Storj
ピアツーピアクラウドストレージネットワーク

Shawn Wilkinson (shawn@storj.io)

貢献者:
Tome Boshevski (tome@storj.io), Josh Brandoff (josh@storj.io),
and Vitalik Buterin (v@buterin.com)

December 15, 2014
v1.01

概要

エンドツーエンド暗号化を実施するピア・ツー・ピア・クラウド・ストレージ・ネットワークは、ユーザが第三者データプロバイダに依存せずにデータを転送し、共有することを可能にする。中央のコントロールの除去は、セキュリティ、プライバシー、およびデータコントロールの大幅な増加だけでなく、最も伝統的なデータ障害や停止を排除するであろう。ピア・ツー・ピア・ネットワークと基本的な暗号化は、ほとんどの問題のためのソリューションとして働くが、ユーザーが適切にこのネットワークに参加するために、我々は適切なインセンティブを提供せねばならない。我々は、チャレンジアルゴリズムを使用して、これらの付加的な問題に対する解決策を提案する。こうすれば我々は定期的に、暗号ファイルの整合性と可用性を確認ができ、ファイルを維持するものに直接報酬を提供できる。ピア・ツー・ピアネットワークの不在下では、記載された方法は、プロバイダがデータへ直接アクセスすることなく、サードパーティのデータプロバイダのデータをユーザーが制御・移行・検証することができる。

1 はじめに

インターネット上のクラウドストレージは、転送したデータを格納するための信頼できる第三者としてのデータプロバイダにほぼ独占的に依存するようになっている。システムは、ほとんどの場合に十分機能するが、それはまだ信頼に基づいたモデルの固有の弱点に悩まされる。エンドツーエンドの暗号化は、非標準であるため、従来のクラウドは、敏感でプライベートな消費者や企業のデータを公開する人が介在するモデルの攻撃、マルウェア、およびアプリケーションによるハックを含む、多様なセキュリティ上の脅威に開放されている。さらに、現在のクラウドストレージアプリケーションは、ユーザーが選択できる相互運用可能で機能豊富なプロバイダがほとんどないので、データ保存に自社のコアコストより大きなプレミアムを請求することができる。さらに、これらのサードパーティプロバイダは、多くのデータ侵害や使用不能、それにもましてそれらに依存するユーザやアプリケーションに苦痛を引き起こす可能性がある技術的な障害を持っていることがある。これらの欠点は、以前 MetaDisk [1] のホワイトペーパーで詳しく説明した。

必要なのは、分散型でオープンなネットワーク上のエンド・ツー・エンドの暗号化を実装した分散型クラウ

ドストレージプラットフォームである。このプラットフォームは、シビル攻撃や他の詐欺の形態を試みる可能性に対し、攻撃者に抵抗がなければならない。さらに、このネットワークは、平均的なユーザーのコンピュータや専用ハードウェアのレイテンシ、パフォーマンス、およびダウンタイムを考慮する必要がある。提案するネットワークでは、暗号化アルゴリズムは、ユーザによって制御されていないデバイスでの輸送及びその上にあるとき、データを保護する。オープンマーケットは、すべてのデータが同じように取引され、ネットワークの周りに移動することができるようにすることで、大きなプレミアムを排除するであろう。ほとんどのインターネットに接続されたコンピュータは、未使用のハードドライブの空き容量を持っているため、ユーザーはネットワークにこれらのリソースを売ることができる。このネットワーク上のファイルはハッシュを通して、その内容に応じて処理される。ネットワーク上のデータは検閲、改ざん、および/またはデータ障害にかなり耐性があるであろう。ファイルのコピーがある限り、オリジナルがオンラインにあるかどうかにかかわらず、アクセスすることができる。

2 暗号化された塊としてのファイル

私たちは、断片（シャード）を、ネットワークに保存したいファイルの暗号化された 1 部分と定義する。保存されたファイルが標準化されたシャードサイズである限り、どのファーマーも完全なコピーを持たないので、ファイルをシャードに分割することでよりよりデータセキュリティを可能にする。私たちは、ネットワークに彼または彼女のハードドライブのスペースをリースしているユーザーをファーマーと定義する。私たちは、8 メガバイトまたは 32 MB のようなバイトの倍数を、標準化されたシャードサイズと定義する。これらは、何が格納されているかを判断しようとする試みを思いとどまらせるために、予め設定されたサイズで保存される。シャード化（断片化）はまた、ネットワーク全体に分散されるので、大きなファイルをより管理しやすくすることが可能である。

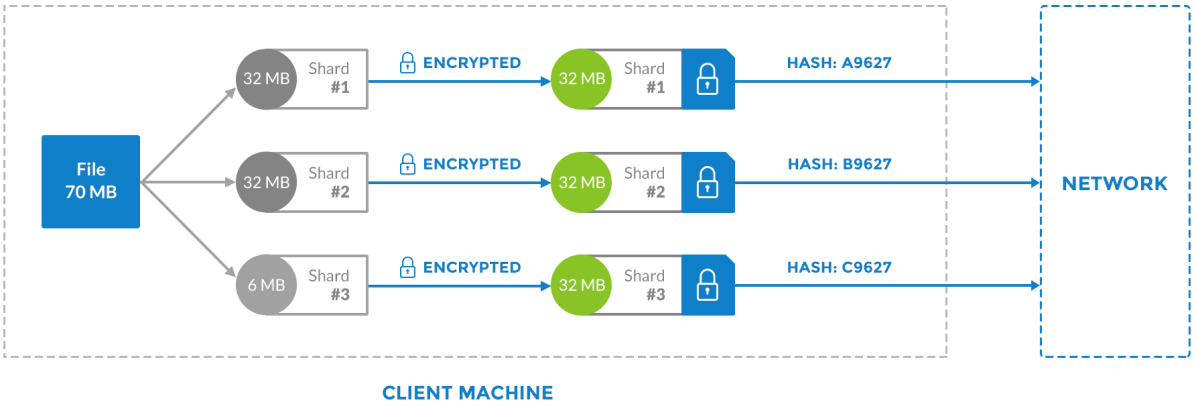

図 *1:* シャード化のプロセスの図式化

1. ファイルは、32 メガバイトのようないくつかの標準的なバイトで複数に分割され、または複数の小さ

なファイルは破片を形成するために結合される。余分なスペースはゼロが満たされている。
2. 各シャードは収束暗号化や、外部の暗号化キー等のユーザー定義の方法で暗号化される。
3. シャードは、ネットワークに直接送信されてもよい。破片は、さらに、ファイルの監査メカニズムに応じて、変更してもよい。

必要に応じて、複数のシャードを一緒に組み合わせることができる。これは、二つの場合に便利である。クライアントが特に敏感か重要なデータを保存する場合、最初に、それが自分のファイルを偽装するためには、他のクライアントデータやゴミデータとのシャードを組み合わせることが有用であろう。さらにいえば、我々はファイルに検証と監査をするが、これは小さな文書を確認するには不要なオーバーヘッドが発生するので、一緒に多数の小さなファイルを結合し、それらをすべて一度に検証することができるのは、価値があるかもしれない。

3 ストレージ証明

もう一つの懸念は、ネットワーク上に保存されたシャードの整合性と可用性である。ファーマーは、それがシャードを持ち、いかなる方法であっても、それを変更していないことを、暗号証明できなければならない。私たちは、以下に述べる方法を介してこれを実現する。

3.1 ストレージ証明対マークル監査

信用不要のデータストレージ・ネットワークを実現するために、我々は、クライアントが彼または彼女がネットワーク上に保存されているデータが利用可能で変更されていないことを監査するための方法を提供しなければならない。私たちは、マークル木 [2] とマークル証明を使用して、これを行う。我々は、データの集合を取得し、図 2 に示すように、マークル木を生成する。

ツリーの葉は 256 バイトのシャードまたはそれより小さくする必要がある。ツリーは格納されるのでなくオンザフライで生成する必要があるので、理想的にはツリーはデータよりも大きくする必要がある。遠隔地でのデータの監査は、単に特定のインデックスとそのインデックスでのサブシャードおよび Merkle 木 SPV 証明からなる応答で構成される。このアルゴリズムは、さらに、Secret Sharing and Erasure Coding[3] に記載されている。

3.2 ストレージ証明対生成済み監査

より多くのオーバーヘッドが必要だが、以前の方法に比べていくつかの利点を有し得る監査のための代替の方法を提示する。ファイルに追加することができ、（決定論的ルートシードから）ハッシュ化されることができるシードのシリーズをクライアントが作成し、一意のハッシュの回答を生成する、ハッシュチャレンジを通じてこれを行う。私たちは、ハートビートとして、このプロセスを参照している。クライアントは、[2] マークルツリーを構築し、これらのハッシュチャレンジを生成し、Satoshistyle の blockchain にマークルルートを挿入する。その後、ファーマーに、葉を覗いたマークル木を与える、または公開する。次に、クライアントは、定期的にそのデータをホストしているファーマーにシードを発行し、ファーマーが応答したハッシュが

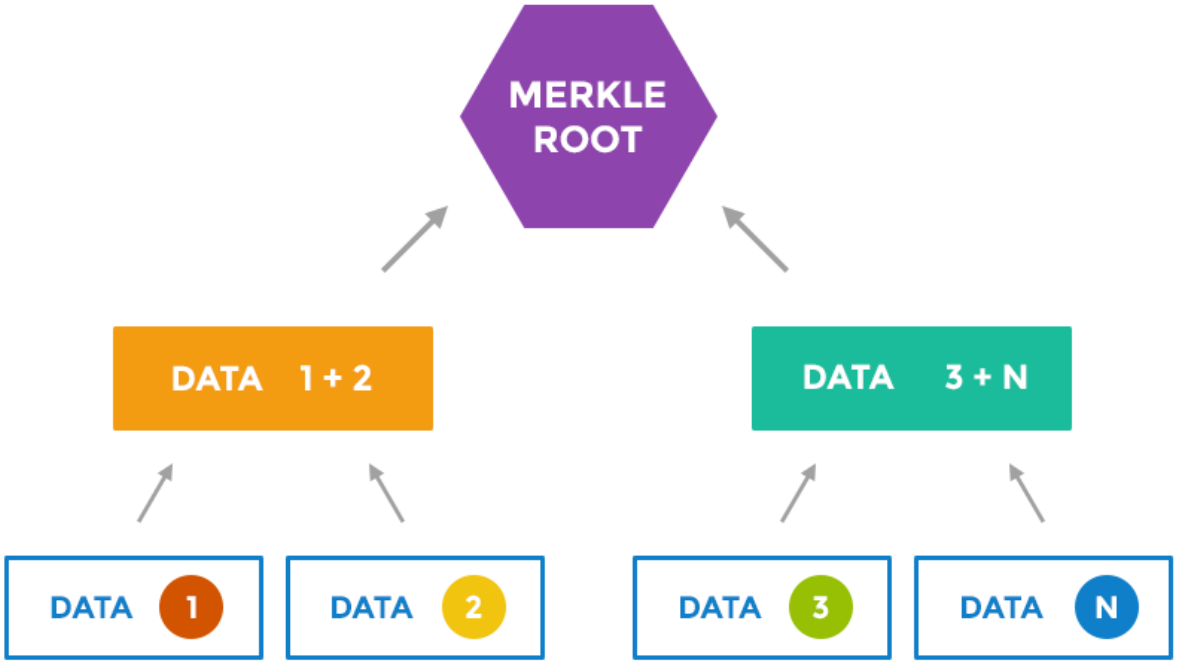

図 2: データマークルツリーの生成

マークルツリーにあることを確認することによって、ファーマーの応答がその生成されたハッシュの答えと一致するかどうかを確認することができる。

彼または彼女は、セクション 8 にさらに記載されたハッシュチャレンジを失敗するため、ファーマーがファイルを変更または削除することはできない。暗号とハッシュの根拠があるので、ハートビートがブルートフォースされることはできない。ハッシュチャレンジが blockchain に挿入されたマークルルートを経由して検証することができるため、クライアントは、ファーマーをごまかすことはできない。このように、どの当事者が本物かを維持するために blockchain を元にしたバックアップ証明の存在 [4] [5] を使用する。

3 つの異なる機構を使用してチャレンジを生成する。

3.3 完全ハートビート

私たちは、ハッシュ応答を生成するため、シードとフルシャードを使用することができる。これは完全なシャードの整合性を検証するが、非常に I/ O 非効率的であり、かつ長いシーク時間につながる。

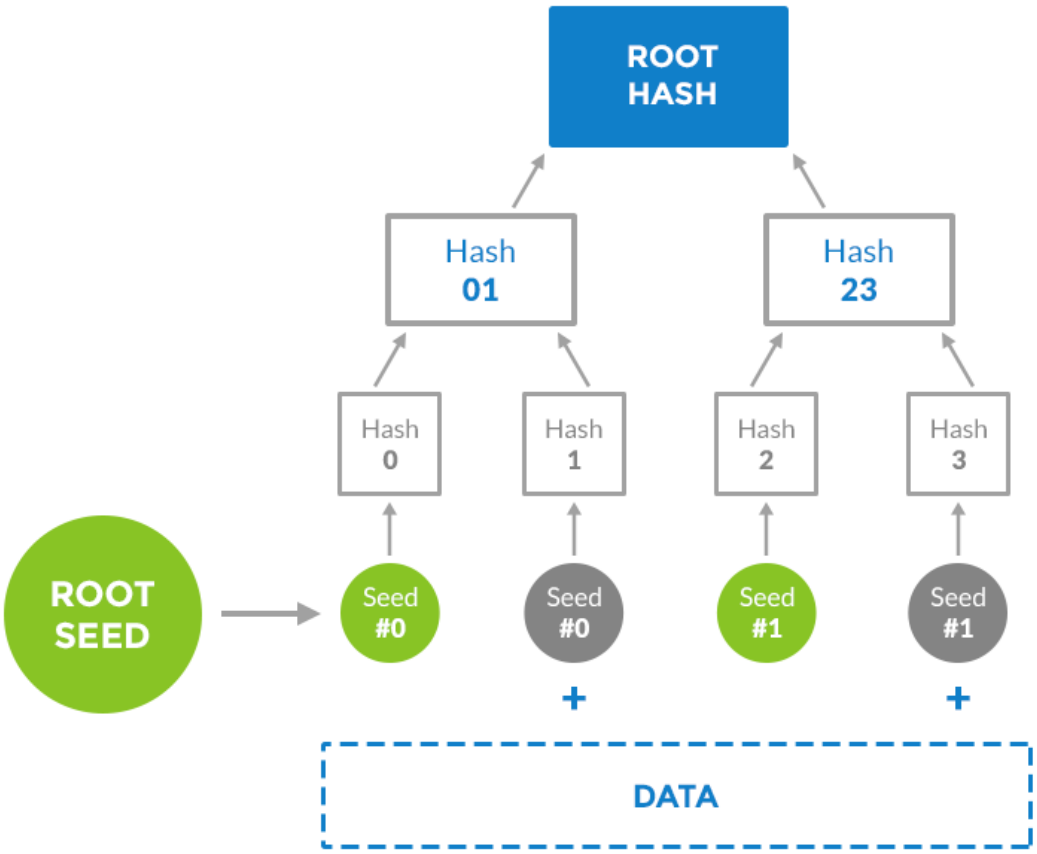

図 *3:* 何かのデータのマークルルート生成

3.4 サイクルハートビート

小さな改善として、シャードが、N 個の別のピースに分割され、サイクルを完了するまで、順番に各部分を確認する、というのもある。この方法は、Filecoin の一部 [6] で説明された。これは、より I/O-効率的であり、私たちは n 個のハートビート後にシャードの整合性を完全に確認することができる。残念ながら、それは、潜在的な攻撃にオープンとなる、というのは攻撃者は依然として $1/n$ のデータの整合性にのみハートビートを完了させればよいからである。

3.5 決定的ハートビート

この方法では、クライアントから渡されたルートシードを使用して決定論的にシャードのシードを生成する。フェイステル順列を使用すると、我々は、高々 $n + 2\sqrt{N}$ 回挑戦した後にデータを確認することができる。シャードへの軽微な変更がハートビートによって検出される、またはシャードの取得の際に修復されるよう

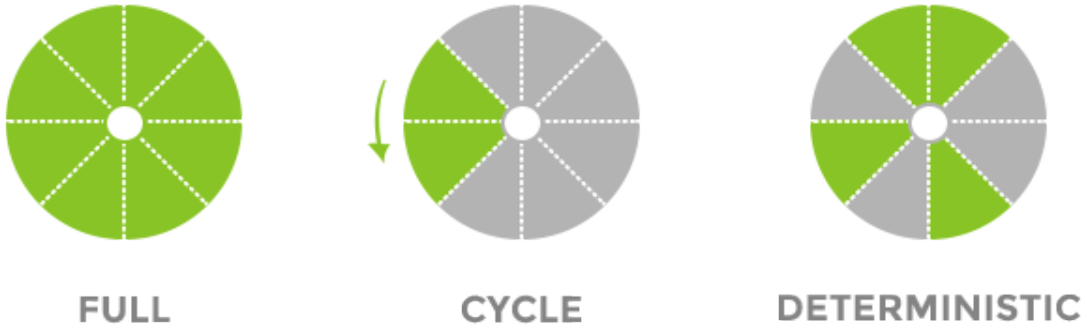

図 *4:* ハッシュチャレンジ生成の 3 つの方法

に、ハートビート・スキームに erasure encoding を追加する。これは、最も効率的な方法であり、パラメータを調整することによって、I/ O 効率のバランスをとりつつ、完全性検証を行うことができる。

3.6 ミックスした方法

各メソッドには、独自の長所と短所がある。我々は、これらの方法の組み合わせを使用することを提案する。ファイルが最初にネットワークにアップロードされたときにフルハートビートを使用することを勧める。サイクルハートビートは定義されたタイムスケール上で定期的に使用されるべきである。決定論的ハートビートは、他のすべてのチャレンジのために使用されるべきである。このようにすれば、我々は各アルゴリズムによって提供されるすべてのメリットを享受することができる。

4 冗長証明

伝統的なクラウドストレージ会社は顧客ファイルを格納するのに、サーバを所有またはリースする。会社は、物理的またはネットワーク障害からファイルを保護するための RAID 方式または複数のデータセンター方式を使用する。 Storj は中央のサービスを提供しない。ファイルは、分散、仮想でネットワーク内に存在する。私たちは、ファーマーが伝統的なクラウドストレージ会社のようにデータ損失に対して同じ安全対策を採用することを、あてにできない。このため、我々は、複数のファーマーと K-of-M erasure encoding 方式を使用してシャードを格納することにより、冗長性を保証する。より具体的には、我々は、物理層でなくネットワーク層の冗長性を考慮する。我々はまた、ファーマーは単に自分のコンピュータをオフにする可能性ゆえにネットワークからのシャードが除去される可能性に対する解決策を提供する。

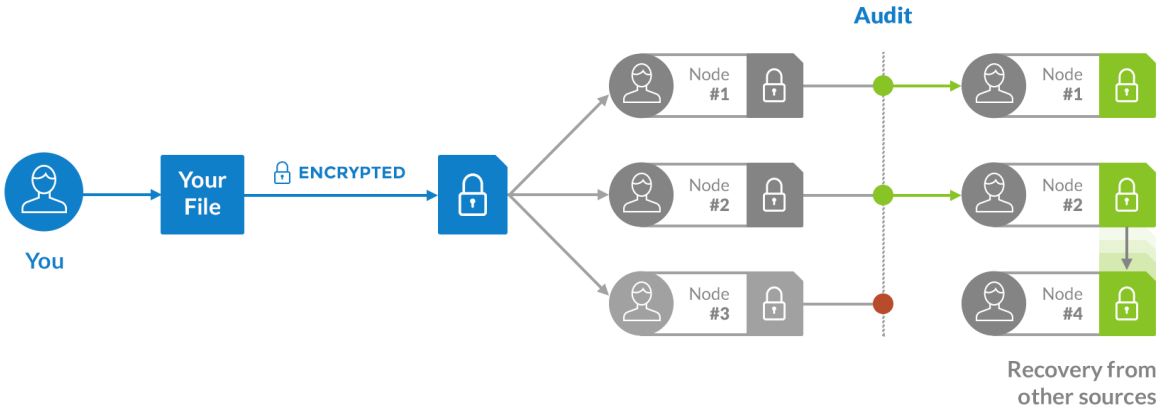

図 5: 監査の視覚化

4.1 簡単なフォールトトレラント

ノードが監査を失敗するか、または到達不能である場合、我々は、ネットワーク上の既存のコピーの 1 つを取り、新しいノードに転送することにより、ネットワークの複製プロセスを開始する。したがって、ネットワークは、各監査後に自己回復することができる。

4.2 シビル冗長攻撃

各シャードは一意に暗号化されている。これは悪質なファーマーが、ただ 1 つしか持っていないのに、ファイルの複数の冗長コピーを持っているふりができないことを意味する。私たちは、収束的にシャードを暗号化する前に、決定論的なソルトの値を追加することでこれを達成することができる。復号鍵が、特定のファイルで知られている場合であっても、悪質なファーマーは、割り当てられていない破片の監査を完了することはできない。各冗長コピーが一意であるため、この方法で、特定のシャードの冗長性を証明することができる。

4.3 K-of-M Erasure Encoding（消失訂正符号、訳注：消失したビット列を復元するための技術 http://www.jdsf.gr.jp/backup/stm/201007.html ）

私たちは、シャードが利用可能であることを確認できるように K-of-M erasure encoding 方式を使用する。クライアントは、ファイルの堅牢性とコストのバランスを達成するために、K と M を選択することができる。計算セクションは、より統計的にファイル堅牢性について説明する。K-of-M erasure encoding も hostage bytes と攻撃の項に記載される改変バイトに対する保護を提供する。

4.4 冗長性のスケーリング

最終的には、ユーザーとアプリケーションは、K-of-M erasure encoding のパラメータおよび配布を通じて冗長性を制御する。単純なデータストレージの場合、ユーザーは自分のデータが均等に 3-4 のファーマー間で分配される推奨設定を選択するかもしれない。これは、基本的なフォルトトレラントに十分であるろう。データが特に重要である場合、ユーザーはハルマゲドンと神の行為に対して、そのデータを保護する必要のある、500 ファーマーにデータを配信してもよい。 K-of-M 方式は、データの堅牢性に影響を与える。

5 ブロックチェーン

ネットワークが、ファイルの場所と完全性について合意に達成することができるようにするには、我々は satoshi スタイルのブロックチェーン [7] を使用する。 blockchain はパブリックな元帳なので、正確な情報を取得するための優れたツールであり、究極のメカニズムとして紛争を解決し、攻撃を断念させる。基本的な類推としては、人里離れた地域にクッキージャーからクッキーを盗むのは簡単だが、それが瓶ではなく何千人もの人々によって観察され、公共広場の真ん中に位置している場合、実行するのは難しい。分散データストアとして blockchain を使用することにより、確立され、長時間の試練を経た分散型のコンセンサスメカニズムを構築し直すことができる。私たちは、blockchain 内に任意のファイルではなく、ファイルのメタデータを保存する。本質的には、我々はファイルハッシュ、シャードのコピーのネットワーク上の場所、およびマークルルートを格納する。この方法の詳細と拡張性は MetaDisk の論文 [1] に記載されている 。

データは、余分なメタデータを持つ標準のトランザクションを経由して blockchain に挿入される。これは、ビットコインの blockchain 上ではひどく高価で、トランサクションの厳しいメタデータの制限により技術的に実現不可能なので、このメタデータを格納するためにそれを使用しない。MetaDisk の論文 [1] で説明したように、私たちは初期メカニズムとして Florincoin [8] を使用する。最終的に、存在証明を通じて、よりスケーラブルな方法で、より直接的にビットコインの blockchain を使用することができるシステムに移行する [4] [5] [9]。 blockchain 技術が進めば、より高速なスループットを提供するために、Factom[9] のや、データストレージに強制力のあるコントラクトを作成するために Ethereum[10] のようなシステムを使用することができるであろう。

6 スピード

増加する冗長性は Storj ネットワーク上のいくつかのユニークな機能を可能にする。 Storj は集中型ではない分散型のネットワークであるため、シャードをホストするネットワークに新しいファーマ追加時に、また、シャードをダウンロードするために接続可能な別のピアを追加する。20 倍の冗長性を設定すると、同時にからシャードをダウンロードするための 20 のピアを持っている。これは、最も近いデータセンターへの高速接続に依存するサーバ・クライアントモデルとは対照的である。

ピア・ツー・ピア・ネットワークの標準機能と同様に、高い転送速度を達成するために地理的に近いピアに接続することができる。暗号通貨を使用して、ネットワークへのノードの接続性を維持するためのインセン

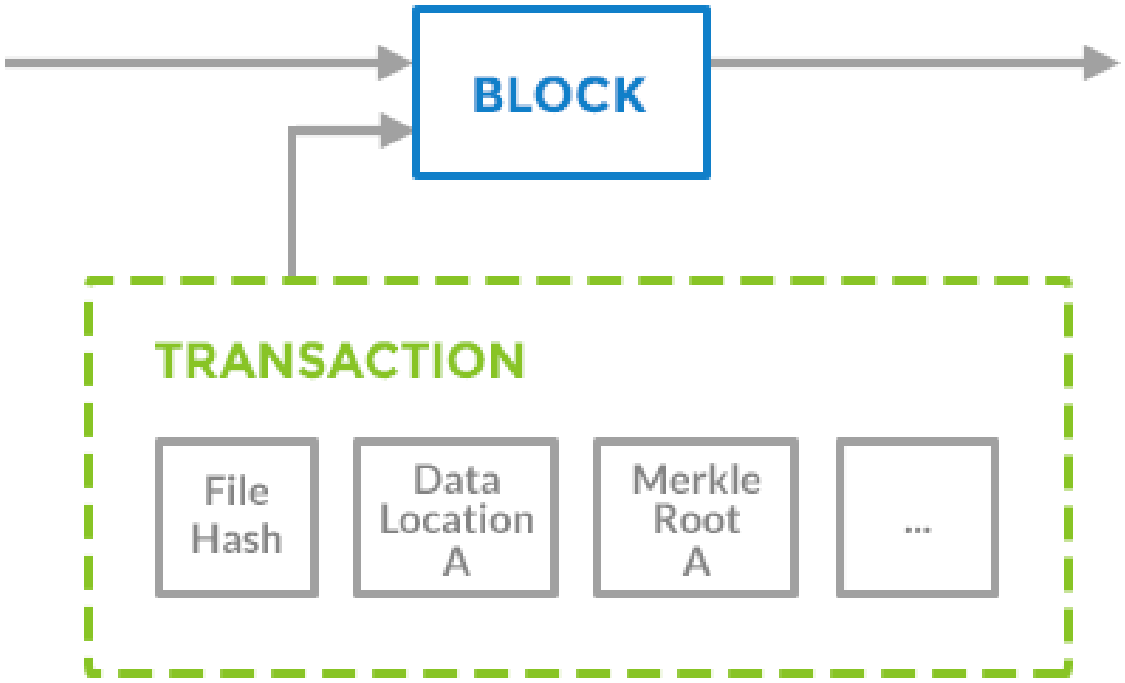

図 *6:* ロックチェーンへのメタデータ挿入

ティブを提供することができ、それで、高速な転送のために不十分なピアが存在するというピアツーピアネットワークで観察された典型的な問題を解決する。本論文で議論されていないが、Storj がコンテンツとデータ配信方式として使用される可能性がある。

7　報酬

ネットワークが正しく機能するために、我々は正しくかつ同意した行動に報酬を与えねばならない。私たちは、Storj ネットワークのベースラインのトークンとして、暗号通貨 Storjcoin X、または SJCX を使用する。クライアントとファーマーは、ネットワーク上の帯域幅及びストレージスペースと、SJCX を交換することができる。

私たちは、帯域幅やストレージスペースの支払いとして、ノード間の暗号通貨の瞬間移動を容易にするために、マイクロペイメント/マイクロトランザクション [11] [12] を使用することができる。クライアントは、ハートビートが完了した後にファーマーに支払う。両方が正直に振る舞うい、合意している場合は、それ以上の手順は必要ない。それ以外の場合は、トランザクションが終了し、合意された最後の量が最終とみなされ

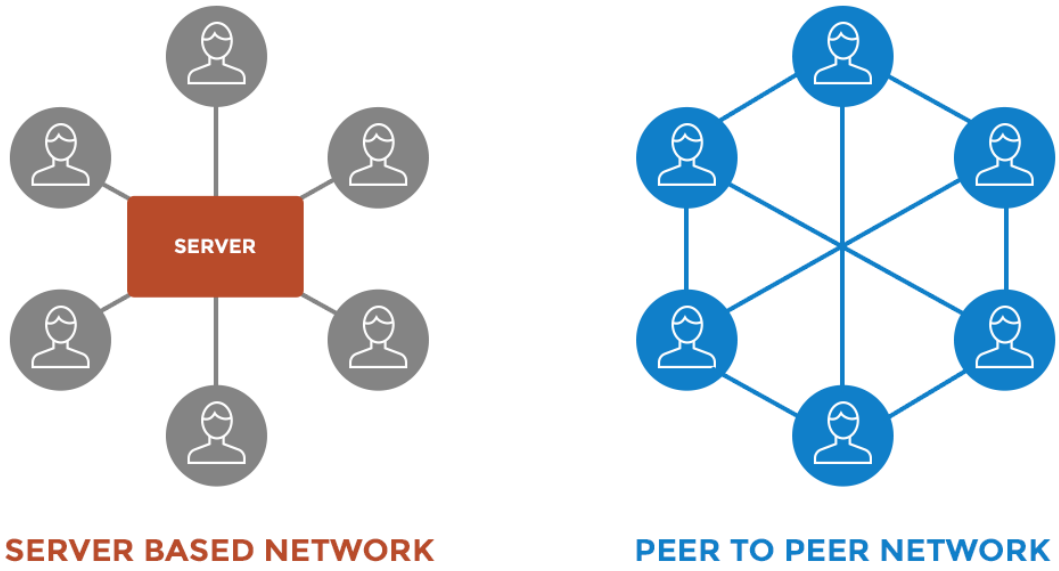

図 7: サーバーベースネットワークのネットワークトポロジ対ピアツーピアネットワーク

る。代わりにノード間の紛争を実行するのに信頼不要な自動化メカニズムを使用できる。両当事者が財政的に合理的に行動している限り、潜在的な損失は１セントの一部の量になるので、これらの方法は必要ではないかもしれない。攻撃者が大きな財政的費用でのネットワーク攻撃のみに関心ある、標的型攻撃の場合には、さらなるメカニズムがネットワーク上の新しいノードを保護するために必要とされるであろう。

7.1 市場の基礎

価格設定を固定実装する従来のクラウドサービスと異なり、Storj ネットワークの価格設定は市場で決定される。データ・リソースの効率的な配分が有効になっており、自由市場によりインセンティブされる。ファーマは売値を出（ask）し、クライアントは買入れ（bid）できる。価格は、帯域幅、位置と速度に基づいて変更することができる。例えば、SSD ドライブを搭載した高速なサーバが標準的な家庭のラップトップよりも多くをチャージすることがある。この方法では、データソースの効率的な使用を達成することができる。ボブがストリーミング用ネットワーク上のビデオファイルを保存したい場合がある。この場合、ボブは、高速 SSD サーバーを使用する場合がある。アリスは、バックアップの目的で Storj ネットワークを使用している場合がある。この場合、標準的な家庭のラップトップは、安く、より効率的なオプションとして役立つであろう。

7.2 ピア発見

任意のソース、追跡サービスや、ネットワークから直接に、ファーマーの品質とサービスのタイプに使用可能なデータを取得できる。これらは単にピア発見での広告である。このようにして、我々は、擬似評判システ

ムを使用する。ボブとアリスがイブが好評であると言うならば、イブはおそらく好評である。イブは稼働時間や速度について嘘をついているかもしれないが、私たちの監査メカニズムは、これを検出し、ピアとしてイブをドロップする。このように、我々はボブまたはアリスの勧告に依存しないが、確かに良い仲間を見つける助けになる。

7.3 市場のスケール

市場はまた、ネットワーク上のストレージ・スペースの需要と供給のバランスを助ける。ストレージ・スペースのネットワーク上の需要の増加がある場合、SJCX トークンの価値は増し、ファーマーが需要を満たすために、ネットワークに複数の記憶領域を追加するためのインセンティブを与える。同様に、需要が減少すると、ストレージの価格は減少し、ネットワーク上のストレージの成長を鈍化させる。ネットワークは、十分なネットワークリソースを確保するために、市場の力に依存している。

7.4 ピアの報酬

伝統的なピアツーピアネットワーク内に存在する問題の一つは、ピア不足である。ピアは、単に非常に人気があるファイルのために非常に高速な転送速度を可能にするのに役立つボランティアである。同様に、あまり引っ張りだこでないファイルを共有するピアの欠如は、多くの場合、苦労しても遅い転送速度または、実質的に存在しない可用性につながる。 SJCX に基づく固有の報酬メカニズムは、破片を保持するためのピアを支払うことによって、その問題を解決する。マイクロペイメント/マイクロトランザクションを使用して、また、より直接的に、シャードを要求するノードがシャードを転送するピアに支払うことができる。このようにして、より多くの SJCX を稼ぐために、ピアは自主的に人気のある破片をホストできる。一言で言えば、このシステムは、バンド幅証明アルゴリズムの技術的なオーバーヘッドに頼らず信頼不要だがインセンティブ化された転送を可能にする。しかし、TorPath 上の最近の論文 [13] はそれが可能かもしれない方法について、いくらか光を当てている。

7.5 稼働時間

ファーマーは、高い稼働率を持っていることが奨励される。これは非常に専用のファーム用ハードウェアにとっては簡単だが、平均的な消費者には難しいかもしれない。この幅広いバリエーションの状況のため、我々は代わりにクライアントとファーマーの間でデータコントラクトを設定する。それで、それぞれがアップタイムの期待値を設定できる。ラップトップ上でファーミングソフトウェアを実行しているボブは、コンピュータを定期的にダウンすることを可能にするデータコントラクトを受け入れられる。しかし、ボブは、ハードウェアを専用にし、常にそれをオンにしているアリスよりもはるかに低いレートを受け取ることになる。シンプルな稼働時間監視サービスは、クライアントがファーマーの稼働時間を発見するのに役立つ。我々は、これらのアップタイムサービスに本質的には依存していない。我々のハートビートアルゴリズムは、より直接的に破片が利用可能であることを保証する一方で、それらは単に、信頼できるピア発見を支援するために使用される。

8　シビル及び悪人による攻撃

私たちが提案したネットワークに対処しなければならない伝統的な攻撃の多くの種類がある。多くの攻撃タイプとその解決策をリストする。

8.1　「グーグル攻撃」

「グーグル攻撃」は、電源/ストレージスペースを計算する膨大な量を制御する大企業またはエンティティによるネットワーク上の協調攻撃を説明するために、ビットコインのコア開発者ピーター・トッドによって名付けられたコイン用語である。我々の研究は、Google が現在、約 8000PB に匹敵するデータを保存していることが示されている [14]。私たちは、ユーザ空間、または平均消費者のコンピュータ上の未使用の空き領域の集合的な概念を導入する。ユーザーが自分の主要なクラウド・ストレージ・プラットフォームとして Storj を使用するようになった場合は、このユーザー・スペースが拡大する可能性はある。我々の研究は、約 250,000 PB [14] であることがあることを示している。したがって、Google は、ネットワークを攻撃するためそのサービスのすべてを停止しても、ユーザ空間によって提供されるリソースを凌駕することはできないであろう。我々は、ユーザスペースの集合がより大きく、常に全体の集中型のクラウドコンピューティング産業のそれよりも大きくなると主張する。

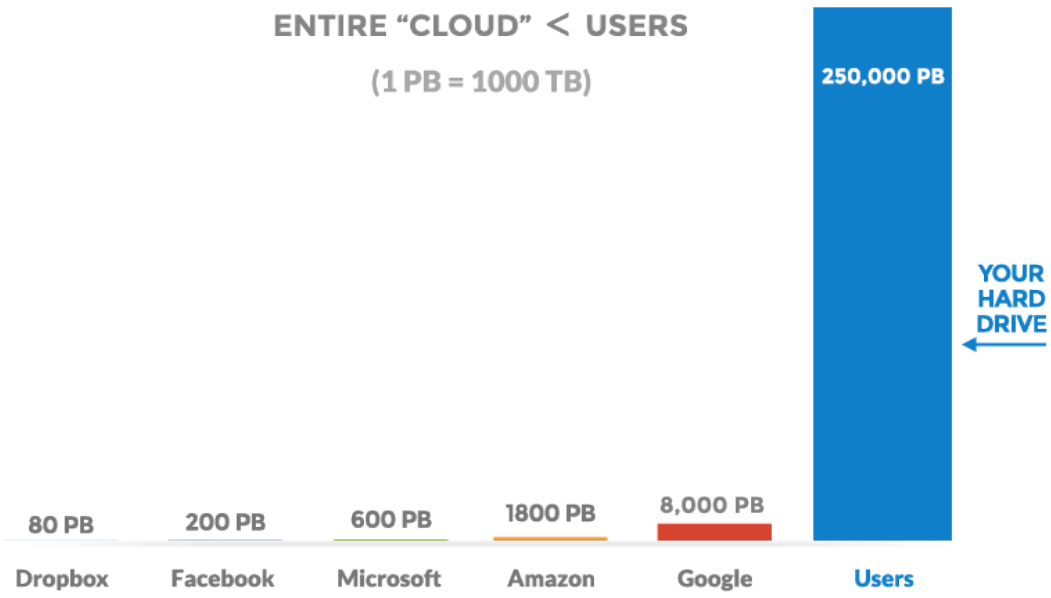

図 *8:* クラウドストレージ対ユーザスペースの可視化

8.2 シビル冗長攻撃

彼/彼女が唯 1 個しか持っていないのに、彼/彼女はシャードの複数の冗長コピーを持っていることをふりをすることで、ネットワークを搾り取りたい攻撃者は、そのようには行うことはできない。各シャードは一意に暗号化されるためである。彼らはユニークな冗長コピーのコピーを持っていない限り、攻撃者が監査を渡すことはできない。クライアントノードは、地理的に分散し、独立したノードにその冗長コピーを配布するための措置をとる必要がある。もしランダムに配布した場合、同一の冗長部分が同じ制御ノード上でホストされている確率は統計的に非常に小さい。ランダム分布と一意に冗長コピーを暗号化する私達の標準的な方法を用いれば、攻撃者は、シビル冗長攻撃を実行することはできない。

8.3 不適切な分配

このケースでは、我々はネットワークを攻撃したい悪意のあるファーマーを考える。この攻撃者は、金融的にインセンティブがないことを気にしない。攻撃者は、ネットワークからデータとその冗長コピーを同時にドロップし、ファイルを回復不能にすることによって、ネットワーク内に不信を引き起こすことを望む。一つ以上の悪質なファーマーと共謀することで、ファーマーはシャードの複数の冗長コピーを受信することを目指す。私たちの最初の防衛ラインは、ネットワークへ敏感なファイルをアップロードするのに単一ノードを使用せず、代わりに複数のノードを経由して破片を中継することである。このようにすれば、いずれかのノードに悪意がある場合であっても、完全にはこの攻撃を行うことができない。ノードが共謀しなければならなず、クライアントが、ランダムにノードを選択した場合には、共謀ノードを選択する確率は小さい。ファーマーが中継ノードの複数と共謀しているという最悪のシナリオの下では、防衛の第 2 ラインは、2 つの追加のメカニズムで構成される。まず、異なるパラメータで erasure encoding と冗長性方式を使用すると、ファーマーはシャードを破壊するのに、いくつのシャードが必要かを知ることができないことを保証する。さらに我々は、すべての検証が、セクション 10 に記載されている GVN で行わる方法をアドバイスすることができる。これにより、ファーマーが複数の冗長コピーを持っている場合、関連するシャードを決定する方法がない。

8.4 クライアントノードを騙す

悪質な人物が、ネットワーク上のファイルをホストしたいが、支払いを避けるために、ファイルの支払いを避け、または意図的に正しい監査を拒否するシナリオを想像しよう。この場合には、二つのノード間の不一致があり、、トランザクションが中止になる。blockchain に埋め込まれたマークルルートは、前述のように、クライアントからのチャレンジの有効性として十分に動かぬ証拠を提供する。

8.5 Hostage Byte(人質となったデータ)

hostage byte 攻撃は最初に Vitalik Buterin で記述され、悪意あるファーマーがクライアントにデータを転送するが、より高い支払のため最後のデータのピースを人質に保持する [3]。ファイルを再構築するために必要な破片が直線的でないように、私たちは、 erasure encoding でこれに対処する。クライアントが erasure encoding との境界を秘密に保つ限り、悪質なファーマーは最後のバイトがどれあるかを知ることができない。

8.6 正直ジェペットアタック

この攻撃ではファーマーは正直に長時間ストレージスペースを大量にアップしている。ある時点で、彼らは、ネットワークの大部分をダウンさせる試みのため糸を引っ張る。我々は 2 つの方法で、これを非常に高く付くようににすることができる。一つは、外注監査でさらに説明する GVN を使用することにより、ファーマーへの支払いを遅らせることができる。ファーマーは支払いを受け取る前に、一定量に対して信頼性を証明しているだろうが、突然悪意をもったり、適切なサービスを提供することができなくなった場合、保留中の資金が没収されることになる。二つ目は、結合したスマート契約は、クライアントとファーマーの間で作成することができるだろう。彼らは両方が従うべきサービスレベル契約（SLA）に同意するだろう、もしファーマーが契約から外れた場合、クライアントに支払われるかもしれない。代わりに資金が h 破棄されるかもしれない。

9 ネットワーク

ネットワークからとネットワークへのファイルのアップロードと取得のステップは次のとおりである：

1. ファイルをピースに分割し、erasure coding と暗号化を実施しシャードを生成する。
2. 分散と erasure encoding スキームに基づき、ネットワーク上の様々なファーマにシャードを転送する。
3. 監査完了に対し、ファーマに支払う。
4. クライアントはデータ取得に何人かのファーマにコンタクトする、その場合はファーマはデータの転送に対して支払われる。

10 アウトソース可能ハートビート

ビットコイン [7] のような従来の分散型ネットワークは、機能するために、コンセンサス・アルゴリズムに依存する。それらは、より直接的に Proof of work [7] などのアルゴリズムを使用し、計算能力がネットワークコンセンサスを決定する。残念ながら、これらの機構は、正直になるためにはネットワーク上のノードの大部分を必要とする。51% 攻撃と呼ばれる、大部分のノードが正直に振る舞わないことにより、Terracoin [15] や他の多くの小さい暗号通貨が破壊された。ビットコイン自体、潜在的に 51% 攻撃 [16] の問題がある。さらに、これらのアルゴリズムは、セキュリティ [17] を確保するために、計算能力を大量に無駄にするが、Peercoin[17] のような暗号通貨でいくらかの進歩がなされた。

ビットコインのような標準暗号通貨ネットワーク上の 51% の攻撃は、ネットワークの大部分にほとんど影響を及ぼさない。攻撃時に取引したユーザーだけが詐欺にあい、残りのネットワークは無事である。攻撃者がネットワークの破片の状態を変更し、データの損失を引き起こす可能性があるので、これはデータ・ベースのネットワークとは異なる。データによっては、そのネットワーク上の壊滅的な影響を与える可能性があり、そしておそらくその完全な故障につながる。このように、我々は、ネットワーク上に格納された破片の状態を決定するために、コンセンサスだけに頼ることはできない。

私たちは、その代わりに、ネットワーク内の最終的な意思決定者としてクライアントを指定する。結局のところ、それは、ネットワーク上のストレージスペースのために払っているクライアントである。クライアントは、最終的な意思決定者であっても、動かぬソースとして blockchain を使用して、ファーマーをごまかすことができない。監査アルゴリズムのおかげで、多くの不正や共謀のノードがネットワーク上にあるかは問題ではない。悪意のあるノードは、監査をパスすることはできず、ユニークで冗長なコピーで共謀ノードとシビル攻撃が不可能であることが保証される。

クライアントがオフラインのときに監査を処理する方法が問題として残る。私たちは、伝統的なクライアントがずっとオンライン時間であることを期待することはできないので、我々はハートビートを検証し、ファーマーに支払うために信頼不要な方法を考案する必要がある。私たちは、個々及び集合的で効果的な検証・支払い方法である、いくつかの方法を提案する。

10.1 クライアントコントロール

最も簡単な方法は、単純にクライアントの制御とアクセスをオンラインハードウェアへ与えることが含まれる。例えば、彼らはわずか月数ドルで基本的な VPS サーバーを購入することができる。コストを相殺するために、彼らが知っている他のユーザーや友人とそれを共有することができる。操作は非常に簡単である - サーバは定期的にファーマーに監査を渡し、所定の応答に対してそれを検証する必要がある。同時に、それはまた、ファーマーへの様々な支払いを扱うことができる。

10.2 認証ノードグループ

もう一つの方法は、ハートビートと支払いを管理する検証ノード（GVN）のグループに頼ることである。ユーザーは GVN に監査と支払いを渡す。 N of M の multisig 方式では、N が GVN 内のノードの数である N of N を用いる。このように、単一で結託していない正直なノードがあらゆる共謀ノードのための支払いを反証し、ブロックすることができる。クライアントの裁量で我々は $M < N$ スキームを使用することがある。SIA [18] は共謀ノードを取り出すために使用することができるいくつかの方法を説明する。クライアントが究極の意思決定者のままであるため、我々はさらに、トランザクションがクライアントに資金を返すよう細工されている必要がある。このトランザクションがブロードキャストされなければ、任意の時点でクライアントが未使用の資金を取得できる。私たちは、さらなる存在証明メソッドを追加することができ [4] [5] [9]、これらのノードグループが実行したすべてのアクションが blockchain を通じて公に監査可能である。

10.3 埋没キー

監査応答自体が資金へのプライベートやアクセスキーである可能性がある。応答の後、ファーマは、ネットワークから、または GVN から直接支払いのために自分の鍵を引き換えることができる。別の方法として、スマート契約によって管理される分散型決済サービスがそれ自身または GVN と強調して解である可能性がある。

10.4 塵と匿名化

GVN で何百万人の検証や支払いをファーマーへの単一の支払いに校合することができる。理想的には、GVN とファーマーはマイクロペイメントチャネルを確立し、ファーマーが検証ごとに瞬時に GVN を経由して支払われる。これは、何百万もの小さな塵のような取引を避けることを可能にする。さらに、GVN は、ユーザに匿名性を提供する、なぜなら、資金がその中に混合されるからである。したがって、ユーザーの資金とそれらのファイルとの間に直接または間接的なリンクがない。

10.5 GVN の拡張

Storj のアイデアを広げるなら、開発されうる、監査や信頼不要の支払いを扱うことができる他のいくつかの方法がある。これらには、Factom[9]、神託 [19]、スマート契約、チューリング完全 blockchains[22]、P2P ネットワーク、blockchains、定足数 [18]、およびスクラッチオフパズル [20] が含まれる。これらの方法のそれぞれは、ハートビートと支払いを処理する有用かつ信頼不要な方法である可能性がある。MetaDisk [1] は、GVN 内のノードの最初の実装として機能するが、これらの他の方法でも GVN のノードとして機能することができる。私達のコアの目標は、最終的な決定者としてのクライアントを持つことである。クライアントは、これらのタスクを処理するか、またはクライアントがそのデータを保護する最も快適と感じるアルゴリズムの上にフルパワーを持つことができる。

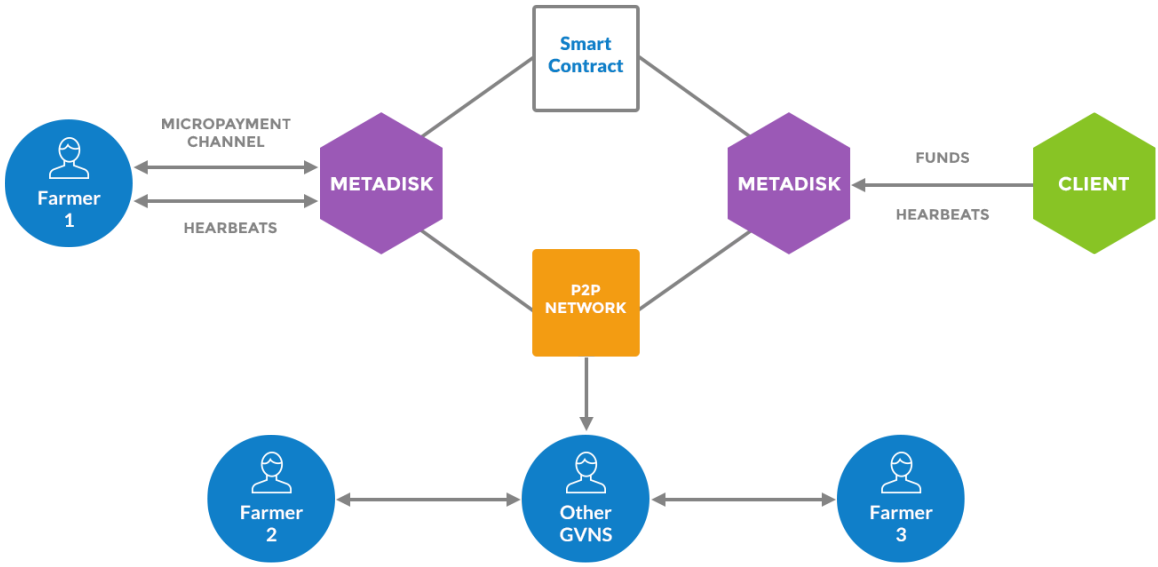

図 *9: GVN* 例の視覚化

11　計算

すべてのシャードがオンラインである確率 p を仮定して、k-of-n erasure coding が失敗するの失敗可能性は、次のように計算される：

$$\Pr_{failure}(n,k,p) = \sum_{i=0}^{k-1} p^k (1-p)^{n-k} \binom{n}{k}$$

Code:

```
def fac(n): return 1 if n==0 else n * fac(n-1)
def choose(n,k): return fac(n) / fac(k) / fac(n-k)
def prob(n,k,p): return choose(n,k) * p ** k * (1-p) ** (n-k)
def prob\_fail(n,k,p): return sum([prob(n, i, p) for i in range(0, k)])
```

したがって、erasure encoding をするのに十分なパラメータを使用して、かつ上述の修復方法を行えば、破片またはファイル損失の統計的可能性は極めて小さい。

12　結論

我々は、ユーザーがサードパーティのデータプロバイダに依存せずに転送してデータを共有することができ、エンドツーエンドの暗号化を実装するピア・ツー・ピアのクラウドストレージ・ネットワークを概説した。私たちは、独自に暗号化された冗長コピーと、定期的にファイルの可用性と整合性をチェックする監査アルゴリズムを使用することにより、シビル攻撃やピアが接続を切断するなどの問題について対処した。 暗号通貨を使用することにより、我々は、ネットワークの成長と、クライアントとファーマーの間でデータの取引のための適切なインセンティブを提供することができる。私たちは、自分のファイルの検証と支払いを処理するアルゴリズムを介してクライアントにフルパワーを与える。これらの方法を用いて、ユーザに、クラウドデータの制御を戻す。

また、ピア・ツー・ピア・ネットワークの非存在下でも、記載された方法は、ユーザが、サードパーティのデータプロバイダで、自分のデータを制御、移行および検証に使用できると仮定している。本質的には、信頼できるデータプロバイダに信頼不要で耐故障なシステムを実行すると、大幅にセキュリティとプライバシーを向上させる。

n	k	p	$\Pr_{failure}\, n, k, p$
9	3	0.5	8.984e-02
9	3	0.75	1.342e-03
9	3	0.9	2.998e-06
9	3	0.98	4.448e-11
9	2	0.5	1.953e-02
9	2	0.75	1.068e-04
9	2	0.9	8.200e-08
9	2	0.98	2.263e-13
18	6	0.5	4.812e-02
18	6	0.75	3.424e-05
18	6	0.9	5.266e-10
18	6	0.98	6.391e-19
18	4	0.5	3.768e-03
18	4	0.75	3.414e-07
18	4	0.9	6.074e-13
18	4	0.98	2.526e-23
36	12	0.5	1.440e-02
36	12	0.75	2.615e-08
36	12	0.9	1.977e-17
36	12	0.98	1.628e-34
36	8	0.5	1.562e-04
36	8	0.75	4.187e-12
36	8	0.9	4.098e-23
36	8	0.98	3.909e-43
72	24	0.5	1.471e-03
72	24	0.75	1.937e-14
72	24	0.9	3.636e-32
72	24	0.98	1.390e-65
72	16	0.5	3.269e-07
72	16	0.75	8.130e-22
72	16	0.9	2.449e-43
72	16	0.98	1.236e-82

参考文献

[1] S. Wilkinson, J. Lowry. Metadisk: Blockchain-based decentralized file storage application, (2014). `http://metadisk.org/metadisk.pdf`.

[2] R.C. Merkle. Protocols for public key cryptosystems, (April 1980). lIn Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133.

[3] V. Buterin. Secret sharing and erasure coding: A guide for the aspiring dropbox decentralizer, (2014). https://blog.ethereum.org/2014/08/16/secret-sharing-erasure-coding-guide-aspiring-dropbox-decentr

[4] D. Cawrey. How bitcoin ' s technology could revolutionize intellectual property rights, (2014). http://www.coindesk.com/how-block-chain-technology-is-working-to-transform-intellectual-property/

[5] M. Araoz. What is Proof of existence?, (2014). http://www.proofofexistence.com/about.

[6] Filecoin: A cryptocurrency operated file storage network, (2014). http://filecoin.io/filecoin.pdf.

[7] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, (2009). https://bitcoin.org/bitcoin.pdf.

[8] Florincoin, (2014). http://florincoin.org/florincoin.pdf.

[9] Factom, (2014). https://github.com/FactomProject/FactomDocs/raw/master/Factom_Whitepaper.pdf.

[10] Ethereum: A next-generation smart contract and decentralized application platform, (2014). https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper.

[11] Bitcoin micropayments, a new enabling technology, (2014). http://bitcoinmagazine.com/12654/bitcoin-micropayments-new-enabling-technology/.

[12] Bitcoin client bitcoinj implements bitcoin micropayments, (2013). http://www.coindesk.com/bitcoin-client-bitcoinj-implements-bitcoin-micropayments/.

[13] A torpath to torcoin: Proof-of-bandwidth altcoins for compensating relays. https://www.petsymposium.org/2014/papers/Ghosh.pdf.

[14] How big is the cloud?, (2014). http://blog.storj.io/post/95376799893/how-big-is-the-cloud/.

[15] Terracoin attack analysis ' , (2014). https://forum.feathercoin.com/index.php?/topic/5796-terracoin-attack-analysis/.

[16] After reaching 51% network power, bitcoin mining pool says ' trust us ' , (2014). http://arstechnica.com/security/2014/06/after-reaching-51-network-power-bitcoin-mining-pool-says

[17] S. King. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, (2014). http://www.peercoin.net/assets/paper/peercoin-paper.pdf.

[18] D. Vorick, L. Champine. Sia: Decentralized, compensated, self-repairing computer storage, (2014). http://www.siacoin.com/whitepaper.pdf.

[19] Orisi, the distributed oracles system for cryptocurrency contracts, (2014). https://github.com/orisi/wiki/wiki/Orisi-White-Paper.

[20] A. Miller, A. Juels, E. Shi, B. Parno, J. Katz. Permacoin: Repurposing bitcoin work for data preservation, (2014). https://www.cs.umd.edu/~elaine/docs/permacoin.pdf.