A tool for implementing privacy in Nano

Abstract—We present a work in progress strategy for implementing privacy in Nano at the consensus level, that can be of independent interest. Nano is a cryptocurrency that uses Open Representative Voting (ORV) as a consensus mechanism, a variant of Delegated Proof of Stake. Each transaction on the network is voted on by representatives, and each vote has a weight equal to the percentage of their total delegated balance. Every account can delegate their stake to any other account (including itself) and change it anytime it wants. The goal of this paper is to achieve a way for the consensus algorithm to function without knowing the individual balances of each account. The tool is composed of three different schemes. The first is a weighted threshold secret sharing scheme based on the Chinese Remainder Theorem for polynomial rings \cite{1} and it’s used to generate, in a distributed way, a secret that will be a private key of an additive ElGamal cryptosystem over elliptic curves (EC-EG) \cite{2}, which is additive homomorphic. The second scheme is the polynomials commitment scheme presented in \cite{3}, which is an improvement of \cite{4}, and is used to make the previous scheme verifiable, i.e., without the need of a trusted dealer. Finally, the third scheme is used to decrypt a ciphertext of the EC-EG cryptosystem without reconstructing the private key and, because of that, can be used multiple times.

Index Terms—Nano, block lattice, privacy, weighted threshold, secret sharing, additive homomorphic encryption

I. INTRODUCTION

Bitcoin appeared in 2008 \cite{5} and is widely considered to be the first decentralised cryptocurrency. Its ingenious design, that uses a blockchain as a ledger to store the transactions that happen on the network and a Proof of Work algorithm to reach a decentralised consensus about the state of that blockchain, allowed it to thrive and even today it is the most well known and most valuable cryptocurrency.

However, this design also has limitations that prevent Bitcoin from being what it was meant to be: a peer-to-peer electronic cash system. These are its inability to scale, its restricted maximum throughput, slow confirmation times, ledger size and lack of privacy.

Many cryptocurrencies were created in the last years that tried to solve these limitations, but so far none of them was able to solve them all. Most of them are based on the Bitcoin design so this can be a difficult task. Two examples of this are Monero \cite{6} and ZCash \cite{7}, that solve the privacy issue, but still share the same other limitations of Bitcoin.

Perhaps the most efficient way to solve the design flaws of Bitcoin is to create a new design altogether. This is what the cryptocurrency Nano did \cite{8}. This is a cryptocurrency that uses a block-lattice instead of a single blockchain and also uses a different consensus mechanism called Open Representative Voting (ORV), a variant of Delegated Proof of Stake. These will be explained in detail in the next section and also how they solve Bitcoin’s issues of scalability, maximum throughput and confirmation times of transactions.

One issue that Nano doesn’t solve is privacy, and as far as we know, there isn’t any work about it, at least at the protocol level. Since Nano is fundamentally different from Bitcoin, any implementation of privacy will therefore be different as well.

This paper’s goal is to develop a tool that can be used to implement privacy in Nano at the protocol level. The rest of the paper is organised as follows: section 2 does a resume of the design of Nano, describing the characteristics that make it different from Bitcoin; section 3 discusses how privacy can be implemented in Nano and section 4 proposes a tool to implement it at the consensus level. The last section is about future work that can be done.

II. DESIGN OVERVIEW OF NANO

A. The Block Lattice

Unlike Bitcoin and many other cryptocurrencies, that use one unique blockchain as a ledger to store the transactions that happen on the network, Nano uses a different data-structure called the block lattice (Figure 1), where each individual account controls their own blockchain within a wider lattice structure.

Fig. 1. The Block Lattice.

This allows blocks to be added quickly without conflict and sent to the network for confirmation in an asynchronous way.

Every individual blockchain is composed of blocks, which can be one of three different types: send, for sending transactions; receive, for receiving transactions; and change, for changing the representative of the account (explained in more
detail in the following subsection). The first block of a new account must be a receive block.

Transactions occur between accounts with two separate actions:

1) The sender publishes a block debiting their own account for the amount to be sent to the receiving account.
2) The receiver publishes a matching block crediting their own account for the amount sent.

Once a block sending funds is confirmed by the network, the transaction goes into a pending state and cannot be reversed. The receiver can be offline and safely leave the funds in this state until they are ready to publish a matching block receiving the funds to their account.

B. The Consensus Mechanism

Nano has a unique consensus mechanism called Open Representative Voting (ORV), which involves accounts delegating their balance as voting weight to Representatives. The Representatives vote themselves on the validity of transactions published to the network using the voting weight delegated to them.

These votes are shared with their (directly connected) peers and they also rebroadcast votes seen from Principal Representatives (PRs), an account with \( \geq 0.1\% \) (of the total supply) voting weight delegated to it. Votes are tallied and once quorum is reached on a published block, it is considered confirmed by the network.

Every account can freely choose a Representative at any time to vote on their behalf (it can even be themselves), even when the delegating account itself is offline. These Representative accounts are configured on nodes that remain online and vote on the validity of transactions they see on the network. Their voting weight is the sum of balances for accounts delegating to them, and if they have enough voting weight they become a Principal Representative.

The votes these Principal Representatives send out will subsequently be rebroadcast by other nodes. As these votes are shared and rebroadcast between nodes, they are tallied up and compared against the online voting weight available. Once a node sees a block get enough votes to reach quorum, that block is confirmed. Due to the lightweight nature of blocks and votes, the network is able to reach confirmation for transaction very fast, often in under a couple seconds. Also note that delegation of voting weight does not mean staking of any funds - the delegating account can still spend (and receive) all their available funds at any time without restrictions.

C. Block Specification

Nano uses a structure for each block which contains all the information about an account at that point in time, which allows ledger pruning: account number, balance and representative. Table 1 shows the main parts of a Nano block. The link section contains a sub block with information that describes the type of block (send, receive, change) and the destination Nano address for a send block and hash of the block of the sending address in the case of a receive block.

Every block must also contain a small, user-generated Proof-of-Work (PoW) value which is a Quality-of-Service prioritisation and spam prevention mechanism, allowing occasional, average user transactions to process quickly and consistently.

Right now, all fields of a block are transparent. A privacy implementation would need to obfuscate fields, in particular: previous, balance and signature. In the following section we discuss how this could be done.

III. HOW TO IMPLEMENT PRIVACY IN Nano?

As shown in the previous section, Nano uses a consensus mechanism that relies on knowing the total balances that are delegated to representatives (that can include their own individual balances or not) to function, however it doesn’t need to know the individual balances of the accounts that delegate to a certain representative. Moreover, total delegated balances of representatives, i.e., voting weights, are trended over two weeks and, because of that, change slowly through time.

Our idea for implementing privacy in Nano at the consensus level is based on these premises. First, we need a cryptosystem that is additive homomorphic, so that a representative can decrypt the total of a sum without decrypting its individual components, which would be the encrypted balances of the accounts. We want this cryptosystem to be as efficient as possible, in computation time (encryption and decryption) and in bandwidth (generating a small size ciphertext).

We also need a scheme for generating a private key for the previous cryptosystem in a distributed way (without a trusted dealer), and we want each part to have a weight on the scheme proportional to the voting weight on the network. This private key will be used to generate a public key, that will be used by all accounts to encrypt the balances and transaction amounts of the network.

From time to time (to be defined), the voting weights will have to be updated and, for that to happen, the new total delegated balances will have to be known. So we need a way to decrypt the total delegated balances without reconstructing the private key, otherwise all individual balances could be also decrypted. Moreover, we want that parts with a certain threshold of total weight can make this decryption.

For efficiency purposes, this scheme will only be run by Principal Representatives, which can be at most 1000.
IV. THE SCHEME

Based on the requirements stated in the previous section, we now present a scheme for implementing privacy in Nano at the consensus level, which has three different components: a weighted threshold secret sharing scheme, which will generate an asymmetric cryptosystem in a distributed fashion; a way to make this scheme verifiable, and, finally, a way to decrypt the ciphertexts of the cryptosystem without reconstructing the private key. In this section we follow the same notation used in the papers in which the schemes are based.

A. A weighted threshold secret sharing scheme

Most secret sharing schemes are based on either polynomial interpolation [9], or on the Chinese Remainder Theorem, originally proposed by [10] and improved by [11].

However, neither of these schemes is specially suited for a weighted threshold scheme. In the first type every share has equal weight and if you have more weight you will simply have to have a proportional amount of shares, which isn’t very efficient or scalable.

In the second type, the setup required for the scheme, a sequence of large coprime numbers that fulfil a certain condition, is difficult and slow to obtain in practice.

Because of this, we chose to use the scheme of [1] for our purposes, which can be seen as a generalisation of the Shamir’s scheme or the counterpart of the Asmuth-Bloom’s scheme for a ring of polynomials instead of integers. The access structure of the scheme is of the form:

$$\Gamma = A \subseteq [n] \bigg\| \sum_{i \in A} w(i) \geq t$$

(1)

where $w$ is the weight function evaluated over $Z$ with $w_i = w(i)$ for all $i \in [n]$ and

$$1 \leq w_1 \leq w_2 \leq \ldots \leq w_n < t.$$

(2)

For a given scheme parameter prime $p$ and $m_0(x) = x$, $P_i$, for all $i \in [n]$, the scheme proceeds as following:

1. $P_i$ chooses a polynomial $m_i(x) \in F_p[x]$ satisfying the condition $d_i = w_i$, with $d_i = \deg(m_i)$, and broadcast it to the other parts, that verify the condition.

2. $P_i$ chooses a secret $s_i \in F_p$ and a polynomial $\alpha_i(x)$ of degree at most $t - 2$. Then, he computes $f_i(x) = s_i + \alpha_i(x)m_0(x) = s_i + \alpha_i(x)$, that will have at most degree $t - 1$.

3. $P_i$ computes the secret private share $s_{ij} = f_i(x) \mod m_i(x)$, with $j \in [n]$ and sends $s_{ij}$ privately to the $j$-th party, including himself.

The secret, and generated private key, will be $s = \sum_{i=0}^{n} s_i$ and the private share of $P_i$ will be $s_i(x) = \sum_{j=1}^{n} s_{ij}(x)$, and could be reconstructed if $k$ parties $i_1, \ldots, i_k \subseteq [n]$ with $\sum_{j=1}^{k} w(i_j) \geq t$, form and solve the following system of congruences:

$$\begin{cases}
X(x) \equiv s_{i_1} \pmod{m_{i_1}(x)} \\
X(x) \equiv s_{i_2} \pmod{m_{i_2}(x)} \\
\vdots \\
X(x) \equiv s_{i_k} \pmod{m_{i_k}(x)}
\end{cases}$$

They can solve this system of congruences and get a solution $X_0(x) \in F_p[x]$. By the Chinese Remainder Theorem for polynomial rings over a field, the solution is unique if only polynomials of degree less than $\sum_{j=1}^{k} d_{ij}$ are considered. Since

$$\sum_{j=1}^{k} d_{ij} \geq t > t - 1 \geq df$$

(3)

and $f(x)$ also is a solution of the above system of congruences, they have $f(x) = X_0(x)$. Then, the secret can be recovered by computing

$$s(x) = X_0(x) \pmod{m_{i_0}(x)}.$$  

(4)

B. A verifiable weighted threshold secret sharing scheme

In this section we show how we can make the previous scheme verifiable. This can be achieved if the secret private shares $s_{ij}(x)$, with $i, j \in [n]$ and $i \neq j$, can be verified, i.e., one can make sure that they really come from $f_i(x) \mod m_i(x)$ and not from some other polynomials, otherwise the secret private share would be useless in reconstructing the secret.

Let $P_j$ be the prover and $P_i$ the prover of the previous statement, with $i \neq j$. Since $s_{ij}(x) = f_i(x) \mod m_i(x) \Leftrightarrow f_i(x) = m_i(x)k_i(x) + s_{ij}(x)$ for some polynomial $k_i(x)$, if the prover commits to $f_i(x)$ and $k_i(x)$ and reveals $f_i(z)$ and $k_i(z)$, $z \in Z_p$ chosen by the verifier, with a proof that the evaluations are correct for the committed polynomials, then, the verifier can check that

$$f_i(z) = m_i(z)k_i(z) + s_{ij}(z).$$

(5)

The probability that this equation holds for a committed $f_i(x)'$ or $k_i(x)'$ would be at most (number of intersections of $f_i(x)$ and $f_i(x)'$)$\#F_p = (t - 1)/\#F_p$, where $t - 1$ is the maximum degree bound, which is negligible for a large prime $p$. So, the verifier can be convinced that the statement is true if the equation holds.

The polynomial commitments required for this proof can be achieved with the scheme proposed in [3], an improvement to [4], that allows multiple univariate polynomial commitments for a single degree bound and a single evaluation over a field family $F$.

This is enough for what we need, since $f_i(x)$, as stated in subsection A, can have at most degree $t - 1$. Since $m(x)$ has at minimum degree $1$, $k_i(x)$ can have at most $t - 1$ as well. For efficiency purposes, we will use the construction in the algebraic group model.
The scheme is composed of the following tuple of algorithms PC = (Setup, Commit, Open, Check), defined as following:

**Setup.** On input a security parameter $\lambda$ (in unary), and a maximum degree bound $D \in N$, PC.Setup samples public parameters (ck, rk) as follows. Sample a bilinear group $\langle\text{group}\rangle \leftarrow \text{SampleGrp}(\lambda^3)$, and parse $\langle\text{group}\rangle$ as a tuple $(G_1, G_2, G_T, q, G, H, e)$. Sample random elements $\beta \in F_q$.

Then compute the vector

$$\sum := (G, \beta G, \beta^2 G, \ldots, \beta^D G) \in G_1^{D+1}$$

These elements can be calculated in a distributed way, where $P_i$ picks a $\beta_i G$, with $i \in [n]$, and broadcasts it. The elements will be $\beta G = \sum_{i=1}^n \beta_i G, \beta^2 G = 2 \sum_{i=1}^n \beta_i G, \ldots, \beta^D G = D \sum_{i=1}^n \beta_i G$.

Set $\text{ck} := \langle\text{group}, \sum\rangle$ and $\text{rk} := (D, \langle\text{group}\rangle, \beta H)$, and then output the public parameters (ck, rk).

These public parameters will support polynomial operations over the field $F_q$ of degree at most $D$.

**Commit.** On input ck, univariate polynomials $p := [p_i]_{i=1}^n$ over $F_q$, PC.Commit outputs commitments $c := [c_i]_{i=1}^n$ that are computed as follows:

If for any $p_i \in p$, $\text{deg}(p_i) > D$, abort.
For each $i \in [n]$, output $c_i := p_i(\beta)$. Set $\text{ck} := \langle\text{group}, \sum\rangle$.

Note that because $p_i$ has degree at most $D$, the above terms are linear combinations of terms in ck.

**Open.** On input ck, univariate polynomials $p := [p_i]_{i=1}^n$ over $F_q$, evaluation point $z \in F_q$, opening challenge $\epsilon \in F_q$, PC.Open outputs the evaluation proof $\pi \in \mathbb{G}_p$, that is computed as follows. Compute the linear combination of polynomials $w(x) := \frac{p(x) - p(z)}{x - z}$. Set $w := w(\beta) G \in \mathbb{G}_p$. The evaluation proof is $\pi := (w)$.

**Check.** On input rk, commitments $c := [c_i]_{i=1}^n$, evaluation point $z \in F_q$, alleged evaluations $v := [v_i]_{i=1}^n$, evaluation proof $\pi$, and randomness $\epsilon \in F_q$, PC.Check proceeds as follows. Compute the linear combination $C := \sum_{i=1}^n c_i$, then compute the linear combination of evaluations $v := \sum_{i=1}^n \frac{c_i}{w}$ and check the evaluation proof via equality $e = (C - v G, H) = e(w, \beta H - z H)$.

This protocol can be made non-interactive by using the Fiat-Shamir heuristic [12] and zero-knowledge by having the prover to send $v G$ instead of $v$.

### C. Decrypting without reconstructing the secret

As already stated, the secret distributed in the previous scheme will be the private key of an asymmetric cryptosystem, and will generate the public key that will be used to encrypt the balances of the network. This section proposes a scheme to decrypt these ciphertexts without reconstructing the private key.

According to [2], the most efficient cryptosystem for this purpose is the additive ElGamal over elliptic curves (EC-EG), whose key set-up consists in choosing an elliptic curve $E$ over $F_p$ together with a 163-bit prime $p$ and generator $G$. Its security is based upon the Elliptic Curve Discrete Log Problem (ECDLP). $x \in F_p$ is the private key of the system and $Y = xG \in F_p$ is the public key. There is a function $\text{map}(\cdot)$ that maps values (e.g. plaintexts) into points on the curve, and vice versa.

The encryption of plaintext $m$ is done the following way:

- $M = \text{map}(m)$ is a point on $E$
- $r$ is a random value in $F_p$
- ciphertext $C = (R, S)$, where $R = rG$ and $S = M + rY$

The decryption process is done the following way:

- $M = -xR + S = -xrG + M + xrG$
- $m = \text{rmap}(M)$, where $\text{rmap}(\cdot)$ is the reverse function of $\text{map}(\cdot)$

This decryption process requires knowing the private key $s(x) \pmod{m_0(x)}$, however in our scheme this is not known and we don’t want it to be known, so that we can use it to decrypt multiple ciphertexts. So, we need to decompose $x$ in relation with the private shares of our scheme.

According to the Chinese Remainder Theorem we get $x = s_1 M_1 x_1 + \ldots + s_k M_k x_k = D_1 + D_2 + \ldots + D_k$, where $s_1, \ldots, s_k$ are the private shares, $M_\alpha$ is the product of all $m_\alpha$ but $m_\alpha x_\alpha$ comes from $M_\alpha x_\alpha \equiv 1 \pmod{m_\alpha}$ and $m = m_1 m_2 \ldots m_k$.

The notation is the same used in subsection A.

However, the solution to the system of congruences only gives us the polynomial that contains the secret, as stated in subsection A, but we need the secret to decrypt, which is equal to the sum of the coefficient of degree zero of the polynomials modulo $p$, so we get:

$$M = -xR + S$$
$$= -xrG + S$$
$$= -(D_1 + \ldots + D_k \pmod{m_0}) \pmod{p}rG + S$$
$$= -(D_1 \pmod{m_0} + \ldots + D_k \pmod{m_0}) \pmod{p})rG + S$$
$$= (x_1 + x_2 + \ldots + x_k) \pmod{p})rG + S$$

(6)

$P_i$ knows $x_i$, but he doesn’t know the others secret shares, otherwise he could reconstruct the secret, so we need a way to send a partial decryption without revealing the secret share. This can be done the following way:

$P_i$ chooses a random element $y_i \in Z_p$ and broadcasts $(x_i + y_i) \pmod{p}$ and $y_i R$. $P_j$, with $j \neq i$, can compute the decryption share removing the $y_i$ in $((x_j + x_i + y_i) \pmod{p})R$, like this:

- If $((x_j + x_i + y_i) \pmod{p}) \geq y_i$, then $y_i$ in $((x_j + x_i + y_i) \pmod{p})R - y_i R$;
- Else, $((x_j + x_i + y_i) \pmod{p})R - (p - y_i) R$.

However, $P_i$ doesn’t know $y_i$. In fact he can’t know it, otherwise he would discover $x_i$. So a way to check this condition without revealing $y_i$ is needed, which can be done using a protocol for solving the millionaires’ problem like in [13]. After that, $P_j$ and compute the decryption share of $P_j$, which will have weight $w_i$.

To decrypt $M$, $P_j$ will need the decryption shares of parts whose weight is at least $k$. This protocol will be run by the Principal Representatives to update their own weights. Any
other account that wants to know what its total delegated voting weight will have to ask Principal Representatives for decryption. After this, the protocol will have to be run again and a new cryptosystem will be generated with the new voting weights.

V. DISCUSSION

The scheme proposed in this paper can be used to implement privacy in Nano at the consensus level. However, this approach have several issues that still need to be solved.

One is that in the additive homomorphic variant of the EC-ELGamal scheme it is necessary to reverse the mapping function in order to map an elliptic curve point $M$ back to a numeric value $m$. In Nano, the balance field has 128 bits, which is large, so either this size will have to be reduced, another additive homomorphic cryptosystem will have to be used [2], or an efficient mapping algorithm, or set of algorithms, found [14].

Another issue is that the decryption protocol can create a new attack vector on the network, namely with thousands of accounts asking Principal Representatives for decryption and thereby clogging the network. Therefore, some kind of mechanism will be needed to prevent this, like a proof of a minimum balance.

Also, since Nano’s network is asynchronous, nodes don’t have a sense of time neither they do share the same state of the network at a given time. Because of this, it can be difficult to reach consensus about these values (time and state) for the update of the voting weights. One way to do this can be to define a probabilistic condition that must be fulfilled for the update to take place. For example, if a block on the network is produced with a hash that has $x$ number of zeros, if the representative of the account that produced that block is a Principal Representative, he can be the initiator of the protocol for updating the voting weights. He sums the encrypted delegated balances of the Principal Representatives based on his state of the network and sends the totals to the rest of the PRs. If they agree, the decryption process begins based on that state of the network.

Another question that comes with this scheme is if it is possible to maintain the cryptosystem after the update of voting weights without harming its security, changing only the private shares of the Principal Representatives.

If not, a new cryptosystem, with a new private and public key, will have to be generated and the difficulty lies in the network making this transition, namely how the nodes know what is the new public key and how do they prove that the balance encrypted with the previous public key is equal to the balance encrypted with the new public key.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have presented ongoing work towards implementing privacy in Nano. A secret sharing scheme was presented that goes some way in implementing privacy at the consensus level. However practical and theoretical limitations still need to be investigated further as was shown in the many issues and problems outlined in the discussion.

Future work will focus on solving the identified issues and developing a robust implementation of the scheme. The security and performance of the secret sharing scheme outlined in this paper will be evaluated and compared to other secret sharing schemes. After implementing privacy at the consensus level, research will be focused on implementing privacy at the transaction level, which will probably be easier, since there is a lot of work on this area.

REFERENCES