The Dusk Network And Blockchain Architecture

Scalable consensus and low-latency data transmissions for privacy-driven cryptosystems

Emanuele Francioni
Dusk Foundation
Amsterdam, The Netherlands
emanuele@Dusk.network

Fulvio Venturelli
Dusk Foundation
Amsterdam, The Netherlands
fulvio@Dusk.network

ABSTRACT

In order to satisfy a broad set of data transfer scenario, the Dusk network adds an additional layer of security to the IP protocol suite (used mostly in a peer-to-peer fashion). Through the adoption of a mix of established strategies and novel techniques, the Dusk network has been conceived specifically to protect the privacy of the communicating peers from any form of eavesdropping while satisfying a variety of challenging use cases varying from fast communication (e.g. voice calls) to large data transfer (e.g. file transmission). Dusk circumvents the notorious unreliability of crowd-sourced infrastructures by embedding economic incentives into the core mechanism of the network itself. Such incentives are designed to encourage peers to partake in the network in a permission-less, anonymous and private fashion.

KEYWORDS

Dusk, blockchain, cryptocurrency, privacy, consensus, segregated byzantine agreement

1 INTRODUCTION

The Dusk network makes use of a decentralized and privacy-oriented digital currency that evolves the CryptoNote protocol[12] through the groundbreaking discoveries in the field of Byzantine consensus and pseudo-random functions of world renown cryptographers such as Silvio Micali, Michael Rabin, Alexander Yampolskiy and Evgeniy Dodis. Dusk radically departs from any other blockchain by employing an adaptive consensus mechanism, called Segregated Byzantine Agreement (or SBA★), which does not require the computational intensity of proof-of-work and is a fairer alternative to proof-of-stake. Built on such consensus algorithm, Dusk is poised to be the first to simultaneously achieve previously conflicting goals of guaranteeing transaction untraceability and unlinkability, safeguarding user privacy, reaching transactional "finality" after a bound number of rounds within a single block election and achieving virtually unbounded user scalability without any significant performance degradation.

The Dusk network requires a heightened security setup designed specifically to:

(1) Obfuscate IP addresses of the communicating peers
(2) Prevent linkability and traceability of accounts
(3) Guarantee network performance
(4) Implement efficient payment mechanism for high QoS applications such as secure and anonymous voice calls

WEB3 Symposium, April 2018, Amsterdam, The Netherlands

An important difference with CryptoNote, is that Dusk does not make use of proof-of-work mining and therefore drops completely CryptoNight and deviates substantially from the hashing algorithms therein adopted. In particular, Dusk uses what we call Segregated Byzantine Agreement (SBA★) protocol which enhances classic BA★ by implementing specific measures to protect peer privacy. SBA★ has been developed specifically to power the Dusk Blockchain and help meeting the aforementioned requirements. These efforts do not solely relate to the application layer but extend to the networking layer as well. This is why the Dusk protocol makes use of:

- **Stealth addresses**: to protect transaction recipient anonymity
- **RingCT signature**: to protect transaction sender's identity
- **Anonymous Network Layer**: to protect the IP address of the network peers; to provide secure data transfer mechanism; to implement off-line data retrieval strategy; to power the anonymous gossip network for transaction propagation and verification
- **Non-Interactive Verifiable Secret Sharing Scheme**: to conceal all but highest priority time-locked transactions from the participants to the Block Generation sortition
- **Cryptographically Committed Provisioners**: to protect the information about stake; to implement a division of responsibilities between Block Generators and the electable Block Voters and Verifiers; to boost network efficiency by acting as state channel guarantors; to incentivise participation to the network; to protect the balance information of transacting nodes; to prepare SBA★ for future expansion with non-balance and non-payment related weights such as storage contributed to the network (as in proof-of-storage), availability expressed in elapsed time since joining the network (as in proof-of-idle), etc.

2 PRELIMINARIES

2.1 Diffie-Hellman Hardness Assumption

In any group, a discrete logarithm log_b a is a number x $\in \mathbb{Z}$ such that $b^x = a$.

Most of the cryptographic building blocks related to this work are linked to the Diffie-Hellman assumption which uses the hardness of discrete logarithms in cyclic groups [13]. Considering a multiplicative cyclic group $\mathbb{G}$ of order p and generator [2] g, we can formulate the following assumption: given g^a and g^b for uniformly and independently chosen a ,$b \in \mathbb{Z}_p$ then g^{ab} performs like a random element in $\mathbb{G}$ of order p.

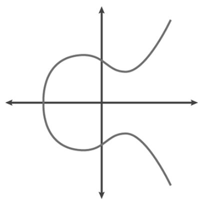

Figure 1: A generic elliptic curve

As a consequence of such assumed randomness, the Decisional Diffie-Hellman (DDH) Problem relates to distinguishing the following two probability distributions:

- $(g^a, g^b, g^{ab})\ \forall a, b \in \mathbb{Z}$
  // (g^a, g^b, g^{ab}) are defined as a Diffie-Hellman Tuple
- $(g^a, g^b, g^c)\ \forall a, b, c \in \mathbb{Z}$

2.2 Hiding Recipients: Stealth Addresses

Inspired by the CryptoNote white-paper[12], stealth address technology is at the basis of Dusk recipient hiding technique. Already widely tested in other privacy-oriented digital currencies, it is the proven choice for concealing the true recipient address of a transaction while keeping uniqueness within the context of the ledger (meaning no other address can be linked to a stealth address). Additionally, a derivation of an unbound number of receiving addresses is also possible without any of them allowing traceability back to the recipient's main address. As an anonymous key agreement protocol, Dusk uses the Elliptic Curve Diffie-Hellman (ECDH) due to the desired property of allowing two parties to generate a shared secret by solely knowing each other's public key, and the generator point of the Elliptic Curve used in the Twisted Edward equation. Following is a detailed explanation of how Dusk implements Stealth Address technology.

2.2.1 Elliptic-Curve Cryptography

The system makes use of Elliptic-Curve Cryptography (ECC), hence approaching public-key cryptography through the algebraic structure of elliptic curves and thus allowing for the creation of smaller and more efficient cryptographic keys. ECC gives the same security levels of, for example, RSA, but using a much smaller security key.

The structure of an elliptic curve is a plane curve satisfying the equation $y^2 = x^3 + ax + b$, which returns us the graph in Figure 1.

In ECC, a Galois Field is created by taking the modulo of all points using a large prime number, creating a finite number of values for the used equation. The following axioms are furthermore taken into account:

(1) A point can't be multiplied or divided by another point.
(2) Any point on the curve can be added or subtracted to another point (or itself).
(3) Adding a point to itself allows for scalar multiplication.

2.2.2 Private and Public Keys

The Dusk Blockchain utilizes an *Ed25519* curve, which is a Twisted Edwards curve with the following Elliptic Curve Parameters:

q : a prime number; $q = 2^{255} - 19$; *This is the number of points in the curve.*

d : an element of $\mathbb{F}_q$; $d = -121665/121666$; *Value used in the curve equation below*

E : an elliptic curve equation; $-x^2 + y^2 = 1 + dx^2y^2$; *The Twisted Edwards curve/equation we are using*

G : a base point; $G = (x, -4/5)$; *The*****generator**** *point. This is a base - starting point used for all Elliptic modulo operations.*

l : a prime order of the base point;
$l = 2^{252} + 27742317777372353535851937790883648493$;
The order of the base point G. This defines the maximum size of scalars and the maximum number of points that can be used.

$\mathcal{H}_s$: a cryptographic hash function $\{0,1\}^* \to \mathbb{F}_q$;

$\mathcal{H}_p$: a deterministic hash function $E(\mathbb{F}_q) \to E(\mathbb{F}_q)$;

All private and public keys in Dusk will be using 64 hex characters.

2.2.3 Accounts and Addresses

The following procedure will be used to create an address.

(1) We pick a random /textitprivate spend key, by generating 256 random bits, and reducing $\mod l$. We call this b.
(2) b is hashed with hashing algorithm H(*Keccak_256*). We interpret the result of the hashing as an integer, reduce it $\mod l$ as before. We call this key a.
(3) We generate our public spend and view keys $B = bG$ and $A = aG$
(4) We hash (network prefix (0xEF) + B + A) with H.
(5) Append the first 4 bytes of this operation to (prefix + B + A), obtaining a 69 bytes value (1 + 32 + 32 + 4)
(6) Convert this to *cnBase58*.

We will explain how stealth addresses work by first going trough a brief explanation about key exchanges on an ECC scenario, in the next section.

2.2.4 The Elliptic Curve Diffie-Hellman

The Elliptic Curve Diffie-Hellman (ECDH) is an anonymous key agreement protocol, a variant of the Diffie-Hellman protocol adapted to work with Elliptic-Curve Cryptography.

Thanks to ECDH, two parties can generate a shared secret over an unsecured connection only by knowing each other's public keys, and the generator point of the Elliptic Curve used in the ECC equation.

To demonstrate this, we will use Alice (with private key a and public key A=aG) and Bob (with private key b and public key B=bG). (Where G is the generator point)

As previously stated, points on a curve can be added together, and Alice could calculate a point C = A + B, but this could also be potentially done by anyone eavesdropping the conversation, since A and B are publicly available.

Now, let's remember that A and B are points on the elliptical curve, and that we can add a point to itself.

Alice can now calculate a new point D=aB, and Bob can get D'=bA.

We can now prove that D=D', and thus Alice and Bob share a secret by operating on ECC and knowing each other's public keys and the generator point G:

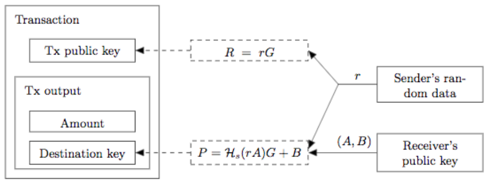

Figure 2: A *stealth transaction*

(1) Given a common generator point G;
(2) Alice has a=5, A=5G (private and public keys)
(3) Bob has b=7, B =7G
(4) a · b = 35
(5) Alice computes D=aB=5B=5·7G=35G
(6) Bob computes D'=bA=7A=7·5G=35G
(7) D=D'

Point D has a corresponding scalar d, in the example above equal to 35, which is a shared secret between Alice and Bob

2.2.5 Stealth Addresses

Let's consider the diagram in Figure 2 from the CryptoNote whitepaper.

The *Dual-key Stealth Address P* is defined as $P = \mathcal{H}_s(rA) \cdot G + B$. The link-ability of the Stealth Address is achieved by using a combination of spend/view-keys, without actually allowing a spending transaction to take place. Let's now assume that Alice has a private *spend-key* z and a private *view-key* y. We call her public *spend/view-keys* Z, Y. On the other side, Bob's public keys are A and B, with Bob's private keys a and b unknown to Alice. In order to build a stealth address, Bob needs to compute r (an arbitrary random scalar chosen by Alice) and R as the corresponding ECC point such as $R = rG$. r is not being shared with anyone and can be discarded after its use - unless Alice wants to prove that she sent a transaction to Bob to an external party. R is added to the transaction so that it can be seen by everyone. A new r is calculated for each transaction, since reusing it in the computation of the Stealth Address, would result in a collision. Therefore, given the equation above: $P = \mathcal{H}_s(rA)G + B$, a Stealth Address can be constructed as follows.

(1) P : the Stealth Address where the funds will be sent
(2) $\mathcal{H}_s$: a Hashing Algorithm returning a scalar value
(3) r : the random scalar chosen by Alice
(4) A : Bob's public view-key
(5) G : The standard Ed25519 base point
(6) B : Bob's public spend-key

Alice calculates point D from ECDH using a randomly chosen r and Bob's public key A.

Bob computes D independently from Alice, due to the properties of ECDH.

Alice computes the scalar $f = \mathcal{H}_s(D)$ - (this hashing step creates unlinkability between Bob's address and the new stealth one) and calculates F=fG, and P = F + B (Bob's public spend-key).

Now, in order to check if he is the transaction's recipient, Bob:

(1) Calculates D' using R as propagated with the transaction (The equality D = D'=aR is still unproven).

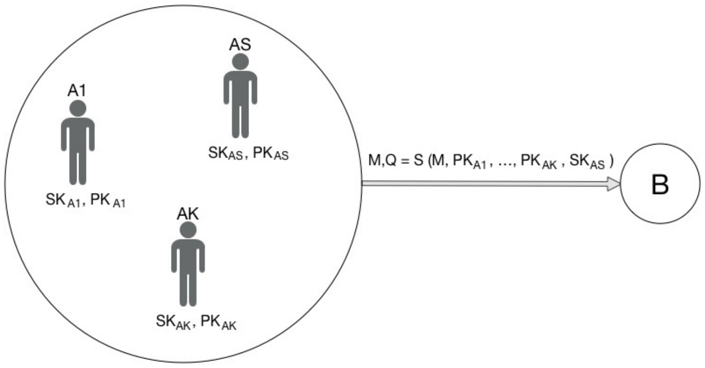

Figure 3: A *ring signature transaction*

(2) Calculates f'=$\mathcal{H}_s(D')$
(3) Calculates F'=f'G
(4) Calculates P'=F'+B

If P'=P, then Bob knows the transaction was intended for him, and can retrieve it like this:

(Bob has to compute the secret key associated with the transaction)

(1) Bob knows f' (computed above), and derives xf'+b (Bob's private spend key)
(2) Knowing that P=xG, we got P=xG=(f'+b)G
(3) Bob then has to check if the transaction on P is spent, Bob computes a key-image and checks on the blockchain if the image linked to that transaction has been spent. Image I=x$\mathcal{H}_p(P)$
(4) Bob can then sign a new transaction using x

2.3 Obfuscating Sender

Ring Signatures are an efficient, established way to obfuscate the input of a transaction by making use of a sender's account keys and a number of decoy keys (called *outputs*) taken directly from the blockchain, using a triangular distribution method. The technology finds its roots in the early days of blockchain research, as Satoshi Nakamoto himself was hypothesizing back in 2010:

"Crypto may offer "key blinding". I did some research and it was obscure, but there may be something there. "Group signatures" may be related." [1]

The procedure allows one of the members of the *ring* to sign messages on behalf of the whole group, and by doing so it renders it infeasible to know exactly which member signed. (Figure 3)

In a group signature procedure, there is no central management or setup - all the signer needs is the public keys of the members she chooses to be part of the ring.

Each signer is associated with a public key PK_x and a corresponding private key SK_x. A ring signature scheme is defined by two procedures:

- ringSign($M, PK_1, PK_2, ..., PK_r, s, SK_s$): which is a method for computing the ring signature Q, which gets the message M, the public keys of the ring members, and the private key of the message signer.

[1] https://bitcointalk.org/index.php?topic=770.msg9074#msg9074

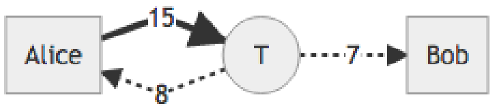

Figure 4: A *ring stealth transaction*

- ringVerify(M, Q): which is a procedure to verify the signature Q, which gets message M and signature Q as arguments, and returns a boolean (correct / not correct) as output.

2.3.1 Signature Generation

Having a message m and its own private key SK_s, together with a sequence of ring member's public keys, signer A_s can create a signature as follows:

(1) Computes a key $k = h(m)$, where h is a collision-resistant hash function.
(2) Chooses a random v
(3) Chooses a random x_i and computes $y_i = g_i(x_i)$
(4) Solves for y_s the equation containing the *Combining function* $C_{k,v}(y_1, y_2, ..., y_r) = v$
(5) Finds x_s knowing its own SK_s: $x_s = g_s^{-1}$
(6) Creates a ring signature: $v; x_1, x_2, ..., x_r$

2.3.2 Signature Verification

A verifier can confirm a ring signature as follows, having the collection of the member's *public keys:*

(1) Compute, for each x_1, $y_1 = g_i(x_1)$
(2) Computes $k = h(m)$
(3) Confirms the equation $C_{k,v}(y_1, y_2, ..., y_r) = v$ for values of y_i
(4) If the equation is correct then the signature is valid, otherwise it's not.

2.3.3 Ring Confidential Transactions

Blockchains such as Monero use a particular breed of ring signatures called Ring Signature Confidential Transactions (*RingCT*), where the privacy is taken one step further by not only giving full anonymity at the sender level (as explained above), but also on the amount spent and destination.

The RingCT technology makes the payment virtually unlinkable to the original spender, it is fast, and it also conceals the amount being transferred. Confidential Transactions include cryptographic proof that, given a set of input amounts, proves that their sum equates the output amounts, without revealing them. In practice, if *Alice* has an output of 15 Dusk and wants to send *Bob* 7 Dusk, she will have to spend the output in its entirety on transaction T, and then send the change (8 Dusk) back to herself. (Figure 4)

This commitment is represented by the formula:

$R_{ct} = xG + aH(G)$

In the formula, a is the amount sent out in the transaction, x is a computed random value. By publishing the value of R_{ct} to the network as an output, the network will be able to verify the legitimacy of the submitted transaction. This technology goes on top of the already untraceable (Stealth) addresses used by the Dusk blockchain, and anonymous networking used to give full anonymity to the nodes involved.

In the Dusk Network, RingCT is used by default for all transactions except those used to participate to the sortition for Block Generator (which are normally *ring signed*). This is merely a detail, though, considering that an external observer would not be able to tell these two kind of transactions apart from each other.

2.3.4 Future Development: Bulletproof Transactions

Initially, non-interactive proof of knowledge based on [Fiat-Shamir heuristic [10] have been taken into consideration in order to conceal transaction information such as sender and amount. However, it quickly appeared that technologies developed on top of its concept such as zk-SNARKs [5] require prohibitive computational power and time of processing for transaction generation. Even the more recent zk-STARKS [6]) proved impractical to integrate in Dusk because of the problematic hurdle of needing extremely bulky verification proof (as big as several hundreds of kilobytes), although solving the problematic reliance on a trusted third-party for system setup and being substantially simpler than zk-SNARKS in terms of cryptographic primitives (they do not make use of Elliptic Curves, nor Public Key cryptography).

The most promising advancement in the field of transaction confidentiality is probably given by the so-called Bulletproof Transactions [4], which would appear to provide a substantial improvement over RingCT in terms of cryptographic proof size. At the time of this writing, first tests show a tenfold reduction in size and verification times (although still not reaching the same level of efficiency if compared to a zk-SNARK proof). Given the "pluggable" nature of Dusk core, the underlying software for transaction generation will be kept flexible through a plugin architecture in order to facilitate the adoption of a future implementation of Bulletproof Transactions as soon as an emerging library proves stable enough for integration.

2.4 Cryptographic Accumulator

The set of algorithms known as cryptographic accumulators have been developed to allow hashing of a finite and potentially large set of values into a single succinct value, called the Accumulator. The algorithm enables efficient computation of a witness for every accumulated input that proves its membership in the Accumulator. A dynamic Accumulator is a special kind of Accumulator which permits efficient input addition and deletion where the computation costs of these operation is independent on the number of values added or deleted. As such, it is a space and computational time efficient data structure initially developed for testing membership.

Formally, a dynamic Accumulator is a tuple of algorithms defined as follows:

- Generate(1^k) $\rightarrow (sk_{acc}, pk_{acc})$: Given a security parameter k return a (private, public) key pair. Notice that the parameter k is sensitive information (trapdoor) which could deterministically recreate the private key
- Eval($(sk_{acc}, pk_{acc}), \chi$) $\rightarrow (acc_\chi, aux)$: Probabilistic algorithm that gets the key pair and a set χ to be accumulated and returns the Accumulator and some auxiliary data
- WitnessCreate($(sk_{acc}, pk_{acc}), acc_\chi, aux, x_i$) $\rightarrow wit_x \vee \perp$: Creates a witness wit_x if $x_i \in \chi$, $\perp$ otherwise

- Verify($pk_{acc}, acc_{\chi}, wit_x, x_i$) $\rightarrow wit_x$is a witness for x_i: This is the actual membership verification
- Add($sk_{acc}, pk_{acc}, acc_{\chi}, aux, y_i$) $\rightarrow acc_{\chi'} \vee \perp$: If the element y_i is not already in the set χ, returns the updated Accumulator $acc_{\chi'}$ with $\chi' \leftarrow \chi \bigcup \{y_i\}$
- Delete($sk_{acc}, pk_{acc}, acc_{\chi}, aux, y_i$) $\rightarrow acc_{\chi'} \vee \perp$: If the element y_i is in the set χ, returns the updated Accumulator $acc_{\chi'}$ with $\chi' \leftarrow \chi \backslash \{y_i\}$
- WitnessUpdate((sk_{acc}, pk_{acc}), wit_x, aux, x_i) $\rightarrow wit'_x \vee \perp$: Updates the witness for x_i in case it has been added or deleted (aux describes which of the two)

Different technologies exist that implement Accumulators with different level of computational efficiency and security requirements. Currently, Dusk is experimenting to achieve the best tradeoff between a decentralized choice and efficiency of Accumulator technology. Following are the technologies under evaluation.

2.4.1 RSA Accumulators

Abundant literature has been developed regarding many different Accumulators, particularly the so-called RSA Accumulators. Considering *strong RSA setting*, RSA Accumulators (e.g. Carmenish's and Lysyanskaya's Dynamic RSA Accumulator [1]) are based on one-way RSA function for a suitably calculated $N = pq$ where p and q are sample primes with polynomial dependence on the security parameter k and therefore are effectively the Accumulator's *trapdoors* which need to be **destroyed** immediately after parameters are generated. As an alternative, the employment of RSA-2048 could be used to circumvent the need for developers to know the security parameters and act as trusted party, considering that the related security parameter k is claimed to be destroyed and no factoring solution to the RSA-2048 number has been found for the past 25 years [2], despite a $200,000 incentive offered.

$acc_{\chi} \leftarrow g^a modN$ // The set of members $a = \{a_1, .., a_n\}$ is compactly represented by the Accumulator $acc_{\chi} = g^{\prod a \in \chi}$
$wit_i \leftarrow g^{(a_1, .., a_{i-1}, a_{i+1}, .., a_n)} modN$

2.4.2 Expressive Bilinear Accumulator

Under such security assumption, dynamic Accumulator schemes from *bilinear pairings* have been developed in the literature. Among those, an expressive *zero-knowledge* set Accumulator has been formalized by Zhang, Katz and Papamanthou [16] capable of providing succinct proofs for a large collection of operations over accumulated sets, among which it is of particular interest the SUM operation, which could find application for rapid and zero-knowledge calculation of accumulated Provisioners stakes.

However, similarly to RSA, bilinear pairing Accumulators present also the drawback of relying on the security parameter k.

2.4.3 Elliptic Curve Multiset Hash

Shepard, Tibouchi and Aranha [14] teach of a new and efficient method to "*associate a hash value to arbitrary collections of objects (with possible repetitions) in such a way that the hash of the union of two collections is easy to compute from the hashes of the two collections themselves: it is simply their sum under a suitable group operation*". This association is the basis for an Elliptic Curve Multiset Hash which constructs a homomorphic multiset hashing on top of efficient *BLAKE2* hash function and binary elliptic curve encoding. This allows for incremental/parallel computation of a hashing function for various applications including efficiently testing for subgroup membership. This makes ECMH an appealing alternative to the other known algorithms to construct Accumulators, especially since the method is unencumbered with the necessity of a trusted setup.

3 SEGREGATED BYZANTINE AGREEMENT

The Dusk blockchain is built upon a novel consensus mechanism, called *Segregated Byzantine Agreement* (SBA★), engineered to provide the best possible tradeoff between security, efficiency and flexibility.

SBA★ complements the idea of *Cryptographic Sortition, Player Replaceability* and *Ambiguity Resilience* (firstly introduced by the BA★ consensus developed by Micali and the MIT CSAIL [9]) with the new concepts of *stealth time-locked transactions* as a method for Sybil resistance and *non-interactive verifiable shared secret* scheme to implement a simple but secure (t,n)-threshold secret sharing for keeping sortition auditable solely by a rotating set of *pre-block Verifiers.*

In order to both improve network and block-generation efficiency and privacy of transacting peers, SBA★ allows only normal transactional nodes to compete for block-generation, while it restricts the computationally and network intensive tasks associated with verification, voting and notarisation (VVN operations) to non-transactional nodes called *Provisioners.* Provisioners harden the Dusk Blockchain by decreasing network communications across VVN operations, improving the time-to-finality through a notarisation process, decreasing the probability of network partitioning and contributing to the overall availability of Dusk Network.

Furthermore, they contribute distributed storage infrastructure in the Dusk On- Offline File Transfer, and enable runtime payments of high QoS transmission by handling *state channels* for communicating peers through a novel mechanism of Secure Tunnel Switching.

BA★

Cryptocurrencies powered by message-passing Byzantine Agreement (BA★), use cryptographic sortition in order to carry out the functionalities of block proposals, validations and subsequent insertion into a tamperproof sequence of blocks.

In its essence, each node of the network, while collecting (and further relaying) pending transactions, runs a computationally lightweight process that yields in a pseudo-random fashion which role such node should assume in the operations of production and validation of a new block. The great advantage of BA★ compared to other consensus mechanisms based upon *proof-of-work* or *proof-of-stake*, lays primarily in avoiding any possibility of forking the Blockchain and thus preventing any ambiguity about which branch will become the dominating one.

This translates into the appealing property of achieving transaction "finality" as soon as consensus is reached for a block. In the best case scenario this happens after 2 rounds of non-interactive

[2]https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

sortition. In the worst case scenario (weak network synchrony controlled by adversaries for a long but bounded period of time), BA⋆ achieves block finality after 9 rounds.

BA⋆ consensus is a very convincing engine for powering open cryptocurrencies which do not require privacy. The election of Block Generator and Block Voter requires the total weights available in the system and each *sortition candidate*'s balance to be public and known by all peers in order to allow blocks proposed by higher priority members to be propagated within the network and validated by multiple voting committees.

This is problematic for a privacy-oriented digital currency such as Dusk which strives to protect the data of the transacting actors. SBA⋆ therefore focuses primarily on the privacy of transactional nodes while keeping auditable (but not public) only essential information about nodes participating in block generation, validation and notarisation.

3.1 Consensus Outline

The BA⋆ algorithm has been developed in a direction unsuitable for Dusk because transactions are supposed to be propagated in clear, nodes do not keep private state except their private keys and there is no measure to prevent every node in the network to learn of every other node's balance, IP address and, ultimately, providing a mere pseudonimity to protect their identity.

We propose here a new approach that evolves BA⋆ into what we call SBA⋆ (as in *Segregated Byzantine Agreement⋆*) which finally renders the consensus mechanism an outstanding choice for privacy-oriented currencies (such as Dusk) due to its quick finality mechanism, minimal amount of computation required and speed in block production.

SBA⋆ foresees different subsequent cyclic phases, all of which are non-interactive, with the sole exception of the *Validation*, where the NIVSS_Reconstruct requires $O(1)$ communications to complete.

Block Generation Sortition

Run by nodes with a time-locked stake called *Block Generator* candidate (or simply a "*candidate*"). During this phase the candidates run the VRF, calculate their priority, run the Non-Interactive Verifiable Shared Secret Protocol and propagate a *pre-block* (which is a block proposal that still needs to get through various round of consensus and gets decorated with additional meta-data along the way) together with the various proofs.

Default Block Generation Sortition

Run by Provisioners in parallel to the Block Generation Sortition. In the eventuality that the Block Generation Sortition produces no *candidate* with priority higher than zero or the Validation phase fails, Provisioners run the classic BA⋆ Sortition algorithm to supply a *default pre-block.*

Validation

Run by a subset of Provisioners called Verifiers. Verifiers run the *Secret Reconstruction Protocol*, validate the priority information of the highest priority candidate and either sign on the pre-block proposal of the *candidate* or a *default pre-block.* Additionally, the *Verifiers* validate the witness wit_P of candidate Provisioners committing their stake to the Accumulator if any. In this case, this information gets added to the pre-block

Voting

A number of rounds each of which run by a different subset of Provisioners called Voters. BA⋆ proves that the amount of round is optimistically 4 for a strongly synchronous network and 9 for a weakly synchronous one with a strong adversarial presence controlling the network (albeit for a finite period of time)

Notarization

The Voters which reached voting consensus on the pre-block are called Notaries. The public key of the *Notaries* are added to the pre-block and they play the role of *Verifiers* in the next block's Validation phase. The pre-block is finally turned into an official block by the *Notaries* by adding the Block Reward information.

3.2 Verifiable Random Function

The cryptographic sortition is implemented through the so-called Verifiable Random Function (VRF). Formally, for a generation algorithm *GEN* producing the key pair (P_k, S_k), a proving algorithm $PROVE_{Sk}(x)$ which outputs a pair of function value and proof of correctness $(F_{Sk}(x), \pi_{Sk}(x))$, a **VRF** is a function for which it exists a verification algorithm $VER_{Pk}(x, y, \pi)$ which verifies that $y = F_{Sk}(x)$ using proof π.

In practice a node running a VRF can prove the output it received by propagating π together with the VRF result. This way a node, by running a VRF and communicating its output, can convince its peers in a non-interactive fashion (meaning without talking to anyone else in the network) whether he has been selected by "the cryptographic sortition" to play a specific role in the current block creation round.

The roles are either *Block Generator* or *Block Verifier.* The elected node will then proceed to perform the steps foreseen by its role independently from all the other nodes (regardless of their role) and thus propagate the result of such operations to the network together with their allotment proof as outputted by the VRF. The nodes forfeit their role and become irrelevant to the consensus as soon as they communicate with the network. This way, the algorithm makes it virtually impossible for an adversary to target nodes participating in the block election or transaction validation. Only a small fraction of the nodes each round get randomly selected.

The likelihood of positive VRF output depends on a weight correlated to the amount of digital currency committed by a node. In *BA⋆* this is the public balance of each node, in SBA⋆ this depends on the sortition role (i.e. Block Generators commit a payment while Provisioners commit a stake). Regardless of the type of commitment, the weight mechanism is used to prevent Sybil attacks since it renders probabilistically and economically disadvantageous for a node to replicate itself over several different Sybil processes, since such behaviour would actually decrease its chance of winning the ballot.

3.2.1 VRF And Sortition Procedure

Through VRF sortition, a number of Block Generator candidates are selected each round. The candidates propagate their proposed

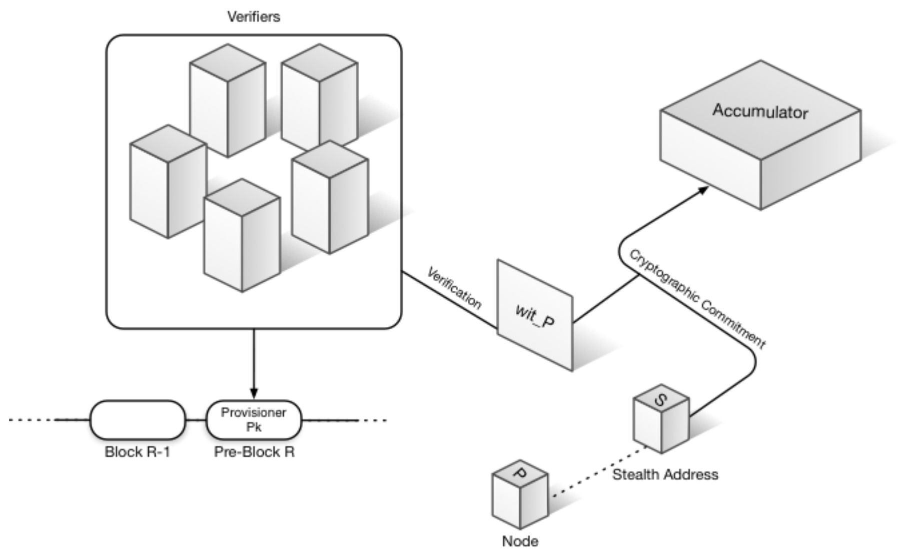

Figure 5: Provisioners setup

block together with the VRF output, which includes the proof of winning the sortition together with the priority given by the digital currency balance of the node. Peers collect and propagate gossiped packets, forwarding solely the message with the highest priority and discarding all the others. The amount of time peers are supposed to collect messages is usually part of the protocol configuration and is suggested to be 5 seconds.

At the end of this period, peers which did not receive a message propagate an empty block (which is a perfectly valid outcome of the consensus). From a security perspective, an adversary could orchestrate an attack by deceiving peers into proposing different blocks. However, this attack would fail unless the adversary would have the highest priority in a round and if the number of honest Provisioners are also less then 2/3rd.

3.3 Cryptographically Committed Provisioners

One approach to protect the stake information was the employment of Order Preserving Encryption and Homomorphic Cypher [3], which would have theoretically allowed for priority comparison without revealing the balance of the different nodes. This approach however has been discarded because it is susceptible to binary search attack. In fact, by exposing priority information, any attacker with access to the oracle could ultimately obtain information about the original balance of a node simply by performing multiple comparisons with the hash of a known number. Additionally, every node in the network is required to perform validation on the packets it receives.

Since the priority of the Block Generator is an input parameter for validating its VRF's output, the measures required to provide access to such datum in a secure setting would result in an unacceptable degradation of performance and increase in complexity.

Instead, in the attempt to obtain the best trade-off between efficiency and anonymity, the system restricts the opportunity to perform VVN operations solely to non-transactional nodes called *Provisioners* organized in elected subsets which form different committees throughout the algorithm rounds. In order to be eligible to be a Provisioner, a node P uses a non-interactive cryptographic commitment to bind a predefined minimum amount of Dusk coins C into a collective anonymous escrow and creating a *spend* transaction toward a *stealth address* obfuscating P's address. Such *spend* transactions are kept concealed until the node is ready to cash back its stack and quit its role as Provisioner.

The system verifies through an efficient (i.e. within *polynomial-time*) *non-interactive zero knowledge proof* wit_M that P actually committed its stack in the Accumulator. This means that as soon as the Provisioner commits its stake to the Accumulator, it announces itself as a Provisioner to the network by gossiping its public key, its Dusk Network Address and wit_P. The drawback of relying on a trusted setup is circumvented by using the RSA-2048.

Also, the notoriously big size of spend proof (estimated to be 45kb in the Zerocoin whitepaper [15]) is not problematic in the case of Provisioners since their stake is meant to be in a long-term escrow. This is easily enforceable at protocol level.

3.4 Non-Interactive Verifiable Shared Secret Protocol

Verifiable Secret Sharing (VSS) is a cryptographic protocol designed to allow a dealer to decompose a secret in n fragments and share them publicly to n peers (players) so that only a subset (*threshold*) t or those fragments are needed to reconstruct the original secret. In a VSS, through the addition of auxiliary information, players can verify reception of a "valid" fragment without acquiring any knowledge of the initial secret.

The Non-Interactive Verifiable Secret Sharing Protocol is the Dusk variation of the Simplified VSS protocol of Gennaro, Rabin and Rabin [11]. In our protocol, the *dealer* does not communicate directly with the *players* but can only rely on multiple untrusted message relayers as it is the case using the Dusk gossip protocol. The protocol foresees a *sharing phase* () and a *Reconstruction Phase*. The constant term s of $f(x)$ is the secret. The second polynomial $r(x)$ is used to generate t independent random strings used to commit to the shares.

The verification performed by player P_i after decrypting its share S_i from the share vector S, with its own private key sk_i is trivially to recompute $\mathcal{A}_i = C(\alpha_i, \rho_i)$ and check that the equation holds. $C(x, r)$ is a commitment hash function such as Skein or SHA-3.

Algorithm 1 Share secret in a verifiable and non-interactive way

```
1: procedure NIVSS_Share($s, V$)    ▷ $s$ is a secret, $V$ a list of $n$ Verifiers $pk$
2:     $f(x) = s + a_1x^1 + .. + a_tx^t$    ▷ $f(x)$ is a random polynomial
3:     $r(x) = r_0 + r_1x^1 + .. + r_tx^t$    ▷ $r(x)$ is a random polynomial
4:     $S = \{\}$
5:     $\mathcal{A} = \{\}$
6:     for each player $i$ in $V$ do
7:         $(\alpha_i, \rho_i) \leftarrow (f(i), r(i))$
8:         $S_i \bigcup = \{Enc_{pk_i}(\alpha_i, \rho_i)\}$
9:         $\mathcal{A}_i \bigcup = \{C(\alpha_i, \rho_i)\}$
       return $\langle S, \mathcal{A} \rangle$
```

Algorithm 2 Reconstruct the secret (View-Key)

```
1: procedure NIVSS_Reconstruct($S, \mathcal{A}$)
2:     $\langle \alpha_p, \rho_p \rangle \leftarrow Dec_{pk_p}(S[p])$
3:     $\text{Sig}^p{}_{\alpha_p||\rho_p} \leftarrow \text{Sig}_{sk_p}(\alpha_p||\rho_p)$
4:     $\text{Gossip}_V(\langle \alpha_p, \rho_p, \text{Sig}^p{}_{\alpha_p||\rho_p} \rangle)$    ▷ propagate share solely to other verifiers
5:     $\mathcal{F} \leftarrow \{\}$    ▷ List of all shares (i.e. points of $f(x), r(x)$)
6: loop onShareReception:
7:     $\langle \alpha_i, \rho_i, \text{Sig}^i{}_{\alpha_i||\rho_i}, \mathcal{H}^{\rangle}(\lfloor\rfloor), \text{Sig}_{pk_c}(b_c) \rangle \leftarrow$ INPUT
8:     if $\mathcal{H}^p(b_c) \neq \mathcal{H}^I(b_c)$ then    ▷ If different pre-blocks...
9:         return $Complaint_{pk_c}$    ▷ ...candidate is dishonest
10:    else if $\text{Vf}(\text{Sig}^i{}_{\alpha_i||rho_i})$ then
11:        $\mathcal{F} \bigcup = \{(\alpha_i, \rho_i)\}$
12:        if $length(\mathcal{F}) \geq t + 1$ then
13:            interpolate $\hat{f}(x)$ and $\hat{r}(x)$ that touch all $\mathcal{F}$
14:            return $\hat{f}(0)$
15:        end if
16:    else
17:        return $Complaint_{V_i}$
18:    end if
19: end loop
20:    return await onShareReception
```

3.5 Block Generation Sortition

A node that wishes to participate in the Sortition (and become a *Block Generation candidate* or simply "*candidate*") is first required to lock an arbitrary amount of Dusk. A *time-locked transaction* is a special transaction where the output is un-spendable for a pre-determined period of time. In order to participate to the Sortition lottery for proposing a block, the candidate performs a time-locked transaction toward a Stealth Address $s\_addr_{pk^2}$ where the recipient party's key is the public key p_k (which is thus used twice). Such a transaction is indistinguishable from a normal one. When propagating the result of the sortition, the *Block Generator candidate* runs the dealer part of the NIVSS Protocol to encrypt the meta-data composed by the *view-key* for the time-locked transaction and by the locking period expressed in number of blocks into a shared secret. Thus, it propagates such secret, together with the proof for the NIVSS and the hash of the meta-data signed with the node's private key s_k.

Following is a description of the Block Generator Sortition. Whereas:

- VRF is the Elliptic Curve Verifiable Random Function 3.2 as described in NSEC5 For Elliptic Curves [8]
- (pk, sk) are the public/private keys of the node with a candidate Block Generator
- *seed* is a public random seed chosen and propagated together with the last block
- Θ is the maximum number of candidate Block Generators per sortition
- π is the proof allowing anyone knowing pk_i to check that the hash corresponds to the input parameters of the VRF
- vk_{tx} is the Block Generator's view-key of a stealth time-locked transaction
- w_P is the sum of all balances of all Provisioners
- W is the amount of circulating Dusk
- V is the list of n validators identified by their public key pk_i

In Algorithm 3 we find the pseudocode for the Block Generator sortition procedure for a node N.

Algorithm 3 Sortition for generating the pre-block with priority j

```
1: procedure BlockProposerSortition($seed, \Theta, w_P, W, V$)
2:     $\langle hash, \pi \rangle \leftarrow VRF_{node.sk}(seed)$
3:     $p \leftarrow \frac{\Theta}{W - w_P}$    ▷ $p$ is the maximum probability to be selected in $N$ extractions
4:     $j \leftarrow 0$
5:     $(n, t) \leftarrow time\_locked(s\_addr_{node.pk^2})$    ▷ tuple of $n$ Dusk and $t$ nr of blocks
6:     while $\frac{hash}{2^{len(hash)}} \notin [\sum_{k=0}^{j} Pr(k; n, p), \sum_{k=0}^{j+1} Pr(k; n, p))$ do
7:         $j$++
8:         $\langle S, \mathcal{A} \rangle \leftarrow$ NIVSS_Share($vk_{tx}||t, V$)
9:         $\sigma_{n,t} = \text{Sig}_{node.sk}(vk_{tx}||t)$
       return $\langle hash, \pi, j, \sigma_{n,t}, S, A \rangle$
```

A node runs the sortition process to become a Block Generator by calculating the pseudo-random *hash* and the proof π by feeding the VRF its secret key s_k and the *seed* calculated and propagated when last block was chosen. The VRF returns a *hash* which is essentially a value with uniformly distributed probability between 0 and $2^{len(hash)} - 1$ where the *len* function is the bit-length of the *hash*. The priority j for the node is calculated by considering each unit of Dusk in the *time-locked transaction* performed toward the stealth address $s\_addr_{pk^2}$ as an independent runner for the Block Generation lottery. This means that for each node N participating in the sortition with a *time-locked payment* of n amount of Dusk there are exactly n runners at every selection round.

The probability for obtaining k selections out of n extractions follows the binomial distribution $Pr(k;n,p) = \binom{n}{k} p^k(1-p)^{n-k}$ where the sum of all probabilities $\sum_{k=0}^{n} Pr(k;n,p)$ is naturally 1. The set representing all possible probability values $[0,1)$ gets split into adjacent intervals $I^j = [\sum_{k=0}^{j} Pr(k;n,p), \sum_{k=0}^{j+1} Pr(k;n,p))$ for $j \in 0,1,\ldots,n$. If $hash/2^{len(hash)}$ falls in the intervals I^j, then j is the priority of the node's sortition and it is verifiable by knowing the VRF's output *hash* (proven by π) and the amount n. While in *BA*★ n is the information about a node's entire balance, in *SBA*★ this amount is instead encrypted by the node and shared among the *Verifiers* using the NIVSS algorithm and kept verifiable solely by a threshold t of the n members of the Verifier Committee.

Differently from *BA*★, in *SBA*★ the probability of obtaining a total amount of Θ positive extractions is 1 only if all nodes participate to the sortition with their whole balance. This is seldom the case and therefore the probability that no candidate gets selected to propose a block is greater than zero. To obviate to this eventuality and still produce a block in case a dishonest Block Generator gets caught, Provisioners run their own parallel Block Generator sortition as a fallback scenario for those cases. Also, to mitigate the potentially reduced probability to generate a successful sortition with multiple candidates, Θ is chosen to be substantially higher than the τ parameter of *BA*★.

3.6 Verification — Pre-Block Propagation

During the gossip procedure, each node relays solely the pre-block with the claimed highest priority (Provisioners will also gossip the default pre-block to the other Provisioners), while dropping all other pre-block proposals. In *SBA*★, protection from Sybil attacks is granted by the *time-locked payment* made by the *Block Generator candidate* which is not in clear. Therefore nodes and Provisioners other than *Verifiers* could only perform validation on the VRF result *hash* and the proposed pre-block. They do not engage in priority validation, which is entirely demanded to the *Verifiers*. This has the positive side-effect to perceivably decrease network latency during gossip operations.

Verifiers run the VerifyBlockProposerSortition in order to reconstruct the *view-key* and the *time-locked transaction* propagated by the Block Generator candidate and be able to validate the claimed priority. Depending on the outcome, they either sign and propagate the candidate's pre-block or the default pre-block. This does not really require consensus since the propagated pre-block is a mere result of the validation operation, which gets further audited by the different *Voter Committees*. As such the probability for propagating mismatching pre-blocks is negligible. In the future we will explore the possibility to use Probabilistic Checkable Non-Interactive Zero Knowledge Proofs to propagate a very efficient proof of validation without revealing the candidate's *view-key* any further than the *Verifiers*.

Whereas:

- j_c is the claimed candidate's priority
- t_{block} is the time when the pre-block has been produced
- pk_c is the *candidate's* public key
- ctx represents an object encapsulating the state of the ledger
- b_c is the *tuple* of the pre-block candidate hash $\mathcal{H}(block)$ and the transaction list tx_{block}
- $b_{default}$ is the default pre-block propagated by the appropriate Provisioner's committee

Algorithm 4 Verify and propagate pre-blocks

```
1: procedure PropagateVerifiedBlock(hash, $\pi, ctx, pk_c, j_c, b_c, S, \mathcal{A}$)
2:     $\langle vk_{tx}, t\rangle \leftarrow$ NIVSS_Reconstruct($S, \mathcal{A}$)
3:     $n \leftarrow extractAmount(vk_{tx})$
4:     if $\neg VerifyVRF_{pk}(hash, \pi, ctx.seed)$ then
5:         Gossip $Complaint_{pk_c}$, validation data and signed $b_{default}$
6:         break
7:     end if
8:     $p \leftarrow \frac{ctx.\Theta}{ctx.W - ctx.w_p}$
9:     $j \leftarrow 0$
10:    while $\frac{hash}{2^{len(hash)}} \notin [\sum_{k=0}^{j} Pr(k;n,p), \sum_{k=0}^{j+1} Pr(k;n,p))$ do
11:        $j{+}{+}$
12:    if $j \neq j_c \vee \neg(VerifyTx(b_c.tx_{block}))$
           $\vee(ctx.t_{last\_block} > b_c.t_{block})$
           $\vee\neg(VerifyBlock(ctx.seed_{pk}, b_c.\mathcal{H}(block)))$ then
13:        Gossip $Complaint_{pk_c}$, validation data and signed $b_{default}$
14:        break
15:    end if
16:    Gossip($\langle node.pk, Sig_{node.pk}(pk_c, hash, \pi, j_c, b_c, vk_{tx}, t)\rangle$)
```

Verifiers already operate an implicit Reduction procedure through the asynchronous function onReception within the NIVSS_Reconstruct procedure, which checks the shares as well as the block and the hash propagated by the *candidate*. As taught by Turpin and Coan [7] two-step technique, such an implicit reduction of the problem to reaching consensus on a binary choice (either the candidate's pre-block hash or the default pre-block) is important to ensure liveness.

3.7 Voting On Blocks — Voter's Sortition

While nodes compete for generating the block, Provisioners other than the *Verifiers* run the Sortition procedure so to be appointed the task of *Voters* and reach Byzantine consensus over different rounds of voting.

Block election happens through a set of steps, each one requiring a different committee being formed by peers sorted through VRF output with the voter's role. During each step, sorted nodes (and only them) gossip their signed hash of the block together with block round, step in the process and proof of sortition as outputted by the VRF. Since voting is a task reserved to Provisioners, gossip during this phase stays confined within the Provisioners' boundaries and is relayed solely by Voters. As a result, the voting operation completes much faster than all the others since there is no need to wait for messages to reach all extremities of the network.

Provisioners organised in subsequent elected *Voting Committees* try to reach consensus (i.e. counting enough votes for either the

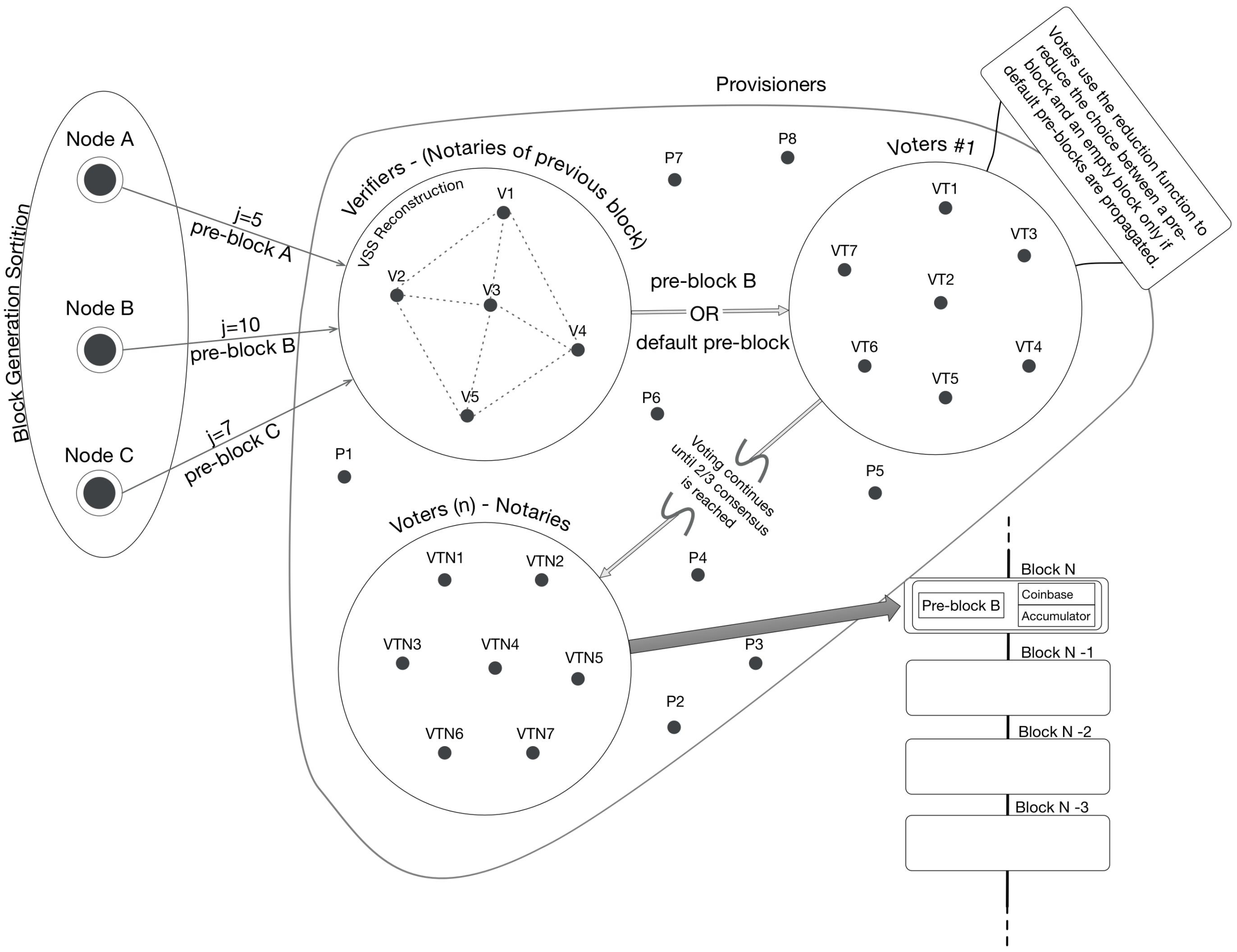

Figure 6: SBA⋆ at a glance

pre-block hash or the empty block) by progressively raising the amount of votes for a pre-block or an empty block.

At each subsequent step, the votes cast during the step before remain accounted for, while the new elected Provisioners cast new votes until the required majority is reached. Intuitively, the convergence is guaranteed by the fact that Voters which have already declared consensus for a pre-block will not vote for any other value in the same round and will keep proposing the same result until consensus converges.

3.8 Notaries — Block Rewards

As soon as the *Validators* reach consensus over a non-empty *pre-block*, they turn into *Notaries* by running a supplementary procedure aimed at generating a new block by hashing the pre-block with a set of **coin-base** transactions. A *coin-base transaction* is basically a transaction with no input which mints new Dusk coins and spend them to the address of the *Block Generator*.

As opposed to Block Generators, Provisioners do not gain their reward by winning the sortition procedure. Rather, at the end of each block an amount of Dusk coins is *coin-based*, dependent on the stake amount committed by the Provisioner to the Accumulator, independently from whether they participated in the Block Committee or not. This amount is thus spent toward the Provisioner's address.

Counterintuitively, the rewards paid are inversely proportional to the staked amount (i.e. bigger stakes get proportionally less rewarded, in respect to smaller stakes). This measure is novel and to the writers' knowledge not a viable option outside of the Dusk Blockchain, where the probability to win the sortition lottery and therefore take an active part to the *SBA⋆* algorithm is not associated with a reward, except the sole payment of the transaction fees.

The motivation is twofold. Together with preventing the *rich get richer* scheme, the intention is to create a counterposition between **power** (intended as the capability to influence block generation

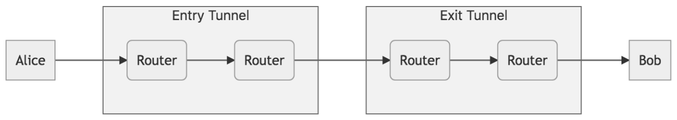

Figure 7: Dusk tunneling

by being selected as part of the Block Committee) and **money** (intended as the financial benefit acquired from running Provisioners). Considering that *SBA*★ is already protected from Sybil attack by making it probabilistically disadvantageous to dilute a stack into several balances, similarly, by reversing the proportion between rewards and stake, the system prevents financially motivated participants to benefit from organizing themselves into few Provisioner pools at the expense of decentralization.

4 DUSK ANONYMOUS NETWORK LAYER

In the vast majority of blockchain implementations, network communication protocols limit themselves to just embracing the privacy standards we have in place today for our daily internet needs: TCP/IP, UDP, SSL for encrypting communication channels to name a few.

While this can be considered acceptable for centralized environments or platforms where user privacy is not the main proposition, the anonymity and privacy requirements established with the Dusk network impose the adoption of technology offering a much higher level of privacy protection.

To achieve this goal, Dusk is enabling full anonymity over its decentralized network by integrating an advanced, custom bi-directional routing, fully compatible with I2P's Garlic-Routing technology for all its networking communications, but extending the underlying protocol not only for the deployment of additional functionality (such as fully anonymous file transfer) but also for allowing the default anonymous gossip protocol which powers the entire Dusk network.

The proposed architecture has been designed to make it computationally infeasible for an eavesdropper to tell apart Dusk related traffic from other network activities. Additionally, it should be very hard for any network node to associate a Tx_{id} with the IP address of the original initiator.

Compared to similar solutions, the Dusk approach offers the following benefits:

1. Makes use of *packet routing*, instead of *circuit routing*. This means transparent load balancing of all networking message across peers, instead of a single tunnel.
2. Multiple packets are joined together in inconspicuous messages, making it exponentially difficult for an attacker to expose network communications.
3. True decentralization: it uses a distributed directory to have an overview of the network, as opposed to relying on a centralized bulletin board.
4. Uni-directional tunnels guarantee that incoming and outgoing traffic is kept decoupled; a measure engineered to enhance transmission unlinkability through data stream separation. (Figure 7)

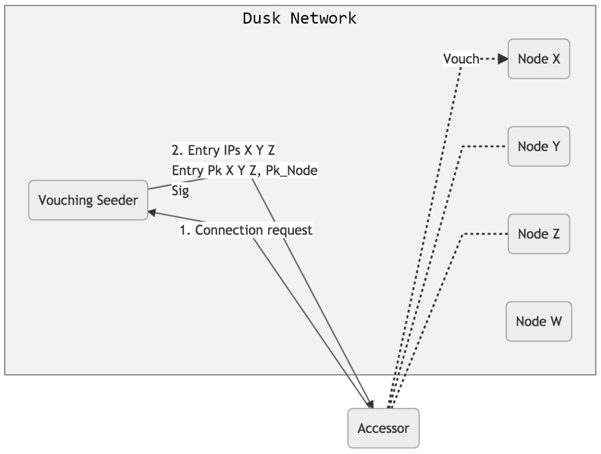

Figure 8: Dusk Network Bootstrap

When negotiating access to the network, the accessor node (Alice) selects *Entry tunnel* to route (*encrypted*) messages through.

Each node in the network acts as a *de facto* router, by relaying the message multiple times, until it gets delivered to an *Exit Tunnel*, for which the last node has been chosen by the message recipient (*Bob*).

4.1 Bootstrap

Prior to forming the *Entry Tunnel*, the accessor connects to a blockchain's *seed server* or *vouching seeder* (inert network nodes specifically designed to facilitate the bootstrap of peers by relaying configuration parameters, the blockchain's current snapshot) in order to obtain a list of active nodes. The *vouching seeder* replies with a message containing three parts:

1. the collection of candidate entry node IPs
2. the collection of candidate entry node public keys and the public key of the accessor node
3. the seeder's signature of 2 and its public key

The part 2. and 3. of the message are called the *vouch*

The accessor node thus selects an arbitrary end point X chosen among those offered by the *vouching seeder*. Thus, the accessor requests *Entry Tunnel* forming with X by sending the *vouch* so that X can verify the *vouching seeder*'s signature and the public key of the accessor, thus preventing potential IP routing leaks. If the verification fails, the node refuses access, marks the node as malicious and puts its IP into a distributed blacklist.

The accessor will then receive a set of active endpoints where connection can be initiated, and will be assigned a hash which will act as a mask for his real IP address. Similarly, the other nodes in the network will be reachable solely through their own hash-mask. The vouching seeders are trusted servers, reachable via a secured *https url* encoded into the Dusk main core.

4.2 Gossip Communication

Once connected to the arbitrarily selected *Entry Node*, *Accessor* will become a full fledged member of the Dusk Network, and will

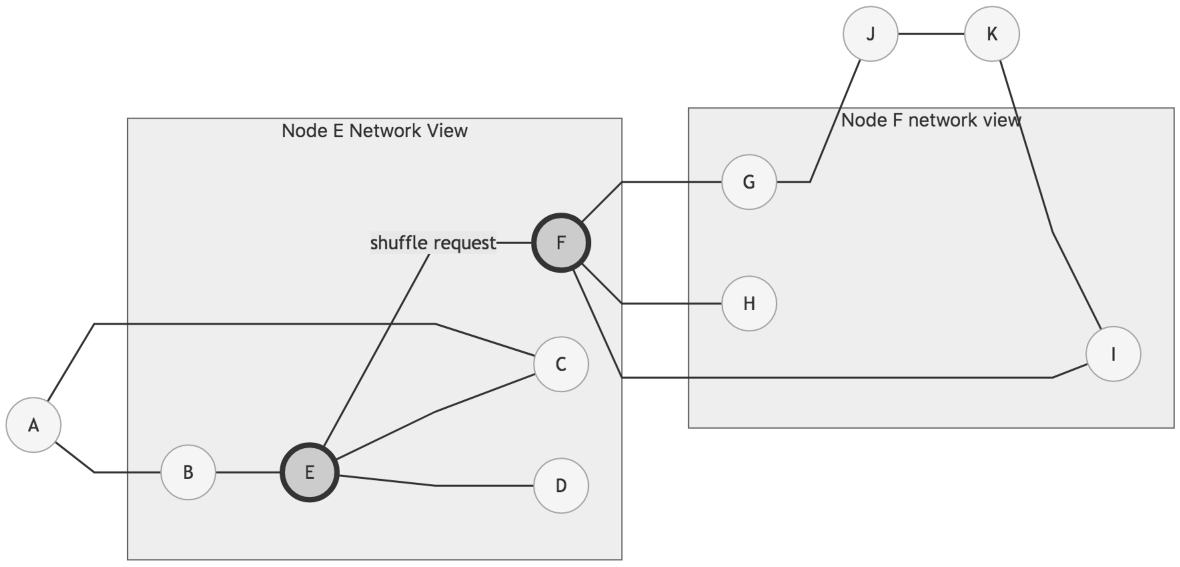

Figure 9: Example gossip setup

receive from the *Vouching Seeder* and the *Entry Node* an initial snapshot of *Dusk addresses* it can gossip to. This internal, partial view of the network is constantly maintained and refreshed by using a *peer-sampling service*, which is itself gossip-based. By using this service, the node will periodically ask other nodes for an updated view of the network, and will receive in return a set of addresses to update its internal view with. For efficient message spreading, it is also imperative that the internal view of each node is as random as possible, and for this reason the *peer-sampling service* is also designed to maintain high entropy within the network by occasionally shuffling nodes between requesting peers.

Let's assume the structure in Figure 9, with nodes E and F being two nodes about to shuffle addresses between each other.

In the scenario above, Node E's internal node tables is $[C, D, B, F]$ - which is the list of node addresses he knows about. Node F's internal node table will be instead $[G, I, H, E]$. When shuffling nodes, the top of the head of each table is exchanged between peers, so Node E will send to Node F the addresses $[C, B]$ and in return will receive $[G, I]$. The new internal tables will modify the network structure as in Figure 10.

This procedure ensures that configurations are never stagnant, and high levels of randomness are always kept within the Network.

4.3 Peer-Status Propagation

Synchronising information about peer's Dusk addresses happens with a gossip-based communication scheme, called Peer-Status Propagation, which is slightly more involved than the one used for transaction propagation (which resembles more a *fire-and-forget* approach). Peer-Status Propagation shows resemblance to a TCP three-way handshake and it is a suitable scheme to synchronise information relating to peer-sampling table, a list of recent Provisioners, new peers joining the network or addresses consistently in timeout. Another possibility (not explored in this paper) is to exchange information about responsiveness of different peers in order to stipulate a constantly updated overview of convenient low-latency routes.

The information is exchanged as follows:

Murmur : The node initiating the communication sends out a message to a receiving peer which contains his current view of the network, plus information of the node itself (uptime, version), and meta-data describing part of its internal status which he wishes to transmit.

Followup : the receiving peer computes a difference between his own meta-data and the one that was sent to it by the initiator. It then follows up on the communication with a reply containing the gossip the initiator ignores and a list of peer addresses the initiator does not know about.

Confirm: After receiving the *Followup* message, the Initiator updates his meta-data with the message it just received by the peer, saves the information, and sends out a *Confirm* message with the missing information the peer did not know about, if necessary. This marks the end of the Status Propagation Round.

The increase of network traffic due to the Peer-Status Propagation is constant and not expected to perceivably impact the efficiency of the network. Relaying the **murmuring** is in fact limited to a restricted number of peers (i.e. three/four nodes) and *Stata* synchronization happens through the constant 2-phases Followup and Confirm messages without causing any network spike.

4.4 Transaction Propagation

The Dusk blockchain makes use of advanced gossip network technology in order to propagate transactions and sortition results for Block Generator/Validators. Gossip protocols follow the *endemic* message dissemination system and represent a natural fit for a P2P network needing to frequently synchronize small status variations among its peers through the following advantages:

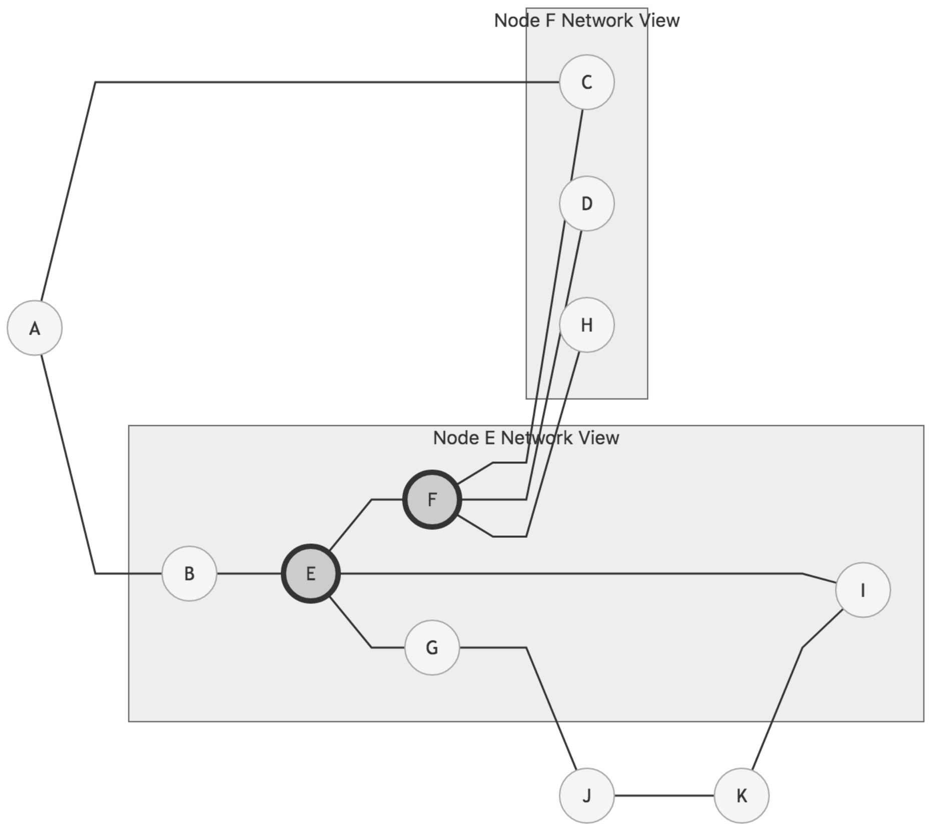

Figure 10: Gossip setup post-shuffle

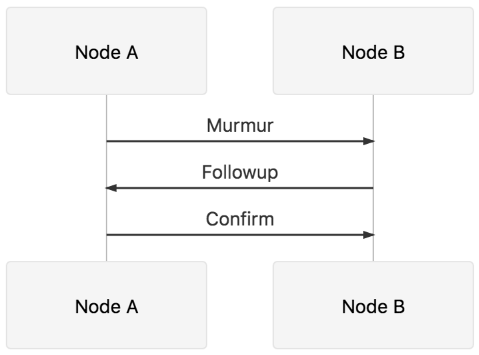

Figure 11: Peer Status Propagation

- **Scalability**: The general complexity of network gossip protocols, is $O(logN)$ which represents the number of rounds to reach all nodes in the network, where N is the total number of nodes. Nodes only send a limited number of messages and do not wait for acknowledgements. Such a system can easily scale although it cannot achieve unbound scalability given the requirement of achieving global dissemination. In the case of the Dusk Blockchain, there is an inherent partition of nodes given by the fact that messages around block proposals, validations and voting is performed solely by Provisioners which relay these messages solely to other Provisioners.
- **Error Tolerant**: Dusk can operate extremely well with unreliable connections and unorthodox configurations. The same packets are sent multiple times to different peers - this way, should an infrastructure problem arise between two points impeding their communication, they will both receive the same message from other nodes in the network.
- **Decentralization Ready**: There is no central role for any of the nodes. Each node works as an independent agent, with pre-established rules about data transmission.

A desirable property the Dusk gossip protocol is obtaining relatively low complexity while guaranteeing safety. For this reason, nodes in the network do not relay more than one message/transaction coming from the same node per $\langle step, round \rangle$ of $SBA\star$.

When a node wants to transmit information (transactions, sortition results, etc.) it selects n random nodes from the set of nodes it knows about (through the *peer sampling service*) and transmits such information to them, who, in turn, relay it further. Information gets periodically sent to N targets, where N is known as the *Fanout*.

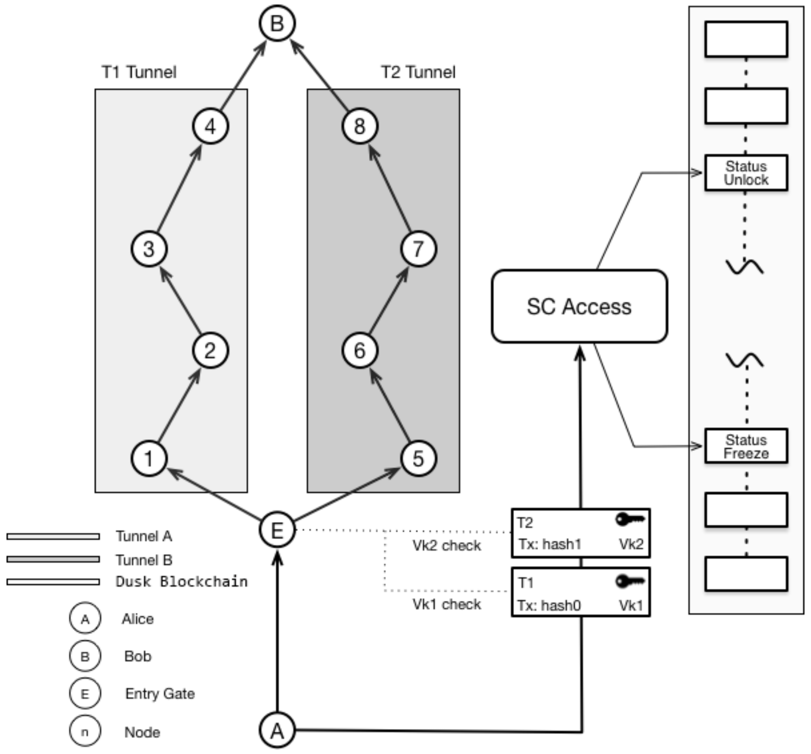

Figure 12: Tunnel Switching

With *Fanout* = 1, we will need $O(logN)$ *Cycles* (which is the number of rouds to spread a rumor) for the information to reach all nodes.

In order to safeguard the anonymity of the peers, most of the messages are relayed through anonymous datagrams following the specification of I2P's Non-Repliable Datagram, which are packets devoid of sender `IP` information using the `UDP` protocol. In the first instance of the Dusk network, we will make use of SAM v3 protocol before considering rolling out a custom-made solution (which might be necessary in case Dusk will need to relay packets bigger than 32Kb).

5 SECURE TUNNEL SWITCHING (STS)

Powered by the Dusk Blockchain, the Dusk Network layer aims to provide a next-generation, cryptographically secure way to perform secure audio, video and data streaming between two distinct nodes on the network. Current technologies limit themselves estaablishing a secure connection between peers - but the Dusk approach we are about to describe goes far beyond that, by making sure that even if a node of the network becomes compromised, the overall security of the platform as a whole remains untouched.

5.1 Setup

We assume that node A (Alice) wants to establish a secure data stream connection with node B (Bob), for which it knows the relevant Dusk address. Before initiating any connection attempt, node A will commit a payment towards a *Smart Contract Access Point (SCAP)* at instant T_1, using an off-chain transaction, for a dynamic value that will be auto-regulated by the Dusk core. This is done to keep the transmission cost stable, and also independent from token fluctuations.

By receiving the transaction request, the *SCAP* will freeze the status on a block on the Dusk chain for node A - to be updated later when the off chain transactions will come to an end.

Upon finalizing the transaction, node A will contact her entry point E, and provide it with the *view key* as proof of payment. In turn, E will check the validity of the key, and if correct will allow the opening of a *garlic routing tunnel* for a duration of time Δ_T towards node B, so that node A can initiate the transmission.

At instant $\Delta_T + \frac{\Delta_T}{2}$, node A will start a new transaction with *SCAS* to renew the duration of such tunnel for an additional interval Δ_T, and if successful will open a new tunnel with B, and will start sending a concurrent, identical data stream on the new one. Node B will consequentially drop the previous tunnel using a procedure that we will describe below. This process will continue until the data transmission will come to an end, by either A choosing to not renovate a tunnel or failing to complete a new transaction with the *SCAP*.

If node A does not provide her Entry point E with a valid view key for the next transaction, the tunnel will be dropped entirely from the Entry point itself.

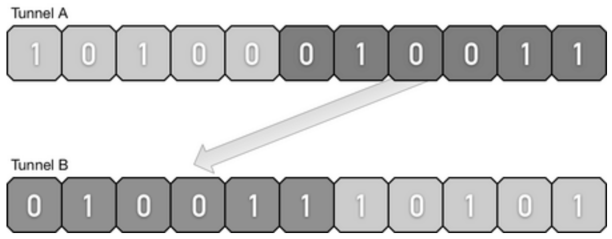

Figure 13: Data Match + Switch

5.2 Tunnel Switching

Once it has received the data stream from sender node *A*, receiver node *B* will parse the raw data (which can be a VoIP call, for example) and will keep waiting for new tunnel connections. Upon receiving a second connection, *B* will match the two data streams by doing a trivial bit matching operation, which will almost certainly show a small time lag due to the fact that the two tunnels use a different set of relaying nodes. (Figure 13)

After successfully matching the streams, node *B* can safely discard the old one, and continue parsing the stream on the most recent one. The procedure will repeat for as long as sender node *A* renews the transaction costs with the *SCAS* to keep streaming data.

This dramatically improves security and anonimity over a conventional *Garlic Tunnel* connection. By switching the data tunnel at regular intervals, a malicious attacker would be unable to predict compromised nodes, perform DDoS attacks, and in general exploit vulnerabilities on the network.

6 ON- OFFLINE FILE TRANSFER

The capability to anonymously and securely send files over the network and allow both offline and online retrieval is a unique feature of the Dusk Network. To implement such a use-case, Dusk combines the capabilities of the anonymous peer discovery and gossip mechanism previously described with those of a third party decentralized and anonymous storage service (e.g. the Orc Object Storage). The workflow to securely send a file over the network is as follows:

- Alice encrypts file.doc using either Bob's P_k (in case the file is of modest size) or using a symmetric scheme such as AES-256 with key A_k to allow for better performances.
- Alice will upload the file to the decentralized storage and will get the `id` of the file once the upload is complete.

Alice will create an Anonymous Non-Repliable Datagram with the following structure:

```
 1: T7CvGQr0/nq8xiLfekdTUGz6rQggGnYYOxuXrMf4vPw=
2: GT7RzkvXJIPjT0xpJXVIhu6QjIzElxKDuvcJKyguwK3HT6GZaRA
9/O4+XCKF67wNeyTfn8RGPM53lp0z+MLW2w==
3: PTlsG6c9kuDyU5vF91SMFAWe55GUi2Pxby+wgb9QYRhvCq5GIUp
dhA==
```

The first line of the datagram is the encrypted `id` of the file on the decentralized storage (using Bob's p_k) and will be used to retrieve such file. Second line is Bob's Dusk Address, also encrypted with using Bob's p_k. Third line is optional and is Alice's symmetric key encrypted with Bob's p_k. This is needed in case symmetric encryption was chosen to encrypt the document. Fourth line is also optional and used in case Alice would like Bob to know she was the sender of the file, in which case she will add her own Dusk address, also encrypted using Bob's p_k.

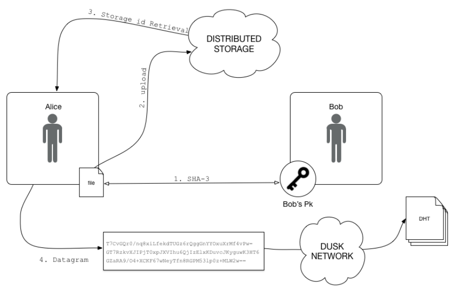
3. Storage id Retrieval
DISTRIBUTED STORAGE
2. upload
Alice
Bob
file
1. SHA-3
Bob's Pk
DHT
4. Datagram
T7CvGQr0/nq8xiLfekdTUGz6rQggGnYYOxuXrMf4vPw-
GT7RzkvXJIPjT0xpJXVIhu6QjIzElxKDuvcJKyguwK3HT6
GZaRA9/O4+XCKF67wNeyTfn8RGPM53lp0z+MLW2w--
DUSK NETWORK

Figure 14: Secure File Transfer

We will now have two separate cases, since Bob can be online when the file is sent, or offline.

- Bob is **online** — Bob receives Alice's datagram, checks if he is the owner of the file by trying to decrypt the second line with his private key s_k. If successful will also decrypt the first line and download the file at the location specified by the ID.
- Bob is **offline** — The entry will be inserted in a distributed hash table (DHT) and kept there for 30 days, which Bob can interrogate as soon as he is online in order to check for files that were addressed to him while he was away. The DHT will be supported by extending the *Provisioner* protocol with a simplified version of the RPC primitives of Kademlia Specifications implemented over Non-Repliable Datagrams.

7 CONCLUSIONS

Dusk Network is an unrestricted, unsurveilled and fully distributed cryptosystem designed for high-rate voice and data communications, enforcing the utmost level of privacy to the partaking peers. The network features a novel blockchain-based digital cash called Dusk, used to directly incentivize participation in the network and promote widespread adoption. Dusk features untraceability through the use of ring confidential transactions, unlinkability through the use of stealth address and protection from Sybil attack and double spending through a novel consensus algorithm called Segregated Byzantine Agreement or SBA⋆. SBA⋆ provides direct block finality by preventing forking while providing virtually unbounded scalability. The network is built on top of an efficient gossip network which utilizes non-repliable datagram and garlic routing in order to prevent IP Address propagation. Finally, the Dusk Network is complemented with an off- online file transfer mechanism and with realtime Dusk payment channel to enable undetectable and fast peer-to-peer data communication through a technique we call Secure Tunnel Switching.

REFERENCES

[1] Jan Camenisch and Anna Lysyanskaya. 2002. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. (Feb. 2002). Retrieved

February 2002 from http://cs.brown.edu/~anna/papers/camlys02.pdf
[2] Harold S.M. Coxeter. 1980. *Generators and Relations for Discrete Groups* (4th. ed.). Springer-Verlag Berlin Heidelberg.
[3] Amitabh Saxena et al. 2016. Application Layer Encryption for Cloud. (Oct. 2016). Retrieved 9 Oct 2016 from https://www.researchgate.net/publication/304549263_Application_Layer_Encryption_for_Cloud
[4] Benedikt Bünz et al. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. (Oct. 2018). Retrieved March 6, 2018 from https://eprint.iacr.org/2018/046.pdf
[5] Eli Ben-Sasson et al. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. (Jan. 2014). http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf
[6] Eli Ben-Sasson et al. 2018. Scalable, transparent, and post-quantum secure computational integrity. (Oct. 2018). Retrieved March 6, 2018 from https://eprint.iacr.org/2018/046.pdf
[7] Russel Turpin et al. 1982. EXTENDING BINARY BYZANTINE AGREEMENT TO MULTIVALUED BYZANTINE AGREEMENT. (Jan. 1982). Retrieved 2014 from https://groups.csail.mit.edu/tds/papers/Coan/TurpinCoan-ipl84.pdf
[8] Sharon Goldberg et al. 2016. NSEC5 from Elliptic Curves. (Jan. 2016). Retrieved March 14, 2016 from https://eprint.iacr.org/2016/083.pdf
[9] Silvio Micali et al. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. (Oct. 2017). Retrieved 28 Oct 2017 from https://web.eecs.umich.edu/~manosk/assets/papers/algorand.pdf
[10] Amos Fiat and Adi Shamir. 1980. *How To Prove Yourself: Practical Solutions to Identification and Signature Problems.* (4th. ed.). Springer-Verlag Berlin Heidelberg.
[11] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. 1998. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. (July 1998). Retrieved 2014 from http://www.eecs.harvard.edu/~cat/cs/tlc/papers/grr.pdf
[12] Michiel Hazewinkel. 2001. CryptoNote v 2.0. (Oct. 2001). Retrieved October 17, 2013 from https://cryptonote.org/whitepaper.pdf
[13] O.A. Ivanova. 1994. Cyclic group. (Jan. 1994). Retrieved April 2018 from https://www.encyclopediaofmath.org/index.php/Cyclic_group
[14] Jeremy Maitin-Shepard and Mehdi Tibouchi. 2016. Elliptic Curve Multiset Hash. (Jan. 2016). Retrieved 25 Jan 2016 from https://arxiv.org/pdf/1601.06502.pdf
[15] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. (Oct. 2013). Retrieved 2013 from http://zerocoin.org/media/pdf/ZerocoinOakland.pdf
[16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. 2017. An Expressive (Zero-Knowledge) Set Accumulator. (Feb. 2017). Retrieved 28 April 2017 from http://legacydirs.umiacs.umd.edu/~zhangyp/papers/accum.pdf