大容量 OP_RETURN 領域をもつ Bitcoin 派生のブロックチェーンにおける任意情報の効率的な保存

Document version 0.4

Fredrick R. Brennan
copypaste@kittens.ph

Ronald Watkins
admin@susukino.com

September 11, 2018

1 導入

ブロックチェーンへの任意情報の記録は Bitcoin のかなり早い段階から行われてきた。この動向はサトシ・ナカモトのころまでもさかのぼることができよう。この人物は Bitcoin のジェネシス・ブロックの `coinbase` 領域を用いて “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks[1].”という文章を記録した。それから、数多くの手段──トランザクションの出力や `OP_RETURN` フィールド、そのほか P2PKH トランザクションの使用といった巧妙なる方法まで──を通して、ブロックチェーン上において任意情報を記録することの可能性が見出されてきたのである。

最近のこの動向に関する顕著な例として、Memo プロトコル[2]があげられる。Memo は Bitcoin Cash において拡張された 220 バイトの `OP_RETURN` 領域を使用してひとつのソーシャル・ネットワークを実現したものである[3]。

この論文は SUMO（*Storage Utility Memory Object*）というシステムを提唱するものである。このシステムは Susucoin やそれに準じた他のブロックチェーンにおいて効率的な情報の記録に使用されるだろう。

---

[1] ブロックチェーンにおける任意情報の記録の例について詳しくはこちらを参照のこと：http://www.righto.com/2014/02/ascii-bernanke-wikileaksphotographs.html

[2] https://memo.cash/about

[3] https://memo.cash/protocol を参照のこと。 SQL データベースの VARCHAR の限界値である 255 バイトに近い。このテーブルの Value カラムにおける限界値はハード限界（hard limit）である。

1.1 目標

この規格文書の作成にあたって次の点が考慮された:

1. **効率性.** ブロックチェーン上においては 1 つ 1 つのバイトもそれぞれ重要である。このプロジェクトの目的はそれらを有効活用することにある。プログラマーにとっての利便性は重要であるが、記録容量の削減もより重要である。そういうわけで容量の削減には最善を尽くした。

2. **容易性.** ウェブからモバイルアプリケーションにいたるまでの多種多様なプラットフォームにおいて標準的な実装の容易性は担保されなければならない。

3. **拡張性.** 標準的な実装は異なる圧縮方法やデータ・タイプ、メタデータに応じて拡張可能である必要がある。

SUMO の目的は情報を永久的にブロックチェーン上に保存し、かつ効率性・容易性・拡張性を兼ね備えたプロトコルを提供することなのである。

2 SUMO プロトコル

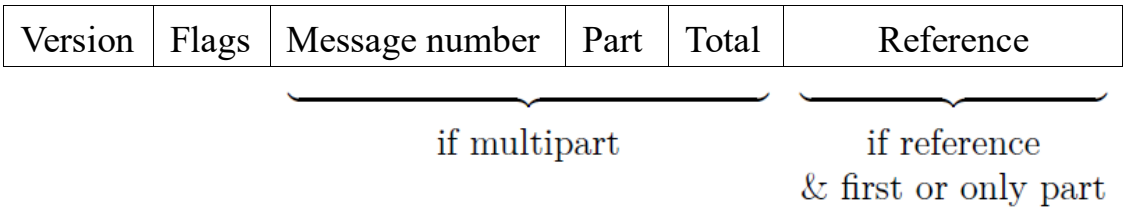
SUMO は実際のブロックチェーンの仕様に即した拡張可能なプロトコルである。

2.1 SUMO のヘッダ要素について

原則として、すべてのヘッダの要素は可変長量（VLQ, Variable-Length Quantities）で `uintvar` として知られているものである。`uintvar` は、その最も基本的な定義から言えば、符号付き 8 ビットの整数型で、符号ビットを用いて後に続く別の情報の存在を示すものだ。このことは実際に意味をもつ情報が 0〜127 の符号無し整数型の範囲であるのに、その記録に 1 バイト分を要してしまうことを示すが、同時にどんな大きさの（CPU アーキテクチャのサポートを超えたものであっても ）記録することが可能であることも示している。

2.2 ヘッダ

SUMO のヘッダの各要素は可変長で、普通は何らかの識別番号と実際に使用される値の 2 バイトであることが多い。ヘッダは以下のようである:



2.2.1 Version

Version は `uintvar` 形式である。最初は 0x01 が当てられるだろう。

2.2.2 Flags

次に述べる各 Flag 情報 は 1 つのバイト列にビットマスクとして定義される。0 は偽（false）を、 1 は真（true）を示す:

- 1 ビット目: この情報の後に続く別の情報が存在するか？ [1]
- 2 ビット目: 内容はブロックチェーン上の別の場所にある他の任意情報の参照であるか？
- 3 ビット目: Payload は複数の部分から構成されているか？
- 4-5 ビット目: この領域は、実装者の裁量によって、さらなる圧縮アルゴリズムを追加するなど、その他の目的に応じて使用できる。
- 6-8 ビット目: 圧縮アルゴリズムを明示する 3 ビットの整数型。

利用可能な圧縮アルゴリズム

- 000: 非圧縮
- 001-111(1-7): 実装者次第

多種多様なブロックチェーンが存在しているので、保存されうるデータの種類を前もって判断することは困難である。多くの場合は短い文章が保存されるので Zstandard などのエントロピーエンコーダーの使用が好ましい。また大部分が ASCII や Latin-1 であるテキストを保存しようとするのなら、 Shoco の使用が好ましいだろう。Shoco は”There was an old man in London named Trent”という文章に対して 26%の圧縮比率を達成している（2018 年 6 月 28 日現在）[2]。

---

[1] 実装者はこの領域を使用して、他の情報をヘッダに追加するなど、さまざまなオプションを実装することができる。

[2] 出典: http://ed-von-schleck.github.io/shoco/

実装者が複数の異なるデータ形式の情報を保存したいと考えるのはごく自然なことである。すべてのクライアントにおいて 3 ビットで示された情報がどの圧縮方法を意味しているかについて共通した認識を持っている限りは、これは問題にならない。送信者(encoder)についてもまた必ずしもすべての圧縮方法を知っている必要があるわけではないが、受信者(decoder)についてはその必要がある。入力された文章が十分に短いとき、然るべき送信者(encoder)ならば、その入力に最適な方法をさがしだすために利用可能なすべてのやり方で圧縮しようと努めるだろう。

日本語と中国語の文章の両方を保存しようとする場合を例に挙げよう。まず Zstandard を Wikipedia データベース上の日本語と中国語の文章によって学習させ、余分な文字情報を処理し、その 2 つの言語の辞書データを作成する。そして日本語辞書付き Zstandard を圧縮方法の 1 番目として、中国語辞書付き Zstandard をその 2 番目として、3 番目から 7 番目を将来使用する領域として定義しよう。その後でブロックチェーンに PNG 圧縮ファイルの保存のサポートを追加したいと考えたとき、確保されていた 3 番目の方法として PNG 圧縮アルゴリズムを定義してみてもいいだろう。

2.2.3 Message Number

Message Number は `uintvar` 形式である。これを使用することで、同一アドレスからの複数の部分にわたるメッセージを区別することができる。ひとつのメッセージを構成するすべての部分情報は必ず同一のアドレスから送信されなければならない。これは `uintvar` 形式であるので、1 つのアドレスから 128 通以上のメッセージが送信された際には、2 バイト分の Message Number が確保されるだろう。

2.2.4 Part と Total

Part と Total は `uintvar` 形式である。これらは複数の部分にわたるメッセージにおいて用いられる。大量の情報量でない限り、ふつう 1 バイト分が確保される。256 バイトが最大の `OP_RETURN` 領域と 5 バイトのヘッダ情報をもつなら、1 バイト分の Part と Total で 32KB 程度のメッセージ(payload)を定義することができる。SUMO プロトコルの実装において、どのくらいの容量の任意情報をサポートするかについてはクライアントの判断に委ねられている。

2.2.5 Reference

Reference は 32 バイトのトランザクション ID を示し、インプット・データが同一の部分情報に含まれる他のデータと繋がりを持つ場合に使用されるものである。複数にわたるトランザクションはバイト値 0x1D を用いてそれらを区別することによって参照することができる。Payload において、バイト値 0x1A を用いることで左端の非ゼロ値へ参照を挿入することができる。

複数のトランザクションを扱うには、トランザクションの後ろにバイト値 0x1A 0x32（ASCII で‘2’を示す）を加えるとよい。参照可能なデータの最大量は親ブロックチェーンの OP_RETURN の最大値に依存する。

2.3 Payload

Payload はヘッダ以外の情報であり、記録される本文の内容を示すものである。Payload は圧縮・非圧縮のどちらでも可で、ヘッダの Flags 情報に依存する。これは圧縮アルゴリズムがその性能外れて悪影響を及ぼしてしまうのを防ぐためである。というのは、圧縮比率が常に 1.0 を下回っているとは限らないということである。圧縮アルゴリズムは保存されているデータ・タイプを特定しないので、あるデータ・タイプに合わせて設計された圧縮アルゴリズムが異なるデータ・タイプをより効率的に圧縮することがあり、またそういうわけで送信者(encoder)によってあえてそれが選択される可能性があるのである。実装をするクライアントは Payload そのものにおいてデータ・タイプを明示しなければならない。例えば、UTF-8 と ASCII それに SHIFT_JIS のそれぞれのテキストを区別しようとするとき:①まず UTF-8 用に BOM（Byte Order Mark）を使用し②0～127 の範囲の全てのバイト情報が ASCII を示しているというやや発見的な保証に頼り③それら以外の情報を SHIFT_JIS と見なすというものだ。

2.4 安全性

安全性は全てのソフトウェアの関心事である。SUMO プロトコルは可能な限りの容量確保を重視する一方で、容易に実装することが可能なように設計され、そのうえ圧縮技術の使用を提供しているが、それそのものの事実によっ

て高圧縮ファイル爆弾攻撃に対する脆弱性を孕んでいる。実装をするクライアントによって RAM の使用量を制限するなどの対策が講じられる必要がある（例：`ZSTD_DCtx_setMaxWindowSize`）。

SUMO プロトコルは可変長に対応したものであるので、整合性の確認を頻繁に行うことが強く推奨されている。メモリ・セーフなコンパイラ言語（Rust, Haskell など）で設計をし、そして他のプログラムには FFI（Foreign Function Interface）を介して組み込むことが望ましい。

3 例示

SUMO プロトコルは多種多様なアプリケーションにおいて実装されるだろう。それは権威主義的な国家や言論統制を画策する多国間にわたる資本家連合が強大化する社会の内にいる個人の自由な言論ないしは意見の交換を重視し代替となるオンライン・プラットフォームへの社会全体での移住のようなものである。

3 ビットの圧縮アルゴリズムに関わるヘッダの Flag 情報の例を示す:

- 000: 非圧縮
- 001: Zstandard （SHIFT_JIS に対して学習ずみの辞書情報をもつ）
- 010: Shoco （英語に対して学習ずみ）
- 011: Shoco （スペイン語に対して学習ずみ）
- 100: Zstandard （数多くの Wikipedia の記事によって学習ずみの大規模な辞書情報をもつ）
- 101: Brotli （LZ77）
- 110: 未使用
- 111: 未使用

4 終わりに

言論の自由は横暴な政治家たちの立法による言論統制の脅威にさらされている。そうした政治家たちというのは、特別利益団体や国家主体の団体でエージェントをして自由な言論への組織的な活動家たちやそれに関わる人々の運動に対して破壊活動を行わしめるものたちのことだ。また多国間にわたる大規模な資本家連合で彼らのプラットフォームにおいてその幹部陣の根深い政治的な偏見の現れとしての制限的な協定や条件を以って言論を統制するのもこれに加担しているものだ。そうしたなかで、ブロックチェーン上における情報の書き込みを通してあらゆる思考や意見、考えの自由な交流を実現することによって、SUMO プロトコルは言論の自由を守り、さらなる個人の自由を支援する多様な取り組みを擁せんとするのである。