

Blockchain Applications

現在讓我們通過將其視為應用程序平台來建立我們對比特幣的理解。如今，許多人使用術語“區塊鏈”來指代任何共享比特幣設計原則的應用程序平台。該術語經常被誤用，並應用於許多無法提供比特幣區塊鏈所提供的主要功能的事物。

在本章中，我們將介紹比特幣區塊鏈作為應用程序平台提供的功能。我們將考慮應用程序構建 *primitives*，它構成任何區塊鏈應用程序的構建塊。我們將研究使用這些 *primitives* 的幾個重要應用程序，例如彩色硬幣 (colored coins)，支付通道(payment (state) channels)和路由支付通道(routed payment channels)(Lightning Network)。

Introduction

比特幣系統被設計為分散的貨幣和支付系統。但是，它的大多數功能都源於可用於更廣泛應用程序的更低級別的構造。比特幣不是使用帳戶，用戶，餘額和付款等組件構建的。相反，它使用具有低級加密功能的交易腳本語言，正如我們在[\[transactions\]](#)中看到的那樣。正如帳戶，餘額和支付的更高級別概念可以從這些基本原語派生一樣，許多其他復雜應用程序也是如此。因此，比特幣區塊鏈可以成為向應用程序提供信任服務的應用程序平台，例如智能合約，遠遠超過數字貨幣和支付的最初目的。

Building Blocks (Primitives)

系統能長期正確運行是因為比特幣提供某些保證，可用作“構建塊 Building Blocks”來創建應用程序。這些包括：

No Double-Spend (沒有雙倍消費)

比特幣分散式一致性算法的最基本保證確保不會花費兩次 UTXO。

Immutability (不可變)

一旦交易被記錄在區塊鏈中並且已經為後續塊添加了足夠的工作，交易的數據就變得不可變。不可變性由能量承擔，因為重寫區塊鏈需要消耗能量來產生工作證明(PoW)。所需的能量以及不可變性的程度隨著包含交易的塊頂部的工作量而增加。

Neutrality (中立性)

分散的比特幣網路傳播有效的交易，而不考慮這些交易的來源或內容。這意味著任何人都可以以足夠的費用和信任創建有效的交易，他們將能夠傳輸該交易，並在任何時候將其納入區塊鏈。

Secure Timestamping (安全的時間戳記)

共識規則拒絕過去或將來時間戳太遠的任何塊。這可確保可以信任塊上的時間戳。塊上的時間戳意味著對所有包含的交易輸入的未使用保證。

Authorization (授權)

在分散網絡中驗證的數字簽名提供授權保證。如果未經腳本中隱含的私鑰持有者的授權，則無法執行包含數字簽名要求的腳本。

Auditability (審計能力)

所有交易都是公開的，可以進行審計。所有交易和塊都可以在一個完整的鏈中鏈接回到創世塊。

Accounting (會計)

在任何交易中（除了 coinbase 交易），輸入值等於輸出值+費用(fee)。在交易中無法創建或破壞比特幣。輸出不能超過輸入。

Nonexpiration (非期限)

有效的交易不會過期。如果它今天有效，它將在不久的將來有效，只要輸入仍未用完且共識規則不變。

Integrity (廉正)

使用 SIGHASH_ALL 簽名的比特幣交易或由另一個 SIGHASH 類型簽名的部分交易不能在不使簽名無效的情況下進行修改，從而使交易本身無效。

Transaction Atomicity (交易原子性)

比特幣交易是原子的。它們要麼有效，要麼被證實 (mined)，要麼不有效。部分交易無法開採，並且沒有事務的臨時狀態。在任何時候，交易要麼被開採，要麼不開採。

Discrete (Indivisible) Units of Value (離散（不可分割）價值單位)

交易輸出是離散的，不可分割的價值單位。他們可以全部花費或未花費。它們不能分開或部分花費。

Quorum of Control (控制法定人數)

腳本中的多重簽名約束強加了在多重簽名方案中預定義的法定數量的授權。 M-of-N 要求由共識規則強制執行。

Timelock/Aging (時間鎖/老化)

任何包含相對或絕對時間鎖 Timelock 的腳本只能在其年齡超過指定時間後執行。

Replication (複本)

區塊鏈的分散存儲可確保在進行交易挖掘後，在經過充分確認後，在網路上進行複製，並變得持久且能夠抵禦電源丟失、資料丟失等。

Forgery Protection (偽造保護)

交易只能花費現有的、經過驗證的輸出。不可能創造或偽造價值。

Consistency (一致性)

在沒有礦工分區的情況下，在區塊鏈中記錄的塊會根據記錄它們的深度進行重組或出現指數下降的可能性的分歧。一旦深入記錄，改變所需的計算和能量就會使變化實際上不可行。

Recording External State (記錄外部狀態)

交易可以通過 op_return 提交資料值，表示外部狀態機中的狀態轉換。

Predictable Issuance (可預測的發行)

Less than 21 million bitcoin will be issued, at a predictable rate.

構建塊的清單並不完整，每個新功能都被引入到比特幣中，會添加更多的構建塊。

Applications from Building Blocks

比特幣提供的構建塊(Building Blocks)是信任平臺的元素，可用於編寫應用程式。下面是當前存在的應用程式及其使用的構建塊的一些示例：

Proof-of-Existence (Digital Notary) 存在證明 (數位公證)

Immutability + Timestamp + Durability. 數位指紋(digital fingerprint)可以與交易一起提交到區塊鏈，這證明文檔在被記錄時存在 (Timestamp)。指紋不能事後修改 (Immutability)，證據將被永久存儲 (Durability)。

Kickstarter (Lighthouse)

Consistency + Atomicity + Integrity. 如果您在籌款活動中籤署一個輸入和輸出（完整性 Integrity），其他人可以為籌款活動做出貢獻，但在目標（output value）獲得資助（一致性 Consistency）之前，不能花費（原子性 Atomicity）。

Payment Channels 支付通道

Quorum of Control + Timelock + No Double Spend + Nonexpiration + Censorship Resistance + Authorization. 具有時間鎖（Timelock）的 multisig 2-of-2（法定數量 Quorum）用作支付通道的“結算 settlement”交易，可以由任何一方（授權 Authorization）在任何時間（非審查阻力 Censorship Resistance）保持（Nonexpiration）。然後，雙方可以創建承諾交易，在較短的時間鎖（Timelock）上花費（No Double-Spend）結算。

Colored Coins

我們將討論的第一個區塊鏈應用是彩色硬幣 *colored coins*。

彩色硬幣是指一組使用比特幣交易記錄“除比特幣以外”的外在資產(extrinsic assets)的創建、所有權和轉移的類似技術。“外在 extrinsic”是指不直接存儲在比特幣區塊鏈上的資產，而不是比特幣本身，而比特幣本身就是區塊鏈固有的資產。

彩色硬幣用於跟踪數字資產(digital asset)以及第三方持有的實物資產(physical assets)，並通過彩色硬幣所有權證券進行交易。“數字資產彩色硬幣”可以代表無形資產，例如股票證書，許可證，虛擬財產或大多數任何形式的許可知識產權（商標，版權等）。“有形資產彩色硬幣”可以代表商品（金，銀，油），土地所有權，汽車，船隻，飛機等的所有權證明。

The term derives from the idea of "coloring" or marking a nominal amount of bitcoin, for example, a single satoshi, to represent something other than the bitcoin value itself. As an analogy, consider stamping a \$1 note with a message saying, "this is a stock certificate of ACME" or "this note can be redeemed for 1 oz of silver" and then trading the \$1 note as a certificate of ownership of this other asset. The first

implementation of colored coins, named *Enhanced Padded-Order-Based Coloring* or *EPOBC*, assigned extrinsic assets to a 1-satoshi output. In this way, it was a true "colored coin," as each asset was added as an attribute (color) of a single satoshi.

More recent implementations of colored coins use the `OP_RETURN` script opcode to store metadata in a transaction, in conjunction with external data stores that associate the metadata to specific assets.

The two most prominent implementations of colored coins today are [Open Assets](#) and [Colored Coins by Colu](#). These two systems use different approaches to colored coins and are not compatible. Colored coins created in one system cannot be seen or used in the other system.

Using Colored Coins

Colored coins are created, transferred, and generally viewed in special wallets that can interpret the colored coins protocol metadata attached to bitcoin transactions. Special care must be taken to avoid using a colored-coin-related key in a regular bitcoin wallet, as the regular wallet may destroy the metadata. Similarly, colored coins should not be sent to addresses managed by regular wallets, but only to addresses that are managed by wallets that are colored-coin-aware. Both Colu and Open Assets systems use special colored-coin addresses to mitigate this risk and to ensure that colored coins are not sent to unaware wallets.

Colored coins are also not visible to most general-purpose blockchain explorers. Instead, you must use a colored-coins explorer to interpret the metadata of a colored coins transaction.

An Open Assets-compatible wallet application and blockchain explorer can be found at [coinprism](#).

A Colu Colored Coins-compatible wallet application and blockchain explorer can be found at [Blockchain Explorer](#).

A Copay wallet plug-in can be found at [Colored Coins Copay Addon](#).

Issuing Colored Coins

Each of the colored coins implementations has a different way of creating colored coins, but they all provide similar functionality. The process of creating a colored coin asset is called *issuance*. An initial transaction, the *issuance transaction* registers the asset on the bitcoin blockchain and creates an *asset ID* that is used to reference the asset. Once issued, assets can be transferred between addresses using *transfer transactions*.

Assets issued as colored coins can have multiple properties. They can be *divisible* or *indivisible*, meaning that the amount of asset in a transfer can be an integer (e.g., 5) or have decimal subdivision (e.g., 4.321). Assets can also have *fixed issuance*, meaning a certain amount are issued only once, or can be *reissued*, meaning that new units of the asset can be issued by the original issuer after the initial issuance.

Finally, some colored coins enable *dividends*, allowing the distribution of bitcoin payments to the owners of a colored coin asset in proportion to their ownership.

Colored Coins Transactions

The metadata that gives meaning to a colored coin transaction is usually stored in one of the outputs using the OP_RETURN opcode. Different colored coins protocols use different encodings for the content of the OP_RETURN data. The output containing the OP_RETURN is called the *marker output*.

The order of the outputs and position of the marker output may have special meaning in the colored coins protocol. In Open Assets, for example, any outputs before the marker output represent asset issuance. Any outputs after the marker represent asset transfer. The marker output assigns specific values and colors to the other outputs by referencing their order in the transaction.

In Colored Coins (Colu), by comparison, the marker output encodes an opcode that determines how the metadata is interpreted. Opcodes 0x01 through 0x0F indicate an issuance transaction. An issuance opcode is usually followed by an asset ID or other identifier that can be used to retrieve the asset information from an external source (e.g., bittorrent). Opcodes 0x10 through 0x1F represent a transfer transaction. Transfer transaction metadata contain simple scripts that transfer specific amounts of assets from inputs to outputs, by reference to their index. Ordering of inputs and outputs is therefore important in the interpretation of the script.

If the metadata is too long to fit in OP_RETURN, the colored coins protocol may use other "tricks" to store metadata in a transaction. Examples include putting metadata in a redeem script, followed by OP_DROP opcodes to ensure the script ignores the metadata. Another mechanism used is a 1-of-N multisig script where only the first public key is a real public key that can spend the output and subsequent "keys" are replaced by encoded metadata.

In order to correctly interpret the metadata in a colored coins transaction you must use a compatible wallet or block explorer. Otherwise, the transaction looks like a "normal" bitcoin transaction with an OP_RETURN output.

As an example, I created and issued a MasterBTC asset using colored coins. The MasterBTC asset represents a voucher for a free copy of this book. These vouchers can be transferred, traded, and redeemed using a colored coins-compatible wallet.

For this particular example, I used the wallet and explorer at <https://coinprism.info>, which uses the Open Assets colored coins protocol.

The issuance transaction as viewed on coinprism.info shows the issuance transaction using the Coinprism block explorer:

<https://www.coinprism.info/tx/10d7c4e022f35288779be6713471151ede967caaa39eecd35296aa36d9c109ec>

Transaction

Hash	10d7c4e022f35288779be6713471151ede967c...	Transaction confirmed
Date	Sunday, August 17, 2014 5:42:41 PM	Confirmations 137057 confirmations
Fee paid	0.0001 BTC	Time Sunday, August 17, 2014 5:...
Assets transacted	1	Block 00000000000000000150ab5...
		Height 316117



Bitcoin	
 akTnsDt5uzpioRST76VFRQM8q8sBF... -0.0001	Fees 0.0001
Free copy of "Mastering Bitcoin"	AcuRVsoa81hoLHmVTNXrRD8KpTqUXe...
 + Issued assets -20	akTnsDt5uzpioRST76VFRQM8q8sBFnQ... 20

Figure 1. The issuance transaction as viewed on coinprism.info

As you can see, coinprism shows the issuance of 20 units of "Free copy of Mastering Bitcoin," the MasterBTC asset, to a special colored coin address:

akTnsDt5uzpioRST76VFRQM8q8sBFnQiwcx

Warning Any funds or colored assets sent to this address will be lost forever. Do not send value to this example address!

The transaction ID of the issuance transaction is a "normal" bitcoin transaction ID. [The issuance transaction on a block explorer that doesn't decode colored coins](#) shows that same transaction in a block explorer that doesn't decode colored coins. We'll use *blockchain.info*:

<https://blockchain.info/tx/10d7c4e022f35288779be6713471151ede967caaa39eecd35296aa36d9c109ec>

Transaction View information about a bitcoin transaction

10d7c4e022f35288779be6713471151ede967caaa39eecd35296aa36d9c109ec	
1HpyyiGaXLq7ZCHk3s9EkVEFoG15oLn2Us (0.01 BTC - Output)	<div> <div>1HpyyiGaXLq7ZCHk3s9EkVEFoG15oLn2Us - (Unspent)</div> <div>0.000006 BTC</div> </div> <div> <div>Unable to decode output address - (Unspent)</div> <div>0 BTC</div> </div> <div> <div>1HpyyiGaXLq7ZCHk3s9EkVEFoG15oLn2Us - (Spent)</div> <div>0.009894 BTC</div> </div> <div>0.0099 BTC</div>

Figure 2. The issuance transaction on a block explorer that doesn't decode colored coins

As you can see, *blockchain.info* doesn't recognize this as a colored coins transaction. In fact, it marks the second output with "Unable to decode output address" in red letters.

If you select "Show scripts & coinbase" on that screen, you can see more detail about the transaction ([The scripts in the issuance transaction](#)).

Output Scripts

OP_DUP OP_HASH160 b895201a7cfd91a9bfb3b42cd114d42e3a634d2 OP_EQUALVERIFY OP_CHECKSIG	OK
OP_RETURN 4f41010001141b753d68747470733a2f2f6370722e736d2f466f796b777248365559 (decoded) j"OA[]u=https://cpr.sm/FoykwrH6UY	Strange
OP_DUP OP_HASH160 b895201a7cfd91a9bfb3b42cd114d42e3a634d2 OP_EQUALVERIFY OP_CHECKSIG	OK

Figure 3. The scripts in the issuance transaction

Once again, *blockchain.info* doesn't understand the second output. It marks it with "Strange" in red letters. However, we can see that some of the metadata in the marker output is human-readable:


```
OP_RETURN 4f41010001141b753d68747470733a2f2f6370722e736d2f466f796b777248365559
```

```
(decoded) "OA____u=https://cpr.sm/FoykwrH6UY"
```

Let's retrieve the transaction using bitcoin-cli:

```
$ bitcoin-cli decoderawtransaction `bitcoin-cli getrawtransaction
10d7c4e022f35288779be6713471151ede967caaa39eecd35296aa36d9c109ec`
```

Stripping out the rest of the transaction, the second output looks like this:

```
{
  "value": 0.00000000,
  "n": 1,
  "scriptPubKey": "OP_RETURN 4f41010001141b753d68747470733a2f2f6370722e736d2f466f796b777248365559"
}
```

The prefix 4F41 represents the letters "OA", which stands for "Open Assets" and helps us identify that what follows is metadata defined by the Open Assets protocol. The ASCII-encoded string that follows is a link to an asset definition:

```
u=https://cpr.sm/FoykwrH6UY
```

If we retrieve this URL, we get a JSON-encoded asset definition, as shown here:

```
{
  "asset_ids": [
    "AcuRVsoa81hoLHmVTNXrRD8KpTqUXeqwGH"
  ],
  "contract_url": null,
  "name_short": "MasterBTC",
  "name": "Free copy of \"Mastering Bitcoin\"",
  "issuer": "Andreas M. Antonopoulos",
  "description": "This token is redeemable for a free copy of the book \"Mastering Bitcoin\"",
  "description_mime": "text/x-markdown; charset=UTF-8",
  "type": "Other",
  "divisibility": 0,
  "link_to_website": false,
  "icon_url": null,
  "image_url": null,
  "version": "1.0"
}
```

Counterparty

Counterparty is a protocol layer built on top of bitcoin. The Counterparty protocol, similar to colored coins, offers the ability to create and trade virtual assets and tokens. In addition, Counterparty offers a decentralized exchange for assets. Counterparty is also implementing smart contracts, based on the Ethereum Virtual Machine (EVM).

Like the colored coins protocols, Counterparty embeds metadata in bitcoin transactions, using the OP_RETURN opcode or 1-of-N multisignature addresses that encode metadata in the place of public keys. Using these mechanisms, Counterparty implements a protocol layer encoded in bitcoin transactions. The additional protocol layer can be interpreted by applications that are Counterparty-aware, such as wallets and blockchain explorers, or any application built using the Counterparty libraries.

Counterparty can be used as a platform for other applications and services, in turn. For example, Tokenly is a platform built on top of Counterparty that allows content creators, artists, and companies to issue tokens that express digital ownership and can be used to rent, access, trade, or shop for content, products, and services. Other applications leveraging Counterparty include games (Spells of Genesis) and grid computing projects (Folding Coin).

More details about Counterparty can be found at <https://counterparty.io>. The open source project can be found at <https://github.com/CounterpartyXCP>.

Payment Channels and State Channels

支付通道(Payment channels)是一種在比特幣區塊鏈之外的雙方之間交換比特幣交易的不可信賴(trustless)的機制。這些交易如果在比特幣區塊鏈上(on-chain)結算, 則是有效的, 若是在連鎖外(off-chain)進行的, 可當作本票(promissory notes)為了最終化批次結算。由於交易未結算, 因此可以在沒有結算延遲的情況下進行交換(ex: 10min 的 PoW 證明), 從而允許極高的交易量、較低(亞毫秒)延遲和精細(satoshi-level)性。

實際上, "通道/頻道/管道/渠道/信道/channel" 一詞是一個隱喻。**狀態管道(State channels)**是以區塊鏈之外的雙方狀態交換為代表的虛擬結構。沒有 "通道" 本身, 基礎資料傳輸機制也不是通道。我們使用 "通道" 一詞來表示區塊鏈之外雙方之間的關係和共用狀態。

要進一步解釋這個概念, 請考慮 TCP 串流 (TCP Stream)。從更高級別的協定的角度來看, 它是一個連接互聯網上兩個應用程式的 "通訊端(Socket)"。但是, 如果您查看網路流量, TCP Stream 只是 IP 資料包上的虛擬通道。TCP Stream Sequence(TCP 串流)的每個結點(endpoint)會組裝 IP 資料包, 以造成位元組流的錯覺。而實際上, TCP Stream Sequence 是所有斷開連接(disconnected packets)的資料包組成的。同樣, 支付管道(Payment channels)只是一系列交易(每個交易是獨立個體, 並非"流")。如果正確排序和連接, 即使您不信任頻道的另一端, 它們也會創建您可信賴的可兌換義務。

在本節中, 我們將介紹各種形式的支付渠道(payment channels)。首先, 我們將研究用於為計量微支付服務構建單向(One-way/unidirectional)支付渠道的機制, 例如 streaming video。然後, 我們將擴展這一機制並引入雙向(Two-way/bidirectional)支付渠道。最後, 我們將看看雙向通道如何端到端地連接以在路由網絡中形成多跳信道(multihop channels), 首先以 Lightning Network 的名義提出。

支付渠道是更廣泛的狀態渠道(State Channel)概念的一部分(State Channel>Payment Channel)，狀態渠道代表了狀態的脫鏈變更(off-chain alteration of state)，通過區塊鏈中的最終結算來保證。支付渠道是狀態渠道，其中被改變的狀態是虛擬貨幣的餘額。

State Channels—Basic Concepts and Terminology

通過鎖定區塊鏈上的共享狀態的交易在兩方之間建立狀態通道。這稱為**融資交易(funding transaction)**或**錨定交易(anchor transaction)**。必須將此單個交易傳輸到網絡並挖掘以建立通道。在支付渠道的示例中，鎖定狀態(locked state)是通道的初始餘額（以貨幣為單位，ex: 10BTC 被鎖住）。

然後雙方交換簽署的交易，稱為**承諾交易(commitment transactions)**，此交易將改變初始被鎖住的狀態。這些交易是有效的交易，因為它們可以由任何一方提交以供結算，而是由各方在通道關閉之前進行離線保管。可以以任一方創建、簽名和將交易傳輸到另一方創建狀態更新。實際上，這意味著每秒可以交換數千個交易。

在交換承諾交易(commitment transactions)時，雙方也會使以前的狀態無效，因此，最新的承諾交易記錄始終是唯一可以贖回的交易記錄。這可以防止任何一方通過單方面關閉具有比當前狀態更有利於它們的過期先前狀態的信道來作弊。我們將在本章的其餘部分中研究可用於使先前狀態無效的各種機制。

最後，通過向區塊鏈提交最終**結算交易(settlement transaction)**，或者通過向區塊鏈提交最後一個承諾交易的任何一方單方面，可以合作關閉該渠道。如果一方意外斷開連接，則需要單方面關閉選項。結算交易代表渠道的最終狀態，並在區塊鏈上結算。

在渠道的整個生命週期中，只需要提交兩個交易：資金和結算交易(funding and settlement transactions)。在這兩個狀態間，雙方可以交換任何其他他人從未見過的承諾交易，也不會提交給區塊鏈。

EX: A payment channel between Bob and Alice，顯示資金，承諾和結算交易，說明了 Bob 和 Alice 之間的支付渠道，顯示了資金，承諾和結算交易(funding, commitment, and settlement transactions)。

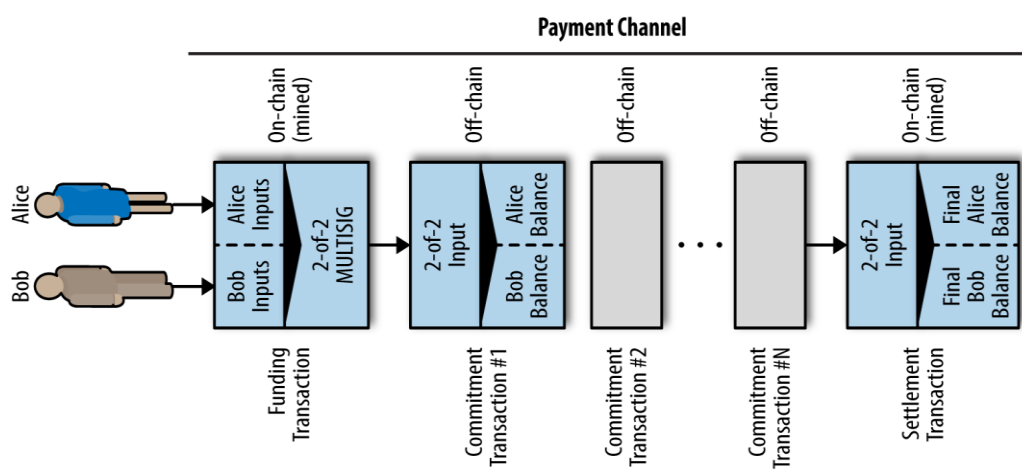


Figure 4. Bob 和 Alice 之間的支付渠道，顯示資金，承諾和結算交易

Simple Payment Channel Example

為了解釋狀態通道(state channels)，我們從一個非常簡單的例子開始。我們演示了一個單向通道(One-way channel)，意味著價值只在一個方向流動。我們也將天真的假設沒有人試圖欺騙，保持簡單。一旦我們解釋了基本的管道想法，我們就會看看需要什麼才能讓它變得不可信(但又安全)，這樣任何一方都不能欺騙，即使他們在嘗試欺騙。

在這個例子中，我們假設有兩個參與者：Emma 和 Fabian。Fabian 提供視頻流服務並以秒數計費。Fabian 每秒視頻收費 0.01 毫比特幣 (0.00001 BTC)，相當於每小時視頻 36 毫比特幣 ($0.00001 \times 3600 = 0.036$ BTC)。Emma 是從 Fabian 購買此串流媒體視頻服務的用戶。Emma 通過支付渠道從 Fabian 購買流媒體視頻，支付每秒視頻節目 Emma 使用支付渠道從 Fabian 購買視頻流服務。

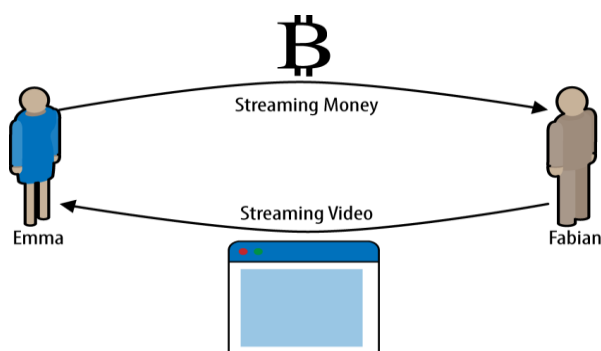


Figure 5. Emma 通過支付管道從 Fabian 購買流媒體視頻，每一秒鐘都要付費

在這個例子中，Fabian 和 Emma 正在使用處理支付頻道和視頻流的特殊軟件。Emma 在她的瀏覽器中運行該軟件，Fabian 正在服務器上運行它。該軟件包括基本的比特幣錢包功能，可以創建和簽署比特幣交易。概念和術語“支付渠道”都完全隱藏在用戶之外。他們看到的是以秒計費的視頻。

為了建立支付渠道，Emma 和 Fabian 建立了一個 2 比 2 的多重簽名地址(2-of-2 multisignature address)，他們每個人都拿著一把鑰匙(key)。從 Emma 的角度來看，她瀏覽器中的軟件提供了一個帶有 P2SH 地址（以“3”開頭）的 QR 碼，並要求她提交長達 1 小時視頻的“存款(deposit)”。該地址由 Emma 壓錢。支付給多重簽名地址的 Emma 交易是支付渠道(Payment Channel)的資金或錨定交易(funding/anchor transaction)。

對於這個例子，假設 Emma 以 36 毫比特幣 (0.036 BTC) 做為頻道存款。這將允許 Emma 消耗長達 1 小時的流媒體視頻。在這種情況下，funding transaction 設置了可以在此渠道中傳輸的最大金額(0.036BTC)，也就是渠道容量(channel capacity)。

funding transaction 消耗了 Emma 錢包的一個或多個輸入(input)(UTXO 概念)，從中獲取資金。它創建了一個輸出(output)，其值為 36 毫微克，支付給 Emma 和 Fabian 共同控制的多重簽名 2-of-2 地址。它可能有額外的輸出，但這將轉回 Emma 的錢包(多餘的 Output 會在 funding transaction 上鏈後轉回 Emma 的錢包)。

一旦資金交易得到確認，Emma 就可以開始播放視頻。Emma 的軟件創建並簽署承諾交易(commitment transaction)，將渠道餘額改為 0.01 毫 BTC 至 Fabian 的地址，並將 35.99 毫 BTC 退還給 Emma。由 Emma 簽署的交易消耗了融資交易(funding transaction)產生的 36 毫 BTC 輸出(output)，並產生兩個輸出：一個用於退款，另一個用於 Fabian 的付款。該交易僅部分簽名 - 它需要兩個簽名 (2-of-2)，但只有 Emma 的簽名。當 Fabian 的服務器收到此交易時，它會添加第二個簽名 (對於 2-of-2 輸入) 並將其返回給 Emma 以及 1 秒的視頻。現在雙方都有一個完全簽署的承諾交易(commitment transaction)，可以兌換，代表渠道的正確最新餘額。任何一方都不會將此交易廣播到網絡。

在下一輪中，Emma 的軟件創建並簽署了另一項承諾交易(commitment transaction)（承諾 #2），該交易消耗了來自融資交易(funding transaction)的相同的 2-of-2 output。第二個承諾交易(commitment transaction)將一個 0.02 毫 BTC 的輸出分配給 Fabian 的地址，將一個 35.98 毫 BTC 的輸出分配回 Emma 的地址。此新交易是兩秒視頻的付款。Fabian 的軟件簽署並返回第二個承諾交易，以及另一個視頻。

通過這種方式，Emma 的軟件繼續向 Fabian 的服務器發送承諾交易，以換取串流媒體視頻。由於 Emma 消耗了更多的視頻，因此 channel 的錢逐漸積累，有利於 Fabian。讓我們說 Emma 觀看 600 秒(10 分鐘)的視頻，創建和簽署 600 份承諾交易。最後一筆承諾交易（# 600）將有兩個輸出（將渠道的餘額 6 毫 BTC 分別為 Fabian、30 毫 BTC 分配給 Emma）。

最後，Emma 點擊“停止”以停止流式傳輸視頻。Fabian 或 Emma 現在可以傳輸最終的狀態交易以進行結算。最後一筆交易是結算交易，並為所有 Emma 消費的視頻支付 Fabian，將剩餘的融資交易退還給 Emma。

Emma 與 Fabian 的支付渠道，顯示更新渠道餘額的承諾交易(commitment transaction)。

最後，區塊鏈中只記錄了兩筆交易：建立渠道的資金交易(funding transaction)和在兩個參與者之間正確分配最終餘額的結算交易(settlement transaction)。

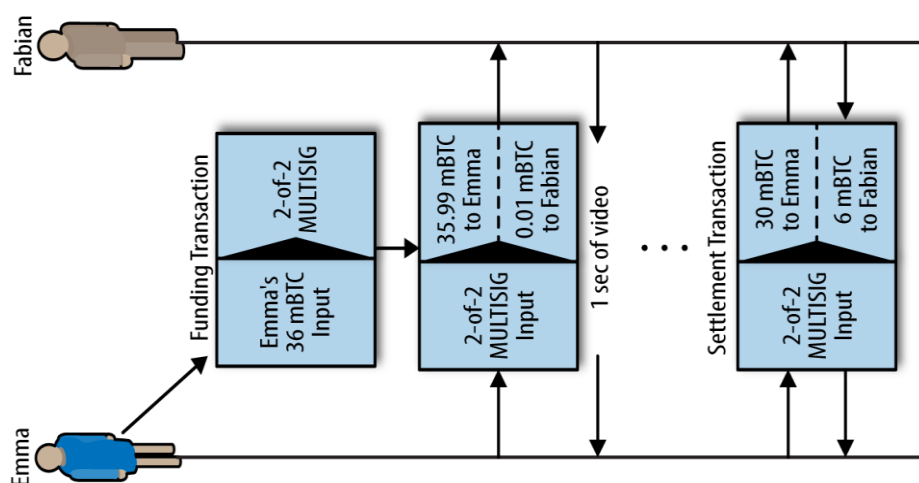


Figure 6. Emma 與 Fabian 的支付渠道，顯示更新渠道餘額的承諾交易

Making Trustless Channels 製作無信任的頻道

我們剛剛描述的頻道有效，但只有雙方合作，沒有任何失敗或企圖作弊。讓我們看一下打破這個渠道的一些場景，看看解決這些問題需要什麼：

- 一旦融資交易(funding transaction)發生，Emma 需要 Fabian 的簽名才能收回任何款項。如果 Fabian 消失，Emma 的資金被鎖定在 2-of-2 中並且實際上已經失敗。如果一方在雙方至少簽署了一項承諾交易之前斷開聯繫，這一管道就會導致資金損失。
- 在頻道運行時，Emma 可以接受 Fabian 已簽署的任何承諾交易，並將其中一個交易傳送到區塊鏈。如果她能 #1 傳輸承諾交易，只支付 1 秒的視頻，為什麼還要支付 600 秒的視頻？艾瑪可以通過廣播事先承諾(commitment)，這是對她有利的欺騙。

這兩個問題都可以通過 timelocks 來解決-讓我們看看如何使用交易級時間鎖 (transaction-level nlocktime)。

除非有保證退款，否則 Emma 不會冒險為 2-of-2 multisig 提供資金。為了解決這個問題，Emma 同時構建了資金和退款交易(funding and refund transaction)。她簽署了 funding transaction，但沒有將其轉發給任何人。Emma 只將退款交易(refund transaction)傳送給 Fabian 並獲得他的簽名。

退款交易(refund transaction)作為第一筆承諾交易(commitment transaction)，其時間鎖定(timelock)確定了渠道生命的上限。在這種情況下，Emma 可以將 nLocktime 設置為 30 天或未來 4320 塊。所有後續承諾交易(commitment transaction)必須具有較短的時間鎖，以便在退款交易之前兌換。

既然 Emma 已經有了完全簽署的退款交易，她可以自信地傳輸已簽署的融資交易，因為她知道她可以在時間到期後最終兌換退款交易，即使 Fabian 消失。

雙方在渠道生命週期內交換的每筆承諾交易將被鎖定到未來。但是，每項承諾的延遲時間會略短，因此最近的承諾可以在之前的承諾無效之前兌現。由於 nLockTime，任何一方都不能成功傳播任何承諾事務，直到它們的時間到期為止。如果一切順利，他們將合作並通過結算交易優雅地關閉渠道，使得不必傳輸中間承諾交易。如果不是，則可以傳播最近的承諾交易(也就是 nLockTime 越小的)以結算賬戶並使所有先前的承諾交易無效(補充: 即使仿造一個 nLockTime 更小的 commitment transaction 也沒用，因為需要雙方的簽名)。

例如，如果承諾交易 #1 在將來被時間鎖定到 4320 個塊，則承諾交易 #2 將來被鎖定到 4319 個塊。在承諾交易 #1 變為有效之前(也就是最終結算之前，或退款交易所設的 Time 上限已到達之前)，承諾交易 #600 可以經由 600 個 Block 之後發生。

每個承諾設置較短的時間鎖，允許在先前的承諾變為有效之前花費它。

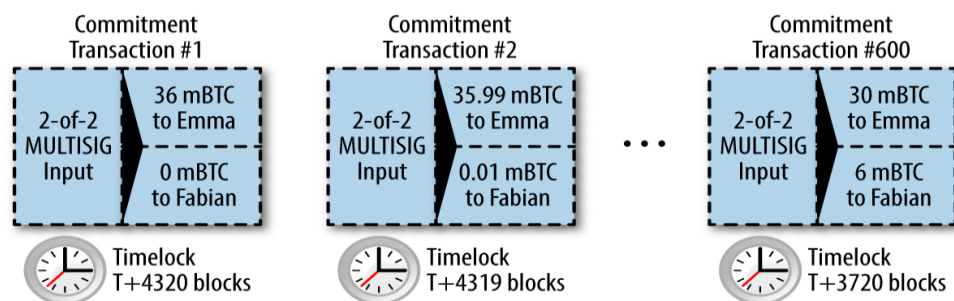


Figure 7. 每項承諾都設定了較短的時間鎖，允許在之前的承諾生效之前花費

每個後續的承諾交易必須有一個較短的時間鎖，以便它可以在其前身之前和退款交易之前進行廣播。更早廣播承諾的能力確保了它將能夠花費資金輸出，並排除任何其他承諾交易被用於花費輸出來兌換。比特幣區塊鏈提供的保證，防止重複支出和強制使用，有效地允許每個承諾交易使其前身失效。

狀態通道使用時間鎖(nlocktime)在時間維度上強制執行智慧合同。在本例中，我們看到了時間維度如何保證最新的承諾交易記錄在任何早期承諾之前生效。因此，可以傳輸最新的承諾交易，花費投入並使先前的承諾交易無效。執行具有絕對時間鎖的智慧合同可以防止一方當事人的欺騙行為。接下來，我們將看到如何使用腳本級時間表(script-level timelocks)、CheckLockTimeVerify 和 CheckSequenceVerify 來構建更靈活、更有用和更複雜的狀態通道。

第一種形式的單向支付渠道在 2015 年由阿根廷開發團隊演示為原型視頻流應用程序。您仍然可以在 streamium.io 上看到它。

時間鎖不是使先前承諾交易(prior commitment transactions)無效的唯一方法。在下一節中，我們將看到如何使用吊銷金鑰(revocation key)來實現相同的結果。

時間鎖(Timelocks)是有效的，但它們有兩個明顯的缺點:

1. 通過在首次打開通道時建立最大時間鎖定，它們限制了通道的壽命。更糟糕的是，它們迫使渠道實施在允許長期存在的渠道和迫使其中一個參與者等待很長時間以便在過早關閉的情況下退款之間取得平衡。例如: 如果您允許頻道保持開放 30 天，則將退款時間鎖定設置為 30 天，如果其中一方立即消失，則另一方必須等待 30 天才能獲得退款。終點越遠，退款就越遠。
2. 第二個問題是，由於每個後續承諾交易必須減少時間鎖(Timelocks)，因此對於可以在各方之間交換的承諾交易的數量存在明確的限制。例如，一個 30 天的渠道，在未來設置 4320 個區塊的時間鎖，在必須關閉之前只能容納 4320 個中間承諾交易。將時間鎖承諾交易間隔設置為 1 塊存在危險。通過將承諾交易之間的時間間隔設置為 1 個區塊，開發人員為必須保持警惕，保持在線和觀看並準備好隨時傳輸正確承諾交易的渠道參與者創造了非常高的負擔。

現在我們已經了解時間鎖如何用於使先前的承諾無效，我們可以看到合作關閉渠道和通過廣播承諾交易單方面關閉它之間的區別。所有承諾交易都是時間鎖定的，因此廣播承諾交易將始終涉及等待時間到期。

但如果雙方就最終餘額達成一致並知道他們都持有承諾交易最終會使這種平衡成為現實，那麼他們就可以構建一個結算交易而沒有代表相同餘額的時間鎖。在合作關閉中，任何一方都採用最近的承諾交易並建立一個在各方面都相同的結算交易，除了它省略了時間鎖。知道沒有辦法欺騙並獲得更有利的平衡，雙方都可以簽署此結算交易。通過合作簽署和傳輸結算交易，他們可以關閉渠道並立即兌換其餘額。

最糟糕的情況是，其中一方可能很小，拒絕合作，並迫使另一方單方面與最近的承諾交易完成。但如果他們這樣做，他們也必須等待他們的資金。

Asymmetric Revocable Commitments 不對稱的可撤銷承諾

處理先前承諾狀態(prior commitment transactions)的更好方法是明確撤銷(revoke)它們。但是，這並不容易實現。比特幣的一個關鍵特徵是，**一旦交易有效，它仍然有效並且不會過期。取消交易的唯一方法是在開采之前將其輸入與另一個交易進行雙重支出。這就是為什麼我們在上面的簡單支付渠道示例中使用時間鎖來確保在舊的承諾有效之前可以花費更多的近期承諾。**然而，即時的一系列 commitment transactions 產生了許多限制因素，使得支付渠道難以使用。

即使交易不能被取消，它也可以“以不希望使用”的方式構造。我們這樣做的方法是給每一方一個撤銷密鑰(revocation key)，如果他們試圖作弊，可以用來懲罰另一方。這種撤銷先前承諾交易的機制最初是作為閃電網絡(Lightning Network)的一部分提出的。

為了解釋撤銷密鑰，我們將在 Hitesh 和 Irene 運營的兩個交易所之間構建一個更複雜的支付渠道。

Hitesh 和 Irene 分別在印度和美國開展比特幣交易。Hitesh 印度交易所的客戶經常向 Irene 美國交易所的客戶付款，反之亦然。目前，這些交易發生在比特幣區塊鏈上，但這意味著支付費用並等待幾個區塊進行確認。在交易所之間建立支付渠道將顯著降低成本並加速交易流程。

Hitesh 和 Irene 通過合作構建融資交易來啟動渠道(注意: 這是個 Two-way 的例子), 每個渠道雙方都用 5 比特幣做為渠道押金(這裡的“押金”並非指真正的押金, 而是指雙方要投入 Payment Channel 的金額數量, 在這個例子雙方都要先投入 5BTC, 以上述 simple channel 的例子只有 Emma 要押錢)。最初的平衡是 Hitesh 為 5 比特幣, Irene 為 5 比特幣。資金交易將信道狀態鎖定在 2-of-2 multisig 中, 就像 simple channel 的例子一樣。

資金交易(funding transaction)可能有來自 Hitesh 的一個或多個輸入(Input) (總和至少 5 個比特幣, 滿足押金需求), 以及來自 Irene 的一個或多個輸入(Input) (總和至少 5 個比特幣, 滿足押金需求)。輸入必須略微超過信道容量(channel capacity, 也就是該支付管道最大可流動金額)才能支付交易費用。該交易有一個輸出(Output), 將 10 個比特幣鎖定為由 Hitesh 和 Irene 控制的 2-of-2 multisig address(現在, 渠道容量(channel capacity)為 10BTC, 意思是在 fund transaction 鎖定出去 10BTC, 而最後的結算交易(settlement transaction)回來也是 10BTC, 不多不少)。如果他們的投入超過其預期的渠道貢獻(例如 input 的 UTXO 總合超過 10BTC), 那麼資金交易也可能有一個或多個輸出返回 Hitesh 和 Irene。這是一項由雙方提供和簽署的投入的單筆交易。它必須在每個方面進行協作並簽署, 然後才能傳輸。

現在, hitesh 和 irene 沒有創建雙方簽署的單一承諾交易(single commitment transaction, 也就是不像上述例子都是某一方在提出承諾交易, 而另一方負責簽屬跟給資料), 而是創建兩個不對稱(asymmetric)的不同承諾交易(commitment transaction)。

Hitesh 承諾交易有兩個輸出 (由 Irene 簽署)。第一個輸出是立即向 Irene 支付欠她的 5BTC。第二個輸出向 Hitesh 自己支付 5BTC, 但要在 1000 個時間段之後才有效。交易輸出如下所示:

```
Input: 2-of-2 funding output, signed by Irene
```

```
Output 0 <5 bitcoin>:
```

```
  <Irene's Public Key> CHECKSIG
```

```
Output 1:
```

```
  <1000 blocks>
```

```
  CHECKSEQUENCEVERIFY
```

```
  DROP
```

```
  <Hitesh's Public Key> CHECKSIG
```

Irene 承諾交易有兩個輸出 (由 Hitesh 簽署)。第一個輸出是立即向 Hitesh 支付欠他的 5BTC。第二個輸出向 Irene 自己支付她 5BTC, 但要在 1000 個時間段之後才有效。看起來像這樣:

```
Input: 2-of-2 funding output, signed by Hitesh
```

```
Output 0 <5 bitcoin>:
```

```
  <Hitesh's Public Key> CHECKSIG
```

```
Output 1:
```

```
  <1000 blocks>
```

```
  CHECKSEQUENCEVERIFY
```

```
  DROP
```

```
  <Irene's Public Key> CHECKSIG
```


這樣，每一方都有承諾交易(commitment transaction)。其輸入由其他方簽名(*other party*)(先有另一方的簽名，但自己還沒簽)。持有交易的一方也可以隨時補上自己的簽名(完成 2-of-2)和廣播。但是，如果他們廣播承諾交易，則它會立即向另一方付款，而他們必須等待短暫的時間到期(例如上例的 1000Block)。

通過延遲贖回，使我們在任一方選擇單方面廣播承諾交易時，使該方處於輕微劣勢。但僅僅延遲時間並不足以鼓勵公平行為。

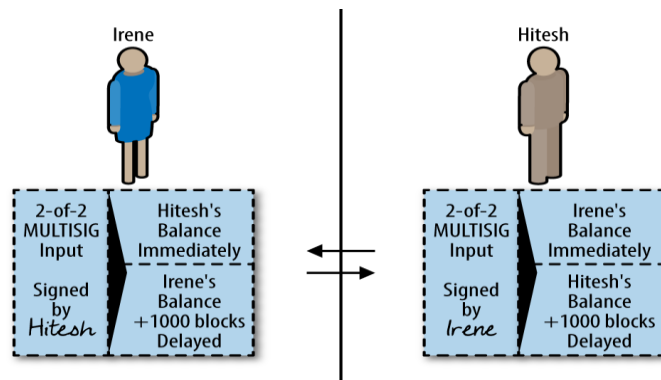


Figure 8. 兩項非對稱承諾交易，延遲支付交易的一方

現在我們介紹這個方案的最後一個要素：一個撤銷密鑰(revocation key)，防止騙子廣播過期的承諾。**撤銷密鑰**允許被冤枉的一方通過獲取頻道的整個餘額來懲罰作弊者。-----> 目前不考慮上述的 nTimelock 機制。

撤銷密鑰(revocation key)由兩個秘密(secrets)組成，**每個秘密(secrets)由每個頻道參與者獨立生成**。它類似於 2-of-2 multisig，但是使用橢圓曲線算法構造(Elliptic Curve Arithmetic)，因此雙方都知道撤銷公鑰(revocation Public key)，但每一方只知道撤銷密鑰(revocation key)的一半(如果參與者有 N 個人，則每一方只知道 1/N 個 revocation key 部分)。

這裡有幾個名詞需先定義：

1. **撤銷密鑰(revocation key)**: 由多個撤銷秘密(revocation secrets)組成，其數量依參與人數而定。
2. **撤銷公鑰(revocation Public key)**: 在這你可以當作是鎖頭，鎖住資金，需要完整的撤銷密鑰解鎖。
3. **已撤銷承諾(revocation commitment)**: 指的是所有舊的承諾交易(又稱 prior commitment transaction)，不論哪個參與者，每當他要提出新的承諾交易時，都要主動撤銷當前的承諾交易，而此撤銷實際做法就是告訴對方另一半“撤銷秘密(revocation secrets)”是什麼，才算撤銷。
4. **下一個承諾交易**: 當參與者要提出新的承諾交易時，要將當前承諾交易的另一半撤銷秘密送給對方(也就是說對方兩把 Key 皆會知道)，然後提出新的承諾交易申請；雙方會先為了新的承諾交易各自產生屬於此新的承諾交易的撤銷密鑰(revocation key)所需的撤銷秘密(revocation secrets)(一人一半)，當對方確認當前承諾交易的另一半撤銷秘密正確(也就是對方擁有了當前承諾交易整把撤銷密鑰(revocation key))，才會同意簽屬此新的承諾交易，而原本當前的承諾交易就會變成“已撤銷承諾”。

在每一輪(round)結束時，雙方都向對方透露了其撤銷秘密的一半，從而使另一方(現在雙方都有一半)在對方將已撤銷承諾廣播出去的情況下獲得了要求罰款的手段。

每個承諾交易記錄都有一個“延遲(delay)”輸出。該輸出的腳本允許提出此承諾交易的那一方在 1000 個 Block 後才能贖回，或者擁有撤銷密鑰的那方可贖回，從而懲罰廣播已撤銷承諾的參與者。

因此，當 hitesh 創建一個承諾交易供 irene 簽署時，他在 1000 塊之後可將第二個輸出支付給自己，或者支付給撤銷公開金鑰（他只知道其中的一半秘密）。Hitesh 會建立好此交易。當他想要更新當前 Channel 狀態，並希望撤銷這一承諾時，他才會向 irene 透露他一半的撤銷秘密。

The second output's script looks like this:

```
Output 0 <5 bitcoin>:
    <Irene's Public Key> CHECKSIG

Output 1 <5 bitcoin>:
IF
    # Revocation penalty output
    <Revocation Public Key>
ELSE
    <1000 blocks>
    CHECKSEQUENCEVERIFY
    DROP
    <Hitesh's Public Key>
ENDIF
CHECKSIG
```

irene 可以自信地簽署這筆由 Hitesh 提出的交易，因為如果 Hitesh 轉發它 irene 將立即取得她的錢。Hitesh 持有這筆交易，但他知道如果他單獨廣播此交易，他將不得不等待 1000 個 Block 才能獲得付款。

當渠道進入下一個狀態時，Hitesh 必須在 Irene 同意簽署下一個承諾交易之前撤銷此承諾交易。要做到這一點，他所要做的就是將他的關於此承諾交易的另一半撤銷鑰匙發給 Irene。一旦 Irene 獲得這交易承諾的兩部分撤銷秘密關鍵，她就可以充滿信心地簽署下一個交易承諾。她知道，如果 Hitesh 試圖通過發布先前已撤銷承諾作弊，她可以使用撤銷密鑰來兌換 Hitesh 的延遲輸出。如果 Hitesh 作弊，Irene 會獲得兩個輸出。同時，Hitesh 只有該已撤銷交易的撤銷秘密的一半，他必須等待 1000Block。Irene 將能夠在 1000Block 之前贖回輸出並懲罰 Hitesh。

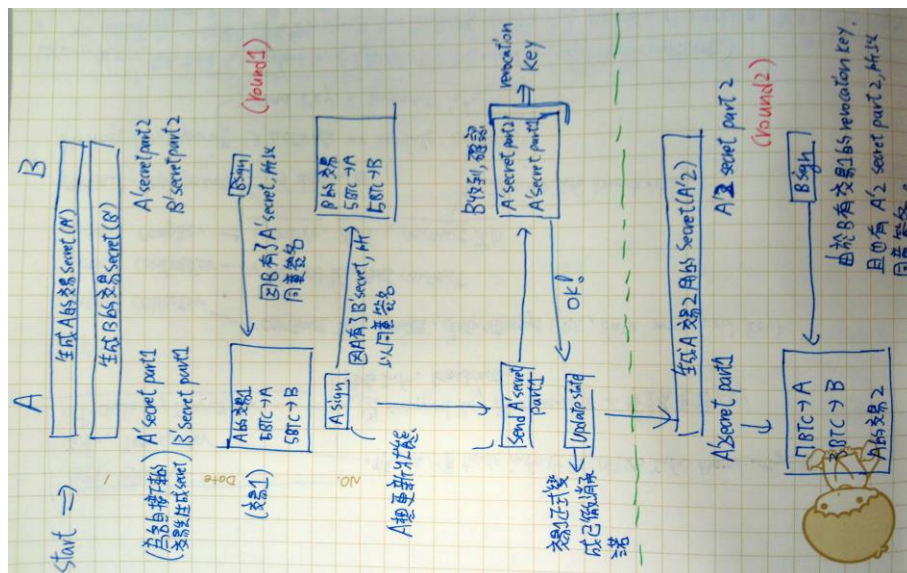
撤銷協議是雙向的，這意味著在每一輪中，隨著渠道狀態的提前，雙方交換新的承諾，交換先前承諾的撤銷秘密，並簽署彼此的新承諾交易。當他們接受一個新的狀態時，他們通過給予對方必要的撤銷秘密來懲罰任何作弊，使得先前的狀態無法使用。

讓我們看一下它是如何工作的一個例子: Irene 的一位客戶希望向 Hitesh 的一位客戶發送 2BTC。為了在整個頻道傳輸 2BTC，Hitesh 和 Irene 必須推進頻道狀態以反映新的平衡。他們將致力於一個新的狀態（State2），其中渠道的 10BTC 由原本狀態的 5BTC 到 Hitesh，3BTC 到 Irene 被拆分成 7BTC 到 Hitesh，3BTC 到 Irene。為了更新此狀態，他們將分別創建反映新渠道餘額的新承諾交易。

和以前一樣，這些承諾交易是不對稱的，因此每一方持有的承諾交易迫使他們在兌換時等待。至關重要的是，在簽署新的承諾交易之前，他們必須首先交換撤銷密鑰以使先前的承諾無效。在這種特殊情況下，由於 Hitesh 狀態由 5BTC 變成 7BTC，情況對他有利，因此他沒有理由藉由廣播先前已撤銷承諾來欺騙礦工 (Miner)。然而，對於 Irene 來說，State1 餘額(5BTC)高於 State2。因此當 Irene 給 Hitesh 她先前承諾交易的

撤銷密鑰 (State1 的) 時，她實際上已經撤銷了她從通道回歸到先前的利潤的能力。因為使用撤銷密鑰，Hitesh 可以毫不拖延地贖回先前承諾交易的兩個輸出。意思是如果 Irene 廣播先前的已撤銷承諾(也就是原先的 5BTC:5BTC，而不是 7BTC:3BTC)，Hitesh 可以行使其獲取所有輸出(Output)的權利(也就是 Hitesh 可以毫不猶豫的直接取走整個 10BTC)。

重要的是，撤銷不會自動發生。雖然 Hitesh 有能力懲罰 Irene 作弊，但他必須努力觀察區塊鏈的作弊跡象。如果他看到先前的已撤銷承諾被廣播，他有 1000 個 Block 時間長度採取行動並使用撤銷密鑰來阻止 Irene 的作弊並通過取得所有 10BTC 來懲罰她。



具有相對時間鎖 (CSV) 的不對稱可撤銷承諾是實施支付渠道的更好方式，也是該技術的重大創新。通過這種結構，渠道可以無限期中保持開放，並且可以擁有數十億的中間承諾交易。

在 Lightning Network 的原型實現中，承諾狀態由 48 位索引標識，允許在任何單個通道中超過 281 萬億 (2.8×10^{14}) 狀態轉換！

以下有幾個問題：

1. 雙方的 Secret 要如何協調產生？
2. 使用相對時間鎖 (CSV)，nTimeLock 的重要性為何？貌似用不太到，依照下面應用的描述，CLTV locktime 鎖定時間用於承諾在多少時間內事情要馬發生要馬不發生(Atomic Swap)。

Hash Time Lock Contracts (HTLC) 哈希時間鎖定合同

支付渠道可以通過特殊類型的智能合約進一步擴展，允許參與者將資金投入到可兌換的秘密，並具有到期時間。此功能稱為哈希時間鎖定合同 (Hash Time Lock Contract HTLC)，用於雙向和路由支付渠道。

我們首先解釋一下 HTLC 的“哈希 hash”部分。要創建 HTLC，付款的預期收件人將首先創建一個 secret R，然後他們計算此秘密的哈希值 H：

$$H = \text{Hash}(R)$$

這會產生一個哈希 H，它可以包含在輸出的鎖定腳本(Output locking script)中。知道秘密 R 的人可以使用它來兌換輸出(Output)。秘密 R 也被稱為散列函數(hash function)的原像(preimage)。preimage 只是用作哈希函數輸入的數據。

HTLC 的第二部分是“時間鎖定 time lock”組件。如果秘密沒有透露，HTLC 的付款人可以在一段時間後獲得“退款 refund”。這是通過使用 CHECKLOCKTIMEVERIFY 進行絕對時間鎖定來實現的。

實現 HTLC 的腳本如下所示:

```
IF
  # Payment if you have the secret R
  HASH160 <H> EQUALVERIFY
ELSE
  # Refund after timeout.
  <locktime> CHECKLOCKTIMEVERIFY DROP
  <Payer Public Key> CHECKSIG
ENDIF
```

任何知道秘密 R 的人，當 hash 結果等於 H 時，可以通過行使腳本中 IF 條件來兌換這個輸出。

如果秘密 R 沒有被透露，在一定數量的區塊之後，付款人可以使用 ELSE 條件要求退款。

這是 HTLC 的基本實現。這種類型的 HTLC 可以被擁有秘密 R 的任何人兌換。一個 HTLC 可以採用許多不同的形式，對腳本略有不同。例如，在第一個子句中添加 CHECKSIG operator 和 public key 會將哈希的兌換限制為已命名的收件人，該收件人還必須知道秘密 R。

Routed Payment Channels (Lightning Network 閃電網絡)

Lightning Network 是一種 P2P 雙向支付通道的路由網絡。像這樣的網絡可以允許任何參與者在不信任任何中間人的情況下將支付從一個 Channel 路由到另一個 Channel。閃電網絡由 Joseph Poon 和 Thadeus Dryja 於 2015 年 2 月首次描述，其基礎是許多其他人提出並詳細闡述的支付渠道概念。

“閃電網絡”指的是路由支付信道網絡(routed payment channel network)的特定設計，其現在已由至少五個不同的開源團隊實現。獨立實現由閃電技術基礎（BOLT）論文中描述的一組互操作性標準協調。

幾個團隊已經發布了 Lightning Network 的原型實現。目前，這些實現只能在 testnet 上運行，因為它們使用的是 segwit，它不會在主比特幣區塊鏈（mainnet）上激活。

Lightning Network 是實現路由支付渠道的一種可能方式。還有其他幾種旨在實現類似目標的設計，例如 Teechan 和 Tumblebit。

Basic Lightning Network Example 基本閃電網示例

讓我們看看它是如何工作的。

在這個例子中，我們有五個參與者：Alice，Bob，Carol，Diana 和 Eric。這五位參與者成對地開設了付款渠道。Alice 與 Bob 有一個付款渠道。Bob 與 Carol，Carol 和 Diana 以及 Diana 與 Eric 聯繫。為簡單起見，我們假設每個參與者都使用 2BTC 作為 Channel 的押金，每個頻道的總容量為 4BTC。

五個閃電網絡參與者，通過雙向支付渠道連接，可以鏈接到 Alice 到 Eric 的支付（路由支付渠道）。

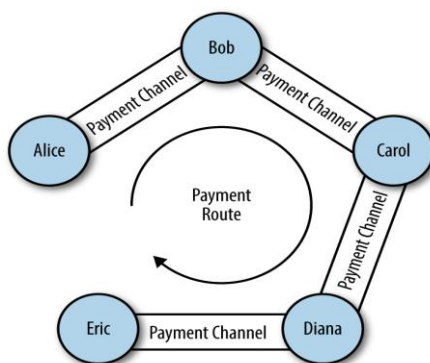


Figure 9. 一系列雙向支付渠道鏈接形成一個 Lightning 網絡，可以將付款從 Alice 路由到 Eric
Alice 想要支付 Eric 1BTC。但是，Alice 沒有通過支付渠道與 Eric 連接。創建支付渠道需要資金交易，必須將其提交給比特幣區塊鏈。愛麗絲不想開設新的支付渠道並投入更多資金。有間接支付 Eric 的方法嗎？

通過連接參與者的支付渠道上的一系列 HTLC 承諾交易，顯示了從 Alice 到 Eric 的付款路由的逐步過程。

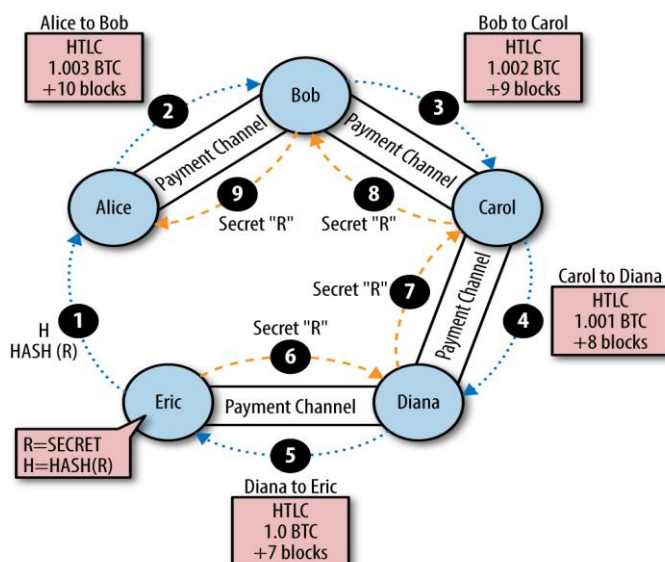


Figure 10. 通過 Lightning 網絡逐步支付路由

Alice 正在運行一個 Lightning Network (LN) 節點，該節點正在跟踪她對 Bob 的支付渠道，並且能夠發現支付渠道之間的路由。Alice 的 LN 節點還能夠通過 Internet 連接到 Eric 的 LN 節點。Eric 的 LN 節點使用隨機數生成器創建一個秘密 R。Eric 的節點並沒有向任何人透露這個秘密。相反，Eric 的節點計算秘密 R 的散列 H 並將此散列傳輸到 Alice 的節點（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 1）。

現在，Alice 的 LN 節點構建了 Alice 的 LN 節點和 Eric 的 LN 節點之間的路由。稍後將更詳細地檢查所使用的路由算法，但是現在讓我們假設 Alice 的節點可以找到有效的路由。(以下的“聲稱”意思等同於領取/解鎖)

然後，Alice 的節點構造一個 HTLC，支付給哈希 H，具有 10Block 長度的退款超時（當前區塊+10），金額為 1.003BTC（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 2）。額外的 0.003 將用於補償中間節點參與此支付路線(也就是費用)。Alice 向 Bob 提供此 HTLC，從她與 Bob 的 Channel 餘額中扣除 1.003BTC 並將其提交給 HTLC。HTLC 具有以下含義:“如果 Bob 知道該秘密(Secret)，則 Alice 將其頻道餘額的 1.003BTC 提交給 Bob，或者如果已經過 10 個 Block 時間長度則退還給 Alice 的餘額”。Alice 和 Bob 之間的渠道平衡現在由具有三個輸出的承諾交易表示：2BTC 餘額還給 Bob(還記得上面假設每個人押 2BTC 在

Channel 上嗎?)，0.997BTC 餘額給 Alice，1.003BTC 在 Alice 的 HTLC 中提交。Alice 的餘額減少了承諾給 HTLC 的金額。

Bob 現在承諾，如果他能夠在接下來的 10 個區塊內獲得秘密 R，他可以聲稱被 Alice 鎖定的 1.003BTC。有了這個承諾，Bob 的節點在 Carol 的支付渠道上構建了一個 HTLC。Bob 的 HTLC 提交了 1.002BTC 來雜湊 H 值，如果 Carol 有秘密 R，Carol 可以兌換它（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 3）。Bob 知道如果 Carol 可以申請他的 HTLC，她必須生產 R。如果鮑勃有 R，他可以用它來向 Alice 索取 HTLC。他還製作了 0.001 比特幣，用於將他的頻道餘額提高到九個 Block ($nTimeLock+1$)。如果 Carol 無法申請他的 HTLC 並且他無法申請 Alice 的 HTLC，那麼一切都會恢復到之前的渠道餘額，並且沒有人會感到茫然。Bob 和 Carol 之間的渠道金額現在是：2 到 Carol，0.998 到 Bob，1.002 由 Bob 提交到 HTLC。

Carol 現在承諾，如果她在接下來的 9 個 Block 內獲得 R，她可以聲稱由 Bob 鎖定的 1.002 比特幣。現在，她可以與 Diana 一起在她的頻道上做出 HTLC 承諾。她提交了一個 1.001 比特幣的 HTLC，用於雜湊 H 值，如果她有秘密 R，Diana 可以贖回（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 4）。從 Carol 的角度來看，如果這樣可行，那麼她的 0.001 比特幣會更好，如果沒有，她就不會失去任何東西。如果 R 被揭示，她的 HTLC 到 Diana 是唯一可行的，此時她可以從 Bob 聲稱 HTLC。Carol 和 Diana 之間的渠道金額現在是：2 到 Diana，0.999 到 Carol，1.0001 由 Carol 致 HTLC。

最後，Diana 可以向 Eric 提供 HTLC，為 7 個塊提交 1 比特幣以雜湊 H 值（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 5）。Diana 和 Eric 之間的渠道金額現在是：2 到埃里克，1 到戴安娜，1 由 Diana 致力於 HTLC。

然而，在路線的這一跳，Eric 有秘密 R。他可以因此聲稱 Diana 提供的 HTLC。他將 R 發送給 Diana 並聲稱 1BTC，將其添加到他的渠道餘額中（通過 [Step-by-step payment routing through a Lightning Network](#) step 6）。渠道金額現在是：1 到 Diana，3 到 Eric。

現在，Diana 有秘密的 R。因此，她現在可以從 Carol 那裡領到 HTLC。Diana 將 R 傳送給 Carol，並將 1.001BTC 添加到她的頻道餘額中（請參閱 [Step-by-step payment routing through a Lightning Network](#) 步驟 7）。現在卡羅爾和戴安娜之間的渠道金額是：卡羅爾 0.999，戴安娜 3.001。戴安娜因參與此付款途徑而“賺取”0.001。

通過路線返回，秘密 R 允許每個參與者聲稱 HTLC。Carol 從 Bob 聲稱 1.002，將他們頻道的餘額設置為：0.998 到 Bob，3.002 到 Carol（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 8）。最後，Bob 聲稱來自 Alice 的 HTLC（參見 [Step-by-step payment routing through a Lightning Network](#) 步驟 9）。他們的渠道金額更新為：0.997 到 Alice，3.003 到 Bob。

Alice 已經向 Eric 支付了 1BTC 而沒有向 Eric 打開頻道。支付路線中的任何中間方都不得相互信任。對於他們的資金在管道中的短期承諾，他們能夠賺取少量費用，唯一的風險是，如果管道關閉或路由付款失敗，退款就會出現小的延誤。

小總結：由上述的例子來看，nTimeLock 與 Hash Locking 的結合技術(HTLC)目的是要確保是正確的人進行交易，而前面所提到的“Asymmetric Revocable Commitments”是用於避免交易途中作惡，而“Asymmetric

Revocable Commitments”本身並不受時間鎖限制，短時間可以有好幾萬筆，但 HTLC 有很明顯的時間規範，避免時間過久導致“Secret R”可以被別人猜出來。

Lightning Network Transport and Routing 閃電網絡傳輸與路由

LN 節點之間的所有通信都是點對點加密的。此外，節點具有長期公開金鑰，它們將其用作識別碼並相互進行身份驗證。

每當一個節點希望向另一個節點發送付款時，它必須首先通過連接具有足夠大量的 **Payment Channel** 來構建通過網路的路徑。節點公佈路由資訊，包括它們打開的管道、每個管道的容量以及路由付款的費用。路由資訊可以通過多種方式共用，隨著閃電網絡技術的進步，不同的路由式通訊協定可能會出現。一些閃電網絡實現使用 IRC protocol 作為方便節點發布路由資訊的機制。路由發現的另一個實現使用 p2p 模型，在該模型中，節點以“flooding model”將通道公告傳播到其對等方，類似於比特幣傳播交易的方式。未來的計畫包括一個名為 [Flare](#) 提案，這是一個混合路由模型，具有本地節點“鄰域(neighborhood)”和更遠距離 beacon 節點。

在我們之前的示例中，Alice 的節點使用這些路由發現機制之一來查找將其節點連接到 Eric 節點的一條或多條路徑。一旦 Alice 的節點構建了路徑，她將通過傳播一系列加密和嵌套指令來連接每個相鄰的支付渠道，從而通過網絡初始化該路徑。

重要的是，此路徑僅為 Alice 的節點所知。支付路徑中的所有其他參與者僅查看相鄰節點。從 Carol 的角度來看，這看起來像是從 Bob 到 Diana 的付款。Carol 不知道 Bob 實際上正在轉發 Alice 的付款。她也不知道戴安娜會向 Eric 轉發付款。

這是 Lightning Network 的一個重要特徵，因為它可以確保付款隱私，並且很難應用監控，審查或黑名單。但是 Alice 如何建立這種支付路徑，而不向中間節點透露任何內容？

Lightning Network 基於稱為 Sphinx 的方案實現**洋蔥路由協議**。此路由協議確保付款發送方可以構建並通過 Lightning Network 傳遞路徑，以便：

- 中間節點可以驗證和解密其路由信息部分並查找下一 hop。
- 除了上一跳和下一跳之外，他們無法了解路徑中任何其他節點。
- 他們無法確定付款路徑的長度或他們自己在該路徑中的位置。
- 路徑的每個部分都經過加密，以至於網絡級攻擊者無法將路徑不同部分的數據包相互關聯。
- 與 Tor（互聯網上的洋蔥路由匿名協議）不同，沒有“退出節點(exit nodes)”可以置於監視之下。付款不需要傳輸到比特幣區塊鏈；節點只是更新 Channel 餘額。

使用這個洋蔥路由協議，Alice 將路徑的每個元素包裝在一個加密層中，從結束開始並向後工作。她用 Eric 的公鑰將消息加密到 Eric。此消息包含在加密到 Diana 的消息中，將 Eric 識別為下一個收件人。給 Diana 的信息包含在加密到 Carol 公鑰的消息中，並將 Diana 識別為下一個收件人。給 Carol 的消息被加密到 Bob 的密鑰。因此，Alice 構建了這個加密的多層“洋蔥(onion)”消息。她將此發送給 Bob，Bob 只能解密和解開外層。在裡面，Bob 找到了一封給 Carol 的信息，他可以轉發給 Carol，但不能破譯自己。在路徑之後，消息被轉發，解密，轉發等，一直到 Eric。每個參與者只知道每一跳中的上一個和下一個節點。

路徑的每個元素包含關於 HTLC 的信息，該信息必須擴展到下一跳，發送的數量，要包括的費用以及 HTLC 的 CLTV 鎖定時間（以塊為單位）到期。當路由信息傳播時，節點使 HTLC 承諾轉發到下一跳。

此時，您可能想知道節點如何可能不知道路徑的長度及其在該路徑中的位置。畢竟，他們收到一條消息並將其轉發到下一跳。它不會縮短，允許他們推斷出路徑大小和位置嗎？為防止這種情況，路徑始終固定為 20 跳並用隨機數據填充。每個節點都看到下一跳和一個固定長度的加密消息來轉發。只有最終收件人才會看到沒有下一跳。對於其他人來說，好像總有 20 多個 hops 要去。

Lightning Network Benefits

Lightning Network 是第二層路由技術。它可以應用於任何支持某些基本功能的區塊鏈，例如多重簽名事務，時間鎖和基本智能合約。

如果閃電網絡位於比特幣網絡之上，比特幣網絡可以在不犧牲無中間人無信任操作原則的情況下顯著提高容量，隱私，粒度和速度：

Privacy: 閃電網絡支付比比特幣區塊鏈支付更私密，因為它們不公開。雖然路線中的參與者可以看到通過其渠道傳播的付款，但他們不知道發件人或收件人。

Fungibility: 閃電網絡使比特幣的監控和黑名单變得更加困難，增加了貨幣的可替代性。

Speed: 使用 Lightning Network 的比特幣交易以毫秒而不是分鐘來結算，因為 HTLC 被清除而不會將事務提交到塊。

Granularity: 閃電網絡可以支付至少與比特幣“灰塵 dust”限制一樣小的支付，甚至可能更小。一些提議允許 subatoshi 增量(也就是 BTC 延伸出來的 subBTC 的概念，有點像是美元跟美分)。

Capacity: 閃電網絡將比特幣系統的容量提高了幾個數量級。每秒可以通過 Lightning Network 路由的支付數量沒有實際上限，因為它僅取決於每個節點的容量和速度。

Trustless Operation: Lightning Network 使用作為對等體運行的節點之間的比特幣交易，而不相互信任。因此，Lightning Network 保留了比特幣系統的原理，同時顯著擴展了其操作參數。

當然，如前所述，Lightning Network 協議並不是實現路由支付渠道的唯一方式。其他提出的系統包括 Tumblebit 和 Teechan。但是，此時，Lightning Network 已經部署在 testnet 上。幾個不同的團隊開發了 LN 的競爭實現，並正在努力實現通用的互操作性標準（稱為 BOLT）。Lightning Network 可能是第一個在生產中部署的路由支付渠道網絡。

Conclusion

我們已經研究了一些可以使用比特幣區塊鏈作為信任平台構建的新興應用程序。這些應用程序將比特幣的範圍擴展到支付範圍以及金融工具之外，以涵蓋信任至關重要的許多其他應用程序。通過分散信任基礎，比特幣區塊鏈是一個平台，將在各種行業中產生許多革命性的應用。