IN PRIVACY
WE TRUST

Official Whitepaper

Current version

1.0 – November 9th, 2018

Table of Contents

About BitcoiNote

BitcoiNote is a decentralized, public blockchain and cryptocurrency project that will make cryptocurrency deals more secure than ever. The private, fast and scalable coin is based on CryptoNight V7 and resolves Bitcoin's issues.

In this whitepaper

BitcoiNote has gone through a period of tremendous growth. User feedback and our ongoing research have inspired the team to map out the future of BitcoiNote. As we listen to user feedback carefully we are convinced that these plans will fulfill the needs and requests of almost all our users.

Over the coming months we are going to introduce a large number of improvements, optimizations, new features and additional platforms. That includes the introduction of the online payment gateway, which will simplify online payments. Wallet improvements will include refactoring of the wallet and the introduction of the wallet on iOS, Android and the web. For miners we are pleased to introduce the merged mining feature and the ability to mine from mobile. Finally the team is working on preparations for the completely new BitcoiNote Express Exchange.

The future of BitcoiNote is almost here! Read along to discover what's coming up.

1 Online Payment Gateway

BitcoiNote strives to simplify online payments. A big step towards this goal is a full-service online payment gateway system. The principle of operation is:

PRINCIPLE OF OPERATION

Initiation

- Buyer initiates checkout process in merchant's website
- Merchant sends **"Create Transaction"** request to Payment Gateway API, receives a transaction ID and stores it for reference. A temporary receiving wallet is created in the process
- Merchant redirects buyer to Payment Gateway's payment page where payment information is displayed. Alternatively: Merchant receives BTCN payment information and displays it to the buyer

Payment

- Buyer uses displayed payment information to send BTCN to Payment Gateway
- Payment Gateway waits for the user's transaction to appear in memory pool (max. 30 minutes)
- Merchant receives first push notification from BTCN Merchant can decide whether this status is enough to process the buyer's order or whether full settlement is required

Settlement

- Payment Gateway waits for the transaction to reach a defined number of confirmations on the blockchain
- Merchant receives second push notification from BTCN
- If not done yet, merchant finishes the user's order
- After one week (refund period), funds sent by buyers are paid out to the merchant's wallet

Registration Process

To register for the Payment Gateway, the merchant needs to create an account with their name, email address and BTCN wallet address. They will receive credentials to access the Payment Gateway API as well as the Merchant Portal.

Checkout Flow

There are two options for the checkout flow.

1 GUIDED CHECKOUT

- Very little implementation work required on the merchant's side.
- Unified user experience: Buyers are aware that they are paying via the BTCN Payment Gateway (with advanced features such as automated refund processing and a private communication channel to the merchant) which may increase the trust level.
- Buyer is directed to a page hosted by the Payment Gateway to complete the payment. On this page, the buyer will be asked for their email address (for purpose of refunding in case of an incorrect payment amount), they will see instructions for executing the transaction, they will see status updates regarding their transaction and they will be redirected back to the merchant's page once the payment is completed or was cancelled.
- Upon email address collection, the buyer can decide whether to share this email address with the merchant or not. If not, the merchant may still contact the buyer until up to one week after the transaction was completed through the Merchant Portal (without seeing their actual email address).
- Merchant can decide whether the payment should wait for full settlement or whether receiving the transaction in the memory pool is enough before the user is redirected back to the merchant's page. In the latter case, the merchant can further specify whether an exception should be made in case of a too-low transaction fee or not.

2

EMBEDDED CHECKOUT

- Full control by the merchant, user never leaves their page.
- Requires more implementation work on the merchant's side.
- Merchant needs to use the Payment Gateway API to get payment and status information, provide the user's email address for refund purposes, display all necessary information to the user, etc.

Registration Process

Regular states

- **Pending**: Transaction was created by merchant (initialized with an amount, a description, required number of confirmations, an optional reference ID and optionally already including the user's email address).
- **Received**: BTCN transaction visible in memory pool (buyer has transmitted transaction, but it's not yet included in a block).

  There is an additional **"at risk"** flag which is set if the BTCN transaction fee is so low and/or the transaction size is so big that the transaction will most likely not be handled by miners anytime soon.
- **Settled**: BTCN transaction received the defined number of confirmations on the blockchain (if zero was specified, this state will be reached immediately after "Received").

Irregular states

- **Cancelled**: Transaction was cancelled – includes a reason: Cancelled by user, cancelled by merchant or timed out (no transaction in memory pool after 30 minutes).
- **Failed**: BTCN transaction was not included in a block within 72 hours and is no longer valid, or the block(s) in which it was included are now orphaned. Includes flags "was received" and "was settled" to indicate whether the transaction had been in the respective states before (to identify the case where, for example, the buyer's order was already processed by the merchant before it became known that the transaction had failed).

Additionally, there is a flag indicating that a transaction has "extraneous funds". In case the buyer didn't provide an email address, or the merchant uses the Embedded Checkout Flow and didn't provide the user's email, the only way to refund the extraneous funds is by the merchant calling the **"Refund"** API with a valid refund address.

There is also a flag **"on hold"** which may be set by the BTCN Payment Gateway staff in case of suspicion of fraud. It will block payout of the transaction to the merchant until this flag has been removed again.

Flags **"partially refunded"** and **"fully refunded"** will be set if a payment was already completed and was then partially or fully refunded. Refunds of extraneous funds or refunds because of a cancelled transaction won't trigger these flags.

In rare cases, there may be an **"unknown error"** flag set which usually requires inspection by support. An example would be a refund transaction being rejected by the BTCN network.

Handling of Payment Errors

Buyer sending too low an amount

If the buyer sent an amount which is too low, Payment Gateway will attempt to handle this case in a way which is transparent to the merchant, if possible. The buyer must send the full amount within the payment window of 30 minutes for the transaction to be processed, even if they split it to multiple payments.

In case of multiple payments, the state of the transaction (received, settled) will be updated only when all the partial payments reached the corresponding status.

In any case, the merchant will receive a push notification about the partial payment, and if a buyer email is recorded, the buyer will receive an email informing them that they still have an amount left to pay, with instructions on how to complete payment.

In case of the Guided Checkout Flow, the user will see on the payment page how much is left to pay and will be asked to submit the remaining amount.

In case of the Embedded Checkout Flow, the merchant must react on the push notification and/or by checking the transaction's status and show the relevant information to the user.

If after the payment window of 30 minutes the buyer did not yet send the full amount, the transaction will be cancelled and the already sent amount will be converted to "extraneous funds". If an email address is recorded, the user will receive an email with instructions how to enter a refund address and get back their funds. If not, the merchant may use the "Refund" API to refund the extraneous funds.

Buyer sending too high an amount

If the buyer sent an amount which is too high, Payment Gateway will process the transaction normally, but the "leftover" amount will be converted to "extraneous funds". If an email address is recorded, the user will receive an email with instructions explaining how to enter a refund address and get back their funds. If not, the merchant may use the "Refund" API to refund the extraneous funds.

Buyer sending additional or duplicate payments

If the buyer sends an additional payment after the transaction was already completed, it will simply be turned into "extraneous funds" and handled like a payment with too high an amount (user is asked for refund address via email and/or merchant may use "Refund" API).
Note that after settlement of a transaction, the corresponding wallet address is monitored only for one week afterwards. If the buyer sends funds to an address which belongs to a transaction older than a week, it will be ignored, and the funds will be lost. The buyer or the merchant would have to contact support to resolve this situation.

Buyer sending a payment with too low a transaction fee

If the buyer submits a BTCN transaction with a very low fee, it is unlikely that this transaction will be processed by miners. In that case, it would sit in the memory pool for 72 hours and then be discarded by the network.
Payment Gateway tries to detect such a case beforehand and will flag the transaction "at risk" if too low a fee is detected. If the transaction is indeed not included in the blockchain and expires after 72 hours, the transaction will reach a "Failed" state.

Blocks becoming orphaned

In rare cases (for example in the event of a chain split), blocks including already confirmed transactions may become orphaned, essentially voiding the transaction. In that case, the transaction will transition to a “Failed” state. The merchant can check the “was received” and “was settled” flags to understand if this means that they lost money by already processing the order and may take steps such as cancelling the user’s order if still possible. Note that the wallet associated with a transaction is monitored only for one week after the transaction was completed.

No payment being submitted

If no payment is submitted at all within the 30 minutes payment window, the transaction will simply enter the “Cancelled” state.

Payment Gateway API

The Payment Gateway API offers the following functionality to merchants:

Create Transaction

Creates a transaction with the given values (amount, description, required number of confirmations, reference ID, buyer email address, postback URL, redirect URLs) and returns the transaction data (most importantly transaction ID, payment information and Status Page URL)

For the Guided Checkout Flow, the merchant would simply redirect the user to the Status Page URL which doubles as payment page.
Once the payment is completed or failed, the Payment Gateway will redirect the user to the specified redirect URLs (if any) with POST fields containing transaction information. Also, when the transaction status updates, the merchant is notified on the specified postback URL in a secure server-to-server call (this is the event which should trigger an order update on the merchant’s side).

For the Embedded Checkout Flow, the merchant would render the payment information to the user, manually update the status displayed to the user and listen for relevant postback notifications in order to continue in the checkout process.

Get Transaction Status

Returns details and status of a transaction (state, Status Page URL, user email if shared, amount left to pay, extraneous funds if any, flags if any, ...).

Get Transaction Status

Cancels a transaction. This may not be possible in all states (especially in the "Failed" state). If the user already paid something, the funds will become "extraneous funds" and Payment Gateway will attempt to process the refund with the buyer (or require the merchant to use the "Refund" API if no buyer email address is recorded).

Refund

Initiates a refund. The execution of the refund is only possible within the refund period and after the corresponding incoming BTCN transactions were confirmed. Note that for all refunds, the transaction fee will be deducted. This has multiple modes:

Refund address
With a refund address specified: Immediately issues the refund to that address (or in case the original BTCN transaction is not confirmed yet: as soon as the transaction was confirmed).

Without a refund address specified (only possible if a buyer email is recorded): Asks the buyer for a refund address via email. Once the buyer set their refund address (must happen before end of refund period), the refund is issued.

Amount

With "refund extraneous" mode: Refunds the extraneous funds – a refund address should be specified (because if the Payment Gateway had known the buyer's email, at that point the refund would already have been initiated automatically). This is usually needed in case the Payment Gateway doesn't have a buyer email recorded and the merchant needs to handle a case of incorrect payment amounts.

With "refund all" mode: Refunds all funds currently held by Payment Gateway.

With specified amount: Refunds the specified amount (up to full amount paid, including any extraneous funds). Specified amount includes the transaction fee of the refund (the buyer receives the specified amount minus the fee).

Refund Period

The refund period is one week. The following activities are possible only during this time:

- The merchant may initiate communication with the buyer through the Merchant Portal (unless the buyer has agreed to share their email with the merchant or the buyer has already contacted the merchant previously)
- The merchant may initiate a refund through the Merchant Portal (after the refund period, the funds will already be paid out to the merchant, so refunds would need to be handled directly between the buyer and the merchant)
- The Payment Gateway processes refunds for overpayment or cancelled transactions with the buyer
- The Payment Gateway keeps monitoring the temporary wallet of the transaction for incoming extra payments and processes refunds of those with the buyer

Note that all refunds are only possible once the original BTCN transaction(s) reached enough confirmations to be spendable again.

Email Communication with Buyer

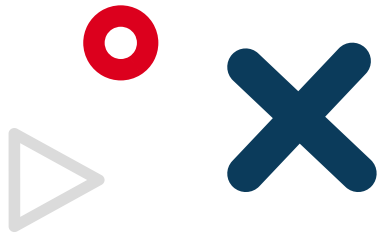
The buyer's email address is collected during the Guided Checkout Flow, and the buyer can decide whether they want to share their email address with the merchant or not. In the Embedded Checkout Flow, the merchant can optionally provide the buyer's email address to the Payment Gateway.
The email address is used for the following purposes:

- Informational email with URL to the Status Page
- Informational email when the transaction was completed
- Alert emails in case an action is required:
  - Not enough paid: Asks the user to submit the remaining amount
  - Too much paid: Asks the user to click a link to enter a refund address
  - Refund initiated by merchant without refund address: Asks the user to click a link to enter a refund address
- Allowing the merchant to contact the buyer:
  - If the user did not agree to share the email address with the buyer during the Guided Checkout Flow, the merchant may contact the buyer only during the refund period (one week), and the communication is proxied through Payment Gateway to keep the privacy of the parties. This way, the buyer can be sure that their email address won't be abused for spam by a rogue merchant.
  - Once the buyer replied to the merchant, the merchant may keep replying to the buyer for an unlimited time in the future. The buyer may always click a link in the emails coming from Payment Gateway to block further communication from the merchant regarding this transaction.
- Allowing the buyer to contact the merchant:
  - The buyer can use a link in the Status Page to contact the merchant. This is done through a contact form which will proxy communication through Payment Gateway to keep the privacy of the parties.
  - Once the buyer contacted the merchant, the merchant may keep replying to the buyer for an unlimited time in the future. The buyer may always click a link in the emails coming from Payment Gateway to block further communication from the merchant regarding this transaction.

Merchant Portal

The merchants have access to the Merchant Portal which allows them to perform the following actions:

- See recent transactions, their status and history
- Contact the buyer:
  - If the Embedded Checkout Flow was used, the email of the buyer is displayed if set.
  - If the Guided Checkout Flow was used, the email of the buyer is only displayed if the buyer agreed to share their email address with the merchant during checkout. If not, the merchant may use a contact form to contact the buyer nonetheless, but only until the end of the refund period (one week).
- Refund transactions within the refund period (one week)
- Change their payout wallet address and personal details
- Get reports about their transactions over time
- Download a transaction list as CSV file
- Contact support



Status Page for Buyer

Every transaction has a Status Page URL associated with it. This doubles as URL to the payment page of the Guided Checkout Flow. In case the buyer closes the page (especially if multiple confirmations are required for settlement and the user must wait for them), they may use this URL to get back to that page later. If the buyer entered their email address, they get this URL sent via email as well.

In the Embedded Checkout Flow, it's up to the merchant to decide whether they want to display this URL to the user or not. They may also create their own status page using the "Get Transaction Status" API.

Online Shop Integration

Plugins for popular online shop platforms (such as WooCommerce) will be provided for simple integration on the merchant's side.
Merchants just need to install the corresponding plugin for their platform, set their API key in the plugin settings and enable it as payment method.

2 Wallet improvements

Wallet Refractoring

To facilitate faster improvement cycles and support for more platforms, the BitcoiNote wallet is refactored and built from scratch using TypeScript.

The choice of TypeScript over JavaScript has been made to reduce the likelihood of bugs and errors, since TypeScript allows compiler-side type safety like what the C++ compiler provides to the current BitcoiNote wallet code.

The usage of modern web technology for the wallet will greatly speed up further developments and simplify error tracking, and it should enable more developers to participate in improvement of the BitcoiNote software on GitHub, as this technology is more easily accessible for developers nowadays and should present a lower barrier of entry.

For cryptographic functions, it still uses the relevant functions of the CryptoNote core which are written in C and partly C++. Depending on the platform, this may be implemented using a native compiled module or

transpiled code using Emscripten.

The refactored wallet will be able to load wallet files from the CLI and old GUI wallet software, but it will use a different format from then on. It is still possible to export the wallet keys, so they can be imported in the old GUI wallet if required.

Note that the refactored wallet would still require the use of a BitcoiNote Core node instance (currently there is a node integrated into the C++ GUI wallet). There are no plans to refactor the BitcoiNote Core as of now. The use of a Core node (either local or remote) will become optional once Light Wallet Support is added (see below).

New Wallet Platforms

The refactored wallet allows easier porting to new platforms. The following platforms will be supported:

- Windows, Linux and macOS (using Electron)
- Android and iOS (as packaged local web app)
- Browser (using Emscripten for CryptoNote core functions)

Note that mobile and browser wallets will be possible only after Light Wallet Support has been added (see below).

Of course, running a wallet in a browser is not secure, as there are many more possible attack vectors for rogue entities than using a desktop or mobile wallet (for example: rogue browser extensions, man-in-the-middle attacks, hacked web servers, etc.). Therefore, the browser-based wallet is designed for testing only and will display a big warning once the balance exceeds 1000 BTCN, asking the user to create a new wallet on a different platform and to transfer all their funds to the new wallet.

Light Wallet Support

Until now, it was impossible to use BitcoiNote on a mobile device. Furthermore, the GUI wallet would require downloading the whole blockchain to the computer before it could be used, as well as continuous synchronization with the network, downloading all new blocks.

There is already the option to set a “remote node” in the wallet settings. This would then not use the embedded node in the GUI wallet and instead talk to a node somewhere on another server, eliminating the requirement of having the whole blockchain stored on the computer. However, this still presents a few problems:

- You would need to have a remote node available in the first place (which needs to be run by someone and doing so costs them network traffic)
- The wallet still needs to download all new blocks and synchronize with the network continuously, even though those blocks won’t be stored on your computer
- If you would have to reset your wallet or import wallet keys (both of which imply a reset of the last known block stored in your wallet), the wallet would need to download all blocks in the blockchain yet again to scan them for transactions regarding your wallet.

Other well-known cryptocurrencies such as Bitcoin or Litecoin solve this using “light wallets”. There is a de-facto standard called “Electrum” for such light wallets. It works by having a service which synchronizes with the blockchain and scanning it for relevant transactions for you, so the light wallet only needs to ask this service for new transactions regarding its address since a certain point in time instead of downloading all relevant blocks and scanning locally. For submitting a transaction, the wallet would sign it locally and send the signed transaction to the online service for broadcasting.

The reason this works for other cryptocurrencies such as Bitcoin and not for BictoiNote is the privacy feature of CryptoNote. In Bitcoin, all the transactions are public – everyone can see a transaction's amount, sender(s) and recipient(s) on the blockchain. This means that the Electrum server can efficiently index transactions by their related addresses and allow returning a list of transactions related to a certain address, and it can keep track of each address' balance (like the Bitcoin Block Explorer allows when searching by address).

However, BitcoiNote's privacy feature means that only the owner of an address knows that a certain input or output in a transaction is related to their address. This is done through ephemeral keys:

To generate an ephemeral key used to send money to:

- *The sender generates a new key pair, which becomes the transaction key. The public transaction key is included in "extra" field.*
- *Both the sender and the receiver generate key derivation from the transaction key and the receivers' "view" key.*
- *The sender uses key derivation, the output index, and the receiver's "spend" key to derive an ephemeral public key.*
- *The receiver can either derive the public key (to check that the transaction is addressed to him) or the private key (to spend the money)*[1].

This implies that knowledge of an address owner's view key is required to link a transaction to that address. For this reason, it is impossible to index the blockchain by address, and this makes it impossible to implement Electrum for BitcoiNote – the wallet always needs to download the blocks in order to scan them locally without exposing the view key.

BitcoiNote solves this problem with a compromise: To use a "light wallet", the user must share their view key with the owner of the light wallet service.

1 CryptoNote Developers: „crypto.h", https://github.com/cryptonotefoundation/cryptonote/blob/8edd998304431c219b432194b7a3847b44b576c3/src/crypto/crypto.h#L199

Note that sharing the view key only allows identifying which transactions belong to the address. It does not allow spending coins. The spend key still stays on the device of the wallet user, and the wallet only transmits already-signed transactions to the online service for broadcasting. The security risk is therefore limited.

The light wallets make use of a "Ledger Agent" which is an online service providing the mentioned functionality of scanning the blockchain on the wallet's behalf.

The wallet opens a session with the Ledger Agent via SSL and transmits the user's view key. The Ledger Agent's SSL certificate is verified first to prevent man-in-the-middle attacks. The Ledger Agent stores the view key in memory only until the session is closed, minimizing the possible consequences of a data breach.

Once a session was opened, the wallet can request transaction updates starting with a certain block hash. The Ledger Agent then sends information about all transactions since the specified block which relate to this wallet, as well as relevant transactions in the memory pool. In case the specified block was orphaned, the Ledger Agent requests previous block hashes known to the wallet to find the point at which the chains split and then update the wallet from that point onwards. If the session is kept open, the Ledger Agent continuously pushes updates to the wallet when transactions come in or change their status.

To submit a transaction, the wallet signs the transaction on the device and then transmits the signed transaction via the open session to the Ledger Agent. The Ledger Agent validates the transaction and broadcasts it to the network.

Since the case where the Ledger Agent must scan the full blockchain for one wallet's request should be eliminated (and caching of data is avoided due to security of the view key), it becomes very important that the creation date (more exactly: block height) of the wallet is known. This information is currently stored in wallet files but often unavailable or unreliable, for

example when keys were imported, or a wallet had to be reset. The light wallets will therefore ask for this information upon importing keys or when loading a wallet file without creation date information, and the wallets will make it simple to get this information from existing wallet files.

To transfer a wallet from desktop to mobile, a QR code can be displayed which contains the wallet keys and creation date. For other cases, the user is required to manually enter the keys and creation date displayed on the source device into the target device. Transfer of an actual wallet file is still preferred because this eliminates the requirement to resync the wallet since the creation date as all transactions are already stored in the wallet file.

Multi-Wallet Support

The refactored wallet application can handle multiple wallets simultaneously, tracking all their transactions and balances. It allows to display them all at once as if they were one wallet, or separately. When sending a transaction, the user can either choose to take funds from all wallets as the application sees fits, or specific amounts from specific wallets.

Headless Mode

The refactored wallet can also be used for automating the wallet operations in a headless mode, through an API. It basically combines the functionality of the CLI wallet and the existing GUI wallet.

One-Click Payment Setup

Currently, it is inconvenient to do payments to users whose address is not already in one's contact list, such as payments to online merchants. It requires manually copying the merchant's address into the wallet, entering the correct amount and in some cases also copying the required payment ID into the corresponding field in the wallet. If such a payment was done

incorrectly (for example by forgetting to set the payment ID), it can often be cumbersome to recover funds from the recipient. In case of an incorrect address, it may even be impossible.

The solution is the One-Click Payment Setup system which allows to fill in all the requirement information for a payment using either of the following options:

1. Clicking a link. This is the simplest way and works by simply clicking a link in the browser which then opens the wallet and presents the "send transaction" page with all values already pre-filled, waiting for the user to click "Send".
2. Copying a link into the wallet. This may be required if directly "opening" the link in the wallet is not supported by the browser or otherwise not working. The user would then click a new "From Link" button in their wallet's "send transaction" page and copy the link into a field, which will then pre-fill the fields.
3. Scanning a QR code (for mobile wallets only) which contains a link. The user would click a new "Scan Code" button in their wallet, scan the code (for example displayed on a website) and see all fields pre-filled on their mobile wallet, waiting for the user to click "Send".

All these options are using a URI scheme "btcn:" with the following format:

```
btcn:[address]
?amount=[amount]
&id=[payment ID]
&label=[recipient name]
&fee=[TX fee]
&message=[description]
```

Example:

```
btcn:N3SVzasFXqJCDMWv9Py45K7gQNLXXgihFgxQyY2ruqBjRLsCkt6zUwDU
3Xb2NDMD6W9ewohbaYMtUWrFGfEKvxMjMub1GAZ?
amount=19.28315&label=Best+Exchange
```

All parameters except for the address are optional. The recipient name is filled into the corresponding field in case the user wants to save it to their contacts (it's not part of the transaction). The description may be displayed by the wallet, but this is optional. Wallets may make it optional to the user whether to use the provided fee or not.

This format is designed to resemble the Bitcoin URI scheme defined in BIP 0021 [2]. It is in fact compatible (except for the scheme name itself, of course), as optional parameters are allowed in BIP 0021.

Payment Descriptions

Now, the only way to add an additional identifier to a payment is the payment ID. Otherwise, it is very hard to understand what a certain payment was used for (especially for incoming payments since their sender is not known). The payment ID is a 256-bit hexadecimal number, which makes it hard to use for transporting human-readable information. The payment ID is also public, anyone can see a transaction's payment ID.

There are several new features which are designed to improve this situation:

Human-readable Payment IDs

Since the payment ID is public, this should not be used to include sensitive information such as usernames, however merchant names and invoice numbers should be fine in most cases (as the sender and receiver address are still private).

The format is simple: If a payment ID's first five bytes equal "FF FF 55 AA 00", the rest of the ID is a string of up to 36 characters encoded as DEC SIXBIT [3]

2 Nils Schneider, Matt Corallo: BIP 0021 „URI Scheme", https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki

3 Digital Equipment Corporation, „rabbit": DEC/PDP Character Codes, http://rabbit.eng.miami.edu/info/decchars.html

(6 bits per character), padded with zeroes (=spaces). This means that only upper-case accent-less latin characters, numbers, spaces and some punctuation characters are allowed (ASCII range 32-95).

Example:

```
FFFF55AA00A25B2CBC10000000000000000000000000000000000000000000000000
```

After the “magic header bytes” in red, the green part is the encoded data. In binary, it would be:

```
10100010 01011011 00101100 10111100 00010000 …
```

Regrouping it to groups of 6 bits gives:

```
101000 100101 101100 101100 101111 000001 0000…
```

Which decodes as:

```
HELLO!
```

Wallets and “btcn:” URIs would support both formats: a 64-character hex string and an up-to-36-character string of characters in ASCII range 32-95 (encodable in DEC SIXBIT).

Encrypted Payment IDs

In field 0x02 of the transaction extra fields (see CNS005 [4] for description of the extra fields), the payment ID is stored. Currently, it is stored as one type byte plus the data. The only valid type is 0x00.

4 Albert Werner, Montag, Prometheus, Tereno, CryptoNote: CryptoNote Standard 005, https://cryptonote.org/cns/cns005.txt

Encrypted payment IDs will be stored with a type of 0x01, like it is done in Monero [5], or 0x02 (see below). It would be visible only to the sender and receiver. To decrypt such a payment ID, the same algorithm is used as in Monero [6].

However, BitcoiNote additionally supports a 256-bit encrypted payment ID (not just the 64-bit ID Monero supports), so that the human-readable IDs defined above can also be used in an encrypted manner. For this type of payment ID, the first byte of extraNonce would be 0x02, while for the 64-bit ID, 0x01 would be used.

Of course, searching by encrypted payment ID in a block explorer is possible only by specifying the encrypted version of the ID instead of the cleartext version.

Encrypted Payment IDs

To simplify identifying payments and remembering what they were used for, the BitcoiNote wallet receives a feature which allows adding a free-text comment to any payment, incoming or outgoing. This comment is private and is only stored locally in the wallet file. A consequence of this is that this data is not shared when using a wallet on multiple devices or recreating a wallet from keys or QR code.

A syncing support through Dropbox or Google Drive across multiple devices may be added in the future to solve this problem.

5 The Monero Project: „tx_extra.h", https://github.com/monero-project/monero/blob/1a4298685aa9e694bc555ae69be59d14d3790465/src/cryptonote_basic/tx_extra.h

6 MyMonero.com, „cryptonote_utils.js": https://github.com/mymonero/mymonero-core-js/blob/92734c193a40b89d55822d62a2a49f39200e1e5e/cryptonote_utils/cryptonote_utils.js#L2557

Paper Wallet Support

To further secure funds, it's possible to use a paper wallet. The principle of a paper wallet is that the spend key never enters an insecure environment. There are two kinds of secure environments in play here:

1. An offline computer, running a separate operating system from a secure live CD, read-only USB flash drive or similar medium.
2. A piece of paper.

To create a paper wallet, the offline computer environment needs to be used to run the paper wallet application. The "Create Paper Wallet" option would be used to create a new wallet in an encoded format which can be written on paper. The paper wallet application will ask for an optional password to encrypt the code with so that a thief couldn't use the code written on paper to access the wallet without also knowing the password.

The code contains the two sections of data:

- Viewing section
  - View key pair
  - Creation date
  - Checksum
- Spending section
  - Spend key pair
  - Checksum

This data is encoded as alphanumeric code in blocks of 5 characters, using numbers and upper-case letters except for 1, 0, I, O and Q (to prevent mistakes in reading), optionally encrypted with the password specified before.
The user should write down or print this information.

To use the paper wallet, the user would go back to their normal environment and use the option "Import Paper Wallet" in their wallet, entering only the viewing section. The user can now view their wallet balance and transactions like normally.

When attempting to send a transaction, the wallet would ask the user to save the transaction data to a file, as it cannot sign the transaction without a spend key.

The user transfers this file to the secure environment (e.g. on another USB flash drive), runs the paper wallet application in the secure environment and uses the option “Sign Transaction”. They would then select the file which they transferred. They would get asked to save the now signed transaction to another file.

The user transfers the signed file back to their normal environment and uses the “Submit Signed Transaction” option in the wallet software to select the signed file copied from the secure environment and broadcast their transaction to the network.

3 Mining Improvements

Merged Mining

BitcoiNote receives merged mining in a way like Fantomcoin's implementation [7].

It makes use of extra field 0x03 in the transaction (merged mining tag).

For merged mining, the miner builds a block for both chains (BTCN and parent) in a way in which the same hash calculation can secure both blocks. If a block is mined which meets either chain's difficulty requirement, the block is assembled and submitted to the correct blockchain (or two blocks are submitted if both requirements are met).

First, the miner must assemble a set of transaction for both the BTCN and the parent chain. They then assemble the final BTCN block and hash it. A transaction with this hash (valid for the parent chain) is then created and added to the parent block's header.

If a miner solves the hash challenge at the difficulty level of the parent chain, the parent block is built and sent to the network (with the parent network ignoring the BTCN hash).

7 FantomCoin developers, original commit with most of the merged-mining-related changes: https://github.com/amphibia/fantomcoin/commit/cdd89a65802387ae5d2efc5ccac77b621b9fae0b

If a miner solves the hash challenge at the difficulty level of the BTCN chain, the BTCN chain is built, including the BTCN transactions, BTCN block header but also parent block header and the hash of the other transactions in the parent block. This is then submitted to the BTCN chain which accepts it as “auxiliary proof-of-work”.

This allows more effective use of hashing power and should draw more miners to BitcoiNote since they can mine BitcoinNote and another coin “for free” at the same time.

Supported coins would be those which use the same hashing algorithm (CryptoNight-Lite V1), currently these are: TurtleCoin, BBSCoin, AEON, BitSum, Iridium, Triton, Worktips

Mobile Mining

To ease access to mining, a mobile miner application is created for Android.

The application is planned to be based on xmrig [8], an open-source software. It must be modified for support of the CryptoNight-Lite V1 algorithm used by BitcoiNote, as xmrig is designed for Monero which uses the CryptoNight V1 (not Lite) algorithm instead.

The application has a settings and status UI and runs and supervises the xmrig binaries in the background. Only ARM (32-bit and 64-bit) architectures are supported, so this application will not work on tablets based on Intel architectures or others.

8 xmrig Developers, xmrig on GitHub: https://github.com/xmrig/xmrig

4 Instant Express Exchange

As a new user, it's currently a bit cumbersome to get BTCN, because you need an account on an exchange, deposit BTC first, then buy BTCN, then withdraw.

To improve this situation, a CoinSwitch-like experience in form of an "Express Exchange" is offered: A website on which any user can exchange BTC to BTCN without account creation, just by providing a BTCN wallet address and sending a BTC payment. The Express Exchange internally interacts with an exchange and buys BTCN on the fly.

On the Express Exchange, the user can specify how much BTC they want to spend, and the website will calculate the amount of BTCN they would receive at the current market situation. It is based on the available sell orders on the market, as well as various fees.

This value is not locked however, it will be recalculated once the BTC funds are received and confirmed, so the initially displayed value is an approximation. This is done because the Express Exchange works with an exchange to buy the user's BTCN, and the exchange rate of BTCN-BTC may be volatile, so it cannot be predicted whether the price will stay sufficiently stable between the initiation and the settlement of the purchase.

Once the user confirms the amount of BTC they want to spend, payment instructions are displayed. The user can, for example, use a QR code to do the BTC transfer with their wallet.

After the BTC payment was confirmed, the Express Exchange recalculates the amount of BTCN the user is eligible to receive at that point in time and issues the relevant BTCN Buy orders on an exchange. Then, the service triggers a withdrawal from the exchange directly to the user's BTCN wallet.

BITCOINOTE WHITEPAPER