Net-assembly Token White Paper

An architecture for a high performance blockchain, and crypto adoption
V0.2.1

1 Legal Disclaimer

Nothing in this White Paper is an offer to sell, or the solicitation of an offer to buy, any tokens. Net-assembly is publishing this White Paper solely to receive feedback and comments from the public. If and when Net-assembly offers for sale any tokens (or a Simple Agreement for Future Tokens), it will do so through definitive offering documents, including a disclosure document and risk factors. Those definitive documents also are expected to include an updated version of this White Paper, which may differ significantly from the current version. If and when Net-assembly makes such an offering in the United States, the offering likely will be available solely to accredited investors.

Nothing in this White Paper should be treated or read as a guarantee or promise of how Net-assembly's business or the tokens will develop or of the utility or value of the tokens. This White Paper outlines current plans, which could change at its discretion, and the success of which will depend on many factors outside Net-assembly's control, including market-based factors and factors within the data and cryptocurrency industries, among others. Any statements about future events are based solely on Net-assembly's analysis of the issues described in this White Paper. That analysis may prove to be incorrect.

2 Who are WE ? KYC

Net-assembly, the legal entity providing this token is based in Paris, France, under the French and European laws.

Created by Simon Cabotse (https://x.com/CabotseSimon), the company has been registered in 2014 in Paris, with the registration number 80809654900023 PARIS and the VAT number FR87808096549. The company's main office is at : Net-assembly, 101 rue de Sèvres lot 1665, 75272 Paris Cedex 6, France

Official Registration documents can be obtained at these addresses, for example :
https://www.societe.com/societe/net-assembly-808096549.html

https://www.infogreffe.fr/entreprise/net-assembly/808096549/6663b90c-995b-4537-aa12-2c2fe7765851
This software publishing company, has published the following applications (among others):

https://www.free-cash-register.net
https://www.jeu-tarot-en-ligne.com
https://www.deliv.shop

3 Project objective

The project is based on the Solana blockchain, chosen for its transaction fees which allow micropayments without transaction fees being prohibitive, for its high speed of recording transactions, and for its wide adoption.
This cryptocurrency is therefore compatible with use for everyday payments, according to its initial design itself (level 1).
The origin of the project is the observation of the unsuitability of this cryptocurrency for development tools which are not the most recent : Solana for old websites (no-code HTML payment form, PHP/ASP.net/Java/etc payment notifications webhooks).
Setting up a Solana online payment with Node.js, React is very simple, but in PHP, ASP.net, Java, Ruby, it is very complex, the simplest being to create a gateway in React.
The Solana project does not allow payment notifications to be obtained in the form of an HTTP Webhook, as requests sent to validators are normally done asynchronously, as allowed by Javascript.
This poses problems for automated order processing if you use PHP, ASP.net, Java, or Ruby as a programming language for your server-side website.
According to https://w3techs.com/technologies/overview/programming_language, PHP, ASP.net and Java together account for 87.1% of websites on the internet.
The project therefore aims to set up a tool allowing a payment page, with payment button and payment QRcode, as well as the possibility of choosing a notification URL (webhook), in order to receive and process payments with the Solana blockchain.

4 How to request payments using our app ?

The users goes to https://crypto.net-assembly.com then enters his **Solana wallet address**, chooses his **currency** (Euro or Dollar), an **amount to pay** expressed in this currency (the amount can also be dynamically configured in the code), an optional notification URL, an optional return URL (end of process), can communicate an email to receive payment notifications by email as well. When the webmaster has entered this data, he obtains a payment link, HTML code, and a QRcode allowing payment by scanning it, allowing direct access to a payment page.

Here is an example of a payment link that the webmaster can obtain:
https://pay.net-assembly.com/Pay/4n9aiKrqNb2UJY3bABwfGNVkwkbYHBuugn4uJWUEh77v/1/1/USD/Achat%20En%20Ligne/PARAM
The payment page supports all available wallets, such as Phantom, Solflare, or Coinbase Wallet.

Even if it is possible to just reuse a default payment request if payment notification are not needed, on https://crypto.net-assembly.com webmaster can get additionnal features :

- Receive payment notification by email and on an notification URL (webhook, there are sample code in PHP, ASP.net, and Java in our Github : https://github.com/paracetamol951/Solana-Token-Payment-Gateway).
- Choose an URL to redirect the payer to after the payment
- Add some customization to the payment page (text, images, links...)

You can customize the amount, the title of the order, and provide an optionnal parameter that will be transmitted with the webhook and email notifications, just replace the parameter 'PARAM'.

If you implemented our payment gateway and would like to be featured in the partners list, please contact us.

If you specified a webhook, a HTTP request will be made on payment on the URL you provided. The data will be sent as a JSON encoded BODY of an HTTP request with the parameters :

```
myParam : the parameter you provided in the payment link or form
sourcePayment : the wallet address of the payer
paidAmountInNeta : the amount paid in NETA token
EUR_Amount : the amount in euros
USD_Amount : the amount in USD
trx : the transaction ID
secret : your secret key
```

The end customer will have to convert their Solana into Neta to be able to make the payment.
The conversion is very simple and can be done directly from the client's wallet application.
If the end customer does not have cryptocurrency, he can purchase the NETA token directly on https://crypto.net-assembly.com
The token is also available on decentralized exchange platforms, such as Jupiter, Meteora, ... (the creation of the pool has already been carried out: https://app.meteora.ag/pools/6ftxsFoL6H9wovWWeDhsq1Wi17JKxkdqVj6m2UUaDzEx)

5 *How to buy NETA ?*

There should be as many ways as possible to buy NETA.
It can be bought :

- on the DEX exchanges, for example on Jupier, or Meteora.
- by sending Solana to the address : 3pT2FyUCWrNwVs5VEGoKkdJCaV6Q2EoydKTYBytJ4KME a robot will automatically look if you have a token account opened, if you dont, it will open the token account for you, then it will use Jupier API in order to convert your Solana into Net-assembly, then it will send back the NETA to the sender address. The operation has an impact on the DEX price of NETA, as the Solana you send is converted to NETA
- by credit card, or with Paypal (at this date this is yet to come). You will receive on the choosen Solana address the corresponding NETA amount according to Jupiter rate. The Euros are not converted into NETA, this has no impact on the NETA price

6 *What synergy can we draw from this?*

This cryptographic project can first be integrated into existing projects.
- In https://www.jeu-tarot-en-ligne.com it will be possible to buy game credits in cryptocurrency.
- A particularly interesting integration is the integration into the cash register software https://caisse.registereuse.fr since this integration will allow all merchants who use this software to accept cryptocurrency payments in their establishment. In the cash register software, the cashier can activate a “Crypto Payment” button which will have the effect of displaying a QR code on the customer display, or on the receipt, which will allow their customer to pay by scanning this QR code with your smartphone. Funds will be received directly to the merchant's crypto wallet, payment success will be automatically recorded in the software when the transaction is confirmed, this payment method can also be integrated in their websites in a few clicks.
Promotion of the platform can be carried out on networks for developers, or also via referencing, answering questions on stackoverflow, Kol deals, etc.

7 Tokenomics

Net-assembly (NETA) is a Solana based token.
https://solscan.io/token/3jW4qfs9ZEqeVdTEN5xhoK81jDyU1tCFJ1uWM1uewgQp
1,000,000,000 tokens have been minted and mint authority has been revoked.
Freeze authority has also been revoked, so that total supply is at a definitive amount of 1,000,000,000.

Liquidity pools, wallets, marketing and project dev. liquidity

55% of the tokens serve as a fund for creating different kind of pools, so that users can buy or sell NETA token : there is Meteora and other dex liquidity pool allocated funds, Solana converter robot liquidity pool (the robot detects when solana arrives on a particular wallet address, and if it receives Solana, it converts it into NETA token using Jupiter, then sends the NETA back to the sender – the address that is connected to this robot is 3pT2FyUCWrNwVs5VEGoKkdJCaV6Q2EoydKTYBytJ4KME), Direct sales liquidity pools for the partnership we could conclude...

35% of the tokens can be used for the needs of the project : developping the project, testing, doing marketing, with the supervision of the managment team, that are the only one to dispose of the private keys of such wallets. The budget is allocated on a daily basis, unlocking maximum 10% the the last 24h total volume.

10% of total supply is shared among the members of the team.

There will be no airdrop ! NETA can only be bought.

8 Security

Security of the technology itself

The technical principle of this payment gateway relies on the usage of existing Solana programs :
- The Memo program, located at this address :
MemoSq4gqABAXKb96qnH8TysNcWxMyWCqXgDLGmfcHr
https://explorer.solana.com/address/MemoSq4gqABAXKb96qnH8TysNcWxMyWCqXgDLGmfcHr
- System programs : Transfer and TransferChecked (11111111111111111111111111111111)
- Compute Budget Program : Set Compute Unit Limit & Set Compute Unit Price (Very common programs to limit transaction fees)

One example of a transaction can be found here :
https://explorer.solana.com/tx/5Mg79Tz7Y887VjujvbxC1hZ4Uy5k2VQJcSTduXc4EZwXpjQqRS6kLAtrLHKBoao4QQfiV6nbz1LuQvJHje1qNmru

As you can see, this technology requires no usage of Smart contracts, or custom Programs, and this is the key to the security we provide.

As we are only using audited programs made by the Solana team itself, there is no risk of security issue, that could happen in Smart Contracts, or customized programs. Thus we do not need to rhave an audit run on our technology, and we can trust one the the best audits : those of the Solana team.

Security of the liquidity tokens

The security of the liquidity pools is operated by Meteora (https://meteora.ag/)
Some other liquidity pool provider might be added, only with the major liquidity pool provider, like Orca, Raydium.

The private keys of the liquidity tokens that have not been allocated in pools yet, are kept in a secured ledger, which is only connected to a computer on imperative need, and is stored in a safe, only accessible by the founder/CEO.

Security for the payer

The payer receives a link, a qrcode, or a form that directs him to a page like this one :
https://pay.net-assembly.com/Pay/5vQLanKy6thd4QxYWCsy9NHWE4GMWMZpSMQG2xHs3xRj/1.5/1/EUR/Pay%20now

In you look at this page, you can see the payer has two ways of paying : one is to scan a Qrcode with a wallet application, the other is to connect his wallet using web3 wallet connector, then to use the « Pay » button.

- When scanning the QR code present in this payment page :
  There is no need to connect a wallet, our gateway plays absolutely no role in the processing of the transaction : the wallet application of the payer is generating the transaction, signing it. Private key is never known by anybody other than your wallet app. The QR code just uses the standard Qrcode specification made by Solana, also called Standard URI scheme :
  https://docs.solanapay.com/spec
  https://docs.solanapay.com/core/transfer-request/merchant-integration
  This scheme lets you specify a memo parameter that is meant to be included in the transaction into a Memo instruction.

- When connecting a wallet, and using the Pay button to generate the transaction :
  With most wallets, you can preview the transaction instructions before signing the transaction. If using Phantom, SolFlare, for exemple, you can see all the instructions that the web3.js application is generating, thus you can verify there are only the type of System instructions (programs) that have been listed before in this same document. This way the payer can ensure there are no malicious instructions hidden in the transaction.

Security for the receiver of the funds

The receiver of the funds has to create the payment request on https://crypto.net-assembly.com or use an existing payment request. He only has to provide his public wallet address, and he will then get a simple link or a qrcode. There is absolutely no risk in communicating a public wallet address, or in copying and pasting a link or an image. So the only concern that the requester should take care about is typing the good USD or EUR amount in the form when creating the payment request, and choosing the good tokens he wants to accept.

Moreover, there are several advantages for the requester of the payment :

- he is the one that types the amount (prevent client amount error)
- he can get payment notification by email (no need to ask for trx ID)
- he can integrate the webhook notifications into a more complex process (more agility for coders)
- just one link to send to ask for a payment (efficiency)
- he can also set a title that will be displayed in the payment page (trust)
- he can accept any Solana token
- less clicks for his client, no need to copy and paste a wallet address, type in an amount : just a few clicks : click on the link, click on « Connect wallet », click on « Pay ».
- looks more professionnal

9 Reminder of underlying architecture of Net-assembly token base 1 layer : Solana.

This paper proposes a blockchain architecture based on Proof of History (PoH) - a proof for verifying order and passage of time between events. PoH is used to encode trustless passage of time into a ledger - an append only data structure. When used alongside a consensus algorithm such as Proof of Work (PoW) or Proof of Stake (PoS), PoH can reduce messaging overhead in a Byzantine Fault Tolerant replicated state machine, resulting in sub-second finality times. This paper also proposes two algorithms that leverage the time keeping properties of the PoH ledger - a PoS algorithm that can recover from partitions of any size and an efficient streaming Proof of Replication (PoRep). The combination of PoRep and PoH provides a defense against forgery of the ledger with respect to time (ordering) and storage. The protocol is analyzed on a \$1\$~gbps network, and this paper shows that throughput up to \$710\$k transactions per second is possible with today's hardware.

Introduction

Blockchain is an implementation of a fault tolerant replicated state machine. Current publicly available blockchains do not rely on time, or make a weak assumption about the participant's abilities to keep time. Each node in the network usually relies on their own local clock without knowledge of any other participants clocks in the network. The lack of a trusted source of time means that when a message timestamp is used to accept or reject a message, there is no guarantee that every other participant in the network will make the exact same choice. The PoH presented here is designed to create a ledger with verifiable passage of time, i.e. duration between events and message ordering. It is anticipated that every node in the network will be able to rely on the recorded passage of time in the ledger without trust.

At any given time a system node is designated as Leader to generate a Proof of History sequence, providing the network global read consistency and a verifiable passage of time. The Leader sequences user messages and orders them such that they can be efficiently processed by other nodes in the system, maximizing throughput. It executes the transactions on the current state that is stored in RAM and publishes the transactions and a signature of the final state to the replications nodes called Verifiers. Verifiers execute the same transactions on their copies of the state, and publish their computed signatures of the state as confirmations. The published confirmations serve as votes for the consensus algorithm.

In a non-partitioned state, at any given time, there is one Leader in the network. Each Verifier node has the same hardware capabilities as a Leader and can be elected as a Leader, this is done through PoS based elections.

In terms of CAP theorem, Consistency is almost always picked over Availability in an event of a Partition. In case of a large partition, this paper proposes a mechanism to recover control of the network from a partition of any size.

Proof of History

Proof of History is a sequence of computation that can provide a way to cryptographically verify passage of time between two events. It uses a cryptographically secure function written so that output cannot be predicted from the input, and must be completely executed to generate the output. The function is run in a sequence on a single core, its previous output as the current input, periodically recording the current output, and how many times it's been called. The output can then be re-computed and verified by external computers in parallel by checking each sequence segment on a separate core.

Data can be timestamped into this sequence by appending the data (or a hash of some data) into the state of the function. The recording of the state, index and data as it was appended into the sequences provides a timestamp that can guarantee that the data was created sometime before the next hash was generated in the sequence. This design also supports horizontal scaling as multiple generators can synchronize amongst each other by mixing their state into each others' sequences. Horizontal scaling is discussed in depth in Section

It is only necessary to publish a subset of the hashes and indices at an interval.

As long as the hash function chosen is collision resistant, this set of hashes can only be computed in sequence by a single computer thread. This follows from the fact that there is no way to predict what the hash value at index 300 is going to be without actually running the algorithm from the starting value 300 times. Thus we can thus infer from the data structure that real time has passed between index 0 and index 300.

This sequence of hashes can also be used to record that some piece of data was created before a particular hash index was generated. Using a `combine` function to combine the piece of data with the current hash at the current index. The data can simply be a cryptographically unique hash of arbitrary event data. The combine function can be a simple append of data, or any operation that is collision resistant. The next generated hash represents a timestamp of the data, because it could have only been generated after that specific piece of data was inserted.

Because the initial process is still sequential, we can then tell that things entered into the sequence must have occurred sometime before the future hashed value was computed.

Given some number of cores, like a modern GPU with 4000 cores, the verifier can split up the sequence of hashes and their indexes into 4000 slices, and in parallel make sure that each slice is correct from the starting hash to the last hash in the slice.

It's possible to synchronize multiple Proof of History generators by mixing the sequence state from each generator to each other generator, and thus achieve horizontal scaling of the Proof of History generator. This scaling is done without sharding. The output of both generators is necessary to reconstruct the full order of events in the system.

Given generators A and B, A receives a data packet from B (hash1b), which contains the last state from Generator B, and the last state generator B observed from Generator A. The next state hash in Generator A then depends on the state from Generator B, so we can derive that hash1b happened sometime before hash3a. This property can be transitive, so if three generators are synchronized through a single common generator, we can trace the dependency between A and C even though they were not synchronized directly.

By periodically synchronizing the generators, each generator can then handle a portion of external traffic, thus the overall system can handle a larger amount of events to track at the cost of true time accuracy due to network latencies between the generators. A global order can still be achieved by picking some deterministic function to order any events that are within the synchronization window, such as by the value of the hash itself.

10 Related links

Project homepage : https://crypto.net-assembly.com/
Github : https://github.com/paracetamol951/Solana-Token-Payment-Gateway
X (Twitter) : https://x.com/Net_assembly_tk
Telegram : https://t.me/netassembly
Medium : https://medium.com/@Net-assembly-token
dAppRadar : https://dappradar.com/dapp/net-assembly
Discord : https://discord.com/invite/uu2RSqq6
Meteora pool :
https://app.meteora.ag/pools/6ftxsFoL6H9wovWWeDhsq1Wi17JKxkdqVj6m2UUaDzEx