CRYPTOCURRENCY INVESTIGATIONS WITH MALTEGO

Tips and Tricks for Efficient Analysis and Visualization of Bitcoin and Ethereum Movement

MALTEGO

whitepaper

Contents

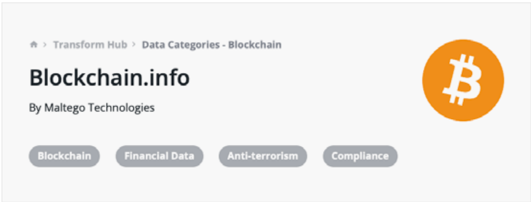
Part 1
Introduction

With the skyrocketing popularity and high adoption of cryptocurrencies, as well as increased interest from the general public, crypto frauds and scams are unfortunately also on the rise today. These illicit activities often entice gullible people on social networks into shady and fraudulent schemes which at the end may cost them a lot of money.

Example Cases

Maltego is a powerful tool for link analysis and investigation, and in real life cases it can be used together with other specific tools (both free and proprietary) which help an investigator in obtaining additional data and building a complete picture.
In this article we will highlight different methods which can be used for investigating crypto fraud and scam schemes, and following cryptocurrency network activity. We will use only free Maltego Transforms (namely, Blockchain.com Transform set, and also complement the investigation by using some free external tools to get additional data related to investigation.

We will go through the investigation process of a few cases: investigating activity and flow of funds related to two scammer's bitcoin addresses, and visualizing activity of a fraud related address in Ethereum network.



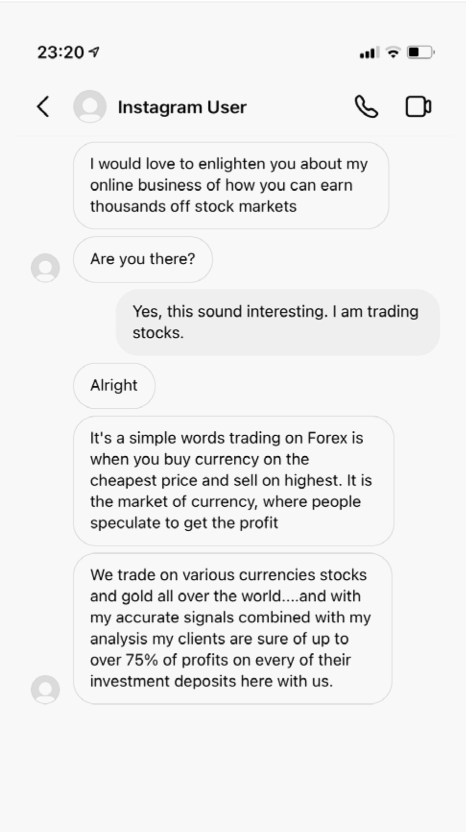
Note: Crypto frauds and scams investigations can be performed from different perspectives. The ultimate goal of online fraudsters is cashing out the stolen funds. In the cryptocurrency world, various exchange services are used for that. This said, the first and foremost problem to investigate is "following the money": basically, this means tracing funds until their final destination in an attempt to take a legal action, as licensed crypto exchanges are subject to regulation and compliance with KYC / AML procedures.

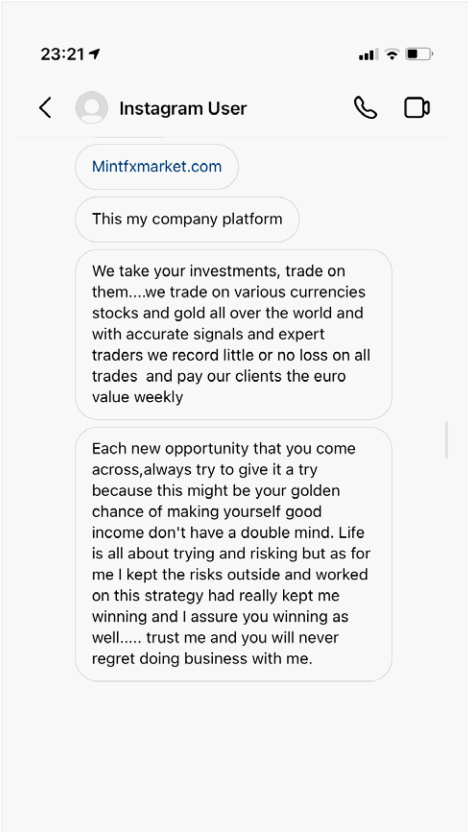
Bitcoin Investigation Concepts

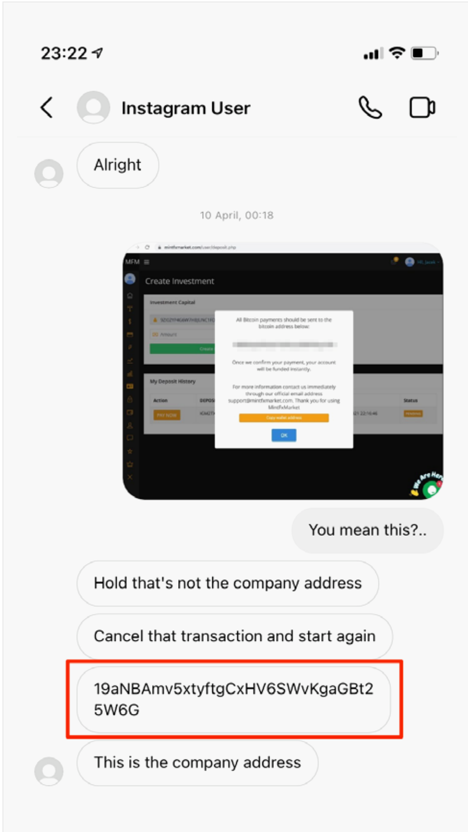
Some of the readers have probably met this kind of scam activity on Instagram and other social networks. A stranger approaches you in the chat and talks you into “investing” in Bitcoin trading or mining, with promises of immediate sky-high returns.

Actual schemes can vary, but at the end they all boil down to talking a victim into sending bitcoins to some Bitcoin address controlled by the attacker. In our case, a scammer uses Instagram chat to encourage victims to register on a website they share, which then requires the victim to send a “deposit” to a specified Bitcoin address. It’s very easy to guess that a victim will never see their money again once they send it.

Below are some screenshots from a chat as it happened on Instagram, where a scammer reveals a Bitcoin address to which a victim is expected to send money:







As the first example, we will investigate this scam address. The actual scam attempt happened in April, so we need to investigate April activity and money flow related to the scammer’s Bitcoin address.

Aims of the Blockchain Investigation

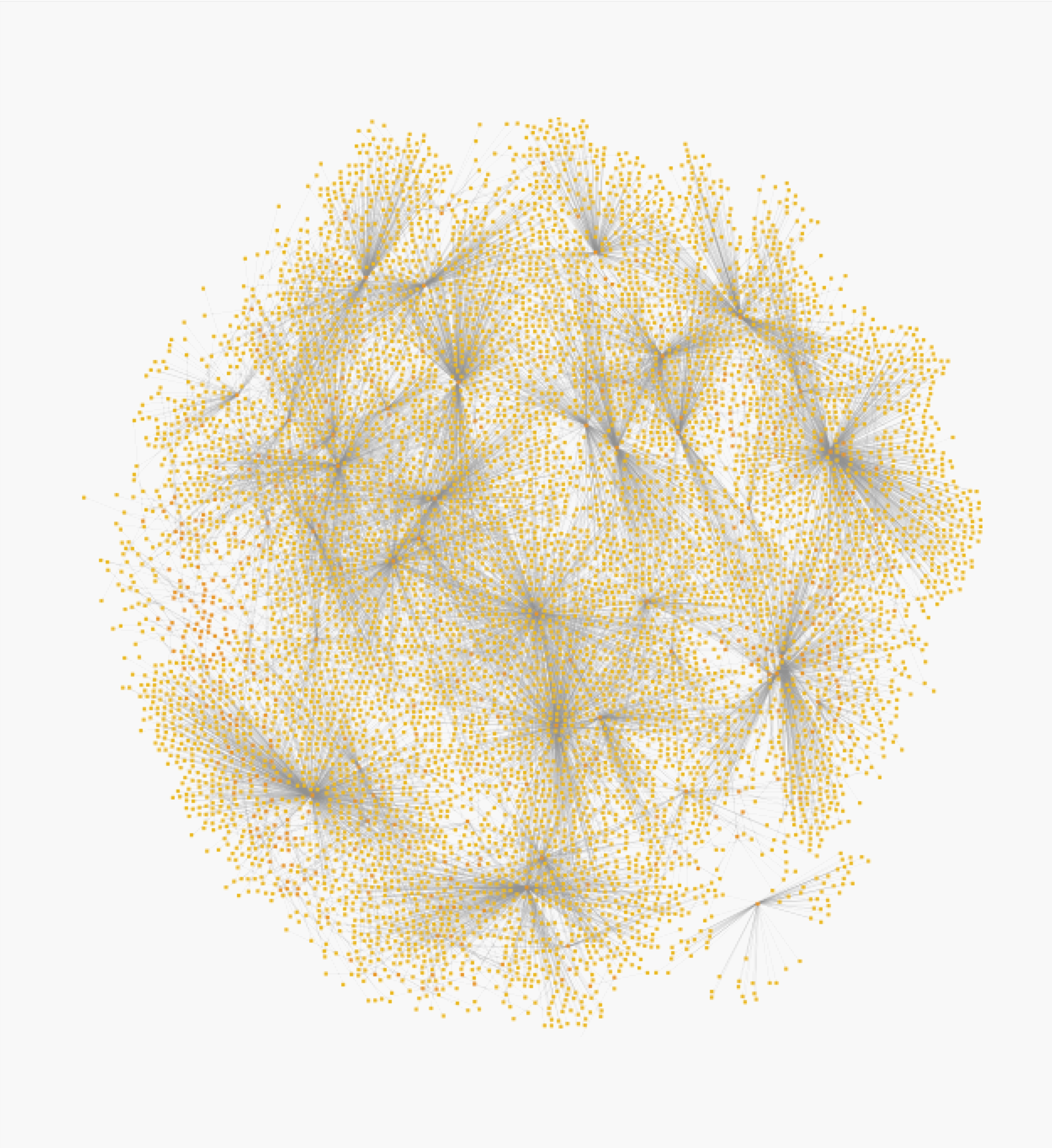
As we already mentioned, the main objective an investigator must keep in mind is to follow the money and understand the relation of the investigated scammer's address with key Entities of interest on the Bitcoin network, like crypto exchanges which are used for cashing out the funds.

Below is an example of one graph we came up with, which shows a logical connection between the investigated wallet and the Binance crypto exchange, which in our case can be a place for cashing out the funds originated from scammer's address:

In the following chapters we will follow a step-by-step process to see how this certain graph is built starting from a single Entity, in our case a scammer's Bitcoin address.

We will also learn how to work with large graphs in Maltego (containing thousands of Entities) representing Bitcoin transactions and narrow down the most interesting and important Entities.

How do you think we will be able to extract valuable information from the following graph with over 8000 Entities?



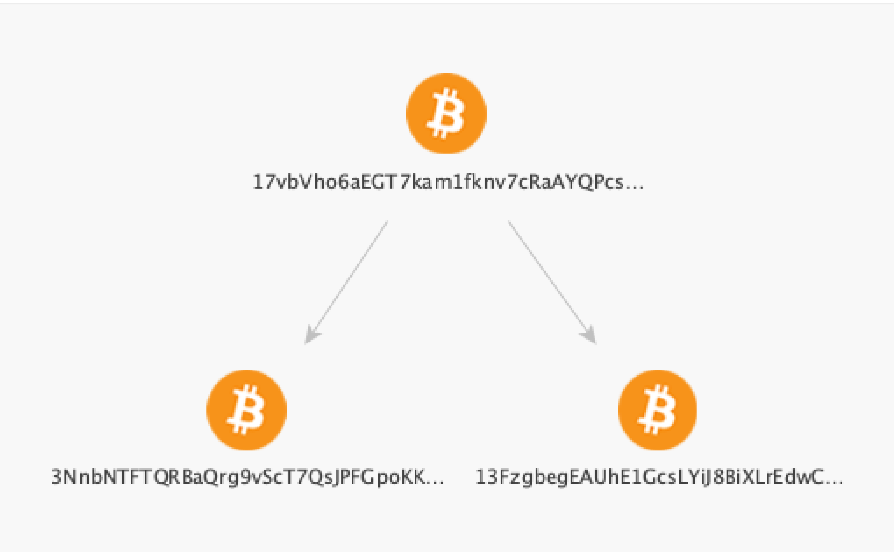
Transaction-based Network Concept

One must have noticed that Bitcoin transactions presented in the Maltego graph follow the same concept: Bitcoin address Entities are connected with each other through transaction Entities.

One may ask, why bother putting Bitcoin transactions and addresses on the graph using different Transforms? Can we not make it simpler and trace the Bitcoin movement directly from address to address? Actually, no, and there's a reason for that.

Remember that the Bitcoin network is transaction-based, and it is essentially a *transaction* which is recorded into the Blockchain. Each transaction may have multiple input and output addresses, so nominally any Bitcoin address (unlike for example, a bank account) does not hold any Bitcoins. Its 'balance', i.e. the spending ability, is only defined by the sum of all input transactions. Let's take a look at a very simple example to illustrate this concept.

Consider a single Bitcoin address to which we apply **To Destination Addresses** Transform:

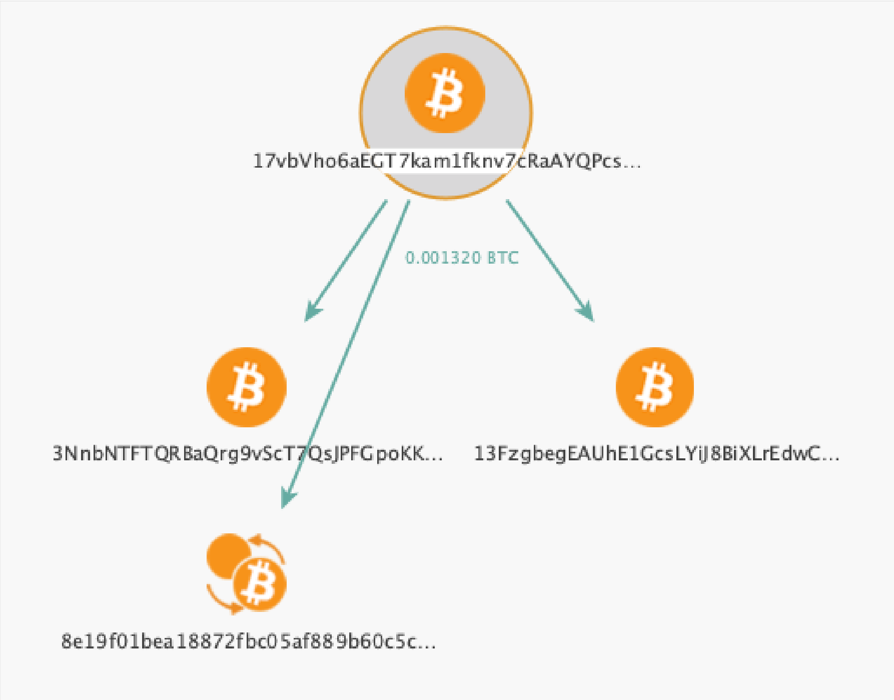


This graph shows that address `17vbV...` has sent some funds to addresses `3NnbN...` and `13Fzg...`, but we don't know the exact amounts sent, and we also don't know if they were sent at the same or different times.

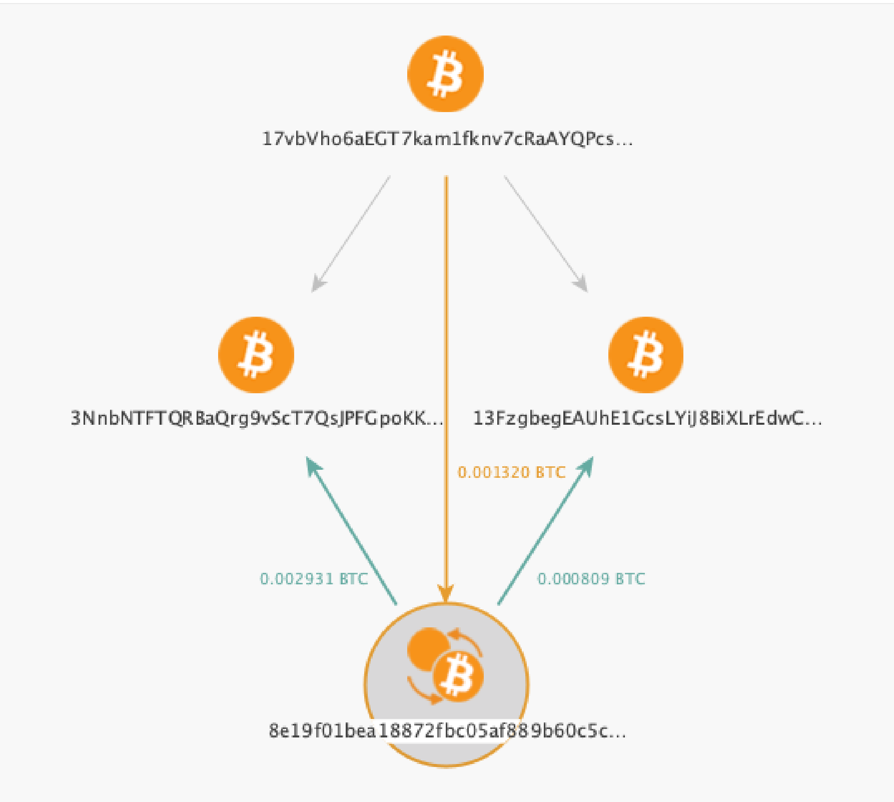
Now let's actually see how we can add transaction Entities to this graph.

In the first step we run the **To Outbound Transactions [Block-Chain.com]** Transform on the `17vbV...` address and get one transaction Entity, which is `8e19f...`:

TIP: In the following text you will find abbreviated bitcoin addresses; e.g.: full address for "`17vbV...`" is `17vbVho6aEGT-7kam1fknv7cRaAYQPcsQFm`.

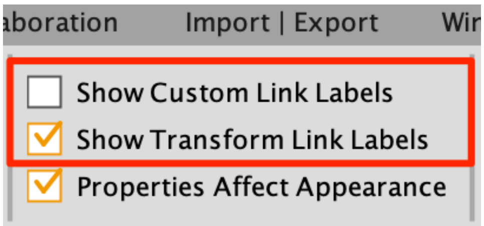


Next, we run the **To Destination Addresses [BlockChain.com]** Transform on this transaction:

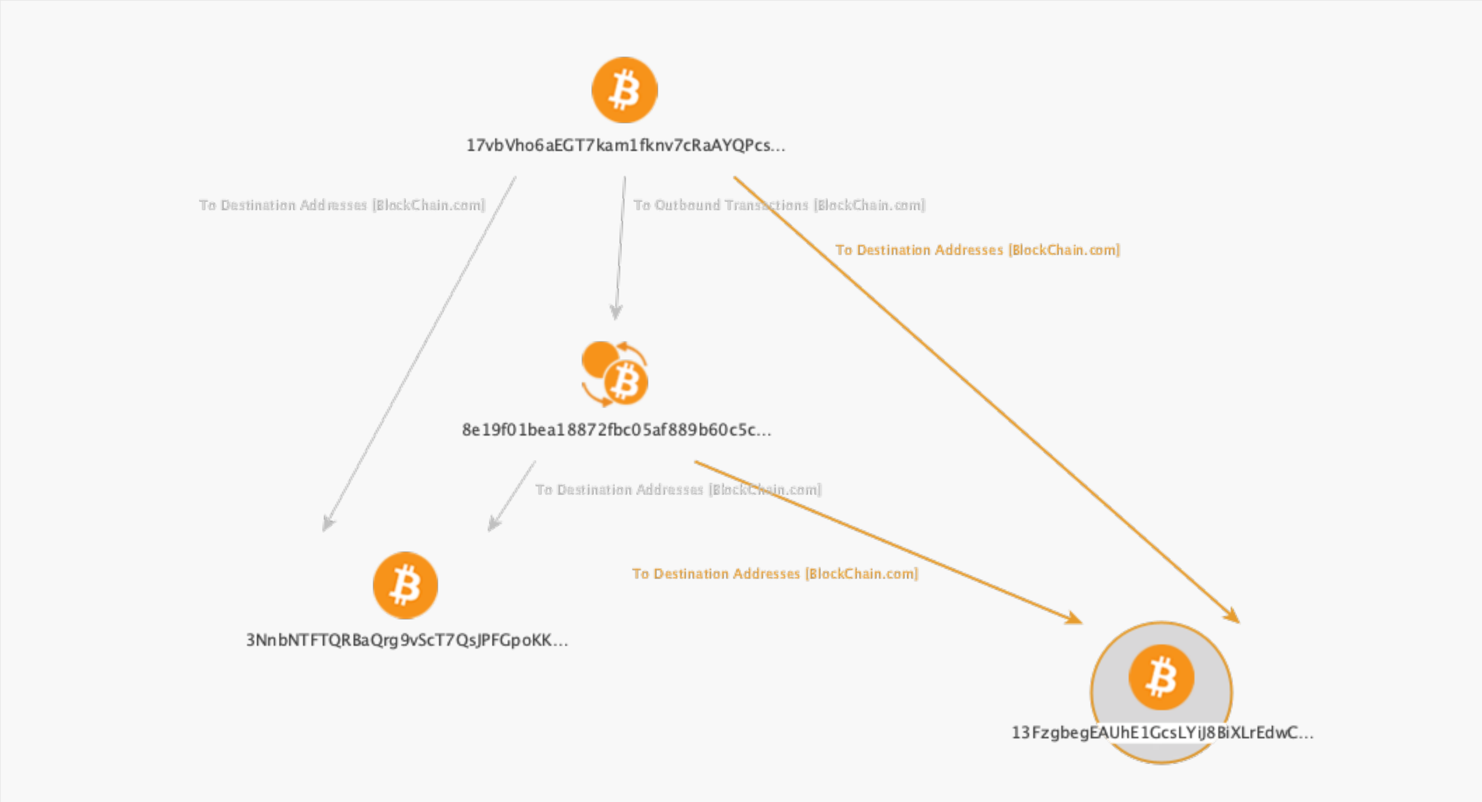


Voila! We now have a transaction on the graph which is connected to the same destination addresses, but we also see the amounts (shown as links labels) and understand that Bitcoins were moved in the course of a single transaction with one input `17vb...` and two outputs `3NnbN...` and `13Fzg...`

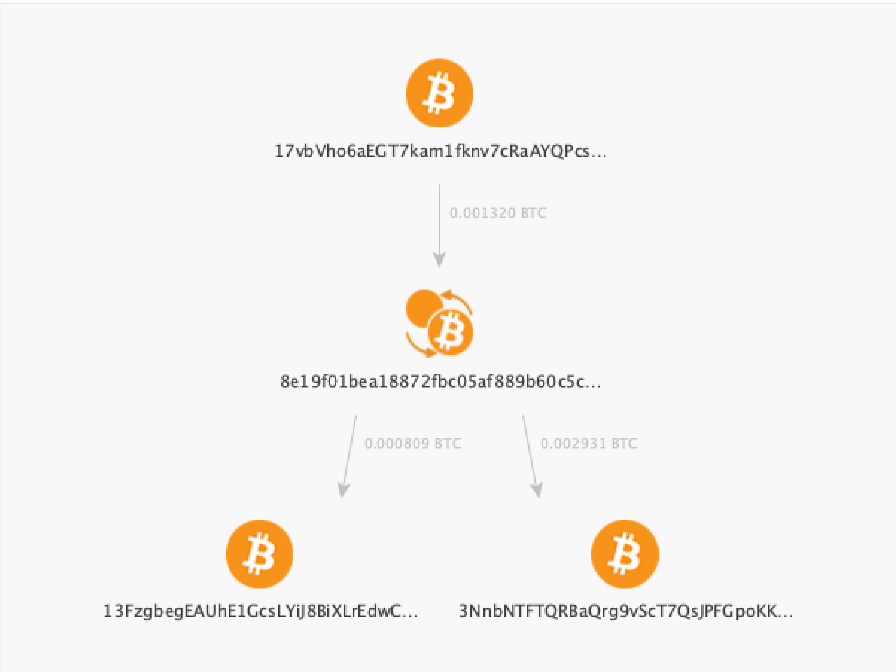
TIP: We can switch the links labels from the custom format which shows transaction amount (uncheck **Show Custom Link Labels**) to Transform names (check **Show Transform Link Labels**) in **View** menu:



Using this tip, the logic of using blockchain Transforms in a proper way (address › transaction › address) is illustrated even more clearly by the corresponding labels:



We don't want to overload the picture with redundant information, as in real life cases we actually don't need to use any 'Address › Address' links on the graph. Therefore, the final correct graph view would be this:



NOTE: For all following graphs in the next parts of this whitepaper, we will follow the same principle of tracing flow of funds: **Address › Transaction › Address.**

Part 2
Building Graphs of Bitcoin Transactions

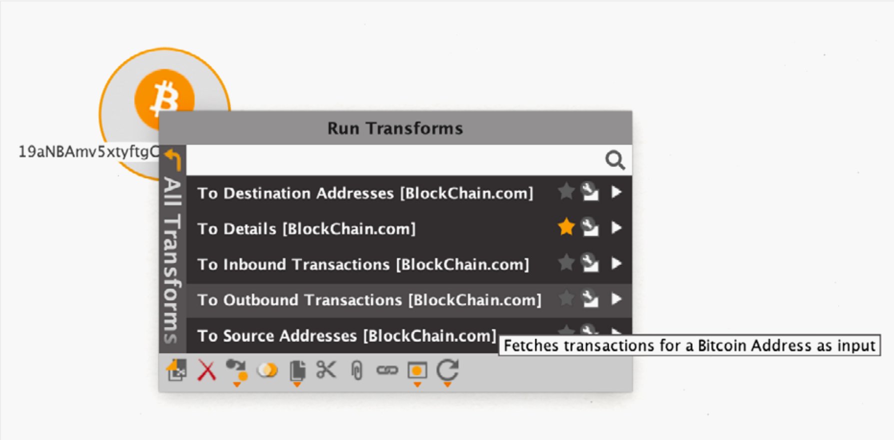
We have already addressed the basic steps involved in building Bitcoin transactions graphs in an earlier article. In this article, we will consider some of the best practices and helpful tips to make an investigation more complete and efficient. We will also utilize some of Maltego's functionality to enhance and speed up the analysis process, which can be helpful in case of a big amount of available data.

Following the money

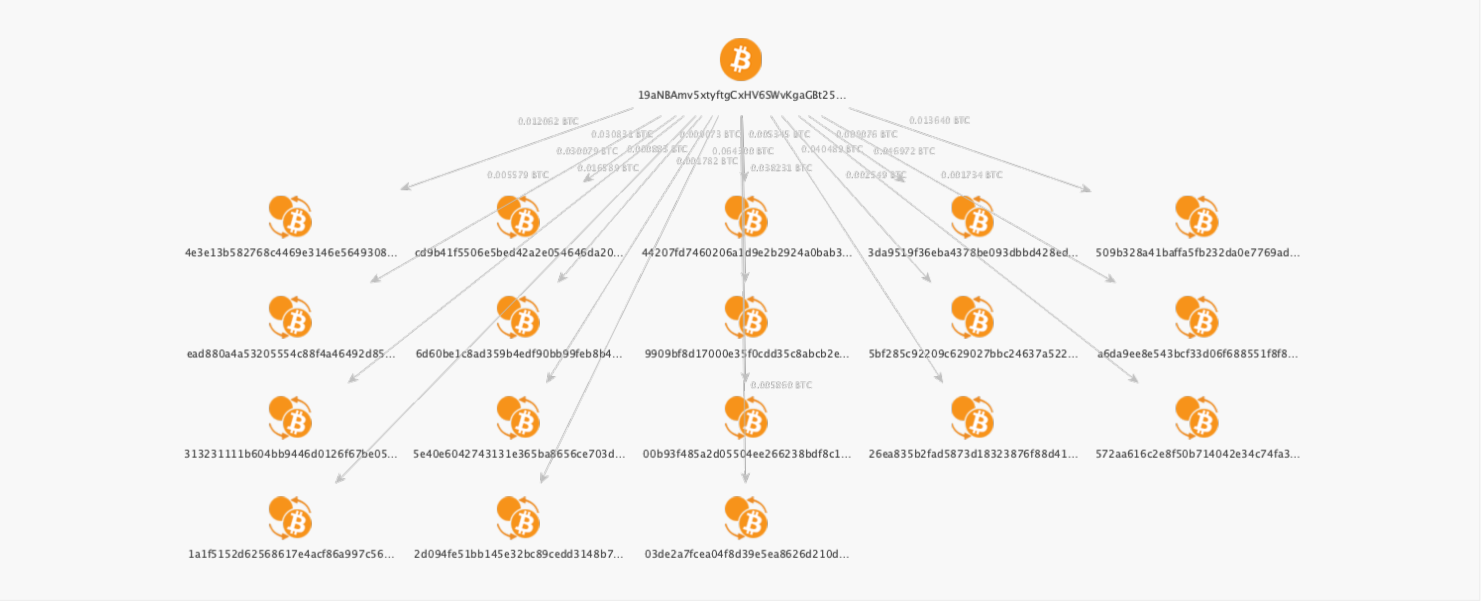
We start here with a known Bitcoin address ₿19aNB... that we got from a scam website, and use the **Bitcoin Address** Entity from among the BlockChain.com Entities:

TIP: In the following text you will find abbreviated bitcoin addresses; e.g.: full address for "₿19aNB..." is ₿19aNBAmv5xty-ftgCxHV6SWvKgaGBt25W6G

As we discovered previously, the Bitcoin network is transaction-based and relations are many-to-many, which means each Bitcoin address might have multiple input and output transactions, and each transaction also might have multiple input and output addresses. Here, moving from the first address, we use the **To Outbound Transactions [BlockChain.com]** Transform:



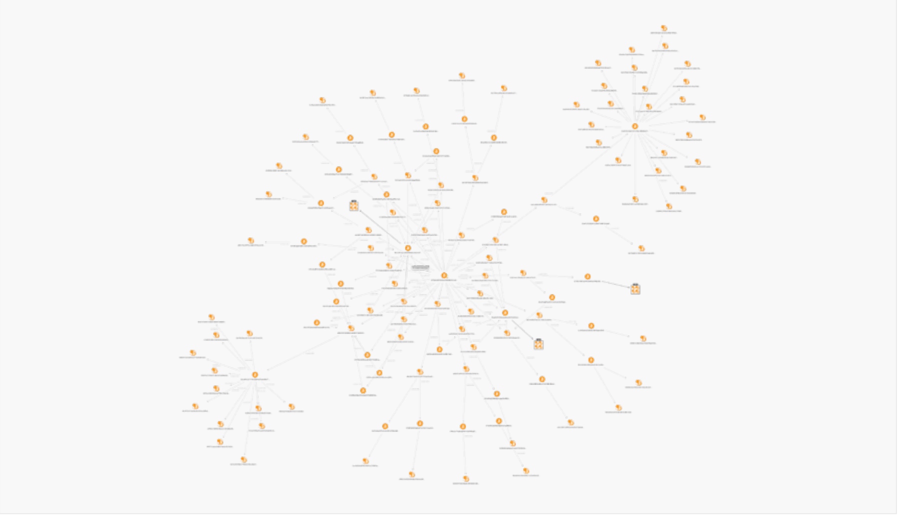
So we get 18 outbound transactions from this address at the time of writing (as new transactions for this address can happen later, running the same Transform again for this address can return more than 18 transactions):



On this graph, transactions are automatically sorted by time, i.e., the top left transaction is the most recent and the bottom right one is the earliest.

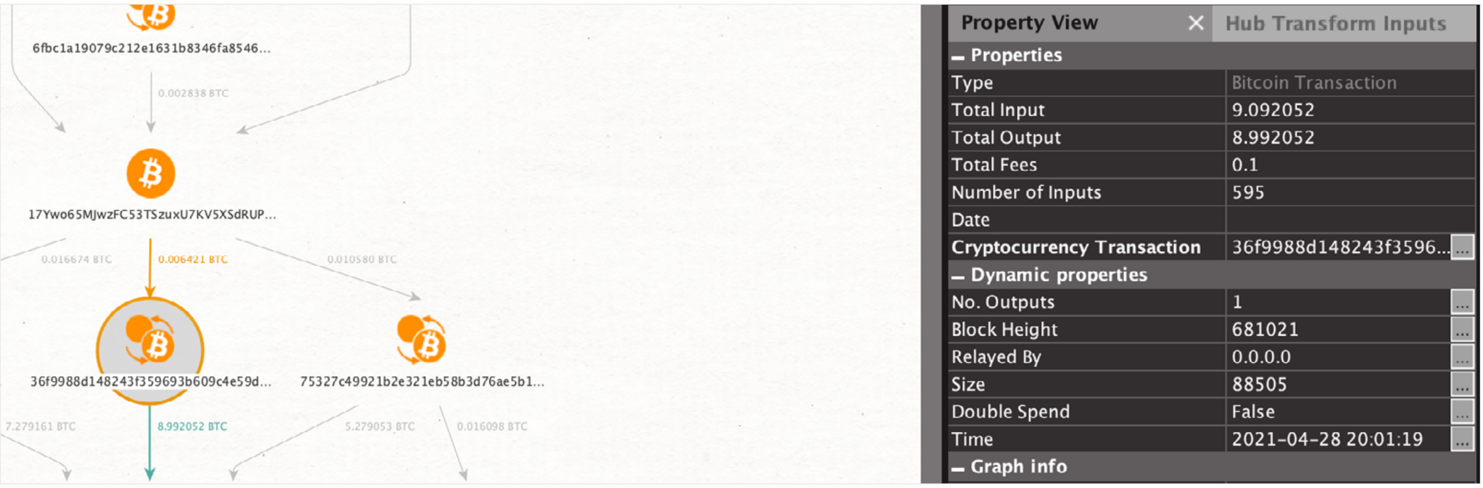
Active Bitcoin addresses might be involved in hundreds or thousands of transactions, and then related to thousands of other addresses as well. Below is one example of a Bitcoin address with only 31 outgoing transactions. In just a couple of steps, we trace all outgoing transactions from this address and then to all destination addresses. Ultimately, we have this picture with hundreds of Entities and links, which can clutter up the graph with information not immediately relevant to the investigation at hand:

NOTE: The important thing here is that we need to focus only on the Entities which are relevant to the investigated case in our graph. Transforms may return extraneous results and we need to handle that, filtering out the unneeded.



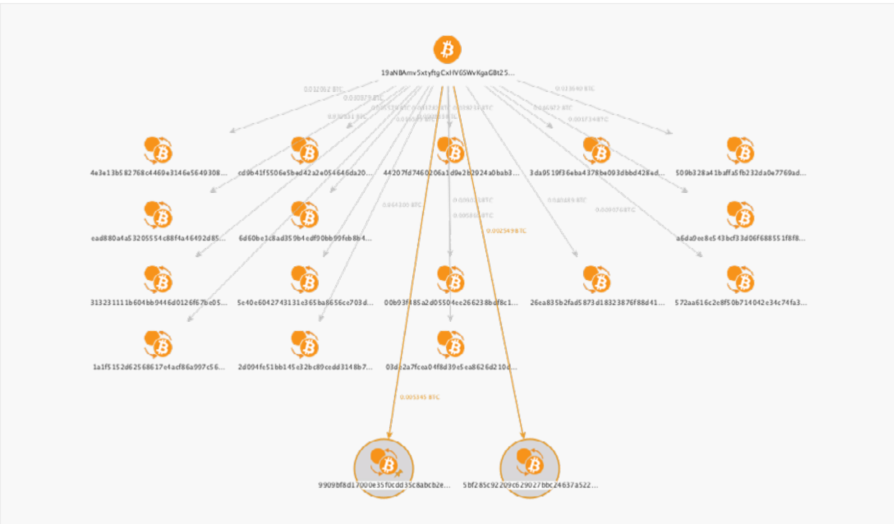
Now let us get back to our Bitcoin address, `19aNB...` We need to find and check April transactions, and we can do it by checking the properties of the Bitcoin Transaction Entities.

Upon selecting an Entity on the graph, its properties are shown in **Property View** window to the right of the main graph window:



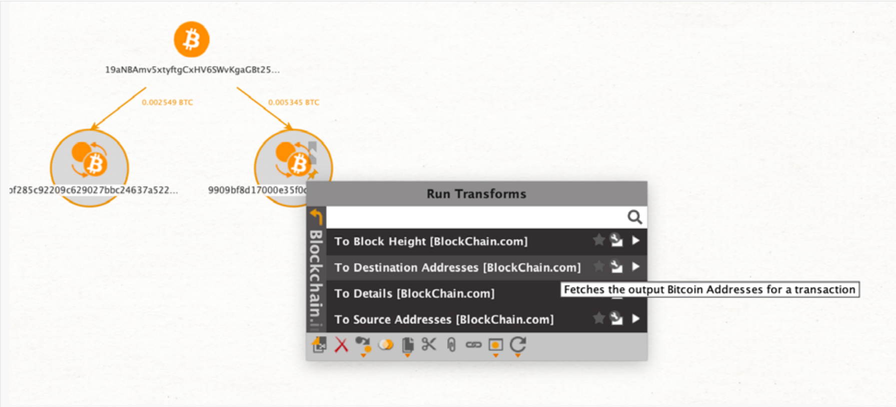
This set of properties varies among the different types of Entities. For blockchain transactions, we have different parameters here, like number and sum of total inputs and outputs, total network fees paid, block height (its position in the blockchain), and, of course, date and time of the transaction.

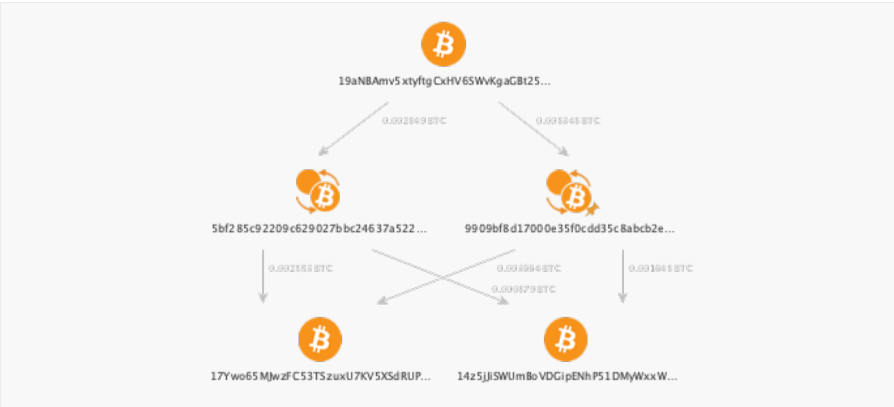
So far, we found only two transactions which took place in April, and we drag them out from the rest:



Now we can actually delete all other transactions which are currently not of interest to us and proceed with building the graph from these two transactions of interest and the original Bitcoin address Entity.

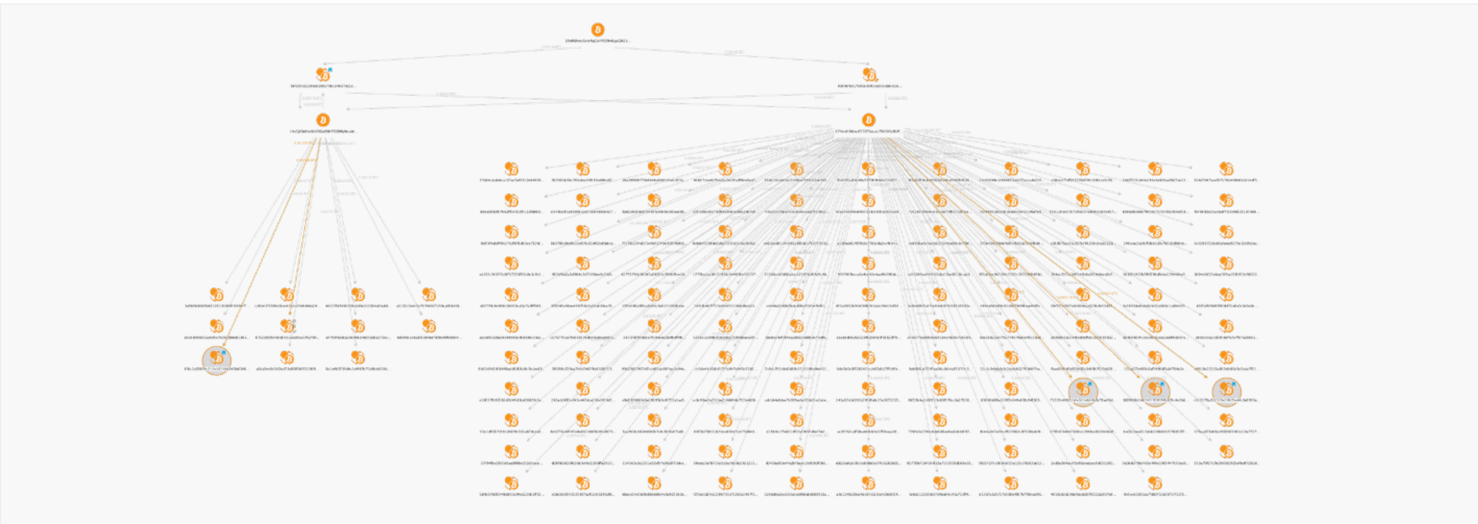
This way we will get all destination addresses for the two selected transactions by running the **To Destination Addresses [Block-Chain.com]** Transform, which will allow us to track further flow of Bitcoins which were sent with these two transactions:





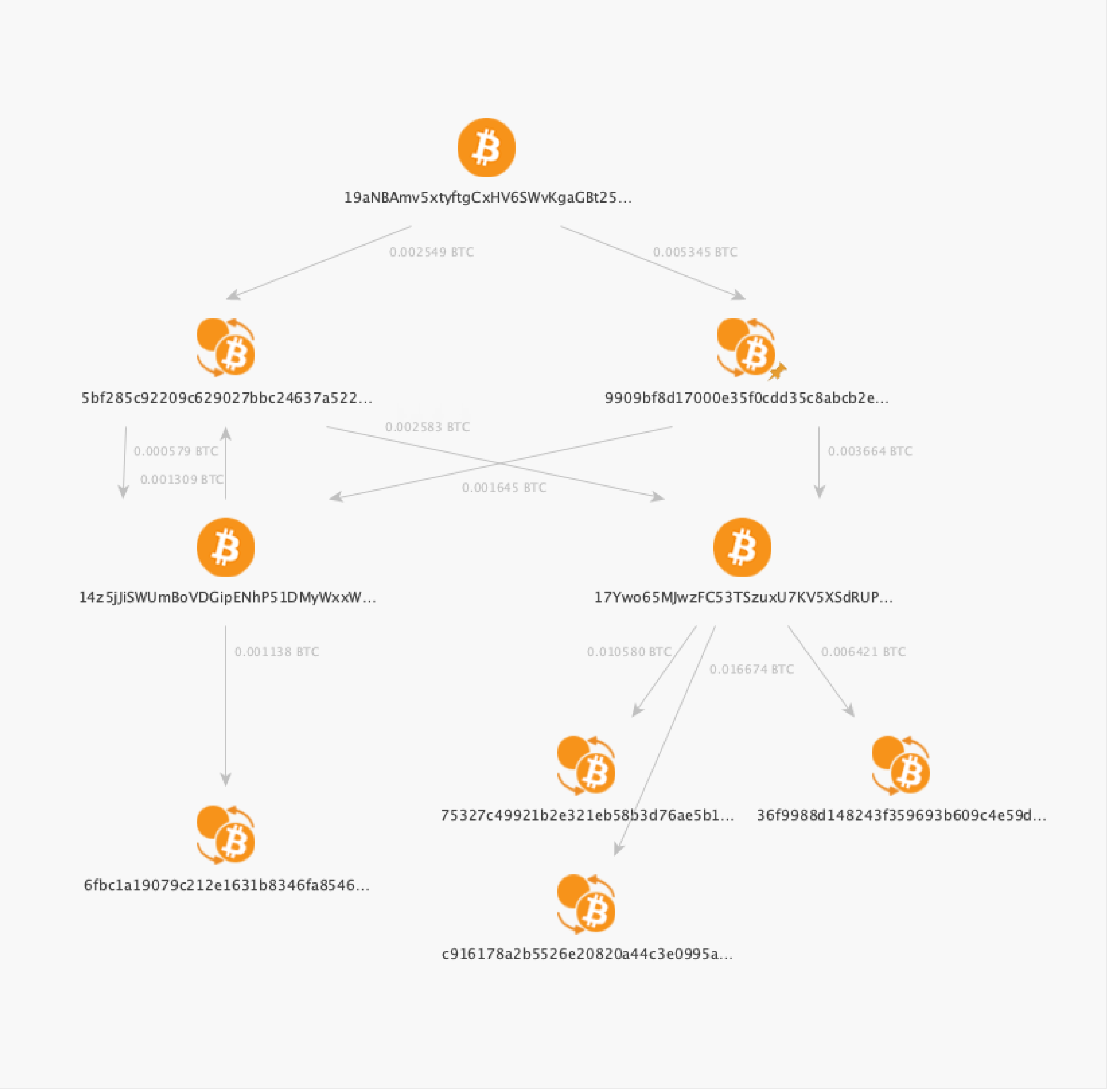
TIP: In Maltego you can easily apply the same Transform to a group of identical Entities by selecting them first, and then choosing a Transform by right-clicking the selected group.

Going one step further: Run the **To Outgoing Transactions [Block-Chain.org]** Transform once again for the newly returned Bitcoin Address Entities, and then select only transactions which are 1) still from April and 2) later than the two already found transactions `5bf28...` and `9909b...`, thus tracking the flow of Bitcoins further:

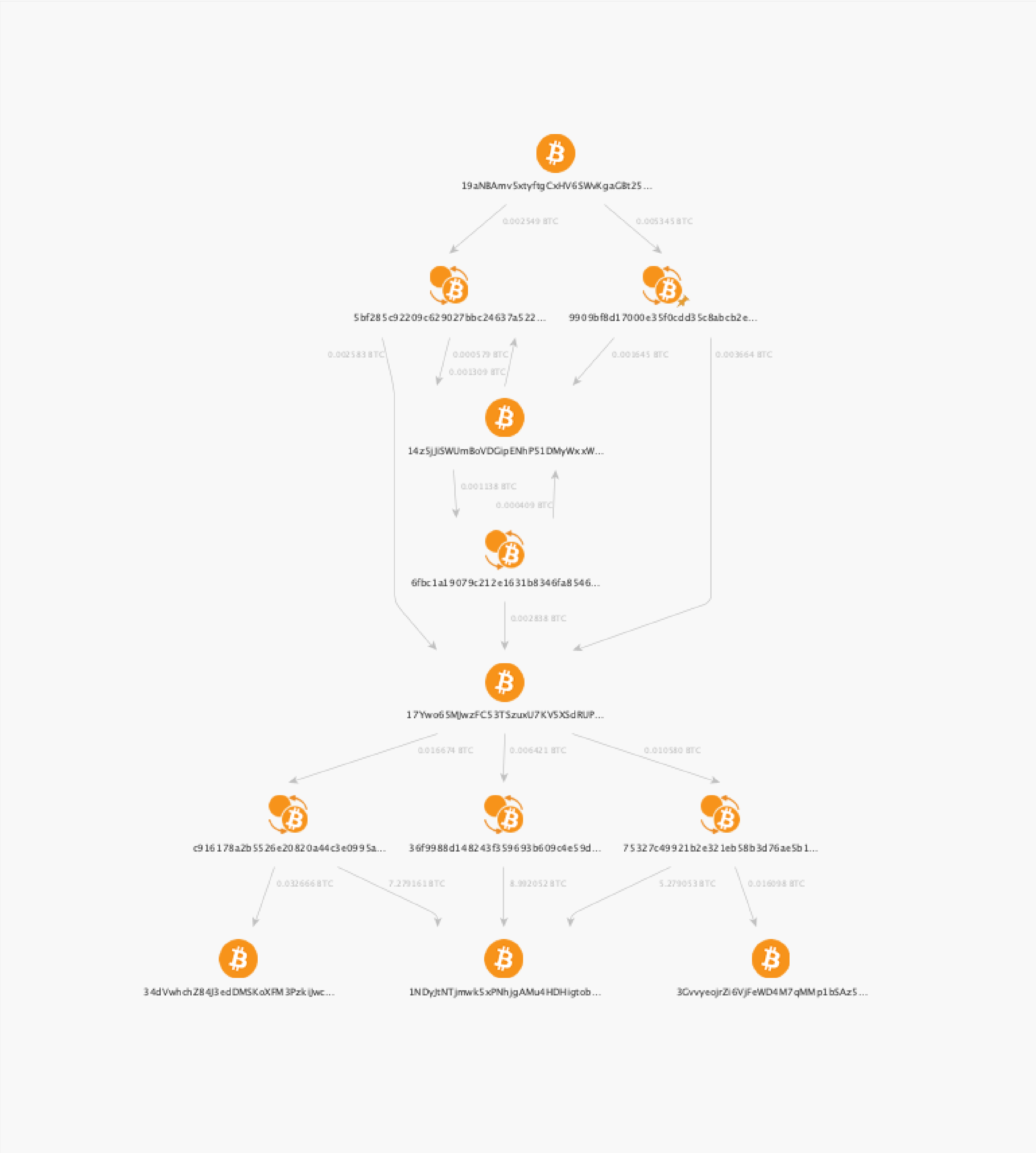


As you see here, without a selective approach we already have too many Entities on the graph, this is why we need to define the investigation path pretty precisely in advance

Next step, let's pull out these 4 transactions that occurred in April but after the first two transactions of interest were conducted, delete all other transactions which are of no interest to us currently, and then get destination addresses for these transactions by running the **To Destination Addresses [BlockChain.com]** Transform.



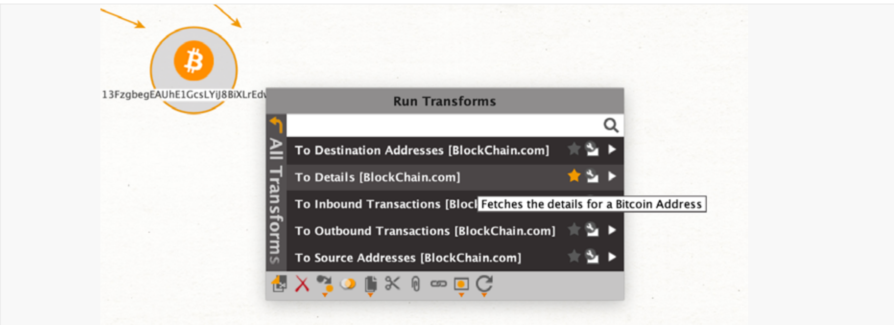
As the next step, we run the **To Destination Addresses [Block-Chain.com]** Transform to all four transactions on the bottom level (remember you can select them and right-click to apply the same Transform to the group of selected Entities), and we get this picture:



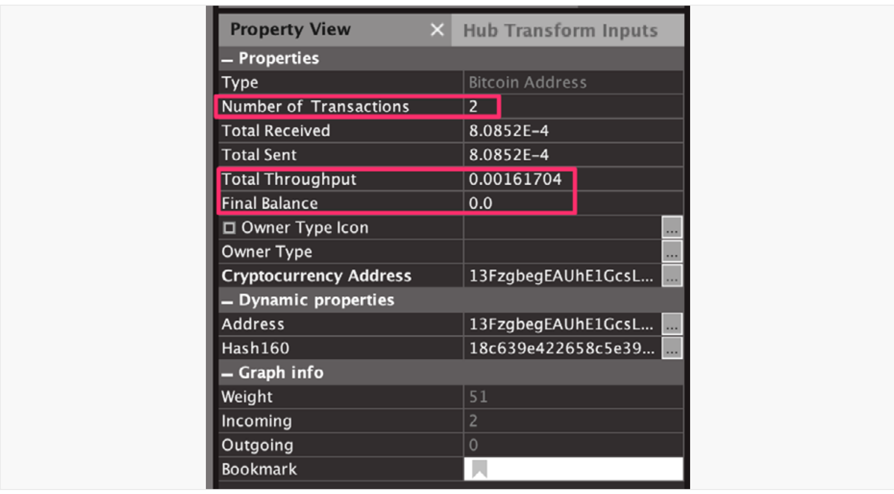
Finding Important Addresses: Step-by-Step Process

Now it is time to check the details of the Bitcoin addresses we got and see if any of these are of special interest to us. So far, we have a few Bitcoin addresses on the graph and we can check their details to see if any of these might be of a special interest to us. Maltego provides different ways to approach this.

In the first method, we can simply use the **To Details [BlockChain.com]** Transform for a selected Bitcoin address or group of addresses:



This Transform fetches some additional information about the address: namely, total number of transactions for the address, monetary amount of input and output transactions, and final balance of the address (that is, amount that was sent to an address but hasn't been spent). This data is not shown on the graph itself, but is available in **Property View** window, and can be viewed upon selecting the Entity on a graph, or even just hovering the mouse over it:



Basically, we should check **Number of Transactions, Total Throughput** and **Final Balance** values to determine if the address is highly active. Checking the three addresses at the bottom level of the graph reveals us the following values:

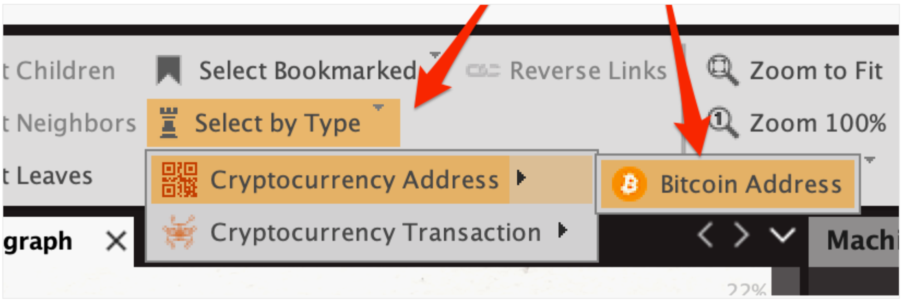
	Number of Transactions	Total Throughput, BTC	Final Balance, BTC
`34dVw...`	2	0.06533218	0
`1NDyJ...`	1017316	28891280	16422.94
`3Gvvy...`	2	0.0321962	0

So far, it is obvious that address `1NDyJ...` is something that could be of the biggest interest because of its much higher activity and throughput. Indeed, quick Googling reveals that this address belongs to Binance crypto exchange, which in our case can potentially be used for cashing out the funds coming from the scammer's address. Of course, in real-life cases we need to perform more detailed tracing, but here we were still able to find the destination exchange associated with the investigated address.

A second method can be used in case we have too many addresses on the graph which makes it not very handy to check the details manually. We can actually use the **weights** of the Entities to find the most active Bitcoin addresses.

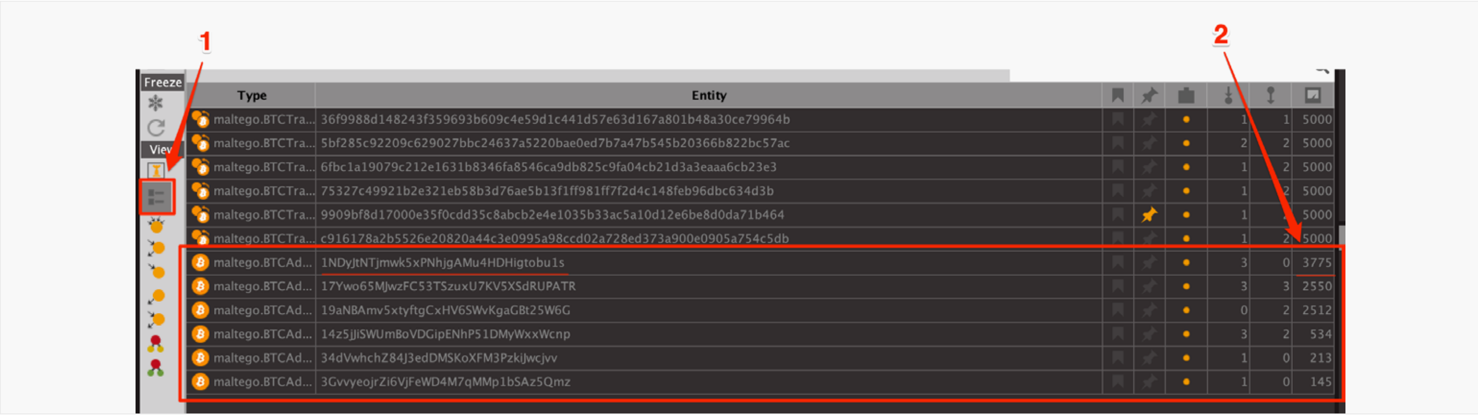
TIP: Blockchain Entities placed on the graph are weighted so that addresses with higher BTC values (or for transactions, higher inputs) will be weighed more heavily. What we need to do is to find the Entity with the highest weight.

First, we still need to run the **To Details [BlockChain.com]** Transform for all Bitcoin addresses on the graph. To do it easily, we select all Entities of a certain type (**Bitcoin Address** in our case) using the **Select by Type** menu in the Investigate tab:



Then we run the **To Details [BlockChain.com]** Transform by right-clicking the selection. This way all Bitcoin addresses on the graph will have their details fetched from the Blockchain and stored in the properties of each Entity. This step is important, because without explicitly fetching the details for *all addresses* Maltego won't be able to correctly assign relative weights to them.

Next, we switch the view to **List View** mode (1), which gives us a table view of all Entities on the graph instead of visual representation:



The rightmost column (2) shows the weight of each Entity. Here we see a group of six Bitcoin address Entities with weights from **145** to **3775.** We instantly find that the same Bitcoin address `1NDyJ...` has the biggest weight, and we would check it first without the need to go manually through numerous Entities.

Now we have a final graph which traces the connection of a scammer's address to the Binance exchange:

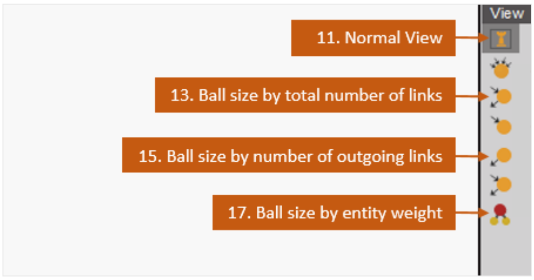
Analyzing Large Graphs

From Weights to Links

In the previous chapter we have seen how we can investigate Bitcoin address activity within a given time period. This gives us good results, but the obvious shortcoming of the method is that we need to check the Entities manually on every step, and we also need to filter Entities by the desired time frame. While filtering generally reduces the number of Entities, manual checks on each step are still required and slow down our analysis.

In certain cases we might not have any additional details like a time frame, and the investigation is performed around a given address regardless of the transactions' time stamp. Also, we can have a really active address with hundreds of transactions, which complicates the possibility of manual analysis.

Just like we previously used **weights** which are based on the BTC values of the addresses, now we will analyze **links** related to an address using views. Generally, views can be used to identify non-obvious information from large graphs, where an analyst can hardly see clear relationships by manual inspection of data. This method, as we will see soon, helps to find key Entities really fast in large graphs with hundreds and thousands of Bitcoin addresses and transactions.



In this method we make a logical assumption that an active bitcoin address belonging to a big Entity like a cryptocurrency exchange should participate in many transactions on the Bitcoin network, and accumulate money flows from many different sources. In other words, we expect it to have both high throughput (in monetary value) and high number of incoming/outgoing transactions. This said, the amount of total transactions and address importance should be positively correlated.

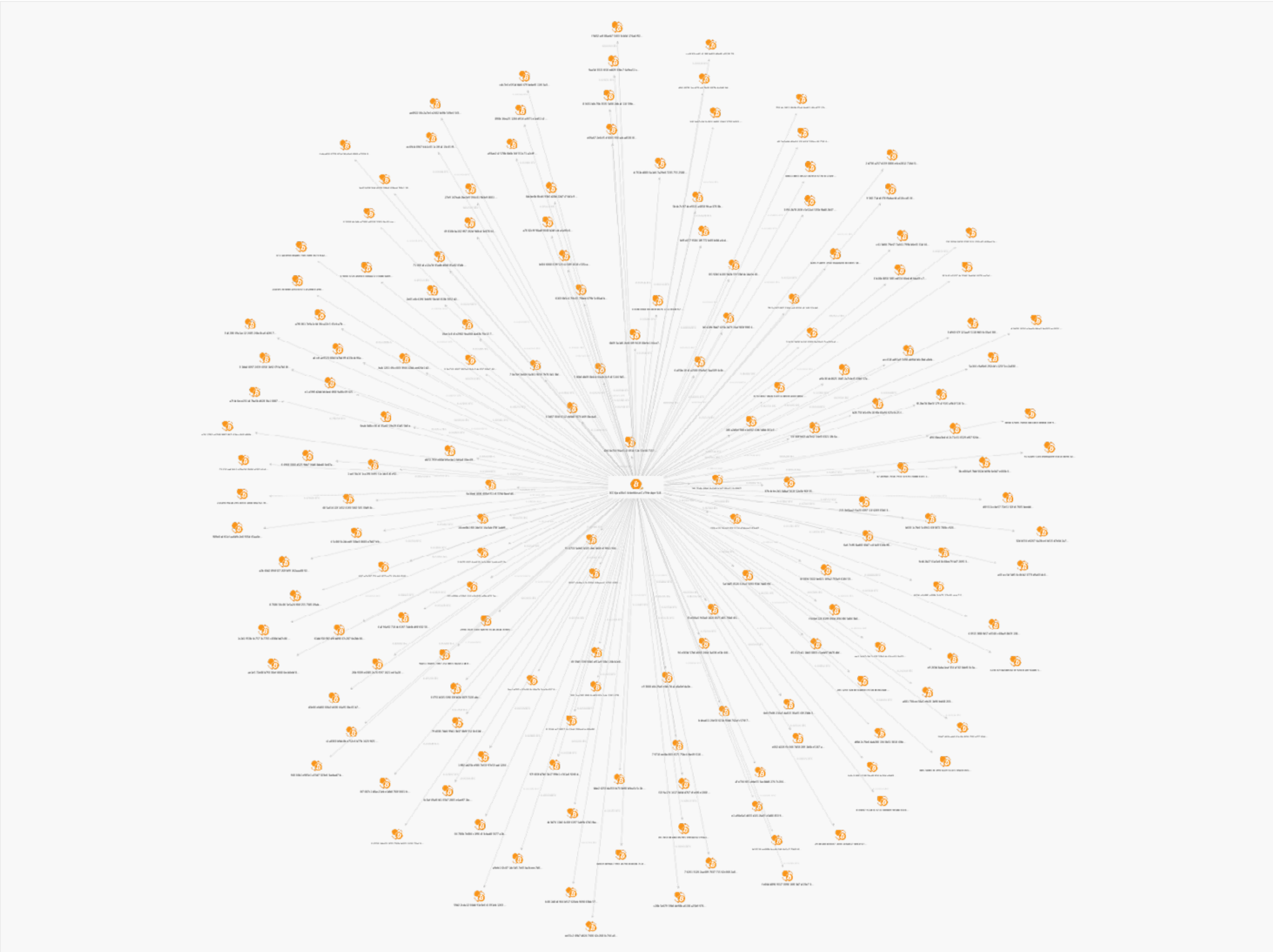
Case Study

Let's consider the following case. We need to find relevant information regarding the activity of the address `3CCKj...` and outgoing money flow, as well as its connections to key network Entities like cryptocurrency exchanges. We don't have any certain time frame to look at, and the address itself is pretty active, having taken part in a total of 362 transactions (at the time of writing, both inbound and outbound).

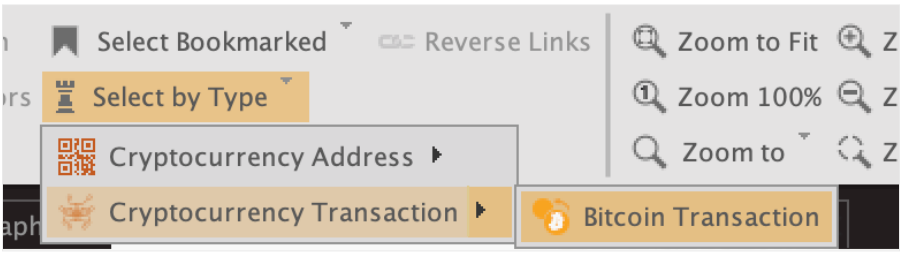
TIP: Abbreviated bitcoin address for "`3CCKj...`" is `3CCKjoo3Xx54vbmMzswCzTHmd-qnr5LBMHv`

Let’s see step-by-step how we can approach this problem in Maltego.

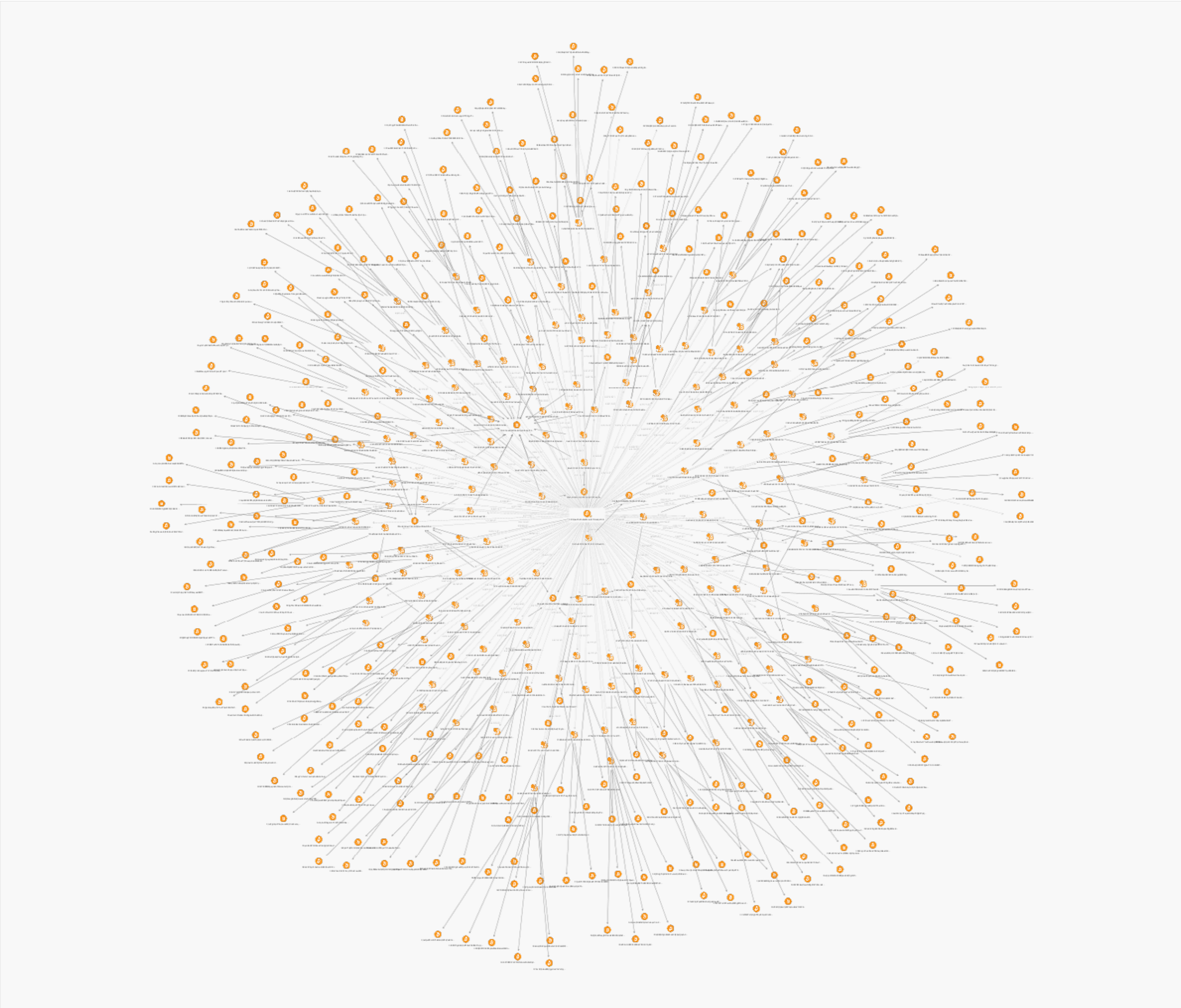
1. In the first step, we add the Bitcoin address Entity to a new graph and run the **To Outbound Transactions [BlockChain.com]** Transform. In the **Organic** layout, the resulting graph looks like this, revealing 166 outgoing transactions:



2. Next, we want to add all destination addresses for these 166 transactions. We use the **Select by Type** option from the Investigate tab so we can first select all transactions on the graph:



Then, we can right-click, choose and run the **To Destination Addresses [BlockChain.com]** Transform, and get a graph of 544 Entities:

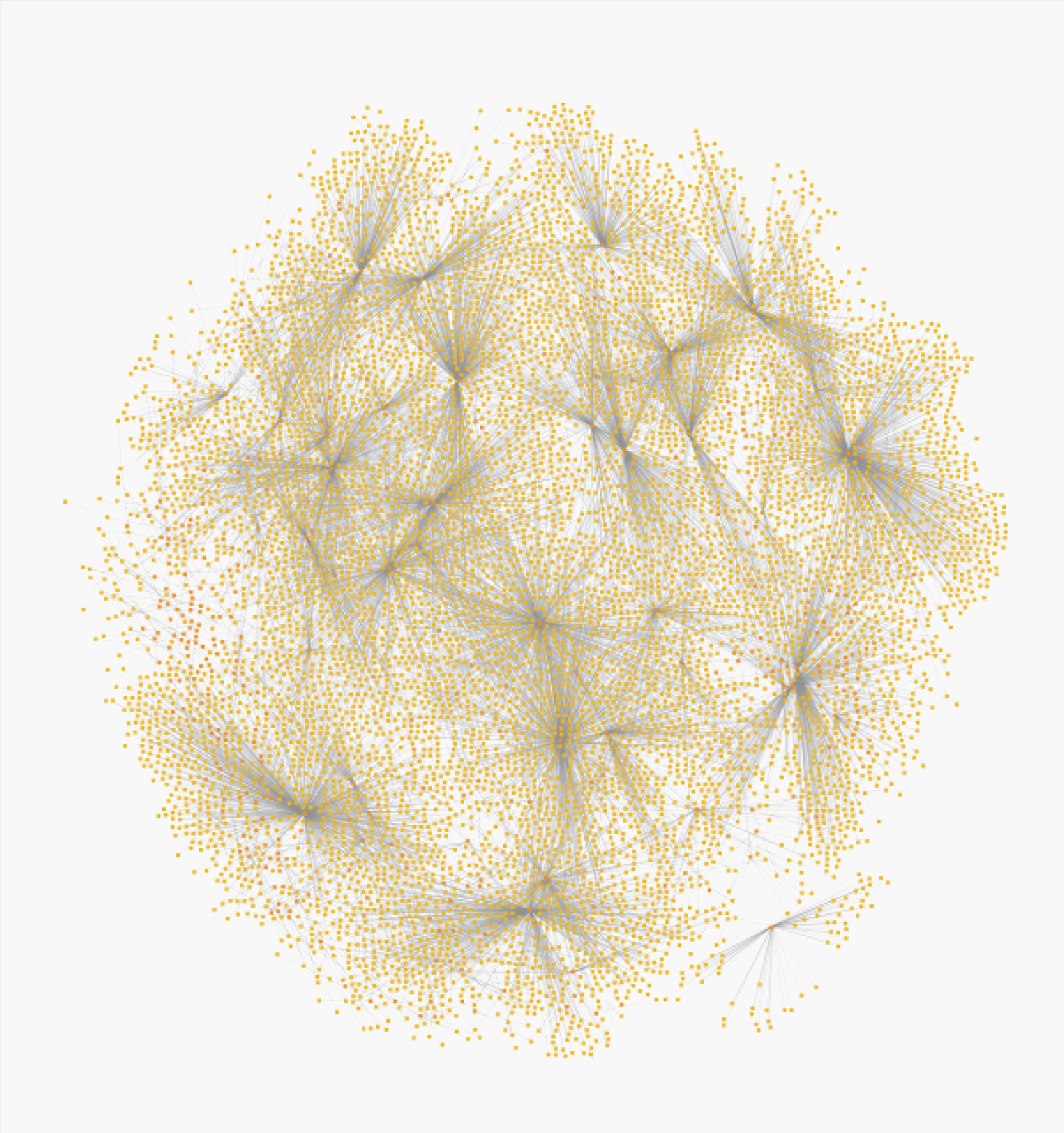


3. Then we take another step to expand the graph one level further, once again retrieving outbound transactions for all addresses on the graph.

In total, we would perform only three iterations to expand a graph starting with only one address. On each iteration, we are taking the following steps:

- Select all Bitcoin address Entities.
- Run the **To Outbound Transactions [BlockChain.com]** Transform for selected Entities.

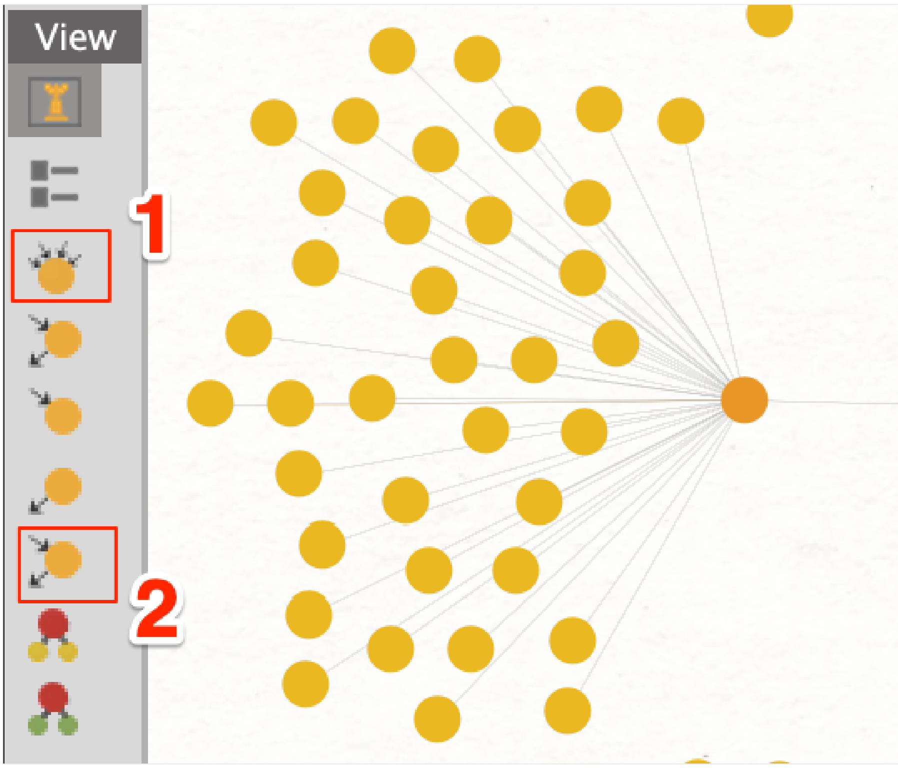
The whole graph after the last iteration contains a total of 8785 Entities and 9000 links:



The final step can actually take up to 10–15 minutes to run because of the large number of Entities. Also notice that with each step, the size of the graph rapidly increases by an order of magnitude (1 › 165 › 544 › 8785 Entities).

Looking at the last graph, one may ask how it is even possible to extract any valuable insight from such a structure?

For finding meaningful insights in such a big graph, let's turn to the **View** sidebar menu, which provides methods of visual exploration of such a graph:



TIP: Generally, **View** options are intended to set different sizes to the Entities based on different criteria. In this case all Entities on the graph are represented by colored circles of different sizes instead of default icons.

The Maltego technical documentation gives definitions and examples on available view options, and we will use two of these (highlighted on a screenshot above):

1. Diverse Descent: With diverse descent, Entities are sized according to the number of incoming links the Entity has. However, incoming links with different grandparent Entities are more highly weighted.
2. Rank: This will size Entities based on its own number of links and the sum of its neighbour's links.

These two view options are somehow alike in the sense that both are taking into account the input links of the Entity, however in slightly different ways. **Diverse descent** gives the highest weight to the Entity that has the most number of incoming links which, in turn, originate from the most number of different parent Entities. **Rank** is based just on the number of links for an Entity and its neighbours.

To make things clear, we can use a simple but quite effective analogy from real life here to explain these principles: River systems!

Below is a map of USA river systems where the river's width is proportional to an average annual discharge. It all begins with tiny streams which flow into bigger ones, then they flow into even bigger rivers and so on, until a huge flow reaches the ocean; at any given point the river width is proportional to the number and volume of its different tributaries.

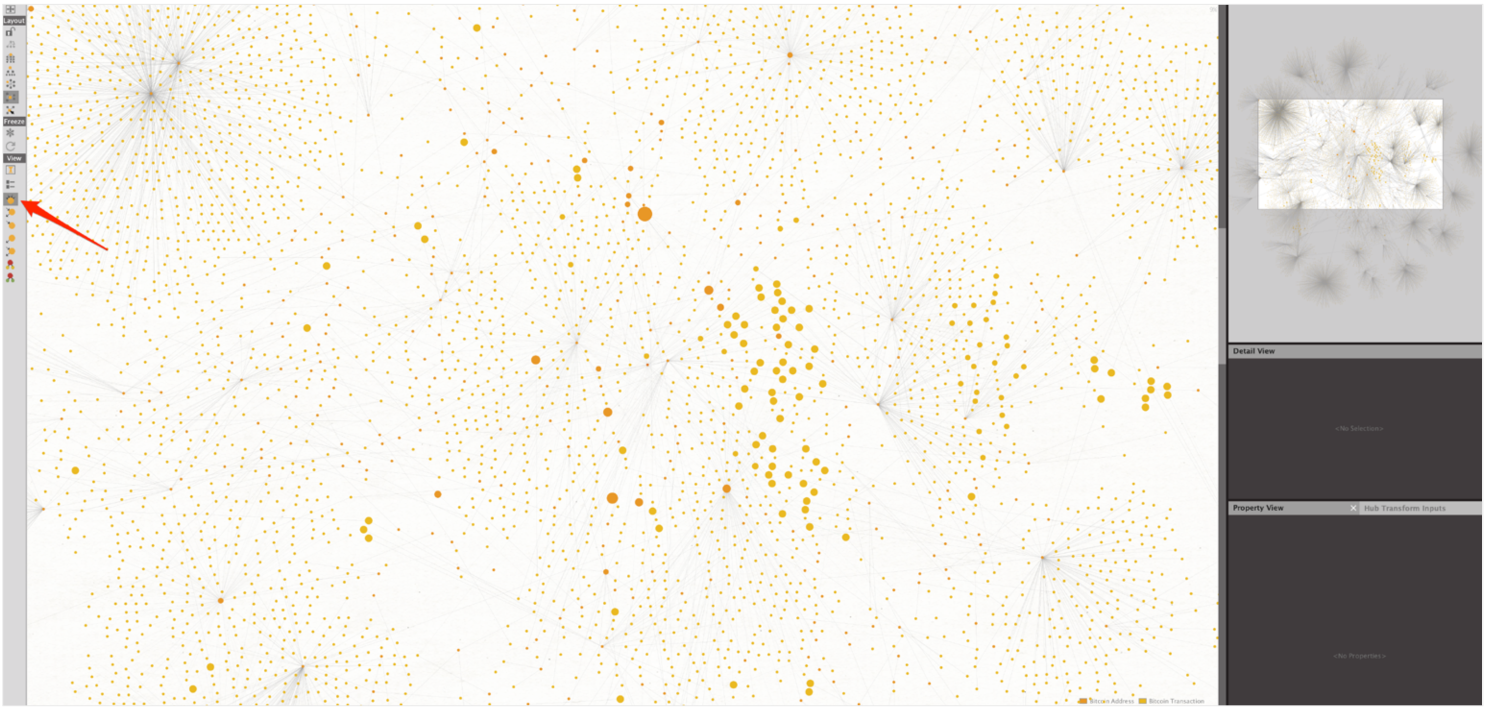
Source: *Heberger, Matthew. 2013. American Rivers: A Graphic. Oakland, Calif.: Pacific Institute.*

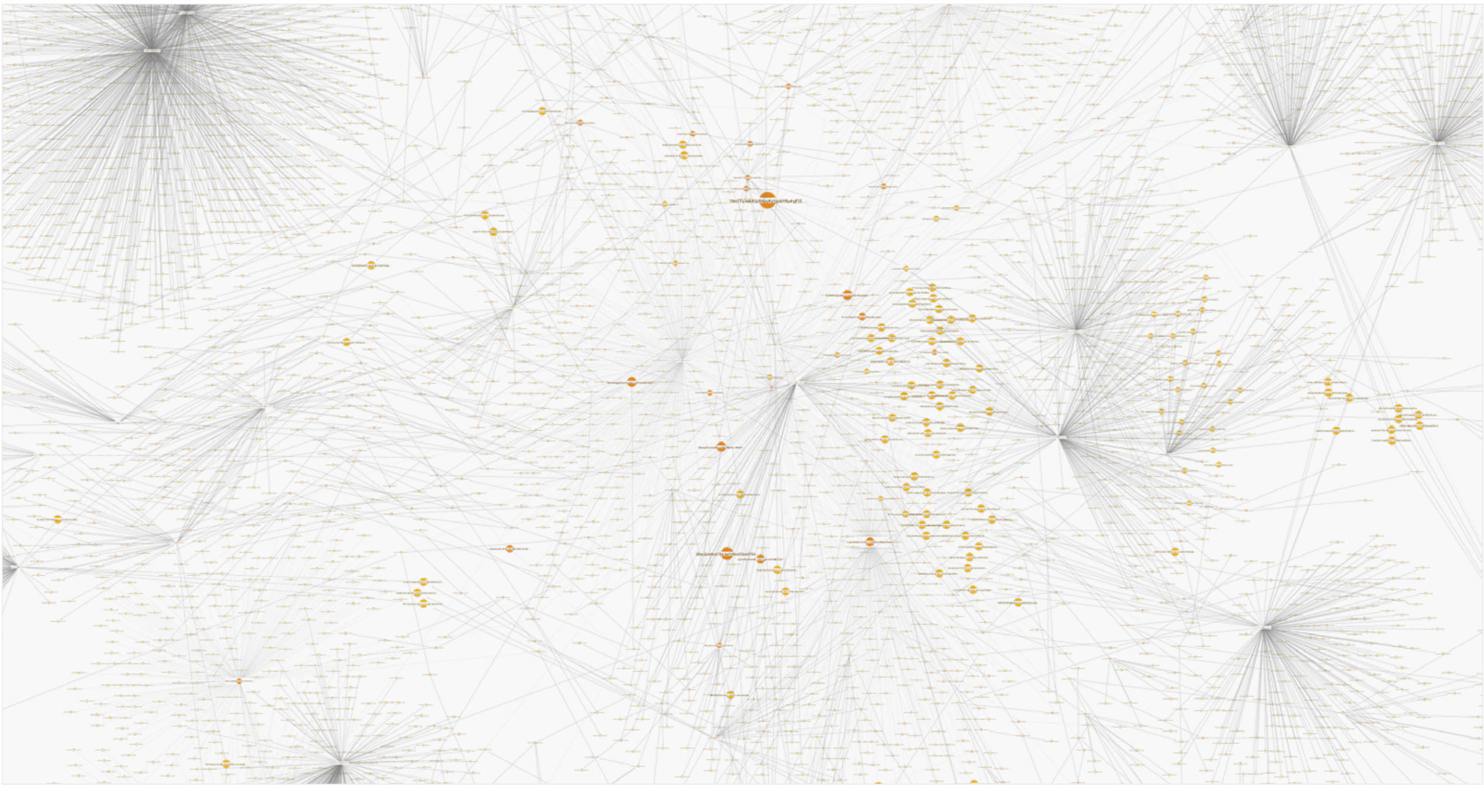
Of course, the whole Bitcoin network structure is not analogous to a river system, because the network does not have a definitive hierarchical structure and is much more complex. However, if we consider a single isolated part of the network which represents Bitcoin flows from addresses with tiny amounts into a big crypto exchange, then it can be surprisingly similar: It starts small with a single address and low amount, and then grows bigger as it approaches a centralized crypto exchange, and this growth depends on the number and volume (meaning value) of "tributaries" (all other addresses that contribute to that money flow along its way).

And so, back to graphs. When using **View** options, we should be able to find Entities that are most important among all others, where *importance* is measured by the amount of different input connections. And this is, as you might have guessed, exactly what we need to locate Entities like exchanges, which naturally accumulate inputs coming from thousands and thousands of individual addresses over the network.

Now, let’s use **View** to find the most important Entities on the latest graph.

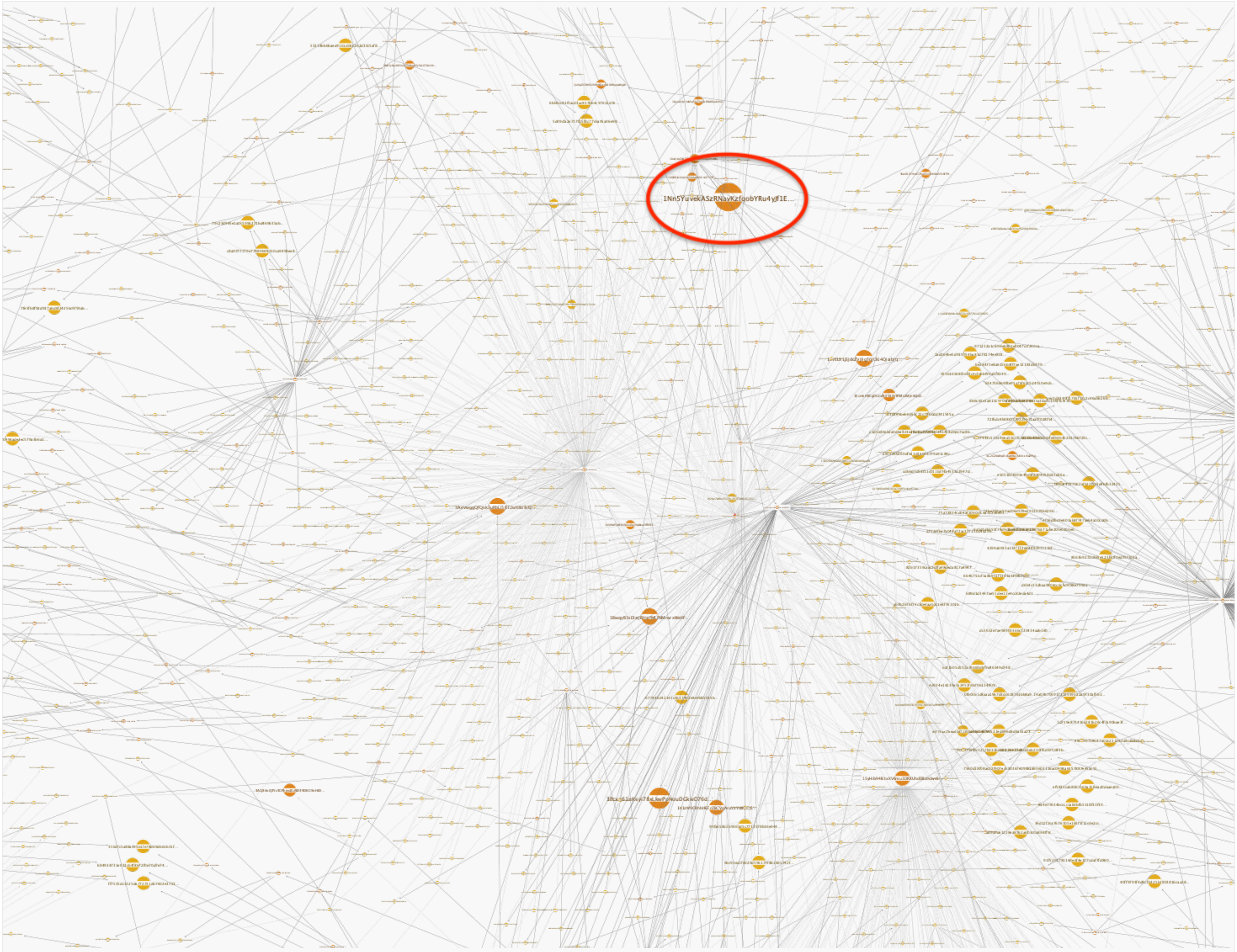
Here we use the **Ball Size by Diverse Descent** option, and then zoom into central part of the graph:





It looks huge, but now we see a not so big group of addresses that are definitely larger than the majority. One of them is the biggest:

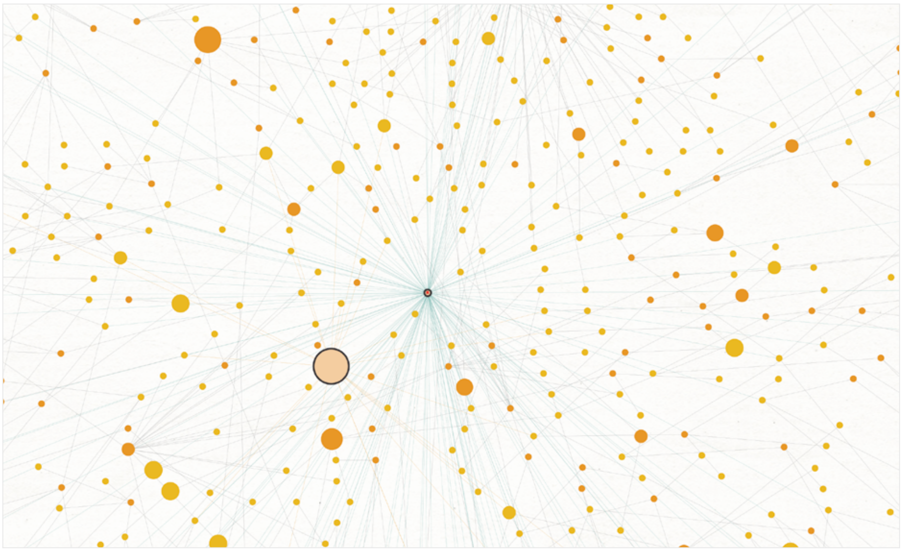
So far, we can make an assumption that this address 1Nn5Y... is the most interesting address on the whole graph. And indeed, this is an address belonging to Binance exchange (one can use the free online analytic tool Vivigle to check the attribution of some known bitcoin addresses).



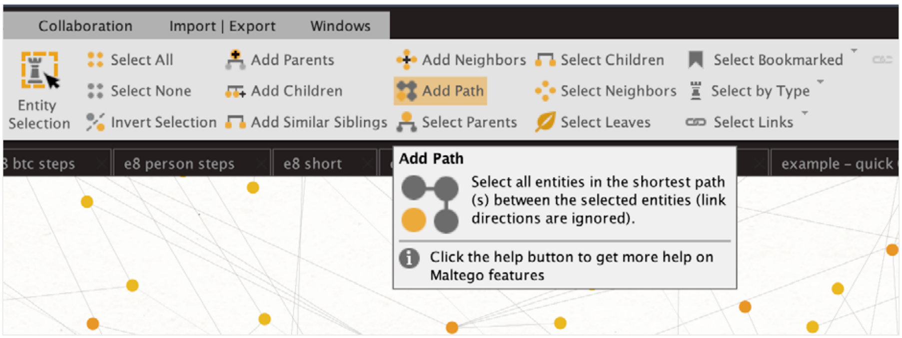
Now that we know there actually is an address on the graph which leads us to a big exchange, we might want to 'pull out' the related Entities from the big graph to investigate this part of the graph in more detail. But of course, we need not only single addresses of interest but also the existing connections between them as well, which might be either direct or indirect.

This can be done pretty easily in just a few steps:

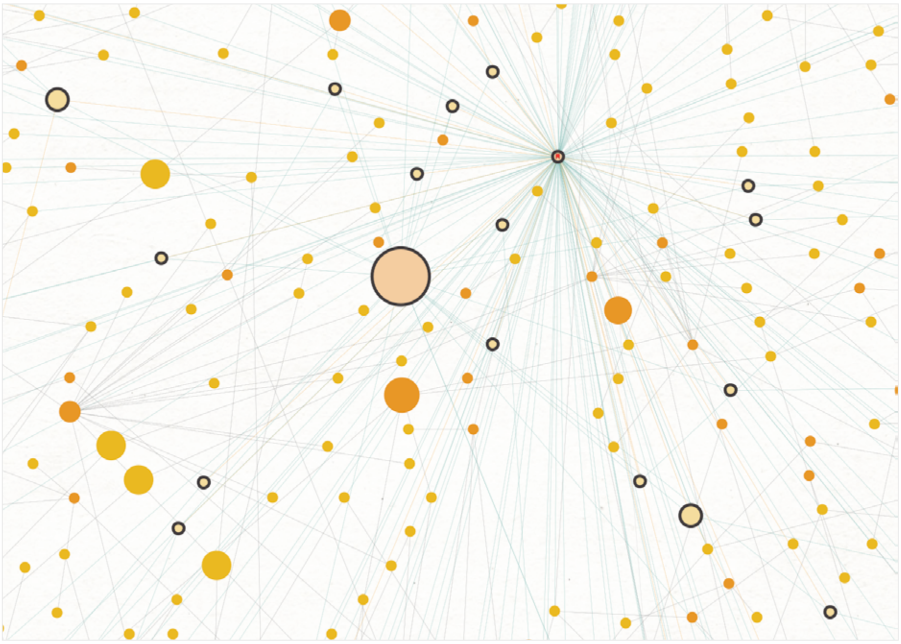
1. Select the starting address 3CCKj... (hopefully you didn't forget to bookmark it from the very beginning!) and the Binance address 1Nn5Y... on the graph (multiple selection on Maltego graph is done by clicking on the Entities while holding Shift key):



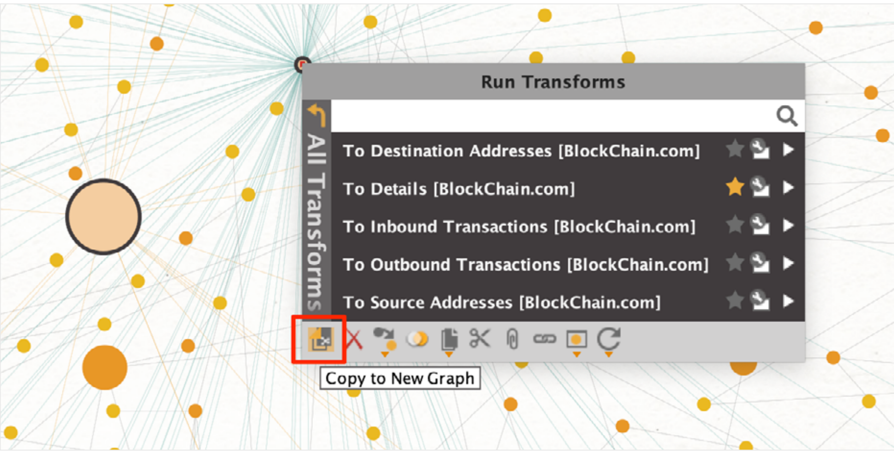
2. Next, go to upper menu Investigate › Add Path:



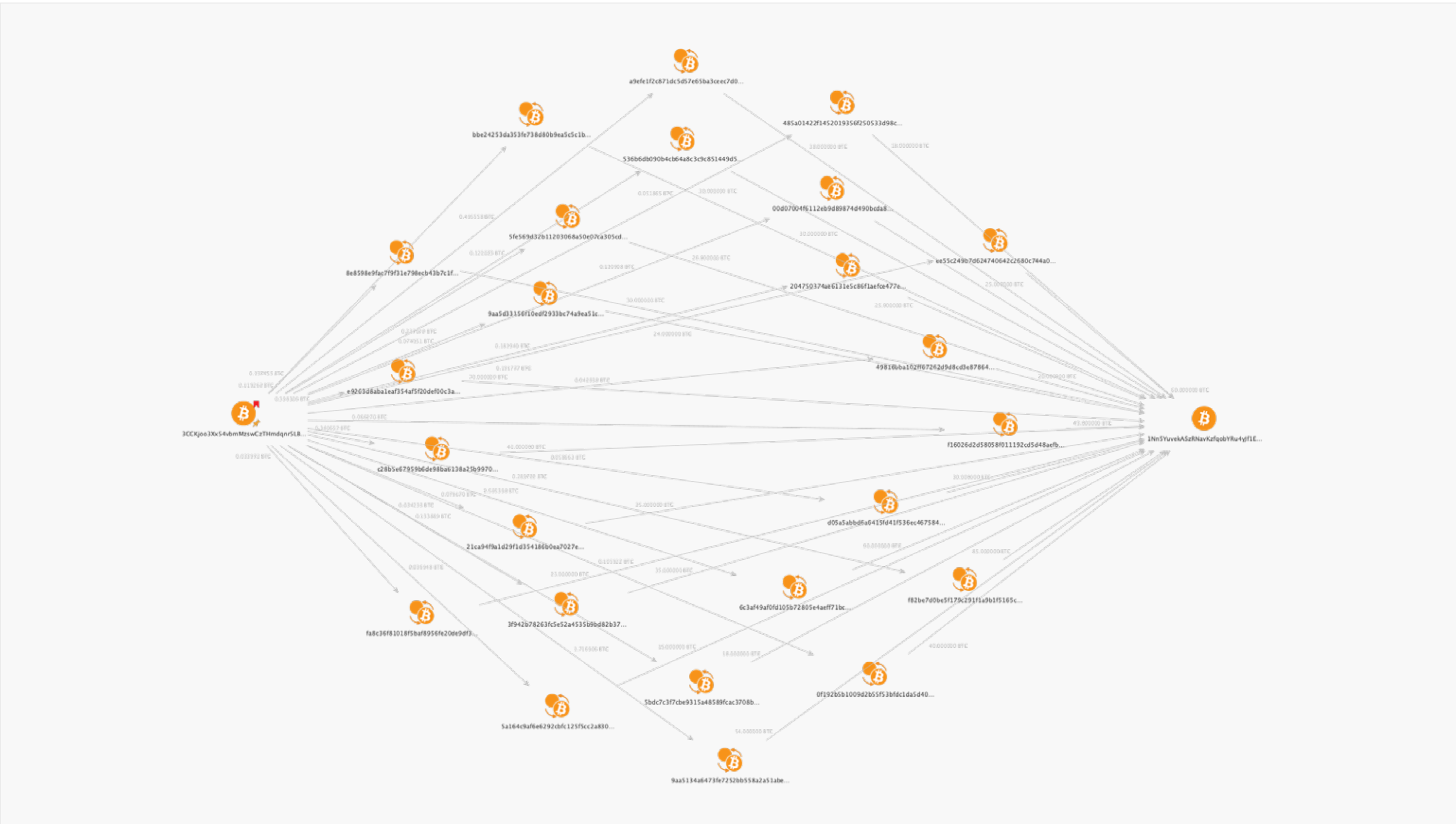
This useful option automatically selects all Entities on the shortest path between the selected Entities (in our case, two addresses). Now you can see quite a few Entities are also selected on the graph around two addresses of interest:



3. Next, right-click any of the selected Entities and press the button at the bottom left “Copy to New Graph”:

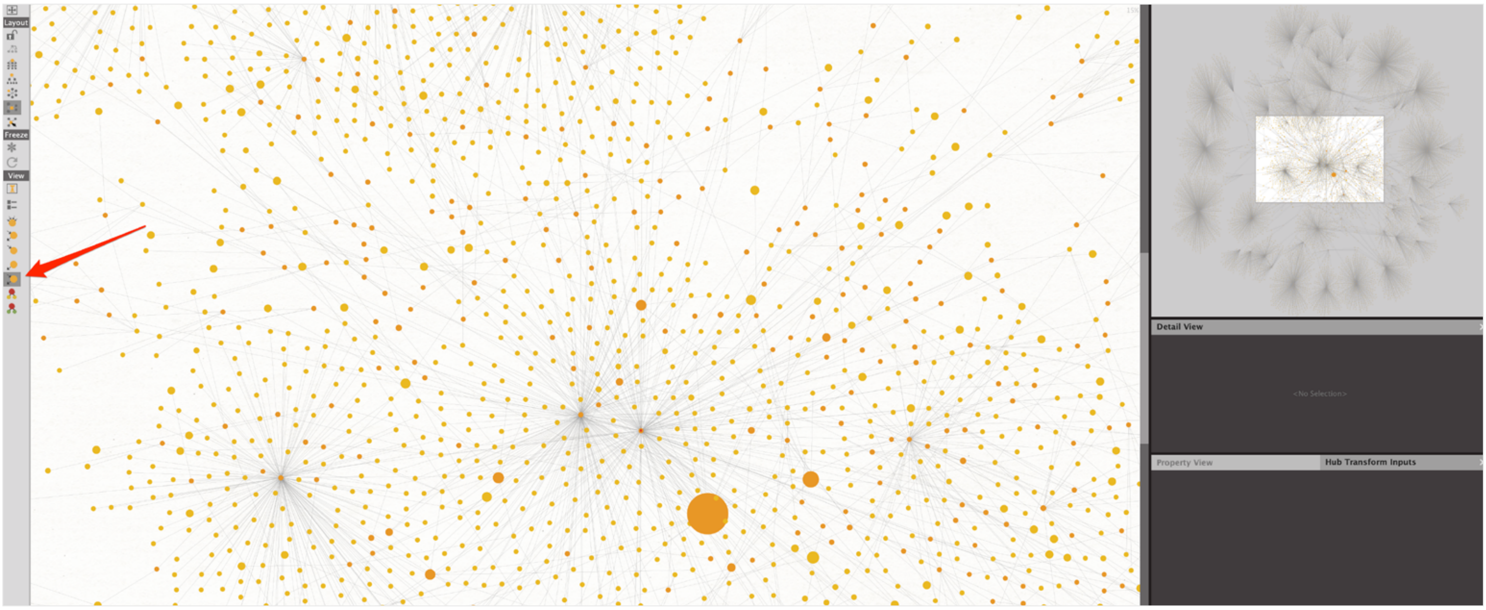


4. Tidy up the new graph a bit if needed (in this case, the organic layout has been applied and addresses were manually aligned to left and right), and voila - now we have a small graph which traces exactly what we need: The flow of funds along with transactions between initial address ₿`3CCKj...` and Binance address ₿`1Nn5Y...`:



The investigation does not have to end here. You can continue tracing, but we have significantly simplified our task by narrowing our graph down to exactly the relevant Entities of interest, effectively from over 8000 Entities to only 26!

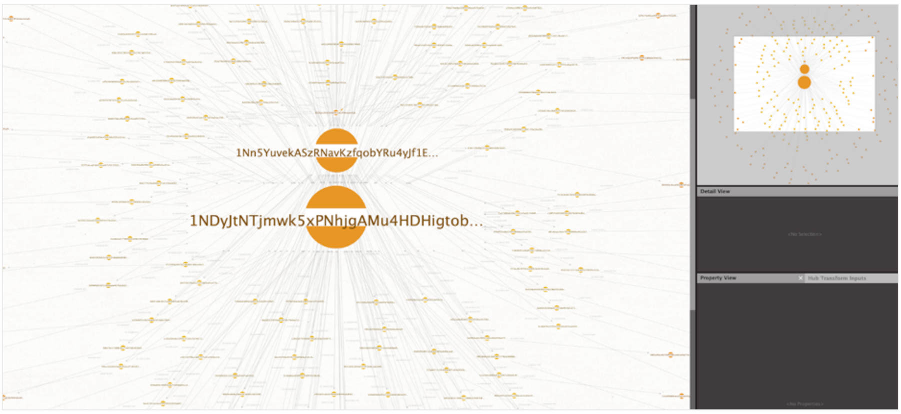
We can also try to use the **Ball size by Rank** view option, which produces slightly different results but still brings up the same address ₿`1Nn5Y...` as the one with the highest rank:



As mentioned earlier, the two view options we discussed use different approaches, but the results can be pretty close. In practice, it makes sense to try both view options and compare the results, then choose the one which best suits the particular case.

Now, what do we do with the smaller graph of 26 Entities we derived from the huge one with over 8000 Entities? We can basically repeat the steps described earlier: Starting from ₿`1Nn5Y...`, we execute the following steps:

- Run the **To Outbound Transactions [BlockChain.com]** Transform;
- Select all newly found transactions, apply the **To Destination Addresses [Blockchain.com]** Transform;
- Switch to the **Ball Size by Diverse Descent** view;
- And we now end up with ₿`1NDyJ...` address, a well-known Binance hot wallet.



Part 3
Visualizing Activity on Ethereum Network

Until now, we've been working with Bitcoin network transactions using **Blockchain.com (Bitcoin)** Transform set, which supports Transforms for looking up connected bitcoin addresses and transactions automatically.

Ethereum network principles are different from those in the Bitcoin network. Bitcoin by design is a digital currency. Ethereum also utilizes blockchain technology, but it is much more robust in the sense that it can be used not only for supporting a digital currency itself, but also for smart contracts and decentralized applications. Smart contracts can also represent such digital assets as tokens. Each type of token serves a certain purpose, may support different protocols, and there can potentially be unlimited varieties of different tokens with different values on Ethereum network.

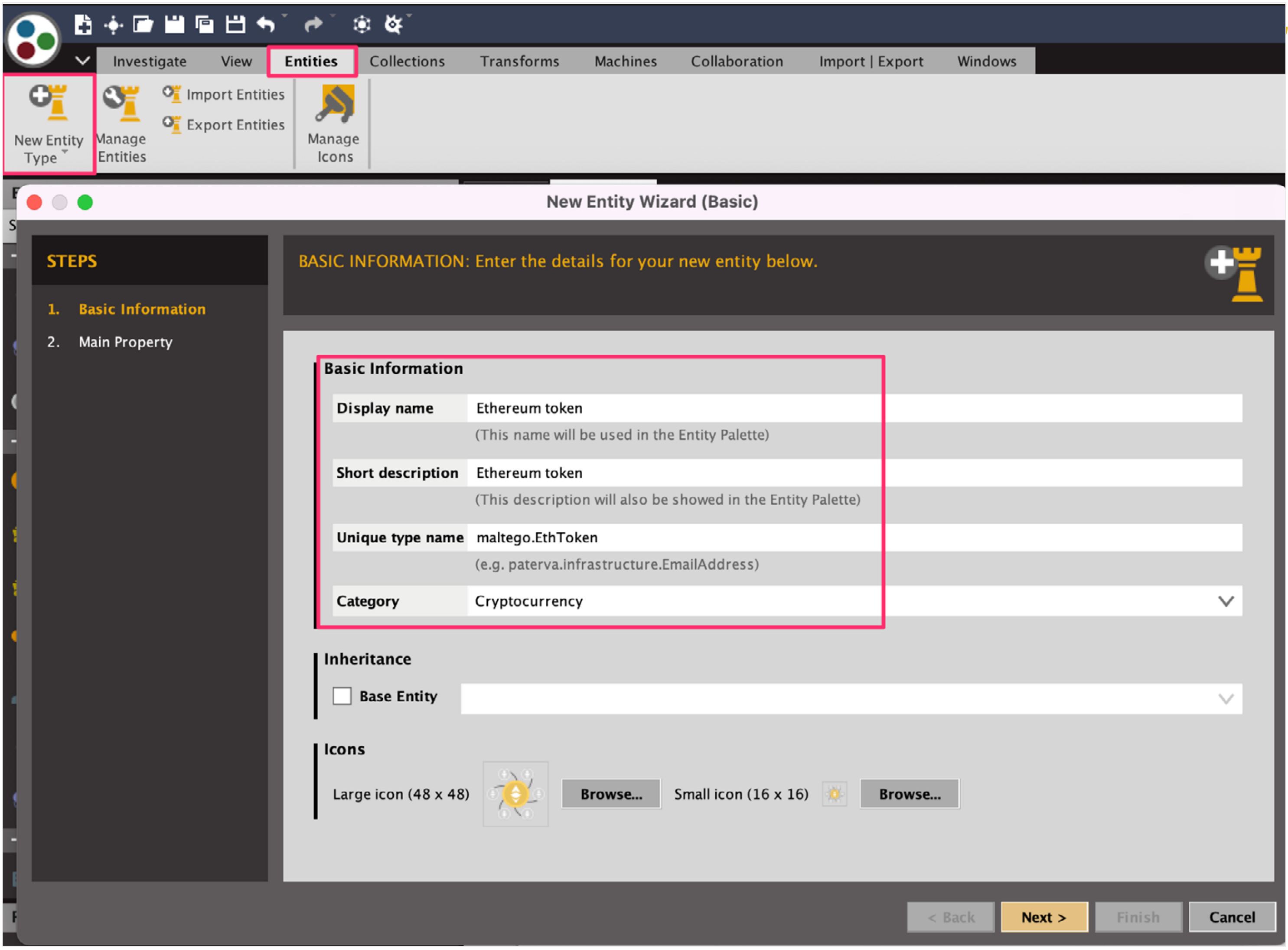
To put it simply, the Ethereum network provides almost infinite opportunities for innovations and creating new digital assets, so the tracing problem turns out to be more complicated compared to the Bitcoin network. You need to understand the type of assets you are tracing, their values, be able to recognize countless swaps where one type of token is exchanged to another one directly or through ethereum as a primary digital currency on the network.

Blockchain.com (Bitcoin) Transform set contains **Ethereum Address** and **Ethereum Transaction** Entities, but does not offer any Transforms for these Entities, so you are left with only a manual visualization process. We will learn further how it is possible to utilize Maltego functionality and visualize Ethereum network transactions with the help of an external data source and powerful Maltego feature of importing external data.

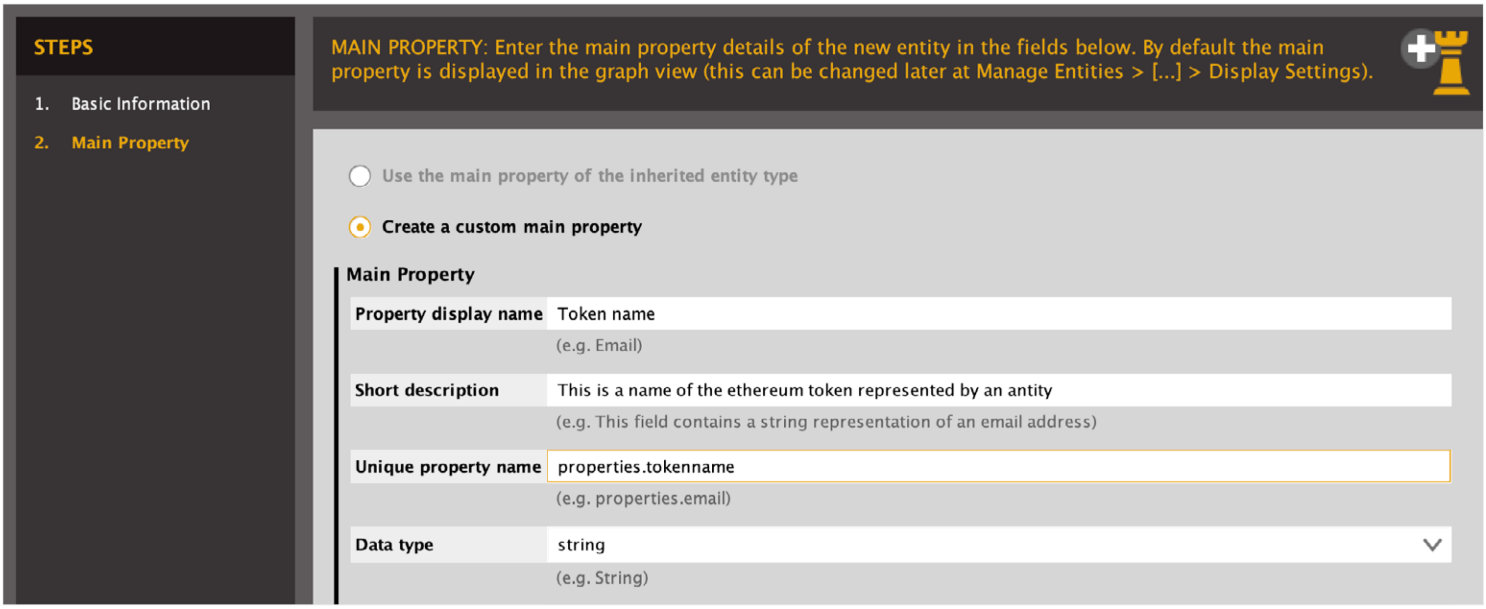
Creating New Entities

We don't have a native Maltego Entity for an **Ethereum token,** so we might want to create a new Entity. For that, we go to **Entities › New Entity Type** menu, fill the fields and choose an icon for the new Entity, all of those are pretty much arbitrary (however let's try to choose descriptive naming):

TIP: Read more about Entity creation in the Maltego Technical Documentation.



The **Next** button will open another dialog box, in which you need to define the main property of a newly created Entity, in our case it will be a name of the token:



Now let's move onto fetching Ethereum network data related to the address.

Working with External Data Sources

Our task will be to visualize the activity related to an Ethereum address `0xB5DbC815D72D05fB6453DAB1263E015CA9F792D3`

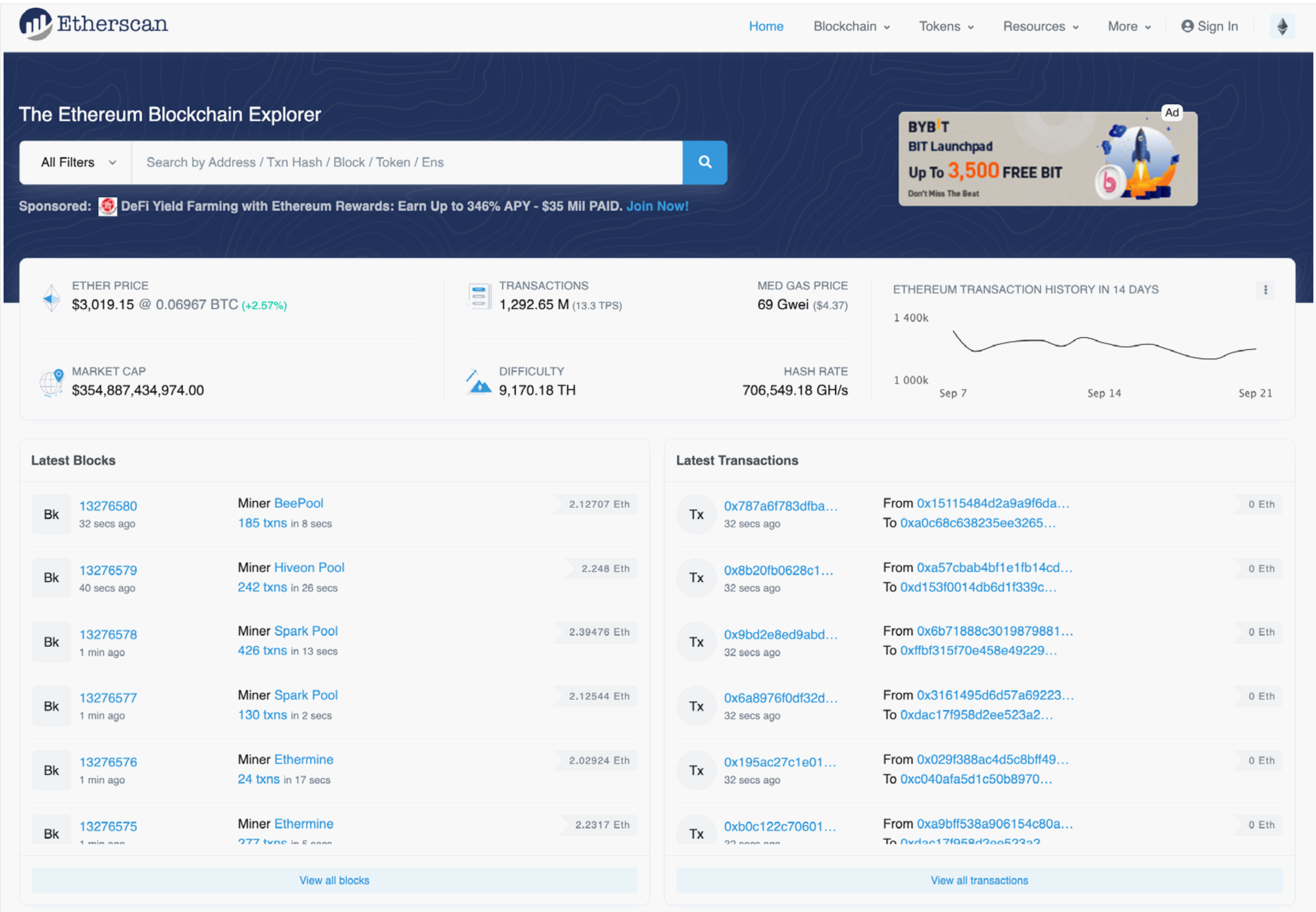
As a third party exploration tool we will be using the free service Etherscan.io, which is an online tool for fetching Ethereum blockchain data. It allows searching for information on transactions with both ethereum and tokens. Basically, it is analogous to the blockchain.info tool, but is specifically built for Ethereum blockchain exploring.

For our use case, we will focus only on token transactions, so as to not mix digital currency and all other assets on one graph.

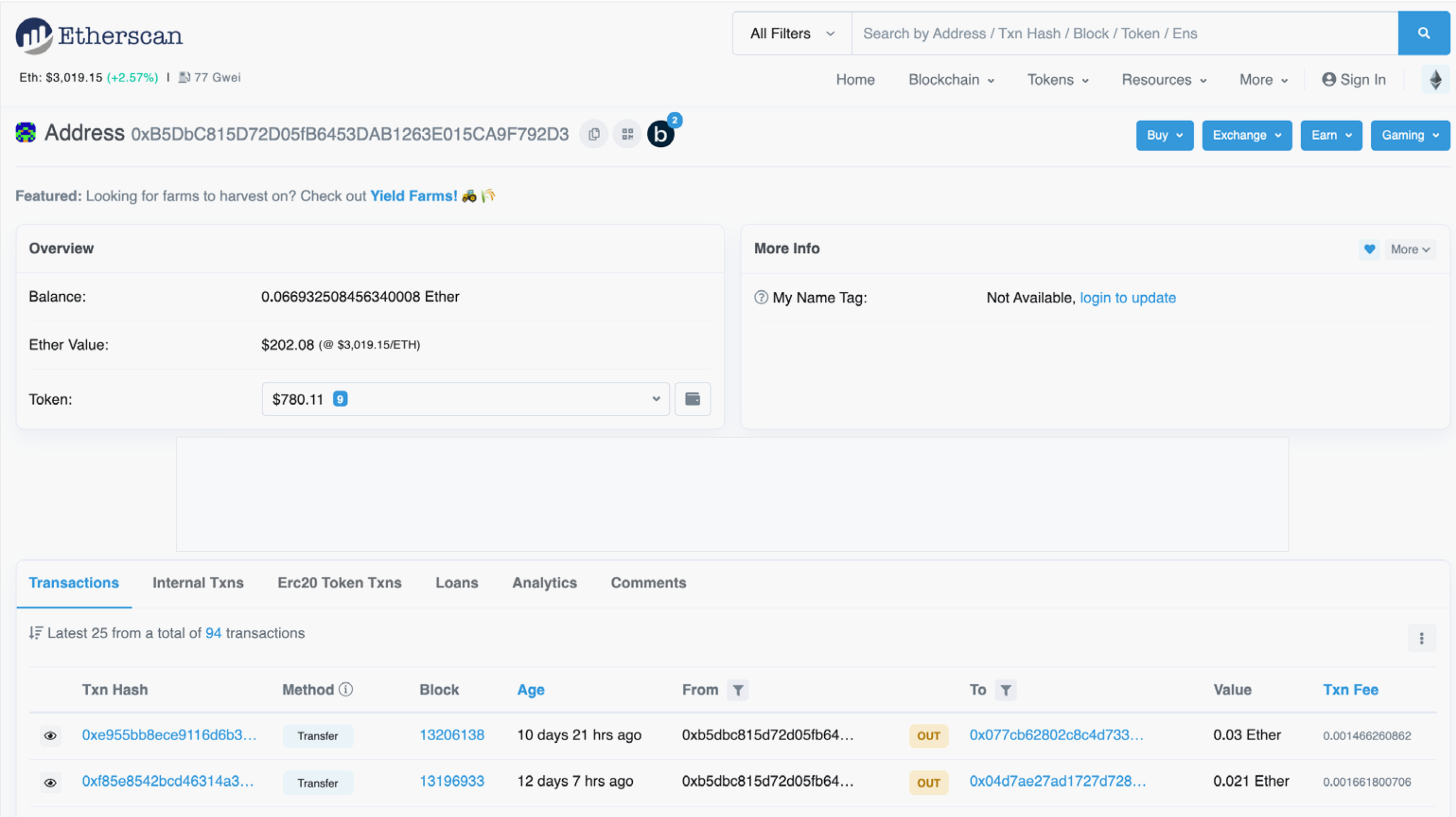
This is how the main page of Etherscan.io looks:

TIP: Read more about working with external data sources in Maltego in the Maltego technical documentation.

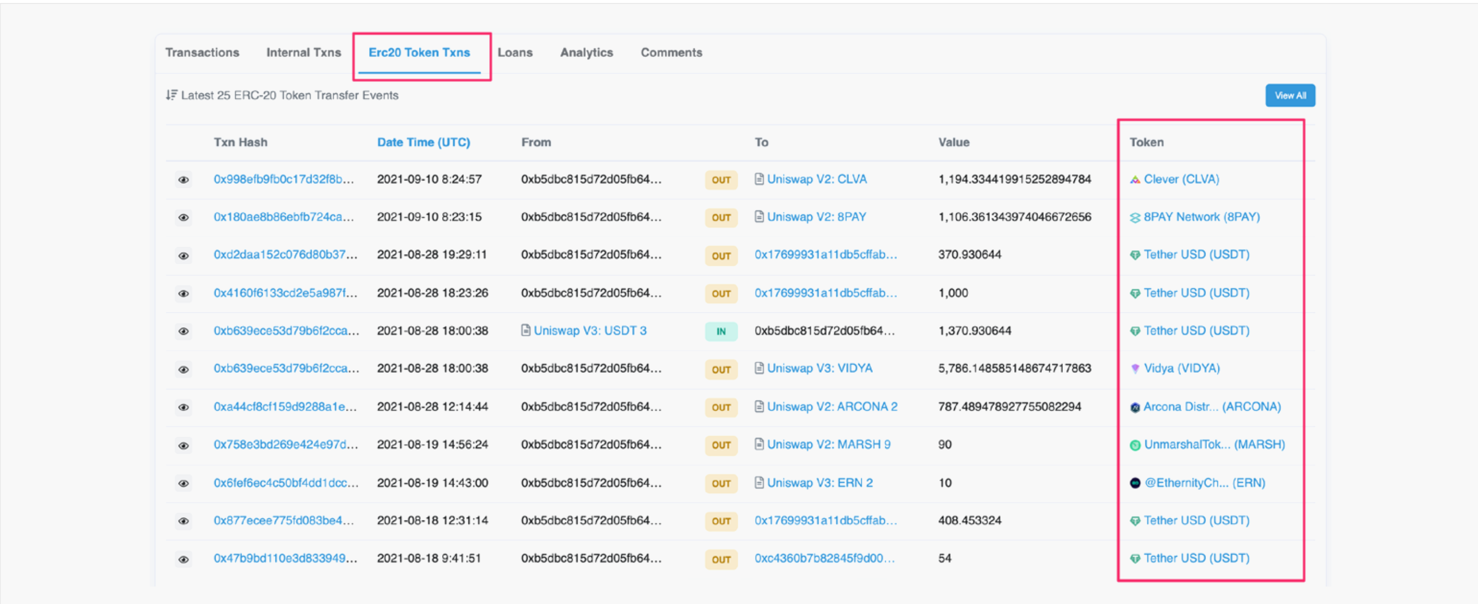
NOTE: *Ethereum network is pretty complex, so in this article we will not explain many concepts of it in much detail, and will focus only on the data relevant to our visualization task performed with Maltego, which is a very basic step in understanding the visualization possibilities.*
For those readers who want to take a really deep dive into technical aspects of Ethereum, I recommend an excellent book "Mastering Ethereum" by A. Antonopolous and G. Wood.



After entering an address of interest, we see a list of transactions associated with it:



However, it is the tab **Erc20 Token Txns** which is of special interest for us, as it lists all the transactions which involve different tokens (ERC20 stands for the official Token Standard used for token transactions on Ethereum network). Right away, you see a few different token names involved in transactions listed on the rightmost column:



Txn Hash	Date Time (UTC)	From		To	Value	Token
0x998efb9fb0c17d32f8b…	2021-09-10 8:24:57	0xb5dbc815d72d05fb64…	OUT	Uniswap V2: CLVA	1,194.334419915252894784	Clever (CLVA)
0x180ae8b86ebfb724ca…	2021-09-10 8:23:15	0xb5dbc815d72d05fb64…	OUT	Uniswap V2: 8PAY	1,106.361343974046672656	8PAY Network (8PAY)
0xd2daa152c076d80b37…	2021-08-28 19:29:11	0xb5dbc815d72d05fb64…	OUT	0x17699931a11db5cffab…	370.930644	Tether USD (USDT)
0x4160f6133cd2e5a987f…	2021-08-28 18:23:26	0xb5dbc815d72d05fb64…	OUT	0x17699931a11db5cffab…	1,000	Tether USD (USDT)
0xb639ece53d79b6f2cca…	2021-08-28 18:00:38	Uniswap V3: USDT 3	IN	0xb5dbc815d72d05fb64…	1,370.930644	Tether USD (USDT)
0xb639ece53d79b6f2cca…	2021-08-28 18:00:38	0xb5dbc815d72d05fb64…	OUT	Uniswap V3: VIDYA	5,786.148585148674717863	Vidya (VIDYA)
0xa44cf8cf159d9288a1e…	2021-08-28 12:14:44	0xb5dbc815d72d05fb64…	OUT	Uniswap V2: ARCONA 2	787.489478927755082294	Arcona Distr… (ARCONA)
0x758e3bd269e424e97d…	2021-08-19 14:56:24	0xb5dbc815d72d05fb64…	OUT	Uniswap V2: MARSH 9	90	UnmarshalTok… (MARSH)
0x6fef6ec4c50bf4dd1dcc…	2021-08-19 14:43:00	0xb5dbc815d72d05fb64…	OUT	Uniswap V3: ERN 2	10	@EthernityCh… (ERN)
0x877ecee775fd083be4…	2021-08-18 12:31:14	0xb5dbc815d72d05fb64…	OUT	0x17699931a11db5cffab…	408.453324	Tether USD (USDT)
0x47b9bd110e3d833949…	2021-08-18 9:41:51	0xb5dbc815d72d05fb64…	OUT	0xc4360b7b82845f9d00…	54	Tether USD (USDT)

Here we might expand the list of transactions using the **View All** button, and in the bottom of the list there's a link for exporting the data in CSV format. This is exactly what we need for further visualization:

0x4434e626111299a9ae...	2021-05-07 16:16:51	0xb5dbc815d72d05fb64...	OUT	Uniswap V2: MCH 2	25	MEME CASH To... (MCH)
0xbb15d394fb45e8fb2f9...	2021-05-04 17:26:39	0x179ce48f234b15a145...	IN	0xb5dbc815d72d05fb64...	1,106.361343974046672656	8PAY Network (8PAY)
0xbb4352b735514d558b...	2021-05-04 17:24:56	0x179ce48f234b15a145...	IN	0xb5dbc815d72d05fb64...	1,138.021339738385319615	POLVEN (POLVEN)
0xdaf2ededb91439852a...	2021-05-04 17:24:56	0x179ce48f234b15a145...	IN	0xb5dbc815d72d05fb64...	341.758924681770393405	Beyond Finan... (BYN)
0xf6659dbe49f30cf66d9...	2021-05-04 17:24:55	0x179ce48f234b15a145...	IN	0xb5dbc815d72d05fb64...	981.245991033662843921	Polkalokr (LKR)
0x34181f5b01bc1df48b2...	2021-05-04 17:23:32	0x179ce48f234b15a145...	IN	0xb5dbc815d72d05fb64...	311.185771688730595667	GamyFi (GFX)
0x2779743f2719dc1199...	2021-05-04 17:23:14	0x179ce48f234b15a145...	IN	0xb5dbc815d72d05fb64...	436.146600543648483533	DCTDAO (DCTD)

Show 50 Records

First < Page 1 of 1 > Last

[Download **CSV Export**]

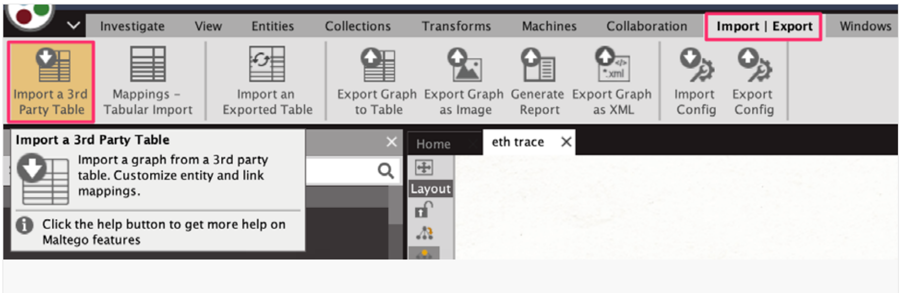
Importing Dataset into Maltego

So far so good, after exporting the data to a CSV file we have a nice looking table of all token transactions, which contains transaction hash, sender, receiver and contract addresses, token names, symbols and values, and the transaction timestamp. You can open it in Excel or another spreadsheet program to clean up the dates and numerical / decimal separators formatting if needed.

This is enough for an informative visualization of Ethereum address network activity.

	A	B	C	D	E	F	G	H
1	Txhash	DateTime	From	To	Value	ContractAddress	TokenName	TokenSymbol
2	0x82c9847654d4276b5f16c70255cbacf9430d04a347d48ae8726bf117628aacf1	09.03.2021 15:31	0xb5b33fbb875be58abba3290532f2e83ba7a1f788	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	16	0x2216e873ea4282ebef7a02ac5aea220be6391a7c	smol	SMOL
3	0x077d5834d6fb77d51079faff16a91ba53cce1b4e6f588fb3fd00a717f8c169ee	09.03.2021 20:03	0xffa3933a870be2b635d671eabe7853764caca020	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	25	0xa4e7414fcba1af15203030c6daac630df8f16aea	MEME CASH Token	MCH
4	0xe24e4bb0a7e61bbd1c36620f2c2104feafa79a4bf365dafa3af87287a49206f2	12.03.2021 22:01	0xf8d184723887b3914587a6e7d0757c4026af1640	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	3,500	0x0f71b8de197a1c84d31de0f1fa7926c365f052b3	Arcona Distribution Contract	ARCONA
5	0xcd954ce6c895fc942191f4e540e6c57c778e5a0abeca45d71e32eab0802cfc85	17.03.2021 04:47	0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	1.04	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE
6	0x0755a7463cb28df4f8850646b630804eec723f4559d41218a850126def37025e	17.03.2021 12:07	0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	57.27171	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE
7	0xfd526868db01f01d04cdcc8136efda17b978dd986bf9d03cdb3c0176269a67c6	17.03.2021 20:31	0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	201.04	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE
8	0x0b8e72890824a4680da142218666da467e0aae905f07921c0497b2392e4b56fc	19.03.2021 12:47	0x9f11b7e400da8591b6bacc6a78b9e36ad0057810	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	500,000	0xfef3884b603c33ef8ed4183346e093a173c94da6	MetaMorph	METM
9	0xda43bf3983fec1fa943a0626506094c57c698225c13b4e57389f14d2cca084d7	20.03.2021 07:58	0xeae8d7f17ee307cbb5261858ccada8cfd00de8cd	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	10,100	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE
10	0xec54996b848bfe9c19662c1d2daf920d40dff76d2da93d7bb3697b8246eaa0cd	20.03.2021 15:17	0xa26cc75e49d044048da95eca5b7c3321f9313128	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	11,000	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE
11	0x1638326d811110263b8559049ae52eb644c15848b6a8c8372f9ba4fa172f8392	20.03.2021 21:41	0x708396f17127c42383e3b9014072679b2f60b82f	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	198.64192	0x88ef27e69108b2633f8e1c184cc37940a075cc02	dego.finance	DEGO
12	0x063545747fcbdd846609176b23adade7b706392ca880b0f19e4f34c6e779e3c6	21.03.2021 15:25	0xeefb29492a7c78a58154682fb52393380f20670e	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	4,750	0xedb7b7842f7986a7f211d791e8f306c4ce82ba32	Polkazeck	ZCK
13	0xbfc70c335aa51bbda18876162bc055daf0afc6a1af9e454c57a4c19bcfba351d	23.03.2021 11:14	0xbefdff940dabfa39ecf04cf4b7978b2eee2fac9f	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	57,149	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE
14	0x37eee5dfdfe28cb69998428529c6c48fbd9954ac8c25e2e3ff06c7b4ea454b0a	25.03.2021 17:16	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x77d50919243dc6a7d7a19360006aeea1caf461e2	500,000	0xfef3884b603c33ef8ed4183346e093a173c94da6	MetaMorph	METM
15	0x35eed217327c994cdc9d08a3c153495ba100f67f7627ae41058bf49bd4088510	29.03.2021 22:09	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x1c87170cdbb3de04dce91bc99c04680d140d1982	40,000	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE
16	0x35eed217327c994cdc9d08a3c153495ba100f67f7627ae41058bf49bd4088510	29.03.2021 22:09	0x0d4a11d5eeaac28ec3f61d100daf4d40471f1852	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	2,553.08	0xdac17f958d2ee523a2206206994597c13d831ec7	Tether USD	USDT
17	0xab6bbb988310ef9cf2b33e07acf707d77462e14a7dcb823f3c5b3652fd00919c	29.03.2021 22:10	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x1c87170cdbb3de04dce91bc99c04680d140d1982	1,000	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE
18	0x4ffefd3615c295326661c8a9fdb79be8a0015c076e9641c57167983debbe5485	29.03.2021 23:53	0x3967905f7805dbaf4f06ba6481741f96d5eac859	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	244,609.26	0xd5930c307d7395ff807f2921f12c5eb82131a789	Bolt Token	BOLT
19	0x7d72e72449cee504b0ab9294644e2be84058d4256dc497071529e5b5c57ffca5	30.03.2021 00:31	0x3967905f7805dbaf4f06ba6481741f96d5eac859	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	306,722.14	0xd5930c307d7395ff807f2921f12c5eb82131a789	Bolt Token	BOLT
20	0x678e1bd0a6d65d1ec3717f34d13ec6b291c46cd25efc7c2e6b97a7a02e4b75dc	31.03.2021 14:27	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x25b13781aca8dd615bf522ad3aaa42f5aee7d5d9	259.35171	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE

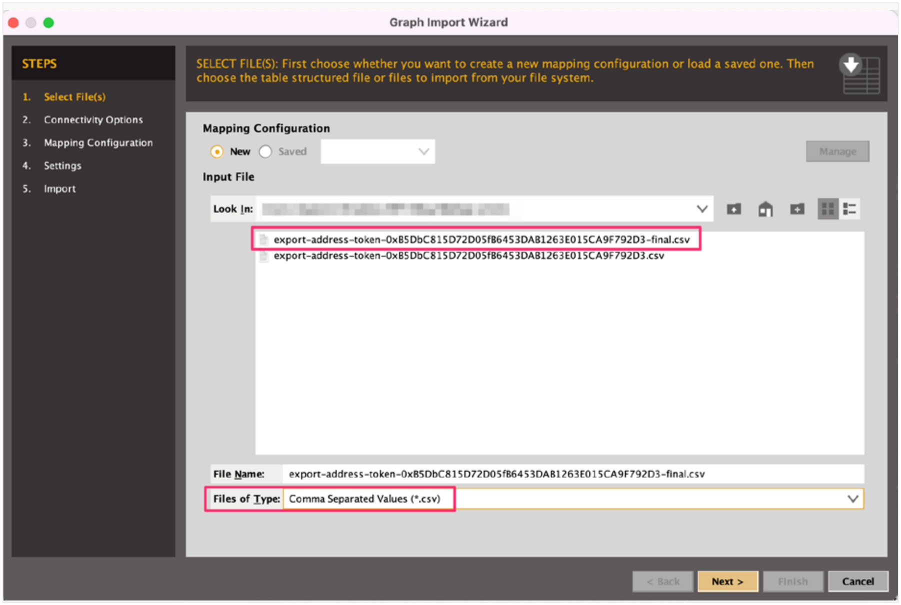
We start with selecting **Import a 3rd Party Table** option in the **Import | Export** menu:



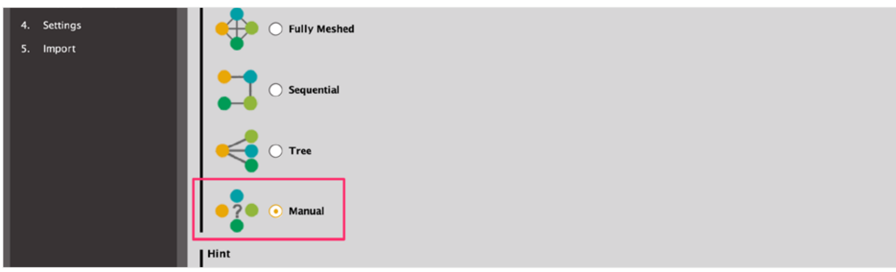
Now we are going to import this table into Maltego and define the relations between all the entries in it. Let's go step by step below.

Mapping data to Entities and links

The first step is to choose the CSV file you need to import (don't forget to also choose CSV type in the bottom dropdown list):

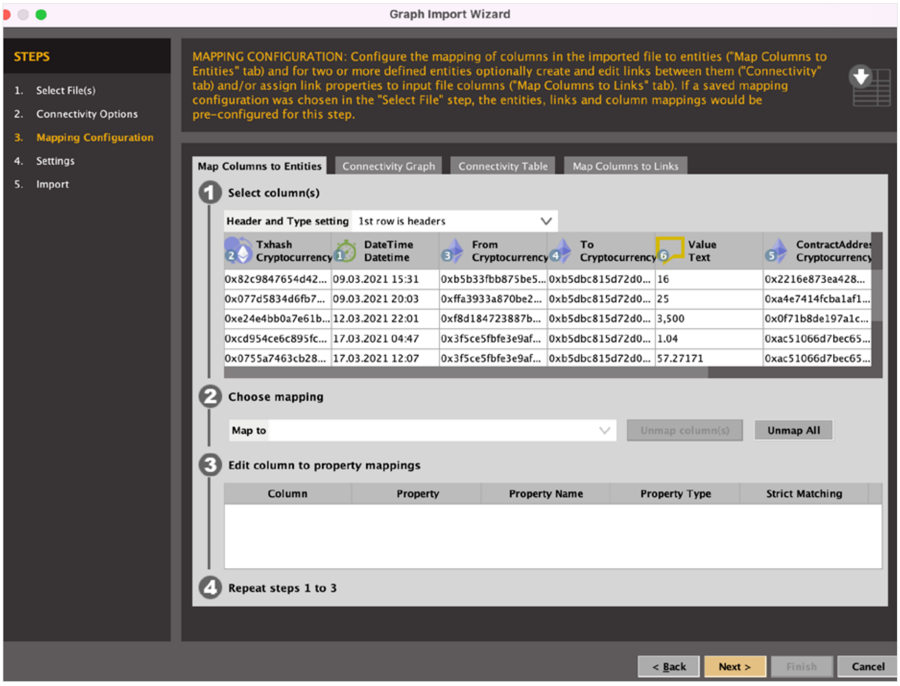


Next, you need to choose the **Connectivity Option**. As all connections between Entities are to be defined manually, you tick **Manual** option here:

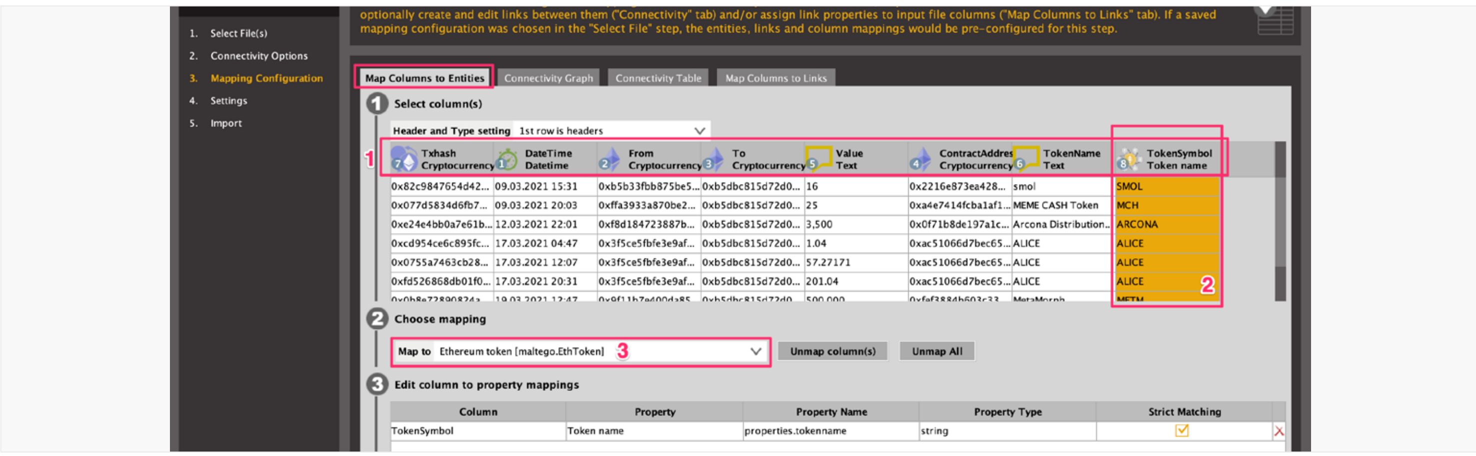


The next part is a creative one, as here you might define the relations between Entities the way you need them. Remember that these relations also affect the appearance of the graph, so you are in full control of how to visually represent the information. This gives you a lot of flexibility, but on the other hand it requires a very well thought out structure of the graph you are building. As we will see further, it is often a trial and error path, and you don't necessarily get a good and clean layout from the very first try.

At this step, Maltego tries to automatically guess (and mostly successfully) the types of Entities in each table column. So, transactions and addresses are already mapped correctly here:

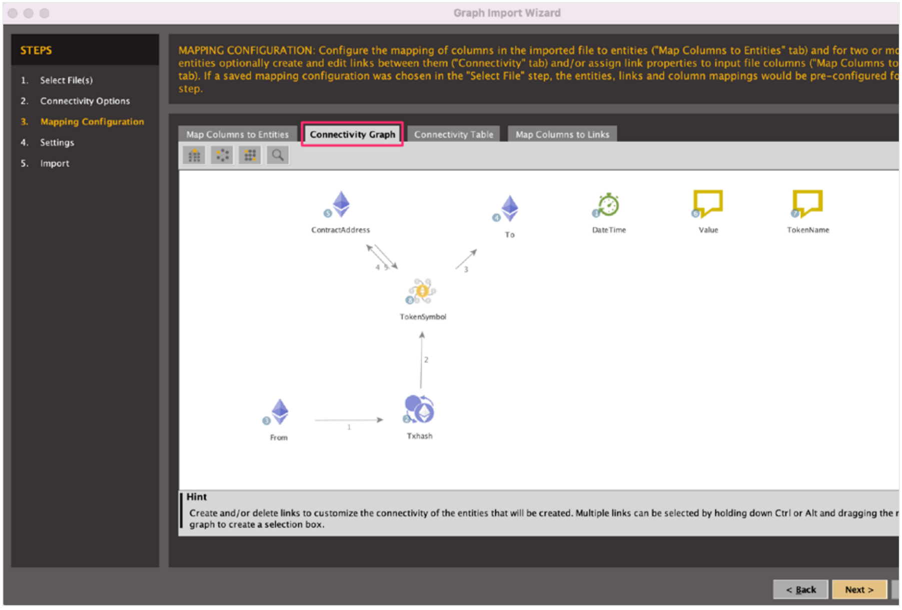


Next step is to re-define mapping where we need it. Specifically, we need to map the **TokenSymbol** column to the **Ethereum Token** Entity we created earlier:



Mappings in all columns are remaining as they were defined automatically by Maltego (1), and the rightmost column (2) with list of token symbols is mapped manually to **Ethereum Token** Entity.

Next step is where we define connections between the Entities **(Connectivity Graph)**. Here you can draw the connections between Entities using a mouse, just like on a regular Maltego graph:



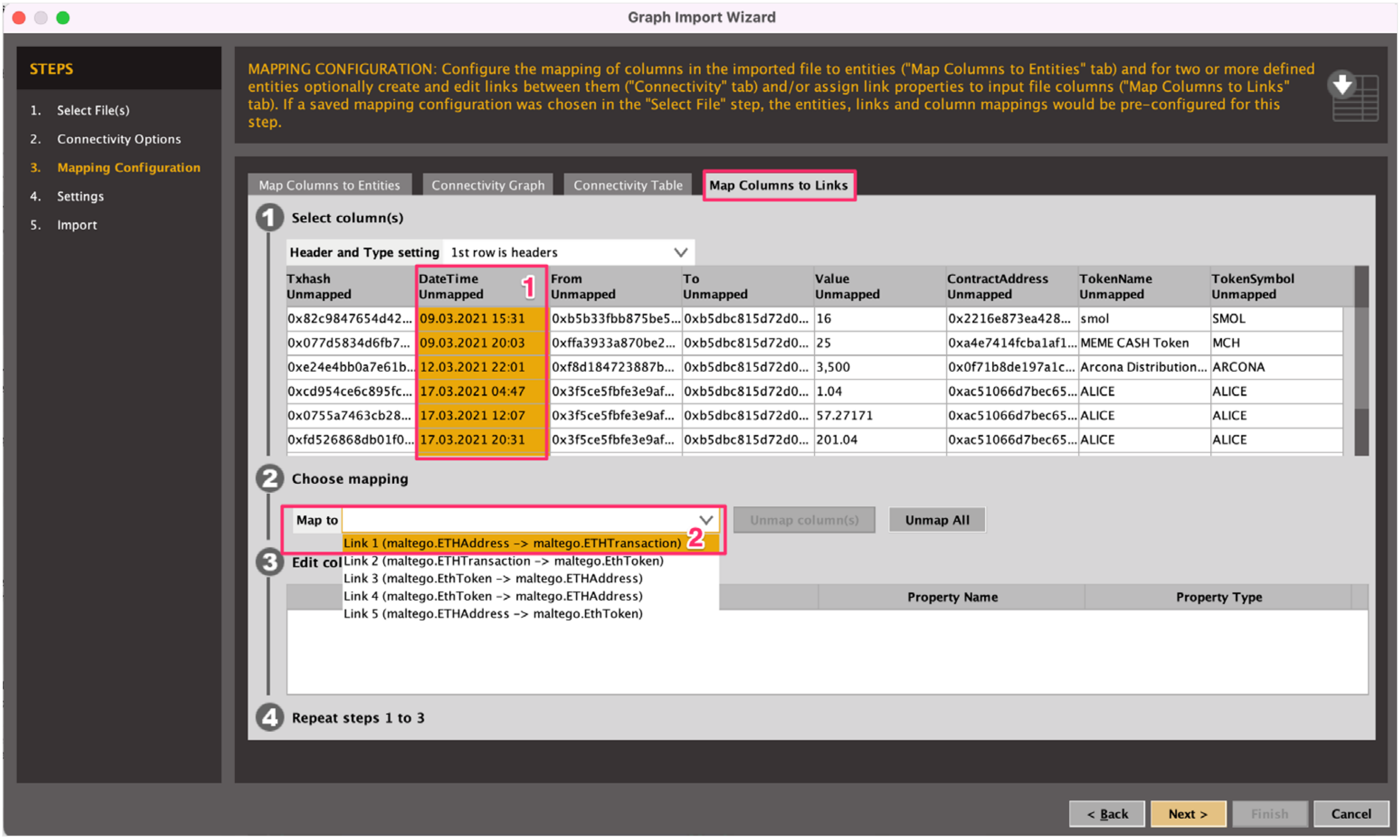
What would these connections look like? You decide. The easiest way to make it work is to define a verbal description (a *mantra*) of what you are trying to visualize. In our case, the *mantra* can be the following:

Address1 (From) has performed a **transaction** on a given **date**, which involved a **token** with a given **contract address,** in a given **amount,** and the token was transferred to **address2 (To).**

This should sound clear, except maybe the **Contract Address** part. Here, you should know that a **contract address** refers to the address location of the actual token contract that manages the logic for the tokens. So, a contract address is a property of a token itself and is not to be confused with sender and recipient addresses.

According to the mantra, we draw connections between Entities in a way that they reflect their relations in the process of the transaction. Two-way links 3 and 4 between **TokenSymbol** and **ContractAddress** mean that there's some interaction between these Entities thats exists, however it might or might not be relevant to the whole investigative graph, we don't know it yet (further you will see how to deal with such relations in a more elegant way).

Also notice that not all of the defined Entities are actually connected here, because some of them will be related to the links, which are defined in yet another step:

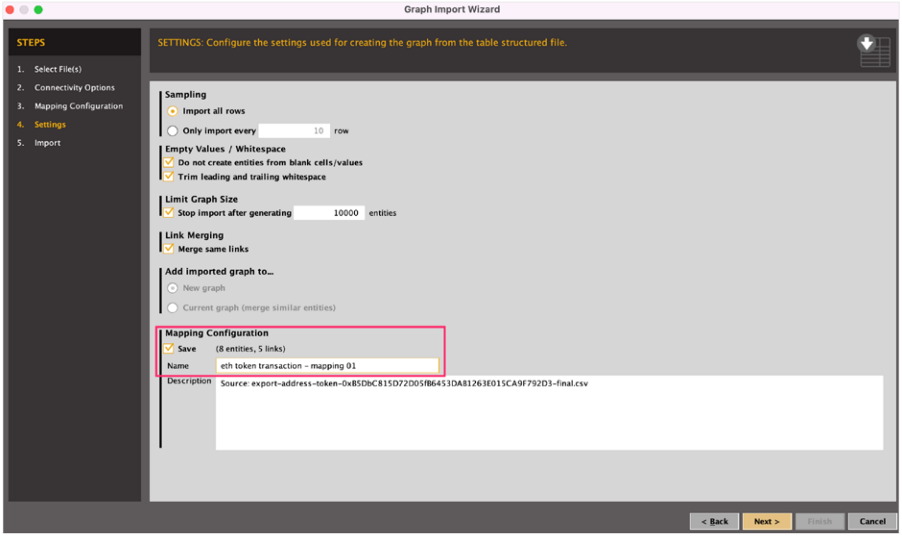


Here we **Map Columns To Links.** The example shown here is a mapping of the **Datetime** column (1) into a link between From address and transaction hash (2fotocas). This means that all such links will be just labelled with DateTime values on a final graph.

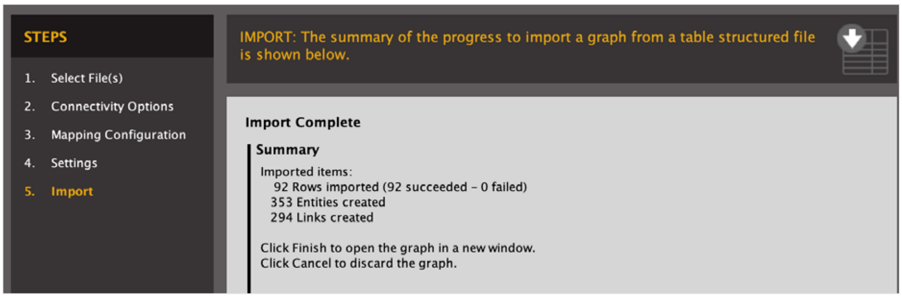
In the same way we will also create mappings for **Value** and **TokenName** columns, and the final mapping will be the following:

Link mapping	Verbal description
From › **DateTime** › Transaction	Sender address on a given **datetime** performs a transaction
Transaction › **Value** › TokenSymbol	Transaction involves **value (amount)** of given token
TokenSymbol › **TokenName** › To address	Token of a particular **name** is transferred to recipient’s address

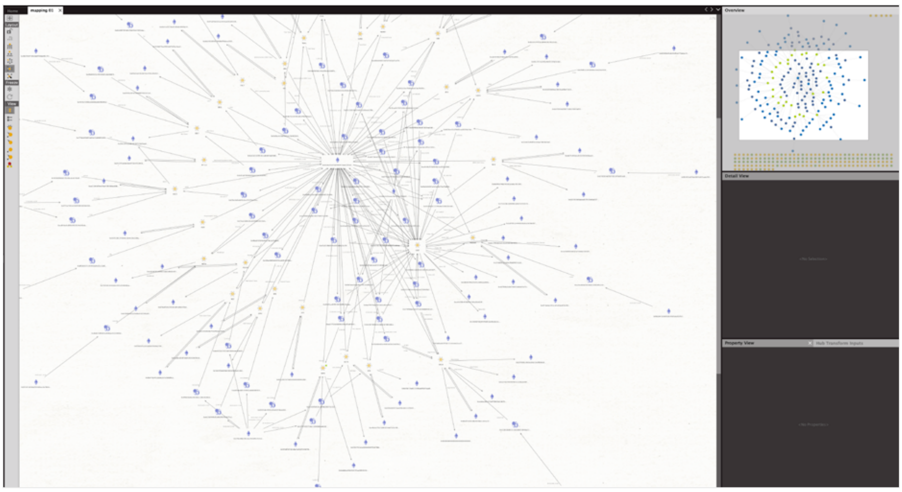
On the next step, we may optionally choose to save the mapping scheme for future use, so you don't have to perform all the steps again once you need to work with a new dataset of identical structure:



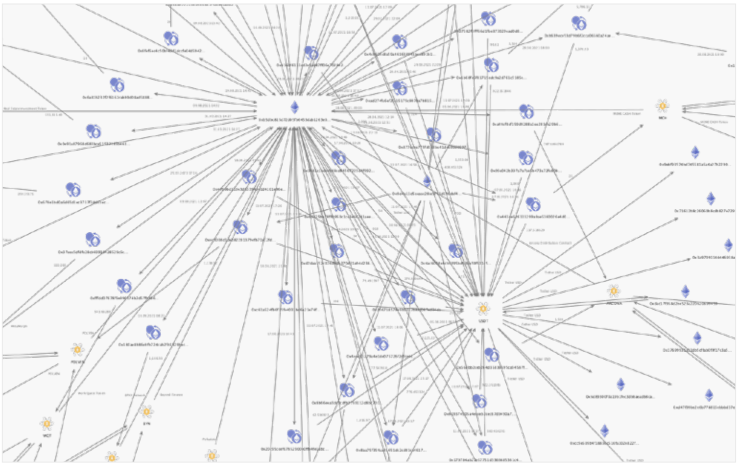
And the final step shows us some statistics as the data is imported into the new graph:

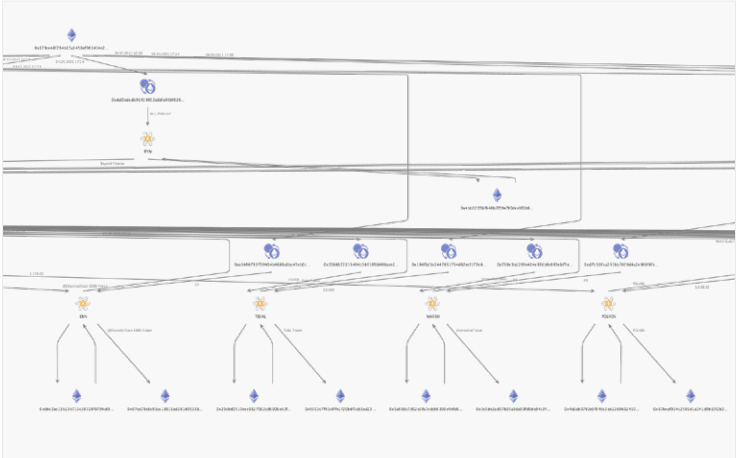


And voila! We have a new graph created, which is based on our dataset and defined mappings between columns, Entities and links:



What you see right away there are a bunch of 'hanging' Entities (at the bottom part of the **Overview** window): these are the ones which do not have any links to others. These can be deleted from the graph.
However, let's take a closer look at what we got, zooming into different parts of the graph and also trying different layouts like **Organic** and **Block:**





Well... it might be better than just a table format, and at least we can trace some relations here. But unfortunately it is still far from perfect. It looks a bit cluttered because of excessive links and labels, and overall not very optimal visual representation.

Let's think about what we can improve it.

Making Things Look Better

The following are some constraints and assumptions that can be important for future visual improvements of the graph.

There are some specific features of Ethereum transactions that could potentially be used for a more efficient visual representation:

- One Ethereum transaction always has only one input and one output, unlike Bitcoin transactions;
- An Ethereum token swap may happen within just one transaction, which in reality will consist of a few transfers, see here for example:

- But at the same time, this is still a single transaction with the same hash, and it happens in one moment of time. That is the reason that the dataset can contain duplicate transaction hashes with the same timestamps:

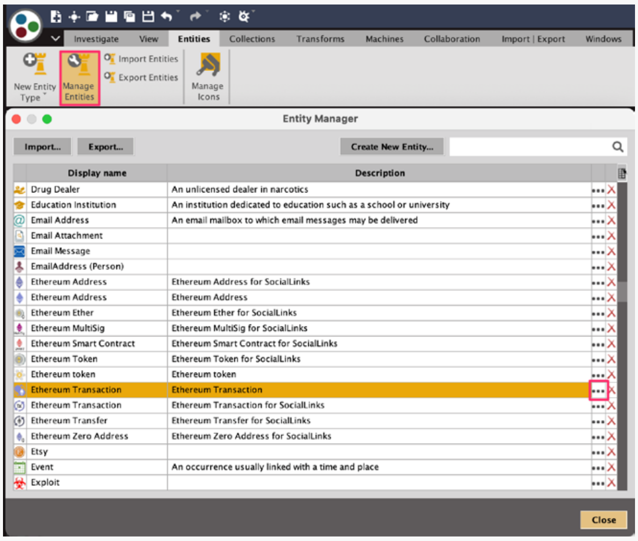
1	Txhash	DateTime	From	To
2	0x82c9847654d4276b5f16c70255cbacf9430d04a347d48ae8726bf117628aacf1	09.03.2021 15:31	0xb5b33fbb875be58abba3290532f2e83ba7a1f788	0xb5dbc815d72d05fb6453dab1263e015ca9f7
3	0x077d5834d6fb77d51079faff16a91ba53cce1b4e6f588fb3fd00a717f8c169ee	09.03.2021 20:03	0xffa3933a870be2b635d671eabe7853764caca020	0xb5dbc815d72d05fb6453dab1263e015ca9f7
4	0xe24e4bb0a7e61bbd1c36620f2c2104feafa79a4bf365dafa3af87287a49206f2	12.03.2021 22:01	0xf8d184723887b3914587a6e7d0757c4026af1640	0xb5dbc815d72d05fb6453dab1263e015ca9f7
5	0xcd954ce6c895fc942191f4e540e6c57c778e5a0abeca45d71e32eab0802cfc85	17.03.2021 04:47	0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be	0xb5dbc815d72d05fb6453dab1263e015ca9f7
6	0x0755a7463cb28df4f8850646b630804eec723f4559d41218a850126def37025e	17.03.2021 12:07	0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be	0xb5dbc815d72d05fb6453dab1263e015ca9f7
7	0xfd526868db01f01d04cdcc8136efda17b978dd986bf9d03cdb3c0176269a67c6	17.03.2021 20:31	0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be	0xb5dbc815d72d05fb6453dab1263e015ca9f7
8	0x0b8e72890824a4680da142218666da467e0aae905f07921c0497b2392e4b56fc	19.03.2021 12:47	0x9f11b7e400da8591b6bacc6a78b9e36ad0057810	0xb5dbc815d72d05fb6453dab1263e015ca9f7
9	0xda43bf3983fec1fa943a0626506094c57c698225c13b4e57389f14d2cca084d7	20.03.2021 07:58	0xeae8d7f17ee307cbb5261858ccada8cfd00de8cd	0xb5dbc815d72d05fb6453dab1263e015ca9f7
10	0xec54996b848bfe9c19662c1d2daf920d40dff76d2da93d7bb3697b8246eaa0cd	20.03.2021 15:17	0xa26cc75e49d044048da95eca5b7c3321f9313128	0xb5dbc815d72d05fb6453dab1263e015ca9f7
11	0x1638326d811110263b8559049ae52eb644c15848b6a8c8372f9ba4fa172f8392	20.03.2021 21:41	0x708396f17127c42383e3b9014072679b2f60b82f	0xb5dbc815d72d05fb6453dab1263e015ca9f7
12	0x063545747fcbdd846609176b23adade7b706392ca880b0f19e4f34c6e779e3c6	21.03.2021 15:25	0xeefb29492a7c78a58154682fb52393380f20670e	0xb5dbc815d72d05fb6453dab1263e015ca9f7
13	0xbfc70c335aa51bbda18876162bc055daf0afc6a1af9e454c57a4c19bcfba351d	23.03.2021 11:14	0xbefdff940dabfa39ecf04cf4b7978b2eee2fac9f	0xb5dbc815d72d05fb6453dab1263e015ca9f7
14	0x37eee5dfdfe28cb69998428529c6c48fbd9954ac8c25e2e3ff06c7b4ea454b0a	25.03.2021 17:16	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x77d50919243dc6a7d7a19360006aeea1caf
15	0x35eed217327c994cdc9d08a3c153495ba100f67f7627ae41058bf49bd4088510	29.03.2021 22:09	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x1c87170cdbb3de04dce91bc99c04680d140c
16	0x35eed217327c994cdc9d08a3c153495ba100f67f7627ae41058bf49bd4088510	29.03.2021 22:09	0x0d4a11d5eeaac28ec3f61d100daf4d40471f1852	0xb5dbc815d72d05fb6453dab1263e015ca9f7
17	0xab6bbb988310ef9cf2b33e07acf707d77462e14a7dcb823f3c5b3652fd00919c	29.03.2021 22:10	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x1c87170cdbb3de04dce91bc99c04680d140c
18	0x4ffefd3615c295326661c8a9fdb79be8a0015c076e9641c57167983debbe5485	29.03.2021 23:53	0x3967905f7805dbaf4f06ba6481741f96d5eac859	0xb5dbc815d72d05fb6453dab1263e015ca9f7
19	0x7d72e72449cee504b0ab9294644e2be84058d4256dc497071529e5b5c57ffca5	30.03.2021 00:31	0x3967905f7805dbaf4f06ba6481741f96d5eac859	0xb5dbc815d72d05fb6453dab1263e015ca9f7
20	0x678e1bd0a6d65d1ec3717f34d13ec6b291c46cd25efc7c2e6b97a7a02e4b75dc	31.03.2021 14:27	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x25b13781aca8dd615bf522ad3aaa42f5aee7
21	0x0e91a87968d683bce51532183b4619ad9b30aee41f6edf8cc5a48dcef0aa156b	31.03.2021 14:27	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x25b13781aca8dd615bf522ad3aaa42f5aee7
22	0x568bb2ceb264d35836595ca94587fa9346d646ce9e3108542a2312e616d08817	08.04.2021 05:23	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x1a1fe6f955b6567a75dddec526320a4a1a21
23	0x972b4eb5215d5fdeecdde79c48673e8f90a2bbbfd4012312ba220565eda21ec5	11.04.2021 22:01	0xe1b86e609da36a80561e47c3724420d4870888e3	0xb5dbc815d72d05fb6453dab1263e015ca9f7
24	0xaa57752a58d68b185ba2f1a85f2e56620e34d4e1eb9a19b5ba8743c739509a0f	11.04.2021 22:11	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x6e8abba440a58421f5eec9892b19c1a72c55
25	0xaa57752a58d68b185ba2f1a85f2e56620e34d4e1eb9a19b5ba8743c739509a0f	11.04.2021 22:11	0x0d4a11d5eeaac28ec3f61d100daf4d40471f1852	0xb5dbc815d72d05fb6453dab1263e015ca9f7
26	0x09e042b397a7a7ae5b473a72fe80b901e95a9fcf546fdf7ea5ca7c3165d58d58	11.04.2021 22:13	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x17699931a11db5cffab0f9f17c3a5fe37dc08
27	0x4cb603c8a59a461638841bec891b1df6368fd9b25e7798f43827d2f71a0132f7	13.04.2021 17:37	0xb5dbc815d72d05fb6453dab1263e015ca9f792d3	0x17699931a11db5cffab0f9f17c3a5fe37dc08

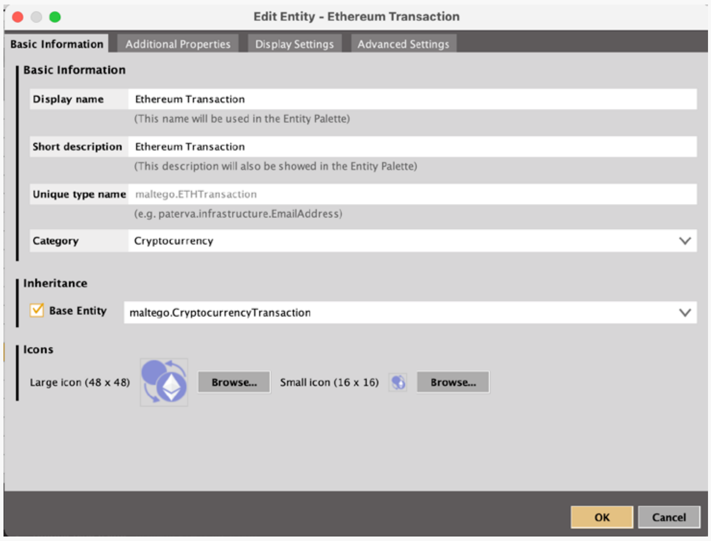
This means that some transactions would be two-way and include different tokens.

We want to implement nicer labelling of the transactions links, so first we will add a new column into our CSV file which merges value AND token symbol:

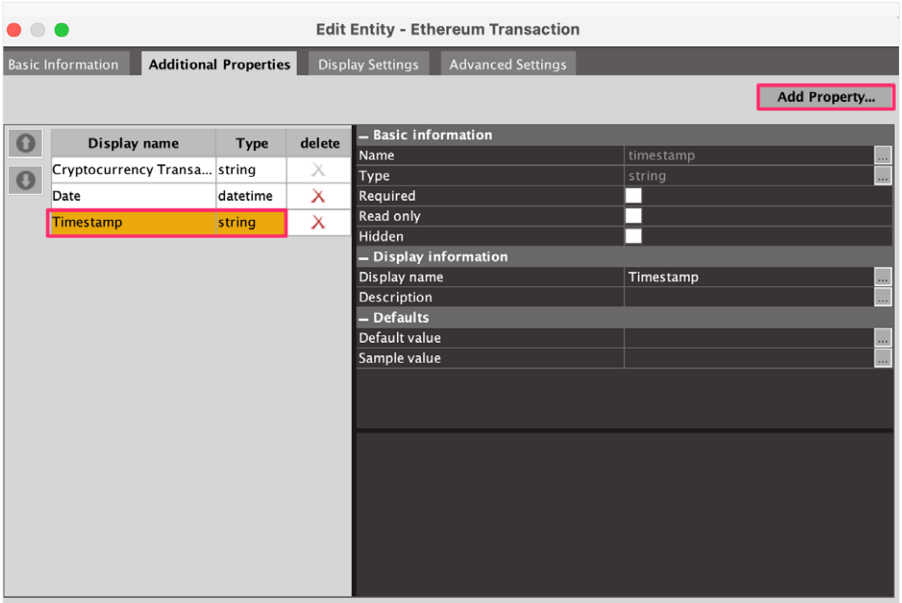
	Value	ContractAddress	TokenName	TokenSymbol	ValueLabel
3e015ca9f792d3	16	0x2216e873ea4282ebef7a02ac5aea220be6391a7c	smol	SMOL	16 SMOL
3e015ca9f792d3	25	0xa4e7414fcba1af15203030c6daac630df8f16aea	MEME CASH Token	MCH	25 MCH
3e015ca9f792d3	3,500	0x0f71b8de197a1c84d31de0f1fa7926c365f052b3	Arcona Distribution Contract	ARCONA	3500 ARCONA
3e015ca9f792d3	1.04	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE	1.04 ALICE
3e015ca9f792d3	57.27171	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE	57.27171 ALICE
3e015ca9f792d3	201.04	0xac51066d7bec65dc4589368da368b212745d63e8	ALICE	ALICE	201.04 ALICE
3e015ca9f792d3	500,000	0xfef3884b603c33ef8ed4183346e093a173c94da6	MetaMorph	METM	500000 METM
3e015ca9f792d3	10,100	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE	10100 VIBE
3e015ca9f792d3	11,000	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE	11000 VIBE
3e015ca9f792d3	198.64192	0x88ef27e69108b2633f8e1c184cc37940a075cc02	dego.finance	DEGO	198.64192 DEGO
3e015ca9f792d3	4,750	0xedb7b7842f7986a7f211d791e8f306c4ce82ba32	Polkazeck	ZCK	4750 ZCK
3e015ca9f792d3	57,149	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE	57149 VIBE
6aeea1caf461e2	500,000	0xfef3884b603c33ef8ed4183346e093a173c94da6	MetaMorph	METM	500000 METM
04680d140d1982	40,000	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE	40000 VIBE
3e015ca9f792d3	2,553.08	0xdac17f958d2ee523a2206206994597c13d831ec7	Tether USD	USDT	2553.082428 USDT
04680d140d1982	1,000	0xe8ff5c9c75deb346acac493c463c8950be03dfba	Vibe Coin	VIBE	1000 VIBE
3e015ca9f792d3	244,609.26	0xd5930c307d7395ff807f2921f12c5eb82131a789	Bolt Token	BOLT	244609.261911796 BOLT
3e015ca9f792d3	306,722.14	0xd5930c307d7395ff807f2921f12c5eb82131a789	Bolt Token	BOLT	306722.141166495 BOLT

Another cool thing to know: Maltego allows editing of the Entities and the way they are displayed, and it is possible to display more than one property for an Entity label on the graph. For that, we will use the Manage **Entities** from **Entities** menu. In the **Entity Manager** window we choose an Entity and click three dots to the right of it, which will bring up the **Edit Entity** window:

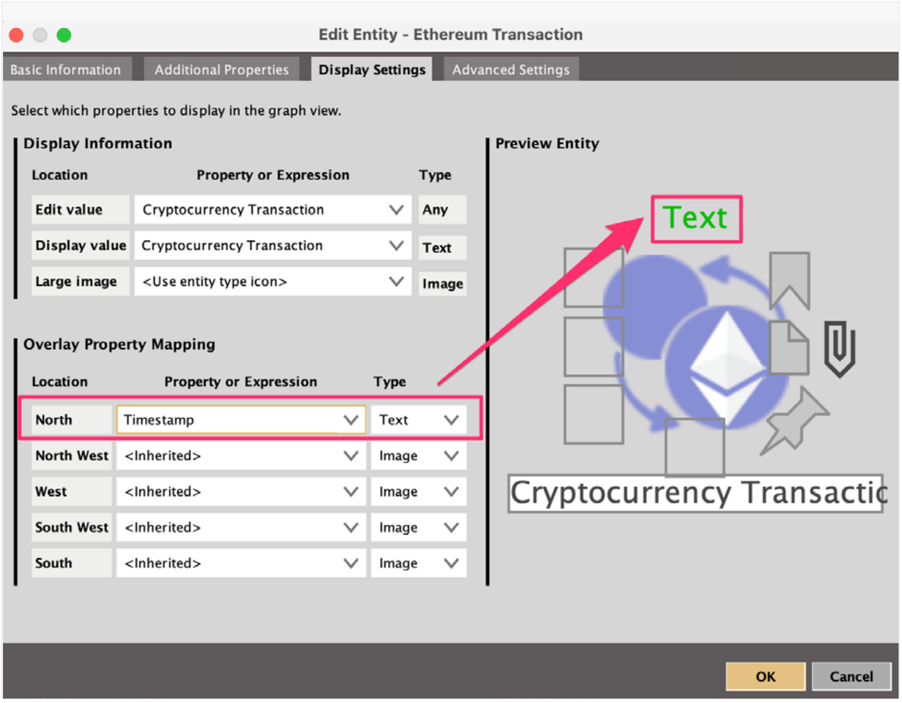




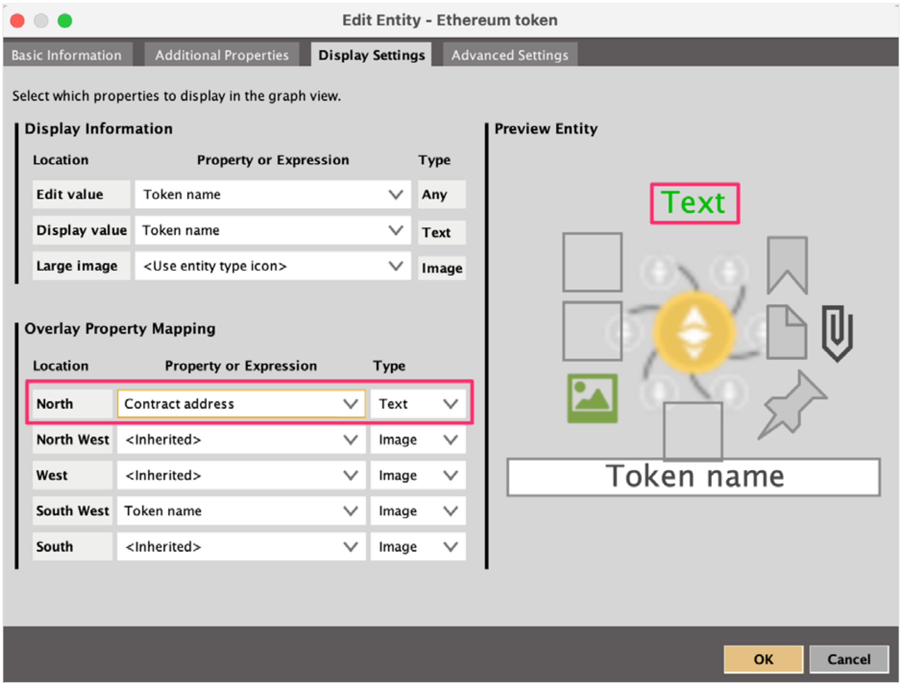
First, we need to add a new property for an existing Entity **(Additional Properties).** In our case, that would be a string representation of a timestamp:



Next, we need to modify the look of the Entity icon **(Display Settings).** What we can do here: By choosing an existing property and assigning it to a certain location around an Entity icon, we may display the value of this property on each icon.
In our case, we want to display a timestamp above the transaction icon:

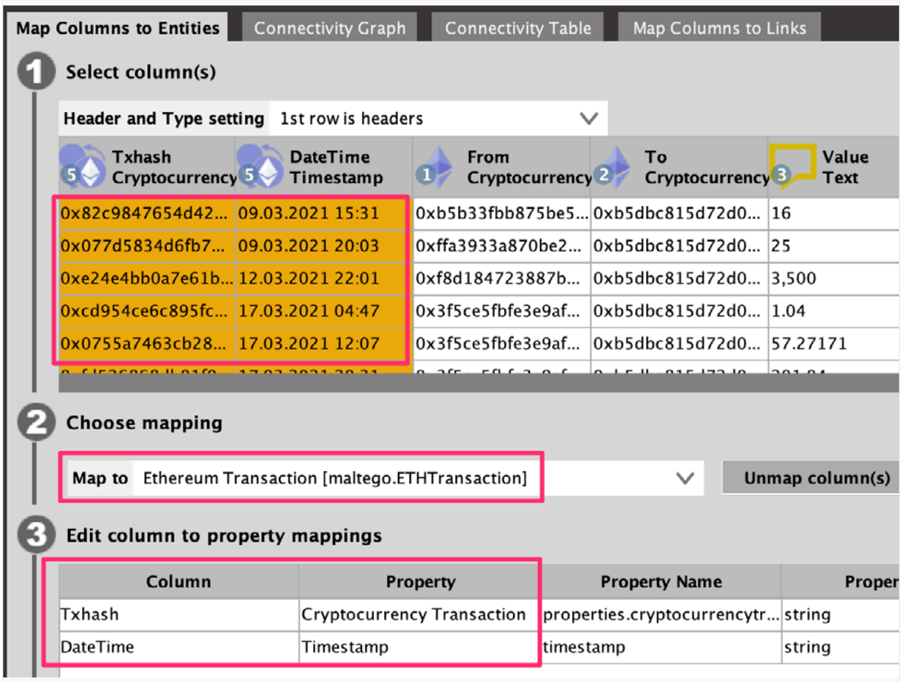


In exactly the same way, we are adding an additional property ContractAddress to an Ethereum Token Entity and setting it to be displayed above the Entity icon:

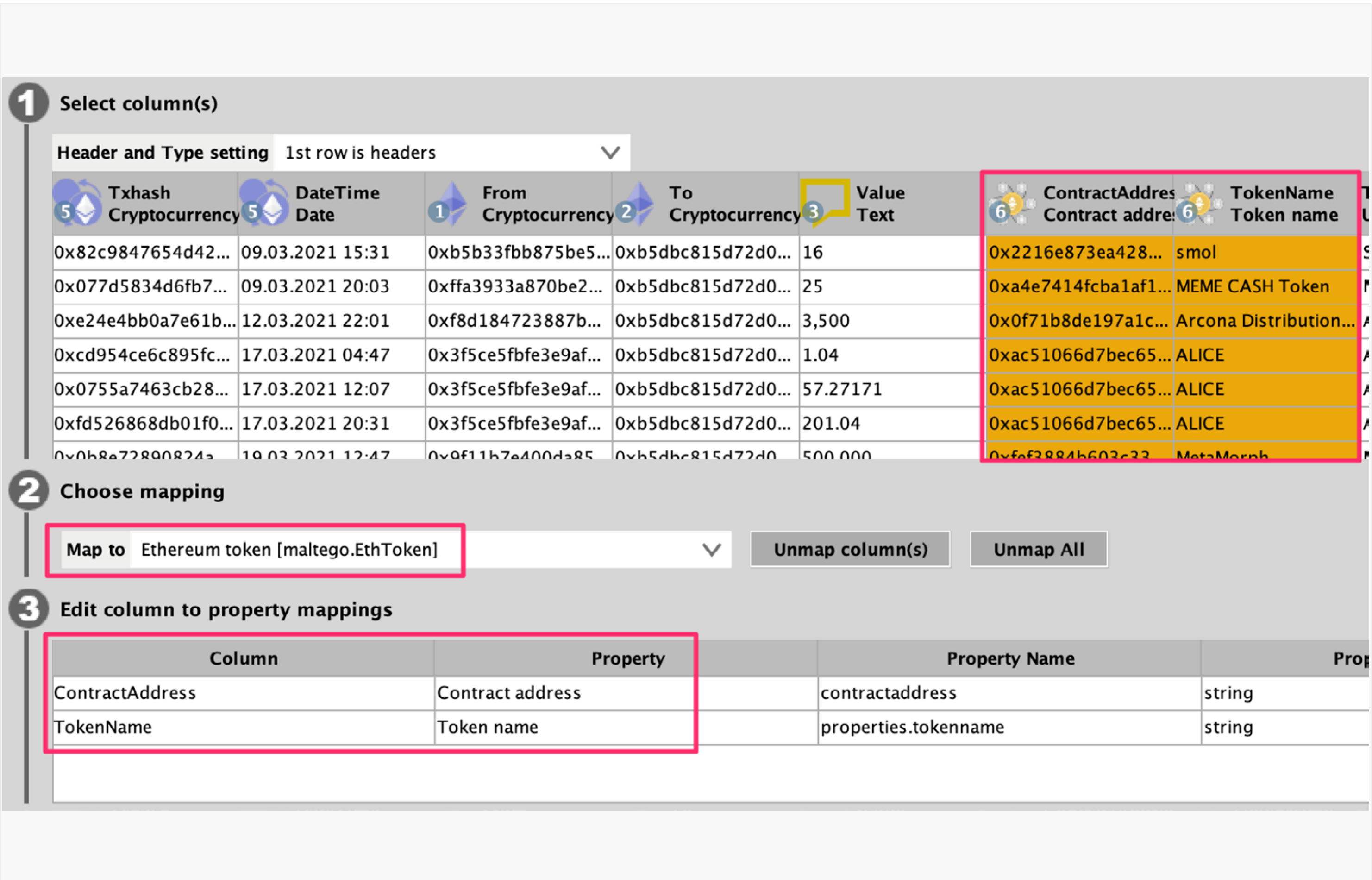


Later you will see how this simple feature can make a graph look much nicer.

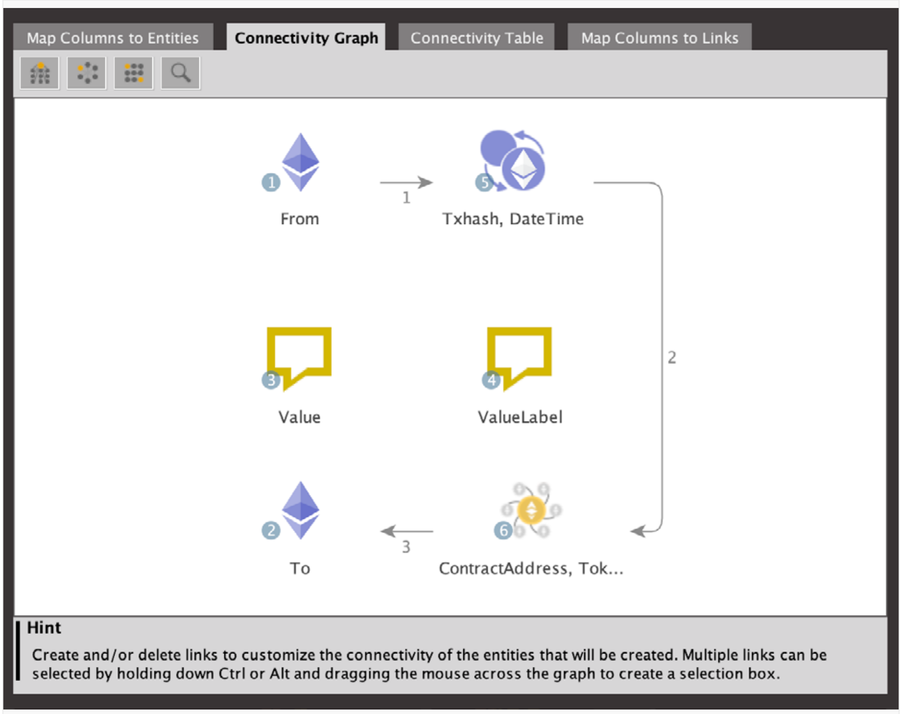
And now, we would need to repeat the data import process once again, this time with a new mapping configuration (and also save this configuration for future use). Here, we map both **Txhash** and **Datetime** columns into one Entity **(Ethereum Transaction),** and specify which properties correspond to each value:



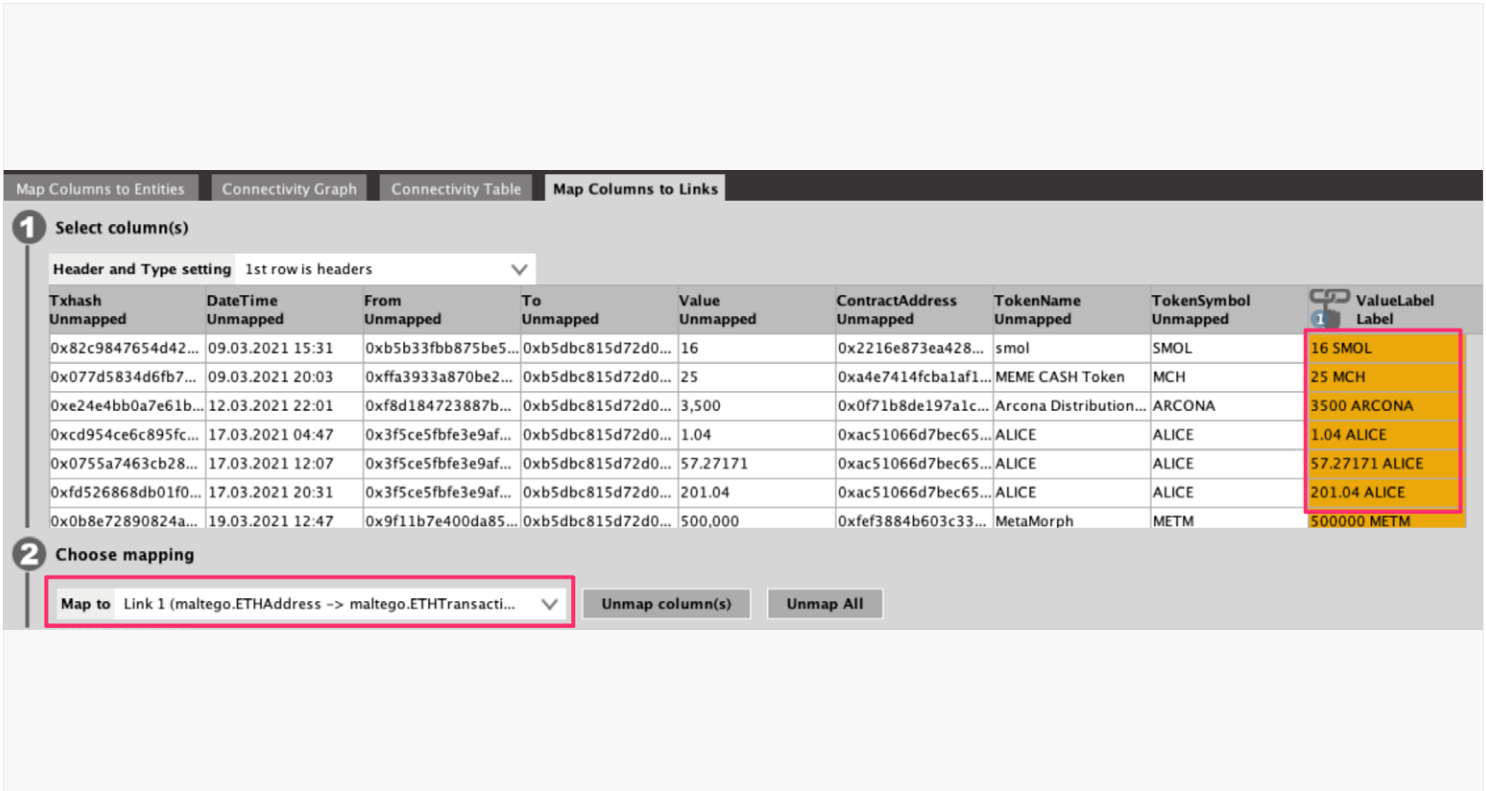
In the same way, we map **ContractAddress** and **TokenName** to **Ethereum token** Entity:



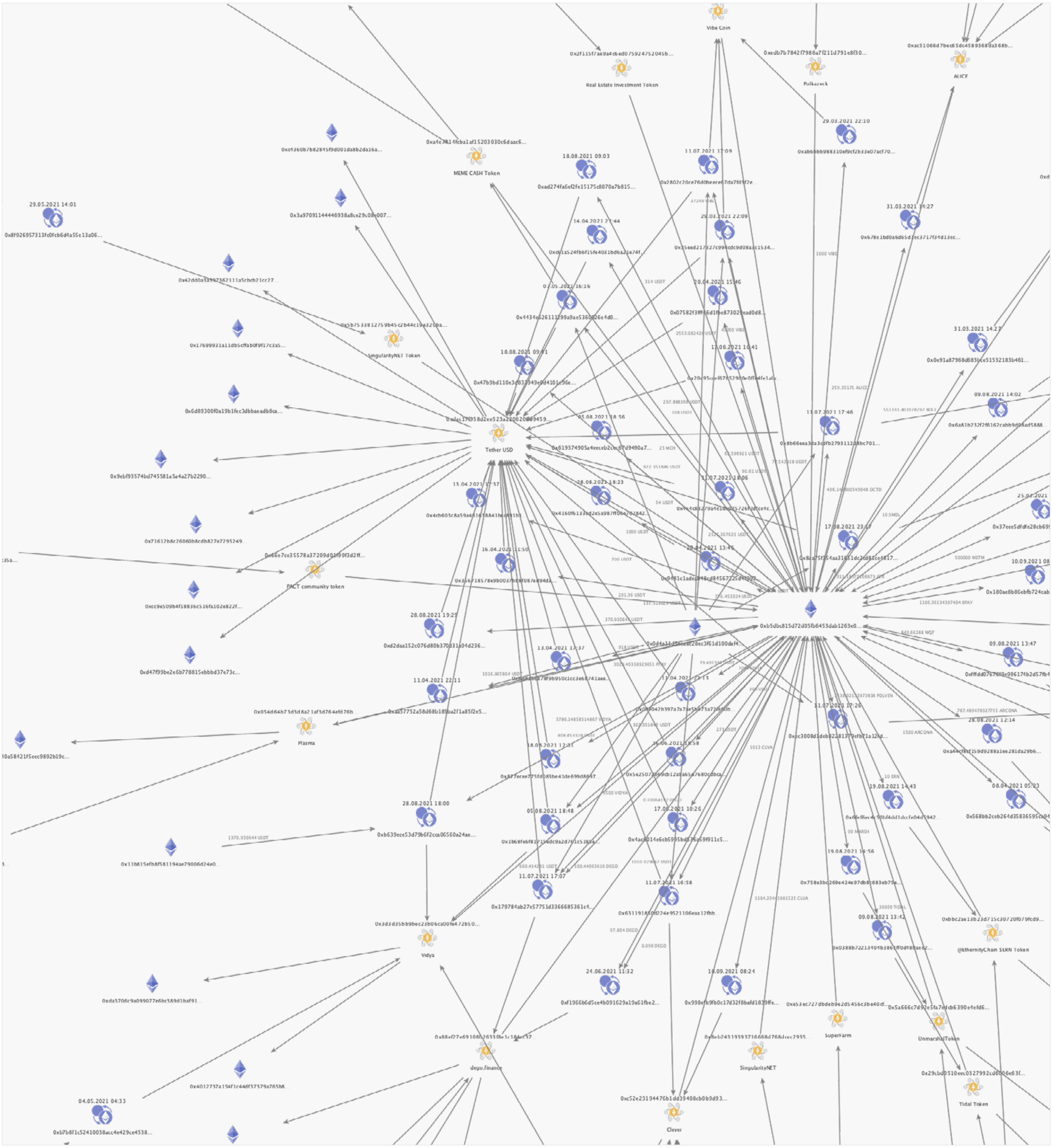
Afterwards, we define a new connectivity graph, which now looks a bit simpler than before – **Sender › Transaction › Token › Recipient:**



And finally, we map a new column **ValueLabel** to a link between **Sender** and **Transaction:**



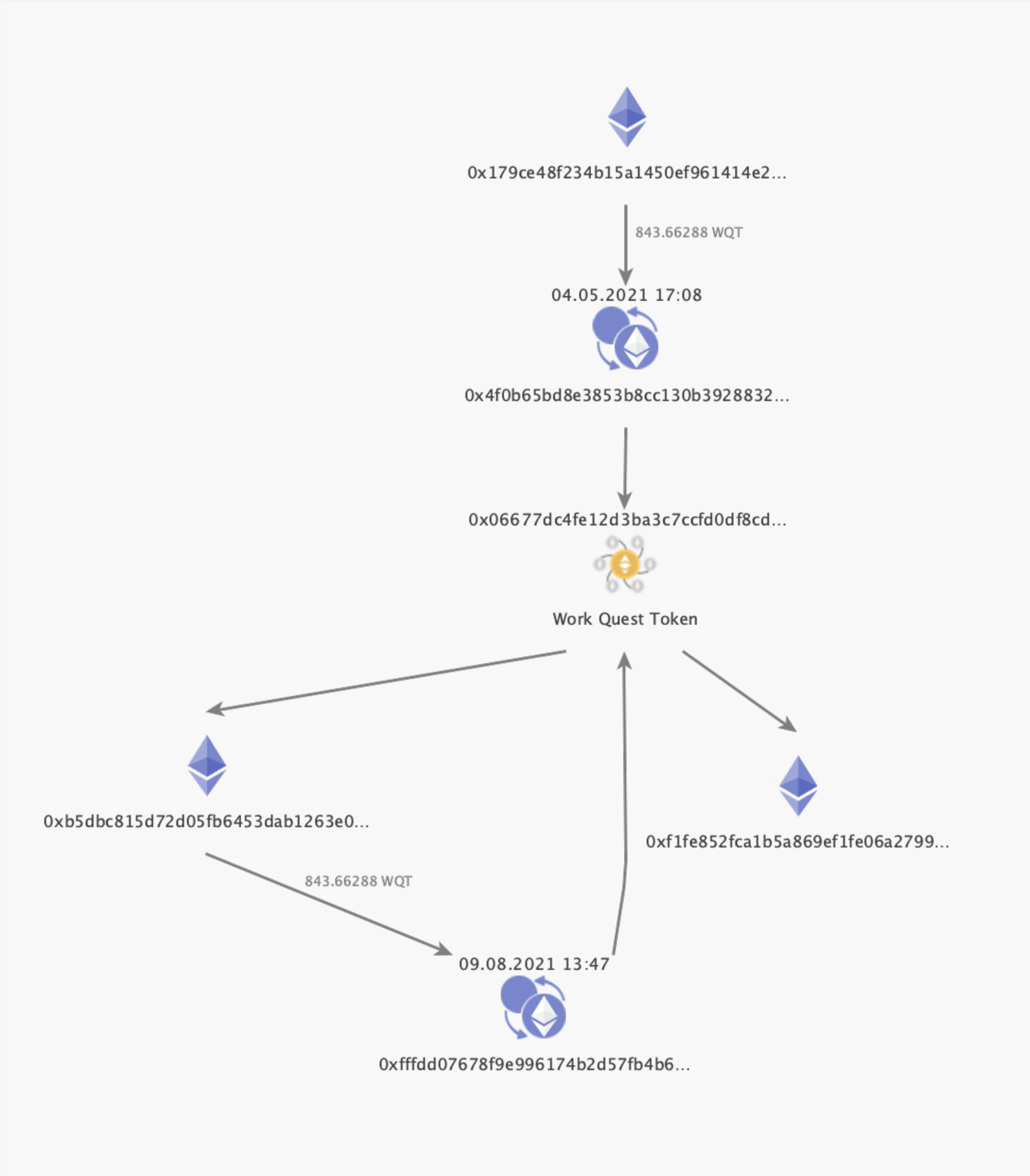
Now it's time to take a look at the final graph after the updated import with the new mapping configuration:



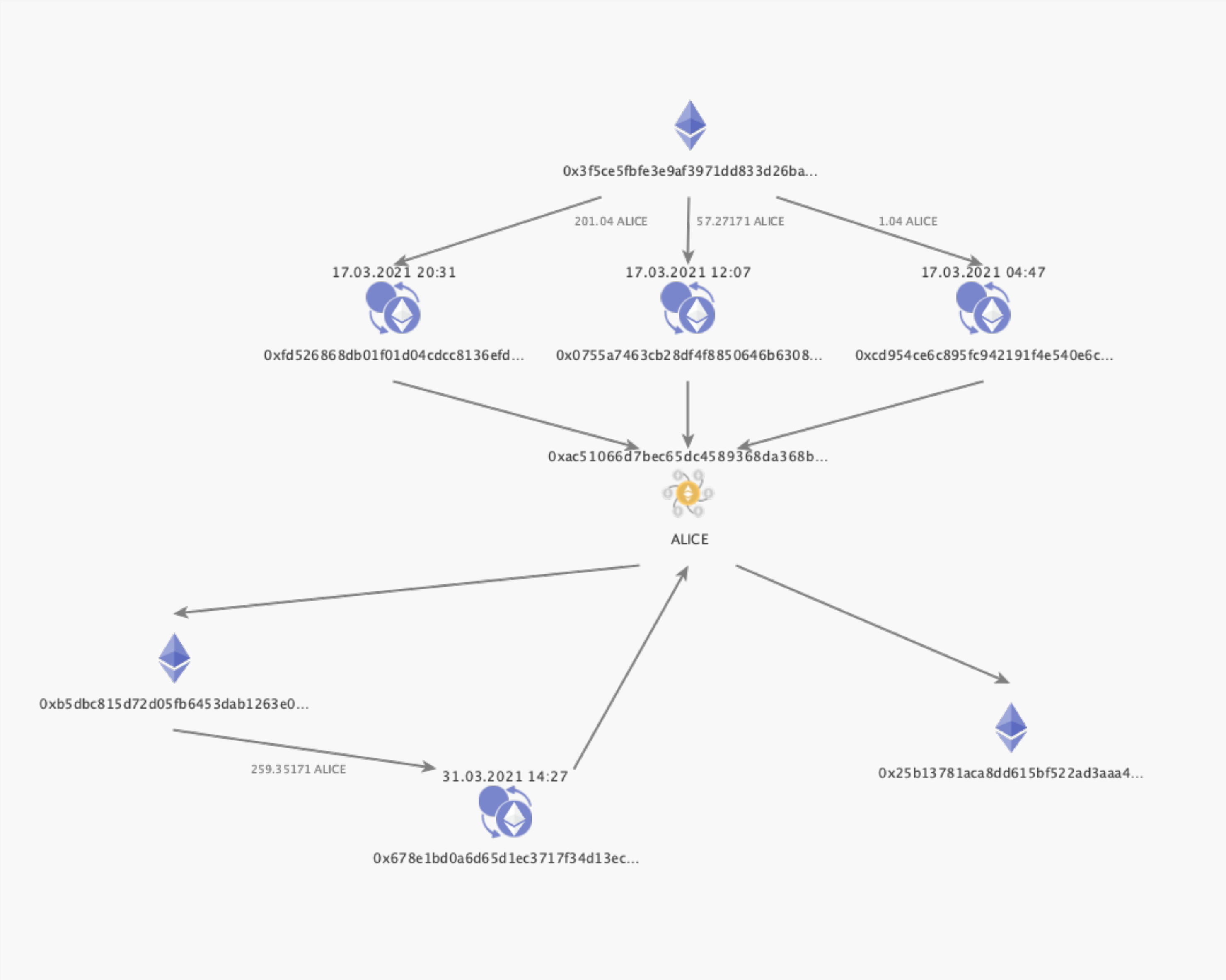
Well, there's still a lot on it, but now it looks a bit cleaner, the links and Entity representations are logical, and overall this format is already suitable for further exploration and investigation.

For example, here's how small subgraphs may look when depicting activity around a particular token.

Work Quest Token (WQT): A certain amount was acquired on May 4th, and then moved further on August 9th:



ALICE Token: It clearly can be seen how tokens were accumulated in three subsequent transactions on March 17th, and then everything was moved to another address on March 31st in one transaction:



Here you can also see how modified display options affect Entity representation: A transaction also shows a date above the icon, and a token shows contract address, which makes the overall picture more easily explainable.

This configuration, of course, may not be the final one and we can continue experimenting with different options depending on investigation goals. In practice, as already mentioned, this is a trial and error process and investigators can spend some time finding out the best visual representation of a particular dataset. However, a good graphic layout always pays off, as it provides an easy way for visual representation of complex information.

Pros and Cons of the Method

So far, we can summarize on the main advantage and drawback of the data import approach:

1. Great thing about it: You can define and build relations within the dataset any way you like. Modify Entities and add properties any way you like. You are not limited to a fixed format of predefined Transforms and have freedom to change the visualization concept depending on what exactly is important for your investigation.

2. Not so great thing about it: You are still limited only to your dataset, as you have no automatic Transforms. You may not have many additional data points which could be handy. For example, in our case we did not have any data on monetary values of token transactions, however it would be really nice to have not only a transaction value in a certain token amount, but also an equivalent in ethereum or fiat currency. But etherscan.io tool does not give an opportunity to include this information in data export.

3. But let's finish on a good note again... You can enrich the dataset the way you want!

And of course, every analyst and investigator has their own methods and tricks to build quality visualizations which help both presenting and investigating the data.

Vladimir Mikhnovich

Vladimir is an expert and consultant in data science, fraud detection, blockchain investigations and open source intelligence (OSINT). His current research interest is in the field of online scam prevention and awareness, which includes both technical and social aspects of modern con artistry involving cryptocurrencies. He runs a consulting company and is also involved in public speaking, writing articles and educating on the subject matters. To learn more about Vladimir's work, visit his LinkedIn profile: www.linkedin.com/kypexin

MALTEGO

Maltego empowers investigators worldwide to speed up and increase the precision of their investigations through easy data integration in a single interface, aided by powerful visualization and collaborative capabilities to quickly zero in on relevant information. Maltego is a proven tool that has empowered over one million investigations worldwide since its first launch in 2008. Due to its wide range of possible use cases ranging from threat intelligence to fraud investigations, Maltego is used by a broad audience, from security professionals and pen testers to forensic investigators, investigative journalists, and market researchers. Learn more about how we can empower your investigations on our website.

MALTEGO.COM