックが無効化されたとしてもその正当性は変わらない。しかし、ブロックチェインが再構成される際に、同じコインを参照している別の取引が先に承認され、コインが無効となり消失する可能性もある。そこで慣習的に、実際にコインを使用するのは、当該コインがブロックチェインに組み込まれたブロックを含み、6 ブロックより後ろにすることが推奨されている (ブロックチェインの分岐の問題は、大抵の場合 6 ブロック以内に解決されることが見込まれていると言える)。

生成取引は、過去のトランザクションを参照せず、組み込まれたブロックに特有のものであるので、ブロックが無効化されるならば、即、コインが消失することになる。当該コインを入力として参照しているトランザクションがあるとすれば、その取引 (および参照により連なる以降のすべての取引) も無効になることから、一般のトランザクションよりもブロックの無効化の影響が大きい。そのため、生成取引に関しては、安全のため、当該ブロックを含み、100 ブロックより後ろでなければ参照できないことになっている。



- 主な遷移のみ示した。

図 3: コインの状態遷移の連なり

あるコイン c の参照とは、c を入力 (の一部) とするコインが作成されることであり、c の「参照元ブロックのマイニング」は、新たに作成されたコインにとっての「自ブロックのマイニング」である。図3に示したように、参照を通してコインは連鎖しており、その状態遷移は重なり合っている。

2.4 トランザクションとデジタル署名

トランザクションでは、コインが宛てられた相手しかそのコインを使用できないよう、アクセス制御を実現しなければならない。トランザクションの出力が、使用者の条件を表し、それを参照する入力は、その条件を満た

していることの証明になっている必要がある。かつ、トランザクションの入力と出力の関係が改ざんできないようにする必要がある。これらの実現のため、ビットコインでは、「デジタル署名 (digital signature)」[7] の技術を用いている。

ここで、デジタル署名の仕組みを簡単に振り返りたい。各自は「公開鍵」K と「秘密鍵」K^{-1} の「鍵ペア」$< K, K^{-1} >$ を持ち、公開鍵 K は公開し、秘密鍵 K^{-1} は隠し持っておく。秘密鍵で暗号化したデータは、ペアとなる公開鍵でしか復号できない。

署名　あるデータ m に対しデジタル署名を施す際は、まず m に「暗号学的ハッシュ関数[4]」H を適用した固定長 (実際の関数により異なるが 160bit, 256bit 等) の値である $H(m)$ を計算する。この値を「暗号学的ハッシュ値」または単に「ハッシュ値」と呼ぶ。ハッシュ値を秘密鍵 K^{-1} で暗号化した $\{H(m)\}_{K^{-1}}$ を「デジタル署名」または単に「署名」と呼び、m とともに相手に送る。

署名の検証　相手も m からハッシュ値 $H(m)$ を計算し、それが署名を公開鍵 K で復号して得られる内容と一致するかを確かめる。このことを署名の「検証」と呼ぶ。一致しているならば、次のふたつのことが言える。

1. 間違いなく本人が署名を行った (秘密鍵は本人のみが使用できるように隠されているため)。
2. 署名された後、データは改ざんされていない。

ビットコインでは、デジタル署名のアルゴリズムとして、秘密鍵の長さを 256bit に設定した「楕円曲線 DSA (ECDSA)」を用いる[5]。公開鍵の長さは 512bit となる。

図 4は、トランザクションの入出力において、どのようにデジタル署名が利用されているかを大まかに示したものである。この機構は、次のふたつの役割を持つ。

1. 出力が、コインの正当な所有者により参照されていることを証明する。
2. 参照する出力の集合を、当該トランザクションの出力の集合に結びつける関係を改ざん不可能 (改ざんされた場合に検出可能) にする。

出力の宛先は、利用者の公開鍵のハッシュ値で指定する。これは、暗号学的ハッシュ関数 SHA-256[6] と RIPEMD-160[7]をその順序で公開鍵[8]に適用した、160bit の二重ハッシュ値である。

ビットコインでは、公開鍵のハッシュ値に限らず、二重ハッシュ値が多用される。ハッシュ関数を二重に適用する理由は、想像するに、ハッシュ関数の入力について特定の条件を満たすことにより、ハッシュ値を衝突させることが可能な攻撃が発見されたとしても、2 段階目について同様に攻撃できる可能性は極めて希になると考えたからだと思われる。

---

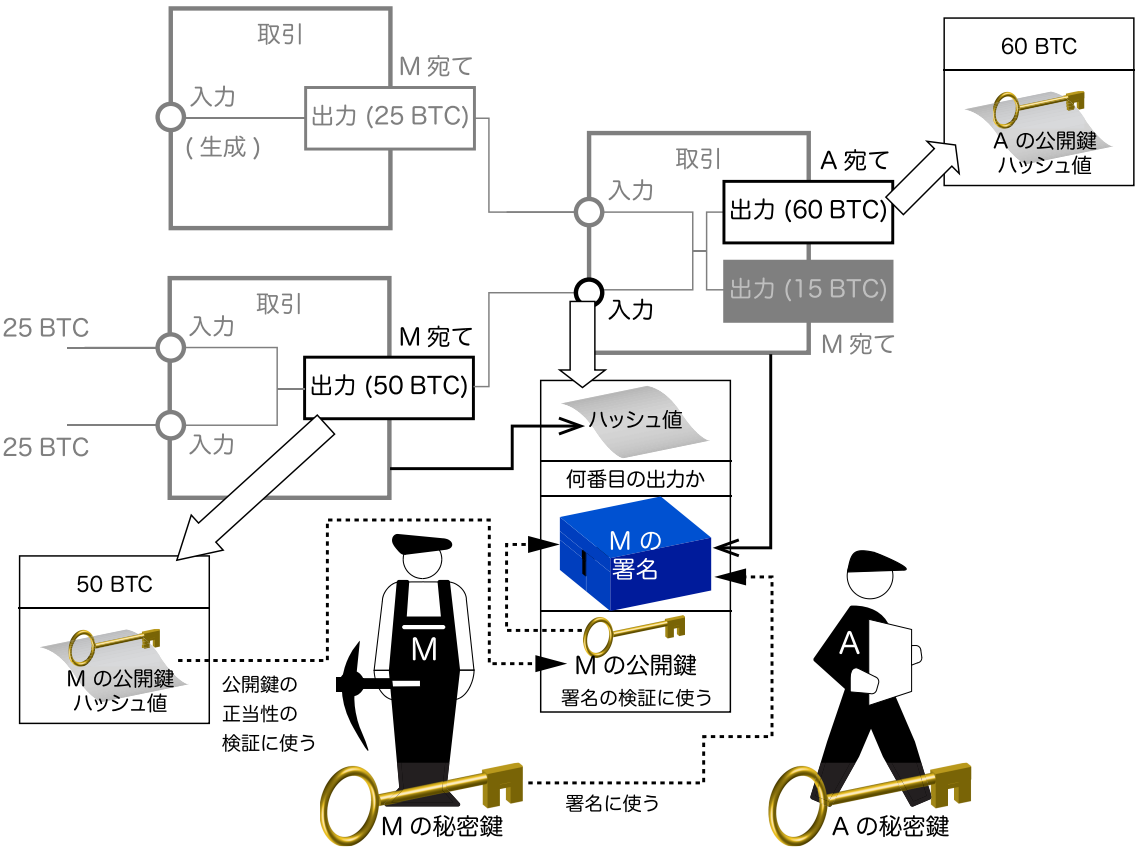
[4]暗号学的ハッシュ関数 H は、データ m が与えられた場合、$H(m) = H(m')$ となるような m' (ただし $m' \neq m$) を現実的な時間内に計算により求めることができず、したがって、ハッシュ値 $H(m)$ が与えられても m や m' を現実的な時間内に計算により求めることができないという性質を持つ。

[5] ECDSA (Elliptic Curve Digital Signature Algorithm) は、「楕円曲線暗号」(楕円曲線上の離散対数問題を利用) をデジタル署名に適用したアルゴリズムである [11]。ビットコインでは、楕円曲線のパラメータとして、国際コンソーシアム SECG (The Standards for Efficient Cryptography Group) が推奨しているものの中から secp256k1 を採用している。

[6]SHA-2 (Secure Hash Algorithm-2) のバリエーション [9]。

[7]RIPEMD (RACE Integrity Primitives Evaluation Message Digest) を 160bit にした改良版 [8]。

[8]正確には、公開鍵の先頭に 1 バイトの鍵長コードを置いたデータに二重ハッシュ関数を適用する。



- 署名時には、「M の署名」と「M の公開鍵」の部分を、参照する出力の「M の公開鍵ハッシュ値」の部分で置き換えておく。

図 4: トランザクションの入力におけるデジタル署名

出力において、公開鍵そのもののデータではなく、ハッシュ値を用いるのは、相手の公開鍵 (512bit) を持たなくとも、ハッシュ値 (160bit) を識別子として用いて指定できるようにするためである。このことは、システム全体の運用を考えた場合、利用者の利便性を増すが、同時に、セキュリティ面での懸念を生む。本来、攻撃のためには秘密鍵の 256bit の空間を探索しなければならないところ、160bit の空間の中で公開鍵ハッシュ値を衝突させることができれば、すなわち、同じ公開鍵ハッシュ値が得られる秘密鍵を見つけることができれば、正当な秘密鍵を持たなくとも送金が可能になるためである。

ビットコインにおいて利用者が用いる宛先の識別子は、上記の公開鍵ハッシュ値に、さらに SHA-256 を二重に適用したハッシュ値の先頭 4 バイトにより誤り検出用のチェック符号を得、併せて Base58 形式[9]で表現したものである[10]。ビットコインでは、この識別子は「アドレス」と呼ばれる。一方、秘密鍵は、ビットコインの用語では「プライベートキー」と呼ばれる。

出力は、取引データのハッシュ値と、当該データにおける出力の番号により参照される。出力をこの形式で参照することは、取引データが変化しないことを前提とすれば、図 2に示したようなブロックの無効化が生じ、ト

[9] バイナリデータを文字形式により表現する体系のひとつであり、英数字のみを用い、かつ、「0」と「O」、「I」と「l」という、人間が識別しにくい文字を除いた 58 文字を用いる工夫がされている [6]。

[10]正確には、公開鍵ハッシュ値の先頭に 1 バイトのネットワークコードを置いた上で、末尾にチェック符号が置かれる。

ランザクションが別のブロックに組み込まれることになったとしても、参照元を直さなくてもそのまま参照関係を維持できるという利点がある。ただし、ブロックの無効化が生じた際に、本稿の大きなテーマであるトランザクション展性を利用した攻撃が行われると、このことは成り立たなくなる (第 4.1節参照)。

送金指示を行う者は、取引データ全体に対して署名することになるが、当然のことながら、署名データは署名の結果であるので、署名データに対して署名は施せない。そこで、署名時には、入力のデータのうち、署名と公開鍵は取り除かれ、参照する出力の、公開鍵ハッシュ値を指定する部分で置き換えるという方法が採られる (詳細を後述する)。

2.5 トランザクションのデータ構造

表 1: トランザクション全体のデータ形式

フィールド	説明	サイズ (バイト)
バージョン番号	現在は 1	4
入力の数	正の整数 (可変長)	1~9
入力のリスト		可変
出力の数	正の整数 (可変長)	1~9
出力のリスト		可変
ロック時刻		4

取引データはバイナリ形式で表現される。表 1に、取引データ全体の構造を示した。ビットコインのデータ形式における可変長整数は、8bit, 16bit, 32bit, 64bit の符号なし整数を、それぞれ 1 バイトと、制御コード 1 バイトを含む 3 バイト、5 バイト、9 バイトで表現する形式である。「ロック時刻」は、ブロック番号 (500,000,000 未満) またはタイムスタンプにより指定され、後述する入力データ内の「シーケンス番号」と組み合わせることで、その時点までトランザクションの内容が更新されるのを許可することを示す (滅多に使われない)。

表 2: 出力のデータ形式

フィールド	説明	サイズ (バイト)
額面	10^{-8}BTC 単位の値	8
スクリプト長	正の整数 (可変長)	1~9
スクリプト		可変

表 2に、取引データ内における出力のデータ形式を示した。10^{-8}BTC は、ビットコインの発案者/初期実装者の仮名に因んで *satoshi* とも呼ばれる BTC の最小単位である。「スクリプト」は、コインの使用の条件やその条件が満たされていることの証明を記述するための、簡単なプログラムである。スクリプトの形式と処理については第 2.6節で述べる。出力におけるスクリプトは、典型的には宛先となる公開鍵ハッシュ値を指定する。

表 3に、取引データ内における入力のデータ形式を示した。参照するコインは、そのコインが出力として作成されているトランザクションのハッシュ値と、当該トランザクション内における当該出力のインデクス番号により指定される。入力におけるスクリプトは、典型的には署名データと公開鍵を指定する。署名を実施する前の段

表 3: 入力のデータ形式

フィールド	説明	サイズ (バイト)
参照トランザクション	二重 SHA-256 ハッシュ値	32
参照出力	出力のインデクス	4
スクリプト長	正の整数 (可変長)	1∼9
スクリプト		可変
シーケンス番号		4

階では、取引のすべての入力におけるスクリプトは 1 バイトの「0x00」[11]により置き換えられ、署名する際に、対象となる入力は、参照する出力のスクリプト (の一部) により置き換えられる[12]。すべての署名が完了すると、入力は署名データを含むスクリプトで置き換えられる。「シーケンス番号」は、トランザクションの更新を許可し、そのバージョンを記述するために用いられる (滅多に使われない)。

2.6 トランザクション処理とスクリプティング

入出力におけるスクリプトは、演算子を被演算子の後ろに置く「後置記法」の形式をもつ簡易なプログラミング言語により記述される[13]。この言語では、演算子には各々 1 バイトのコードが割り当てられる。

出力側:
OP_DUP OP_HASH160 OP_PUSHDATA* < 公開鍵ハッシュ値 > OP_EQUALVERIFY OP_CHECKSIG

入力側:
OP_PUSHDATA* < 署名 > OP_PUSHDATA* < 公開鍵 >

図 5: 公開鍵ハッシュ値に宛てたトランザクションの出力と、参照する入力のスクリプト

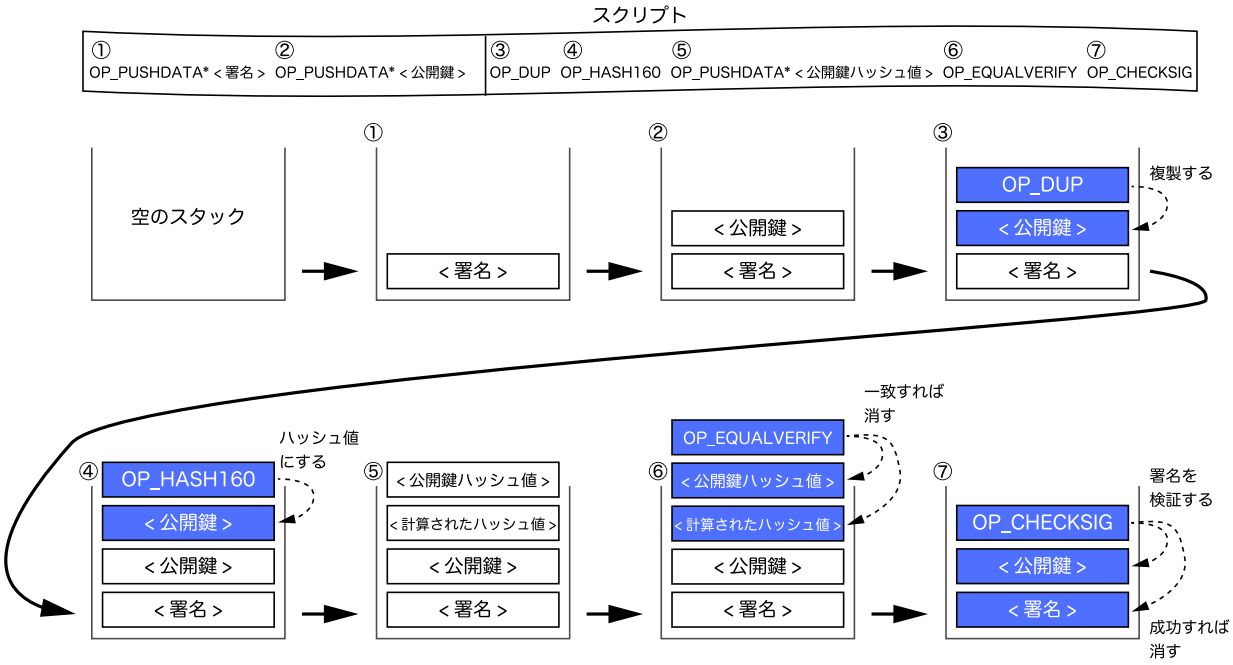
図 5に、公開鍵ハッシュ値に宛てたトランザクション (すなわち、一般的な取引) の出力と、それを参照する入力に用いられるスクリプトを例示した。「OP_」で始まる要素は演算子を示している。「OP_PUSHDATA*」は、例外的に前置形となる演算子で、その直後に続く被演算子をスタック (後から入れたものを先に出すようなデータ構造) に積む操作を示し、被演算子となるデータの大きさに応じて幾つかの表現形式を持つ (このことがトランザクション展性を可能にする要因のひとつとなっている)。スクリプトは、入力側から先に読むように構成されている。

この例を用いて、図 6に、スクリプトの処理を図示した。まず入力側と出力側のスクリプトをその順序で結合する。そして先頭から順にスタックに積み、演算子が現れたら、それに応じた処理を行う。この例では、スクリプトの先頭は署名のデータであり、続いて公開鍵のデータがある。それらをスタックに積んだ後に現れる「OP_DUP」

[11] 0x は、16 進数を表す接頭辞であり、C 言語などのプログラミング言語で用いられている。16 進数では、1 バイトの数 (0∼255) は 2 桁の数字で表される。

[12] 可変長整数により表される長さに続いてスクリプトが置かれる。

[13] 後置記法は「逆ポーランド記法」[10] とも呼ばれ、図 6に示したように、スタックを用いた処理により、簡単な言語処理系で実行できる特徴を持つ。そのため、機器に組み込めるような小型な処理系が要求されるプログラミング言語の形式として向いている。例えば、プリンタに処理系が内蔵されるページ記述言語である PostScript は、後置記法で記述される。その意味で、組込み応用も想定できるビットコインで後置記法の言語を採用したことは、設計上の判断としては妥当と言える。



- 入力側 → 出力側の順にスクリプトを結合する。
- 左から順にスタックに積み、処理していく (実際の処理では、演算子を律儀にスタックに積む必要は無い)。

図 6: スクリプトの処理

は、スタックの一番上に積まれているデータ (この場合、公開鍵) を複製 (duplicate) し、スタックに積む操作である。その次に現れる「OP_HASH160」は、スタックに積まれている公開鍵の 160bit のハッシュ値を計算し、計算された値で置き換える操作である。次に、スクリプトで指定された公開鍵ハッシュ値がスタックに積まれる。その次に現れる「OP_EQUALVERIFY」は、スタックに積まれているふたつのデータが一致するかを確認し、一致していればスタックから取り除く。最後に現れる「OP_CHECKSIG」は、スタックに積まれている公開鍵を用いて署名を検証する。ここまでエラーなく進行すれば、出力のスクリプトで指定された条件が、入力のスクリプトで指定されたデータにより満たされていることが確認できたことになる。

出力側:

OP_PUSHDATA* < 公開鍵 > OP_CHECKSIG

入力側:

OP_PUSHDATA* < 署名 >

図 7: 生成取引の出力と、参照する入力のスクリプト

一般の取引では、公開鍵ハッシュ値により宛先を指定するため、以上のように比較的複雑な処理となる。しかし、生成取引の場合は、出力を自分に宛てるため、自分が所持している公開鍵を直接スクリプトに組み込むことができ、ハッシュ値により宛先を指定する必要はない。また、出力されたコインを使用する入力側でも、署名の

みを指定すればよいようにスクリプトを構成できる。この場合、秘密鍵の 256bit がそのままの強度で用いられるため、衝突攻撃等に対して、より安全でもある。

図 7に、生成取引における出力と、それを参照する入力で用いられるスクリプトを示した。この場合、図 6における①, ②, ⑦ の順で処理することに等しくなる。

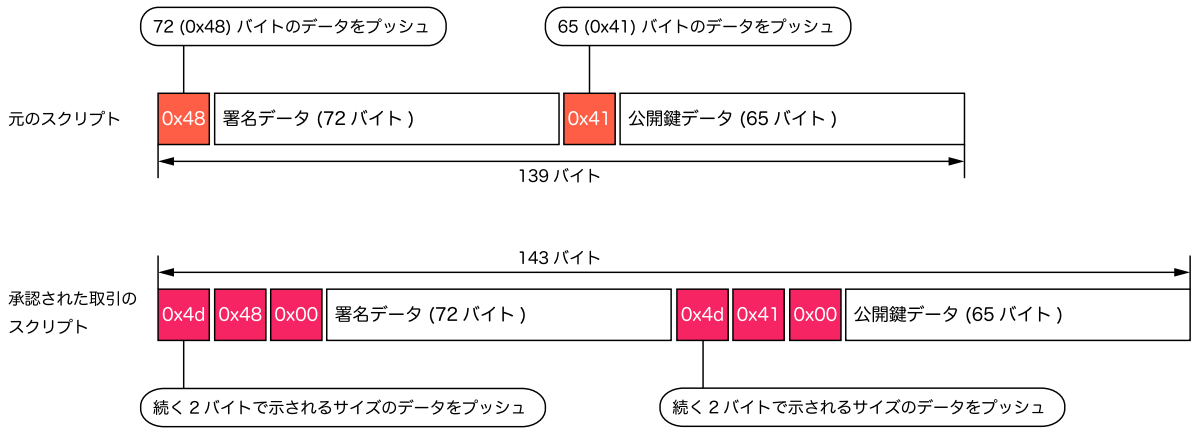
3　トランザクション展性と攻撃

3.1　トランザクション展性

「展性 (malleability)」とは、もともと、金(きん)などが、打撃などによって破壊されずに伸びる性質を示す。ビットコインにおけるトランザクション展性は、取引データが、その意味を変えず、入力で施されている署名も正しく検証されるまま、変化 (典型的には伸張) できる性質を表す。

前述のように、取引データを署名する際には、入力のスクリプトは署名の対象から除外されている。従って、入力のスクリプトの意味を変えず、その表現を変化させることができれば、トランザクションの意味を変えず、署名も検証可能なまま、取引データの内容を改ざんすることが可能になる。

このことが問題となるのは、トランザクションが、そのハッシュ値である TXID により識別されるからである。トランザクション展性が利用されると、同一の取引でありながら、TXID が変化してしまうことになる。



- ふたつのスクリプトの意味は同じ。

図 8: 署名スクリプトの展性

図 8は、実際に攻撃に使われた展性の例である。

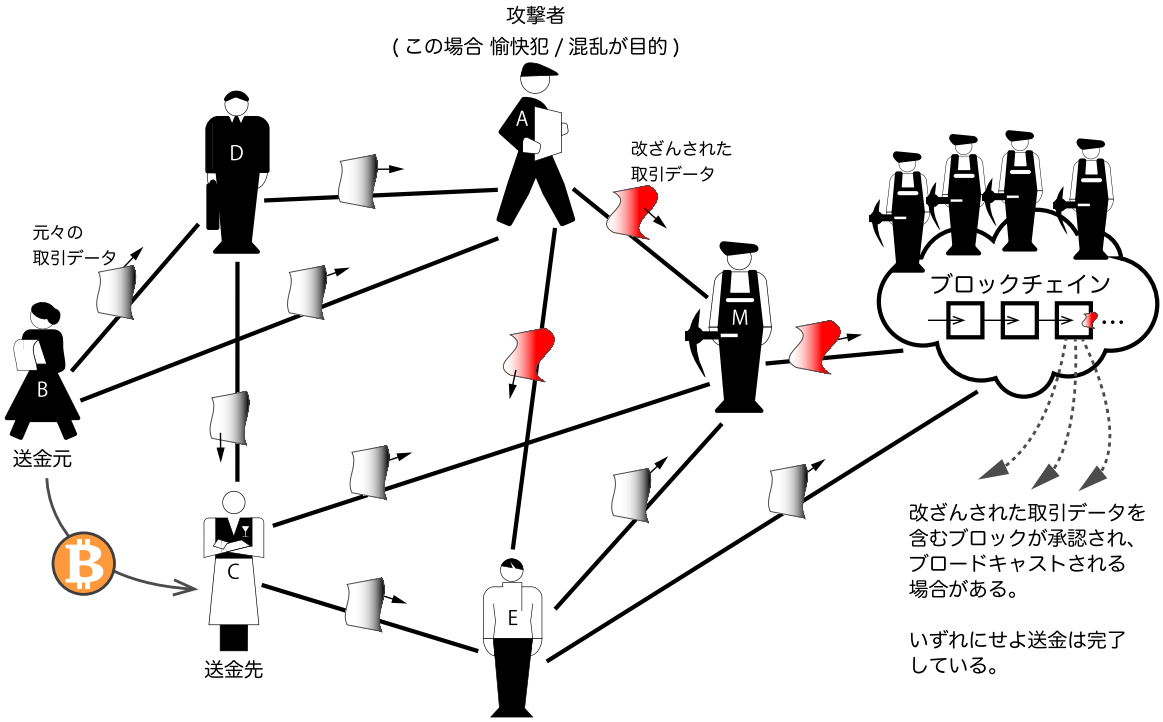
元々のスクリプトでは、署名データと公開鍵データをそれぞれスタックにプッシュすることを指示するために、最も短く書ける方法を用い、1 バイトでそれぞれのデータのサイズを指定している[14]。

改ざんされたスクリプトでは、同じ操作を指示するために、「OP_PUSH_DATA2」(0x4d) と、つづく 2 バイト (下位バイトを先に置く) を用いてデータのサイズを指定している。スクリプトの動作はまったく同じであり、署名の検証は、こちらでも等しく成功する。

[14] ビットコインのスクリプトでは、0x01∼0x4b は、その数が表すサイズのデータをプッシュする演算子となっている。

このことを利用した攻撃に、取引が完了していないと見せかけ、送金元から二重に送金させることを目的としたものがある。トランザクションがネットワークにブロードキャストされ、マイナーに届くまでの間に改ざんが行われ、元々のスクリプトを含むトランザクションではなく、改ざんされたスクリプトを含むトランザクションの方が承認されるとき、攻撃は半ば成功することになる。実際に攻撃が成功したと言えるのは、それにより、送金元が取引の完了を確認できなくなったときである。

3.2 トランザクション展性を用いた攻撃とその実際



- 元々の取引データと改ざんされたデータは同じ出力の集合を参照しているため、どちらか一方のみが承認され、ブロックチェインに組み込まれる。
- 確率的に、改ざんされた取引データの方が承認され、元々の取引データは二重消費と見なされ弾かれる場合がある。
- そのような場合、送金元のソフトウェアが元々の取引データの TXID を用いてブロックチェインの中を探しており、正しくブロックチェインと同期してコインの状態を把握していないとすれば、送金が完了していることを確認できない。

図 9: トランザクション展性を利用した攻撃

図 9 に、トランザクション展性を利用した攻撃の動作を示した。この図では、B から C に送金をしているが、第三者である A が攻撃者であり、B からブロードキャストされた取引データを改ざんして転送している。

この例では、攻撃が成功した場合、取引が完了していないと誤解した B から別のコインで二重に支払いを受けるのは C であり、C が攻撃による利益を得る。従って、通常は C が攻撃者である例を考えることが多い。しかし、ブロックチェインには両方の送金の記録が残るのであるから、攻撃の痕跡は明らかであり、C には自らが疑われるリスクがあることになる。

一方、動機を別とすれば、原理的に、取引データの転送の経路上に位置するのであれば、誰でも攻撃を仕掛けることが可能である。図 9 では、そのような例を描いた。

取引データの転送は、次のように行われる。ネットワークに参加する各ノードは、自分が受け取った取引データのハッシュ値を隣接するノードに送る。受け取った側は、自分が持っていないデータであれば、実際のデータを要求する。その要求に答えるかたちでデータが転送される。このとき、リファレンス実装では、ローカルに保存している取引データと矛盾ないし一致するトランザクション、すなわち、すでに受信済みの取引データにおいて入力が参照しているのと同じ出力を参照している取引データを受け取った場合、そのデータは破棄される。

図 9 では、E と M が元々の取引データと改ざんされたデータの両方を受け取っているが、E は元々のデータを、M は改ざんされたデータをそれぞれ先に受け取っているため、後から受信されたデータは他のノードに転送されない。

M を含む各マイナーは、各々、どちらかの取引データを先に受信し、それを採用し、ブロックに組み込んでマイニングを行う。確率的に、改ざんされたトランザクションの方が承認されることになる。

もし B が正しくブロックチェインと同期してコインの状態を把握しておらず、ハッシュ値のみで自分が送金した際の取引データをブロックチェインの中に見つけようとするならば、送金が完了していることを確認できないことになる。

3.3 チューリッヒ工科大学の研究者による調査と分析

この節では、2013 年から 2014 年 2 月にかけて行われたトランザクション展性を利用した攻撃に関する、[3] で示された調査結果について解説する。

[3] の著者らは、ビットコインのネットワークに自らが制御するノード群を参加させ、2013 年の 1 月から、ネットワーク内にブロードキャストされた取引データ (およびブロック) を収集した。彼らのノード群は、ビットコインネットワークの通信可能なノードのうち平均して約 20% と常に接続しており、ほとんどのトランザクションを収集できたと推定できる。彼らは、ブロックチェインに含まれるトランザクションも合わせて[15]、次のように調査を行った。

1. すべての取引データの入力のスクリプトを取り除き、残った部分のハッシュ値を求める。
2. このハッシュ値が一致する取引を「衝突集合 (conflict set)」としてまとめる。

結果、35,202 個の衝突集合が得られ、そのうち 29,139 個が、正当な取引として承認されうるものだったという。29,139 個のうち 28,595 個が、図 8で示したタイプの展性を利用していた。そのうち、5,670 個で改ざん後のトランザクションが承認されていた。

攻撃の対象となった取引の送金総額は 302,700BTC であり、実際に承認されたのは 64,564BTC だという (送金側のソフトウェアが正しければトランザクションは正常に完了していることが確認できるので、この額の被害があったということにはならない)。そのうち、1,811BTC が Mt.Gox による交換取引の停止 (引き出しのみ) の前に攻撃の対象となり、実際にそれまでに承認されたのは、全世界で 386BTC と推定される。

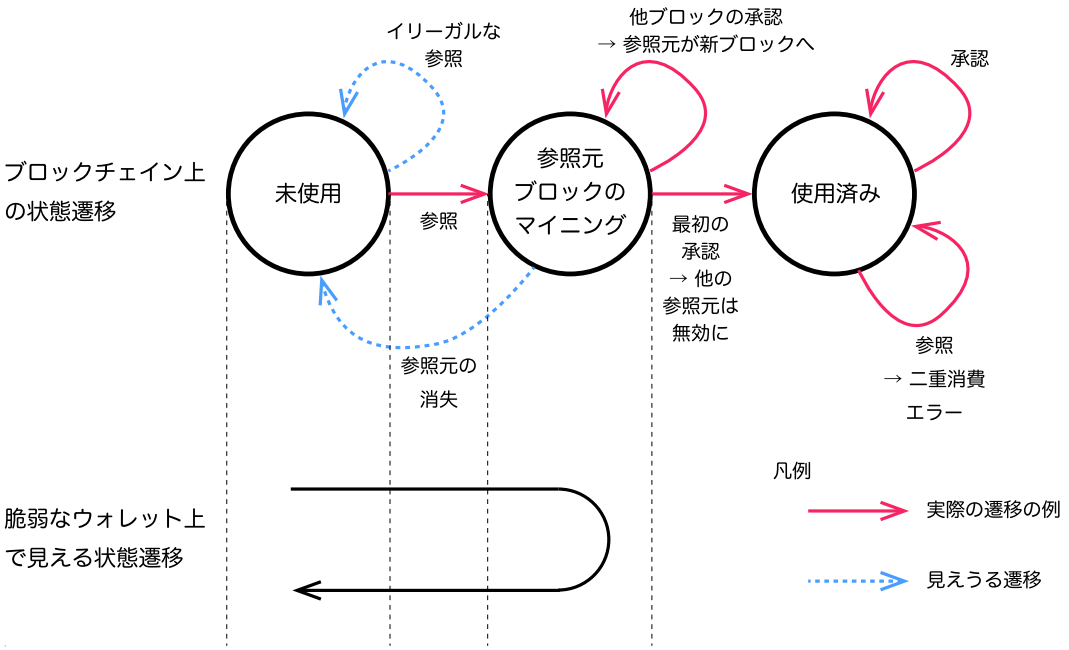
[15] [3] の記述からは、ブロックチェインに含まれるトランザクションも調査の対象であったかが不明瞭なように思われたが、著者らに確認したところ、ブロックチェイン内のトランザクションを含むとの回答を得た。すなわち、彼らのノード群が改ざんされたトランザクションを取得できていなくても、元々のトランザクションを取得していれば、改ざん後のトランザクションがブロックチェインの中にあるなら、成功した攻撃を検出できることになる。

以上のように、ほとんどの攻撃は交換取引の停止後に発生したため、Mt.Gox には影響していないはずである。[3] の著者らは、多くの攻撃は、Mt.Gox からのプレスリリースが契機となってもたらされたと推察している。

攻撃の 78.64% が無効であり、元々のトランザクションに有利であるのは、取引データがブロードキャストされる経路の途中で改ざんされるため、多くのノードは先に元々のトランザクションを受け取り、改ざんされたトランザクションが転送されないためだと考えられる。

4　トランザクション展性を利用した攻撃の影響

4.1　攻撃の波及効果



トランザクション展性を利用した攻撃の場合: 脆弱なウォレット (送金処理を司るソフトウェア) が「参照」することを契機に攻撃が行われ、当該ウォレットからは最終的に「未使用」状態が観測される。

秘密鍵が盗まれている場合: 任意のタイミングで送金が行われ、脆弱なウォレットが「参照」すると実際には二重消費のエラーとなるが、当該ウォレットからは、やはり最終的に「未使用」状態が観測される。

図 10: 攻撃の影響による状態遷移の齟齬

Mt.Gox の混乱の要因　上記のように、トランザクション展性を用いた攻撃による Mt.Gox の損失は、最大に見積もっても 386BTC 以下だと推定できるのに対し、Mt.Gox は約 85 万 BTC (後に約 65 万 BTC と訂正) の損失の要因を当該攻撃に求めていた。好意的に考えるならば、Mt.Gox は激しく混乱していたことになる。その混乱の理由は何だろうか。

第 2.3節で示したコインの状態遷移 (図 2) は、交換所のソフトウェアがトランザクション展性を利用した攻撃に対して脆弱な場合、その影響が波及的に拡大することを示唆する。すなわち、実際に行われた攻撃の規模に対

して、より大きな影響を受けるため、Mt.Gox が攻撃の規模を誤解していた可能性を考えることができる。

図 10に、攻撃の影響により、脆弱なウォレット (送金処理を司るソフトウェア) 上で見えるコインの状態が、実際の状態と食い違う様子を示した。

脆弱なウォレットは、ブロックチェインと正しく同期せず、ローカルな取引記録における TXID に基づいて、ブロックチェイン内のトランザクションを検索することに特徴がある。したがって、トランザクション展性を利用した攻撃が成功した場合、当該ウォレットから見ると、コインは決して使用済み状態に遷移しない。コインを参照している取引データが、ブロードキャスト中、あるいはマイニングの過程の中で消失したか、または、イリーガルな参照を行ったと見なす、ということにならざるを得ない。前者と見なす場合、トランザクションの再送を行うが、これは実際の状態では二重消費と見なされるためエラーになる。これもウォレットの側では消失と観察されるが、何度再送しても消失するので、当該ウォレット (の利用者) が合理的に判断するなら、イリーガルな参照[16]をしているという判断になる。したがって、当該コインは未使用として温存し、ウォレットのソフトウェアを更新したタイミングを見計らうなどして、再度、別の取引で使用を試すことになる。それも実際には二重消費と見なされるので、やはり取引は完了しない。攻撃は過去のものなのだが、コインを未使用と勘違いしているため、完了しない取引が波及的に増えていくことになる。

例えば、コインの集合 $\{\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, \mathbf{c_4}\}$ を入力として参照するトランザクションが、展性を利用した攻撃を受け、改ざんされたトランザクションの方が承認されたとする。すると、これら 4 枚のコインはすべてが使用済み状態に遷移するが、脆弱なウォレットからは、最終的に未使用に見える。

これらのコインを、その後、それぞれ $\{\mathbf{c_1}, c_5\}, \{\mathbf{c_2}, c_6, c_7\}, \{\mathbf{c_3}, c_8\}, \{\mathbf{c_4}, c_9, c_{10}\}$ といった入力の組をもつトランザクションで使おうと試みると、これらすべてのトランザクションが二重消費エラーにより完了しないことになる。この例では、1 回の攻撃で、計 5 つのトランザクションが正常に完了しないことになるし、その後も完了しないトランザクションがネットワークに投入され続けることになる。

また、攻撃の規模を誤解することに加え、別種の攻撃により Mt.Gox の秘密鍵が漏洩し、コインが大量に盗まれ、使用済みの状態になったが、そのことと、トランザクション展性を利用した攻撃の影響が区別できなかったとも考えられる。図 10は、取引が正常に完了しないという一点において、トランザクション展性を利用した攻撃の影響が、秘密鍵が漏洩し送金がすでに行われてしまっている状態と区別できないことも示している。

実際に、2013 年 10 月下旬以降、Mt.Gox からの交換取引にてコインが二重消費エラーとなる例が頻出していたことが、ビットコイン利用者の掲示板である Bitcoin Forum にて話題になっている [4] 。

以上のことは、Mt.Gox における当時の混乱をよく説明できるように思える。

ブロックチェインの分岐と再構成における影響　トランザクション展性を利用した攻撃が、ブロックチェインを分岐させる攻撃とともに使われる場合、特に効果的にシステムを混乱させることができる恐れがある。

大規模なブロックの無効化が発生した場合、ブロックチェインの分岐からの回復時に展性を利用した攻撃が行われ、無効になったブロックに格納されていたトランザクションと意味は同じだがハッシュ値が異なる取引データが新しいブロックに組み込まれると、元々のトランザクションに連なる以降のトランザクションがすべて無効になり、混乱を助長する恐れがある。

大規模なブロックの無効化を発生させうる攻撃の可能性については、別の稿で論じる。

[16]例えば、ビットコインの開発コミュニティにおいてプロトコルが更新され、独自のソフトウェアがそれに追従できていない状況では、実際にイリーガルな参照が行われる場合がある。

4.2 対策

トランザクション展性の問題は、遅くとも 2011 年にはビットコインの開発コミュニティの中で知られており、Mt.Gox にて混乱があったような問題に関しては、現在のリファレンス実装では改修されている。

交換所のソフトウェアは、多くの顧客を相手に大量のトランザクションをさばく必要があるため、Mt.Gox に限らず、ウォレットに独自の実装を採用している場合が多い。特に Mt.Gox は古くから営業を始めた交換所だったため、処理に古いスタイルが残ってしまったと考えられる。

現状、二重支払いを誘発するためにトランザクション展性を利用する攻撃が行われたとしても、その影響を受けるようなソフトウェアは少なく、被害は非常に限定的になると考えられる。

しかし、第 4.1節でも述べたように、ブロックチェインが分岐から回復する際、ブロックから取り出された過去のトランザクションが展性により改ざんされ、ブロックチェインに組み込まれると、当該トランザクションを参照する以降のトランザクションが無効になる問題がある。これはむしろ、ビットコインの設計において本質的な問題のひとつだと考えることができ、対策は必要だと考える。

考えうる対策　ビットコインの開発コミュニティで考えられているのは、署名スクリプトの表現を正規化することである [12]。しかし、このことは簡単ではない。正規化は、スクリプティングによる自由度とのトレードオフになってしまうからである。

ビットコインの基本的な設計を崩さない、かつ、より根本的な解決は、署名スクリプトを署名の対象にすることだと筆者は考える。そのためには、スクリプトから署名データを取り除く必要がある。そのことは、次の方法で可能だと考える。

- 現在のスクリプトで未使用のバイトコードを「署名をスタックに積む」の意味とし、スクリプトには当該コードのみを埋め込む。
- 実際の署名データは、取引データの末尾に新たに設ける独立したセクションに登場順に置く。

署名データの表現を正規化することは、開発コミュニティにおいても検討されている[17]が、より根本的な展性の回避方法としては、署名されている部分のみ (すなわち署名データのセクションを省いた部分) に暗号学的ハッシュ関数を適用したものを TXID とすることが考えられる。

5 おわりに

Mt.Gox の記者会見資料 [15] には、「平成 26 年 2 月初め頃、ビットコインのシステムのバグを悪用した不正アクセスにより、ビットコインの送金 (ビットコインの引出) が正常に完了しない取引が増え、また、かかるバグを悪用した不正アクセスにより、ビットコインが不正に引き出されている可能性があることが判明しました」とある。このバグがトランザクション展性のことを指すとすれば、[3] が指摘したように、この主張は事実を反映していない可能性が高い。好意的に考えて、Mt.Gox には大きな混乱があったと思われる。この稿では、ビットコインにおけるトランザクションの仕組みを振り返るとともに、Mt.Gox の混乱の要因を探り、また、トランザクション展性がより本質的に問題となる場面とその対策について論じた。

本稿で提案した対策は、現状のビットコインプロトコルと互換ではないため、実際の導入は難しいかも知れない。しかし、ビットコインプロトコルに基づく新たな通貨システムを設計する際などに参考になれば幸いである。

[17] 楕円曲線 DSA では、既存の署名データから、同様に検証可能な別の署名データを計算的に求めることが可能だという問題があるが、そのことへの対処も含む。

参考文献

[1] Bitcoin community. Bitcoin. https://en.bitcoin.it/.

[2] Bitcoin Project. Bitcoin - Open source P2P money, as of 2013. Hypertext document. Available electronically at http://bitcoin.org.

[3] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox, 2014. Available electronically at http://arxiv.org/abs/1403.6676v1.

[4] Bitcoin Forum. MTGOX makes doublespend transactions for btc withdrawal request. https://bitcointalk.org/index.php?topic=314901.0.

[5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Available electronically at http://bitcoin.org/bitcoin.pdf.

[6] Wikipedia contributors. Base58 - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Base58.

[7] Wikipedia contributors. Digital signature - Wikipedia, the free encyclopedia. Available electronically at http://en.wikipedia.org/wiki/Digital_signature.

[8] Wikipedia contributors. RIPEMD - Wikipedia. http://ja.wikipedia.org/wiki/RIPEMD.

[9] Wikipedia contributors. SHA-2 - Wikipedia. http://ja.wikipedia.org/wiki/SHA-2.

[10] Wikipedia contributors. 逆ポーランド記法 - Wikipedia. http://ja.wikipedia.org/wiki/逆ポーランド記法.

[11] Wikipedia contributors. 楕円曲線 DSA - Wikipedia. http://ja.wikipedia.org/wiki/楕円曲線 DSA.

[12] Pieter Wuille. Dealing with malleability. https://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki.

[13] 斉藤賢爾. ビットコイン — 人間不在のデジタル巨石貨幣, 2013. WIDE Technical-Report. Available electronically at http://member.wide.ad.jp/tr/wide-tr-ideon-bitcoin2013-00.pdf.

[14] 斉藤賢爾. これでわかったビットコイン — 生きのこる通貨の条件. 太郎次郎社エディタス, 2014.

[15] 株式会社 MTGOX. 平成 26 年 2 月 28 日 民事再生手続開始の申立てに関するお知らせ, 2014. Available electronically at https://www.mtgox.com/img/pdf/20140228-announcement_jp.pdf.

[16] 株式会社 MTGOX. 平成 26 年 3 月 20 日 当社保有ビットコインの残高に関するお知らせ, 2014. Available electronically at https://www.mtgox.com/img/pdf/20140320-btc-announce.pdf.

[17] 株式会社 MTGOX. 平成 26 年 4 月 16 日 民事再生手続開始申立ての棄却、保全管理命令のお知らせ, 2014. Available electronically at https://www.mtgox.com/img/pdf/20140416_001_press_release.pdf.