A tool for implementing privacy in Nano

Rui Morais
Release Laboratory
Universidade da Beira Interior
Covilhã, Portugal
ru.morais@ubi.pt

Paul Crocker
Release Laboratory
Universidade da Beira Interior
Covilhã, Portugal
crocker@di.ubi.pt

Simão Melo de Sousa
Nova-Lincs, Release- Laboratory
Universidade da Beira Interior
Covilhã, Portugal
desousa@di.ubi.pt

***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 an 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 fundamental goal of this paper is to construct a tool 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 Shamir's secret sharing scheme, used to generate a secret amongst a set of distributed parties, which will be a private key of an additive homomorphic ElGamal cryptosystem over elliptic curves. The second is a polynomials commitment scheme used to make the previous scheme verifiable, i.e., without the need for a trusted dealer. Finally, the third scheme is used to decrypt an ElGamal ciphertext without reconstructing the private key, which, because of this, can be used multiple times.**

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

I. Introduction

Bitcoin appeared in 2008 [1] 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 [2] and ZCash [3], that solve the privacy issue, but still share the same other limitations of Bitcoin.

On the other hand, cryptocurrencies based on Delegated Proof of Stake, like Tezos [4], or based on practical Byzantine Fault Consensus [5], like Stellar [6], improve on the maximum throughput and slow confirmation times, but still are only pseudo-anonymous.

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 [7]. 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, as well as 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 goal of this paper is to develop a tool that can be used to implement privacy in Nano at the protocol level. The tool is composed of three different schemes. The first is a weighted threshold secret sharing scheme based on Shamir's secret sharing scheme [8] which is 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) [9], which is additive homomorphic. The second scheme is the polynomials commitment scheme presented in [10], which is an improvement of [11], and is used to make the previous scheme verifiable, i.e., without the need for a trusted dealer. Finally, the third scheme is used to decrypt a ciphertext of the EC-EG cryptosystem without reconstructing the private key, which, because of that, can be used multiple times.

The rest of the paper is organised as follows: in section II a brief resume of the design of Nano is given, describing the characteristics that make it different from Bitcoin; section III discusses how privacy can be implemented in Nano and section IV proposes a tool to implement it at the consensus level. Section V is a discussion of the tool and open issues. The last section consists of the conclusions and future work.

II. Design Overview of Nano[1]

A. The Block Lattice

Nano uses a ledger structure based on parallel blockchains, called a block-lattice (Figure 1), where every node maintains its own local blockchain (only the account owner can add blocks to it), as well as a copy of the other nodes blockchains at a given time [12].

[1]The information in this section is taken from https://docs.nano.org/

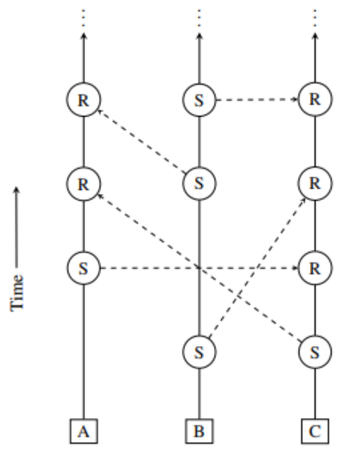

Fig. 1. The Block Lattice.

The block lattice structure allows for asynchronous transactions, since a transaction does not have to be included in a block together with other account transactions. To effect a transaction, an account computes a block and sends it to a set of peers, which in turn send it to other peers. As the block is being propagated through the network it will be voted on by the representatives. Once a certain threshold of votes has been achieved the transaction is considered confirmed.

There are three different types of blocks in Nano: *send*, for sending transactions; *receive*, for receiving transactions; and *change*, for changing the representative of the account. The first block of a new account must be a *receive* block.

After a *send* block is validated by the network, the transaction becomes ”pending” and can’t be reversed. The receiving account can pocket the funds anytime and, for that, it only has to send the corresponding *receive* block to its peers, which will then be propagated through the network and voted on as well.

B. The Consensus Mechanism

Nano uses as consensus mechanism a variant of Delegated Proof of Stake (DPoS) called Open Representative Voting (ORV). Any node can be a representative and any account can delegate its voting weight to a representative at anytime (including itself), which will vote on transactions on its behalf.

Representatives with $\geq 0.1\%$ (of the total supply) voting weight delegated to it are termed *Principal Representatives* (PR), and only their votes are rebroadcast by other nodes, to decrease the bandwidth consumption of the network.

As these votes are propagated through the network, they are summed up and then compared against the total online voting weight available. A block is confirmed if it gets enough votes to reach a quorum, which happens very fast (in less than a second) if the network is not saturated.

When there is a conflict between transactions on the network, representatives change their vote to converge to only one transaction, quickly resolving the fork.

Since transactions have no fees and the supply is fixed, there is no monetary reward paid to representatives for validating transactions and thus it is of utmost importance that the network is the most efficient as possible in terms of resource consumption.

There are though indirect incentives, as maintaining a node is less expensive for a vendor than paying transaction fees using other networks, and can also attract new costumers, thereby increasing revenue.

C. Block Specification

In Nano, a block contains all the information about an account at that point in time, which allows ledger pruning: account address, balance and representative (Table 1). The *link* field describes the type of the block (send, receive or change) and the destination address for a send block or the hash of the block of the sending address in the case of a receive block.

The *signature* proves the authenticity of the transaction and the *work* is a small, user-generated Proof-of-Work (PoW) used as a Quality-of-Service prioritisation and spam prevention mechanism. When the network is saturated, a user can generate a higher PoW value, ensuring thereby that the transaction gets validated before others with lesser PoW value.

TABLE I
NANO BLOCK FORMAT[2]

Key	Size
account	32 bytes
previous	32 bytes
representative	32 bytes
balance	16 bytes
link	32 bytes
signature	64 bytes
work	8 bytes

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 for the *balance*.

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

[2]ahttps://docs.nano.org/integration-guides/the-basics/

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 their 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.

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. Therefore, we also 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.

A. Notation

We follow the same notation used in the papers in which the schemes are based on, in the following way:

- Z is the ring of integers and $[n]$ is the set $1, 2, ..., n$ of n elements.
- F_p and F_q are finite fields of p and q elements respectively, with $p, q \in Z$.
- i and j represent the individual elements of $[n]$, with $i \neq j$, unless stated otherwise.
- P_i and P_j are the participants of a scheme.
- $R[x]$ is the univariate polynomial ring in the variable x over R, R being a ring.
- deg($f(x)$) is the degree of $f(x) \in R[x]$.
- <group> is a bilinear group parsed as a tuple $(G_1, G_2, G_T, p, G, H, e)$.
- G_1 and G_2 are two additive groups of prime order p and G_T is a multiplicative group of the same order p.
- G is an element of G_1 that generates a QR_p subgroup of order q and H is an element generator of G_2.
- $e : G_1$ x $G_2 \to G_T$ is a pairing that satisfies the properties of bilinearity, non-degeneracy and computability [13].

B. A weighted threshold secret sharing scheme

In our weighted threshold secret sharing scheme (n, t) we want to generate a secret s and divide it into n parts, s_i, in a distributed way. With t or more parts, the secret s can be easily reconstructed, and with less than t parts you get no knowledge about it. For this, we are going to use Shamir's secret sharing scheme [8], in the following way:

1) P_i chooses a polynomial $f_i(x) \in F_q[x]$ of degree $t-1$, with $f_i(0) = x_i$ being the subsecret of each P_i.
2) P_i computes $W = \sum_{i=1}^{n} w_i$ point/s of the polynomial $f_i(x)$ for $x = 1,...,W$.
3) Each P_i sends to each P_j w_j of these point/s. These points are called the the subsecret private share/s $s_{ik} = (k, f_i(k))$, where $k = 1 + \sum_{l=1}^{j-1} w_l, ..., \sum_{l=1}^{j-1} w_l$.[3]
4) Each P_i will calculate w_i secret private shares $s_k = \sum_{i=1}^{n} s_{ik} = (k, \sum_{i=1}^{n} f_i(k))$.
5) The secret $x = f(0)$, with $f(x) = \sum_{i=1}^{n} f_i(x)$, can be reconstructed with at least t secret private shares using Lagrange's interpolation as:

$$x = \sum_{k=1}^{t} s_k \prod_{k=1}^{t} \frac{j}{j-k}, j \neq k \tag{1}$$

C. 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 P_j can verify that the subsecret private shares s_{ik} are indeed points of the polynomials $f_i(x)$.

Feldman's scheme [14] allows for this, however it requires commitments for every coefficient of $f_i(x)$, which doesn't scale very well for large polynomials.

Because of this, we use instead the scheme proposed in [10], an improvement to [11], that allows multiple univariate polynomial commitments for a single degree bound and a single evaluation over a field family $\mathbb{F}$.

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 λ and a maximum degree bound $D \in N$, PC.Setup samples public parameters (ck, rk) as follows. Sample a bilinear group <group> $\leftarrow$ SampleGrp(1^λ), and parse <group> as a tuple $(G_1, G_2, G_T, p, G, H, e)$. Sample random elements $\beta \in F_q$.

Then compute the vector

$$\sum := \left(G, \quad \beta G, \quad \beta^2 G, \quad ..., \quad \beta^D G\right) \in G_1^{D+1}$$

These parameters can be calculated in a distributed way, where P_i chooses a β_i and broadcasts $\beta_i G$,$\beta_i^2 G$,...,$\beta_i^D G$.

Values can be verified by checking that $e(\beta_i G, H) = e(G, \beta_i H), e(\beta_i^2, H) = e(\beta_i G, \beta_i H), ..., e(\beta_i^D, H) = e(\beta_i^{D-1}, \beta_i H)$.

The parameters will be $\beta G = \sum_{i=1}^{n} \beta_i G$, $\beta^2 G = \sum_{i=1}^{n} \beta_i^2 G$,...,$\beta^D G = \sum_{i=1}^{n} \beta_i^D G$.

Set ck := (<group>, $\sum$) and rk := (D, <group>, βH), and then output the public parameters (ck, rk).

[3]As an example, consider three participants P_1, P_2, P_3 with weights 4, 3 and 2 respectively. Then P_1 will generate a polynomial $f_1(x)$ with private secret x_1 and calculate 9 points $s_{1k} = (k, f_i(k))$, for $k = 1, ..., 9$. P_1 then sends the points s_{15}, s_{16}, s_{17} to P_2 and the points s_{18}, s_{19} to P_3.

These public parameters will support polynomials over the field $F_q[x]$ of degree at most D.

Commit. On input ck, univariate polynomial p over $F_q[x]$, PC.Commit outputs the commitment c that is computed as follows:

- If $deg(p(x)) > D$, abort.
- Otherwise, output $c := p(\beta)G$.

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 polynomial $p(x)$ over F_q, evaluation point $z \in F_q$, opening challenge $\epsilon \in F_q$, PC.Open outputs the evaluation proof π in $\mathbb{G}_1$, that is computed as follows. Compute the linear combination of polynomials $p(x)' = \epsilon p$ and witness polynomial $w(x) := \frac{p(x)' - p(z)}{x - z}$. Set $w := w(\beta)G \in \mathbb{G}_1$. The evaluation proof is $\pi := (w)$.

Check. On input rk, commitment c, evaluation point $z \in F_q$, alleged evaluation v, evaluation proof π, and randomness $\epsilon \in F_q$, PC.Check proceeds as follows. Compute the linear combination $C := \epsilon c$, then compute the linear combination of evaluation $y := \epsilon v$ and check the evaluation proof via equality $e = (C - yG, H) = e(w, \beta H - zH)$.

This protocol can be made zero-knowledge by having the prover send vG instead of v.

With this protocol, the subsecret private shares can be verified in the following way:

1) P_i uses PC.Commit algorithm to commit to $f_i(x)$, with $D = t - 1$, and sends it to all participants.
2) P_i makes a proof evaluation of $s_{ik}G$ using PC.Open algorithm and sends it to P_j, so that the other participants can verify s_{ik} without getting to know it.

D. 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 [9], the most efficient cryptosystem for this purpose is the additive ElGamal over elliptic curves (EC-EG), whose key set-up consists of an elliptic curve $\mathbb{G}_1$ over F_p with a generator G of order q. Its security is based upon the Elliptic Curve Discrete Log Problem (ECDLP).

$x \in Fq$ is the private key of the system and $Y = \sum_i^n f_i(0)G = \sum_i^n x_i G = xG \in \mathbb{G}_1$ is the public key. The public key can be verified using the previous scheme, with a proof evaluation that $f_i(0)G$ comes indeed from the previously committed $f_i(x)$ polynomial.

There is an efficient and invertible function $map()$ that maps values (e.g. plaintexts) into points on the curve, and vice versa.

The encryption of plaintext m is done the following way:

- $M = map(m)$ is a point on $\mathbb{G}_1$.
- r is a random value $\in Fq$.
- ciphertext $C = (C_1, C_2)$, where $C_1 = rG$ and $C_2 = M + rY$.

The decryption process is done in the following way:

- $M = -xC_1 + C_2 = -xrG + M + xrG$.
- $m = rmap(M)$, where $rmap()$ is the inverse function of $map()$ and m is, in this specific case, the balance of an account.

In this scheme we don't want the private key to be reconstructed during the decryption process, in order to be able to use it to decrypt multiple ciphertexts. For that, we can use the distributed decryption based on [15], in the following way:

1) P_i computes his decryption share/s $D_k = b_k s_k C_1$, where $b_k = \prod_{k=1}^{t} \frac{j}{j-k}$ is public and can be calculated by any participant.
2) $s_k C_1$ can be proved correct with an equality proof of logarithms [16] of $s_k G$ and $s_k C_1$, with $s_k G = \sum_{i=1}^{n} s_{ik} G$.
3) The message can be decrypted as $M = C_2 - \sum_{k=1}^{t} D_k$.

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 is will have to request 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 has 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 the encrypted balance will have to be composed of various ElGamal ciphertexts ($4 * 32$ bits) or another additive homomorphic cryptosystem will have to be used [9].

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 do not have a sense of time, neither do they 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 and 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 arises with our scheme is if it is possible to maintain the cryptosystem after the update of voting weights without compromising 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 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, with the intent to obfuscate also sender's and receiver's addresses.

VII. Acknowledgements

The authors would like to thank the support of the The RELEASE, RELiablE And SEcure Computation Research Group at the University of Beira Interior and the Instituto de Telecomunicações. This work is funded by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/EEA/50008/2020.

Thank you to Paulo Vieira and Lee Bousfield for valuable insights.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] N. Van Saberhagen, "Cryptonote v 2.0," 2013.

[3] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.

[4] L. M. Goodman, "Tezos : A self-amending crypto-ledger position paper," 2014.

[5] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. USA: USENIX Association, 1999, p. 173–186.

[6] D. Mazières, "The stellar consensus protocol : A federated model for internet-level consensus," 2015.

[7] C. LeMahieu, "Nano: A feeless distributed cryptocurrencynetwork."

[8] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[9] E. Mykletun, J. Girão, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks," vol. 5, 07 2006, pp. 2288 – 2295.

[10] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," Cryptology ePrint Archive, Report 2019/1047, 2019, https://eprint.iacr.org/2019/1047.

[11] A. Kate, G. Zaverucha, I. Goldberg, and D. Cheriton, "Polynomial commitments," 07 2010.

[12] Y. Xiao, N. Zhang, W. Lou, and Y. Hou, "A survey of distributed consensus protocols for blockchain networks," 04 2019.

[13] A. Menezes, "An introduction to pairing-based cryptography," 2005.

[14] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, Oct 1987, pp. 427–438.

[15] N. Ruan, T. Nishide, and Y. Hori, "Elliptic curve elgamal threshold-based key management scheme against compromise of distributed rsus for vanets," *Journal of Information Processing*, vol. 20, pp. 846–853, 01 2012.

[16] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," 1997.