Discreet: An Anonymous Peer-to-Peer Economy

Discreet Foundation
getdiscreet.org

June 4, 2021

Abstract

An all-encompassing solution to implementing scalable, programmable decentralized networks capable of providing full privacy is the panacea of the blockchain paradigm. Many projects have tried and failed at this, either erroring in universality, interoperability, network stability, security, or scalability. Yet, the future of the global economy and industry will inevitably require such a system.

Discreet implements this exact solution. Not only is it capable of scaling to meet the demands of an ever-growing population of consumers, it does so effortlessly. Discreet makes use of state-of-the-art cryptographic algorithms researched for decades in the fields of privacy and consensus to achieve an anonymous peer-to-peer digital value transfer system. Additionally, Discreet specifies a fully generic programmable model for decentralized networks through a novel computational scheme. Services such as decentralized finance (DeFi), online gaming and gambling, decentralized autonomous organizations (DAO), Internet-of-things (IoT), e-voting, supply chain, smart grids, and more are trivial to implement with Discreet. Thus Discreet transcends the paradigm of blockchain with a universal solution which no other project currently achieves.

1 Introduction

Secure methods for digital value transfer systems have been a topic of interest surrounding the cryptographic sphere since the early 1990s. Early implementations of systems described in the late 90s and early 2000s like KARMA [1], Wei Dai's "b-money" [2], and Szabo's bit gold [3] have introduced indispensable concepts regarding the fundamentals of cryptocurrencies. Ever since the advent of the blockchain paradigm, kicked off with the release and development of Bitcoin by "Satoshi Nakamoto" [4], the focus on architecture has largely diminished in favor of specialized research and development to tackle individual issues. However, the result of these efforts is thousands of undifferentiated or obsolete projects with large amounts of value attributed to them. In some cases, the issues these projects were ostensibly created to solve are still largely present; or new exploits and problems were born.

1.1 Scalability

One of the biggest challenges most projects face is with scalability: meeting the demand of an ever-growing online world. Bitcoin clocks in at a block time of around 10 minutes, supporting a maximum of 3.3 to 7 transactions per second (TPS) [5]. Additionally, the cost for validating a transaction can hover between one and three dollars. With the number of transactions above 600 million, and the number of unspent transaction outputs (UTXOs) above 70 million, running a full Bitcoin node has become next to impossible for consumers and enthusiasts. While several projects have tried to solve Bitcoin's scalability problem with increases to block size (thereby increasing the TPS) [6] [7], the increases are still nowhere close to what's needed for a viable global cryptocurrency. Moreover, they ignore larger latent issues with Bitcoin's technology.

Lightning networks (LN) [8] have been discussed as another means of solving the scalability problem with bitcoin, with companies like Blockstream heavily promoting its use. Unfortunately lightning networks have the downside of becoming heavily centralized and insecure. Moreover, the "Watchtower" concept [9] is essentially a kludge in a fundamentally broken technology. Requirements for nodes to be always-online will limit adopters to enthusiasts and corporations, thus "recentralizing" Bitcoin. Claims that the technology is in its infancy and will mature into a full solution to scalability are equally implausible.

Some methods for decreasing the storage requirements for Bitcoin, such as pruning, signature aggregation [10], and MimbleWimble [11] are promising and simple. These would not be difficult for the Bitcoin network to adopt. Unfortunately pruning destroys historical security and signature aggregation can erase authenticity verification, essentially weakening the security of the network. Solutions such as sharding and sidechains try to alleviate network scalability but introduce netsplits, which weaken the overall security of the network once again.

Researchers have consolidated many years of research on the security problems with Bitcoin [12]. Many of these originate from the very mechanisms which try to prevent double-spends, and demonstrate the limitations of the consensus mechanism and heuristic Bitcoin chose: Proof-of-Work and longest chain. Further discussion on consensus can be found in later sections.

But what's most unfortunate about Bitcoin is its failure to provide a decentralized platform for digital value transfer. Bitcoin's reusable Proof-of-Work based on Adam Back's Hashcash [13] transfers the problem of a 51% attack from sheer number of nodes to the ability to compute SHA-256 hashes (i.e. computing power) [12]. However, over 51% of this hashing power is centralized in only three mining pools. The geographic distribution of hashing power reveals that over seventy percent is contained in China, with just three provinces having over 50 percent [14]. The environmental impact of Bitcoin and the economic constraints for feasible mining [15] means further deployment of full nodes by consumers is not just unlikely but decreasing over time.

We focus mostly on the issues with Bitcoin in this section primarily to outline why it is infeasible for it to become a global payment network. While it's proven itself well as a digital reserve asset and for large cross-border transactions throughout its history, it is unlikely to continue being such indefinitely. By documenting the issues present and considering their solutions we work towards providing the motivation for Discreet.

1.2 Privacy

The need for privacy in a digital economy with governmental and corporate surveillance is a quintessential concern for cryptocurrency. As regulation is passed to protect the establishment, the question of legality becomes a fight for essential liberties.

It is important to make a distinction between several concepts prior to discussing the technology, however. *Anonymity* refers to the affordance granted to individuals who wish to remain unidentifiable in a system. Often coupled with this, *confidentiality* refers to the affordance granted to individuals who wish to make it impossible to determine the content, payload, sentiment, or value of their messages to non-participants. Finally, it is necessary to define the third component of privacy: freedom from surveillance, regulation, restriction, and external control. This concept is motivated from an ever-growing push for regulation by government of all things which empower individuals and redistribute control. Legislation proposes disallowance, so-called claims of adoption which hide regulation, or direct restriction of cryptocurrency [16] [17] [18] [19] [20]. It should be immediately evident that decentralized banking is anti-establishment in the eyes of government. This is a must-consider for all crypto projects which seek even the most basic integration with any organization. To grant this final component of privacy, cryptocurrency can either directly challenge the establishment, or integrate and suffer the consequences. This mutual exclusion cannot be overlooked as many projects have done, and will be considered with the technologies proposed.

With that in mind, technologies which afford value transfer systems privacy exist. In the following subsections the specific cryptographic methods for each component will be analyzed, with a focus on their viability and adoption in modern cryptocurrencies.

1.2.1 Anonymity

The most primitive method of remaining anonymous is through utilization of credentials which are difficult to link to oneself. This was the idea behind Bitcoin's promise of anonymity, which allowed individuals to perform transactions without revealing any pertinent information about themselves. All that was necessary for a transaction to be authentic was a cryptographic signature by a private key which could be owned by anyone. However, this is not the case anymore.

It is evident that "know your customer" (KYC) methods, as well as the ability to map out and detect individuals based on the connectivity of their transactions, can be performed on Bitcoin and many other cryptocurrencies. This motivates a further distinction between two forms of anonymity— *identity anonymity* and *unlinkability* [21]. Since the former can be guaranteed simply by not requiring personal information from individuals, the focus lies on guarantees for unlinkability.

The primary cryptographic technology adopted by cryptocurrencies for preserving anonymity is through ring signatures [22]. This is a group signature technique that removes the requirement for group cooperation, and instead members of the group can be chosen non-interactively. Each ring signature is composed of a number of randomly sampled public keys in the network and can obscure

the signer's identity while allowing for full validation that the original signer did authorize the signature. The algorithm requires a public key cryptography scheme and is made non-interactive using the Fiat-Shamir transformation [23]. This algorithm also requires an efficient and effective uniform random number generator, as well as a "ring equation." One problem with this construction is that users will be able to double-spend, since it would be impossible to determine exactly which user authorized a transaction.

The groundwork for solving this double-spend problem begins with linkable ring signatures [24]. In this scheme, verification of the signature is coupled with a second algorithm which checks if two valid signatures and messages are *linked*— that is, if the signatures were created by the same anonymous individual. However, an attacker in this scheme can frame a signer as a double-spender by reusing their signature. To solve this, traceable ring signatures were developed [25] [26], which replace the *linked* check with *trace*. There are several modifications to this scheme [27] which enable smaller transaction sizes, more efficient verification schemes, and higher degrees of anonymity. These extensions are the primary technologies used in privacy-oriented cryptocurrencies to establish anonymity.

1.2.2 Confidentiality

Confidentiality in the context of cryptocurrency refers mostly to hiding the values spent and received in a transaction. The primary scheme is to apply partial homomorphic encryption via Pedersen commitments, then create a zero-knowledge range proof to ensure no inputs are negative or overflow. However, it might be the case where some of the inputs in a transaction do not belong to the sender. This is fixed by ring confidential transactions [27] which introduces a scheme combining the previously-described ring signatures and confidential transactions.

Bulletproofs [28] and their more efficient Bulletproofs+ [29] extend this by reducing the overall computational expense and size for the zero-knowledge proofs needed for this feature. These represent the state-of-the-art in terms of preserving confidentiality. Moreover, they give a general model for providing cryptographic proofs on any arbitrary arithmetic circuit, proving useful in privacy-preserving multi-party computation (MPC). This is very important for the validation and execution paradigm of blockchain, primarily for smart contracts.

There is also the field of fully homomorphic encryption (FHE), which when made efficient and resistant to common attacks, can be applied immediately to smart contract execution. This builds on partial homomorphic encryption primarily by representing the scheme as an arithmetic circuit and bootstrapping itself. However, known methods can produce noise in the ciphertexts, or have potential attacks which compromise security. Unfortunately generic multi-party computation with these setups and their application to smart contracts on blockchain is still young. More research into guaranteeing hardness and resistance to attacks will be needed before adopting any FHE technique to blockchain. This does not mean they are not currently secure nor does it mean they won't be heavily explored.

2 Preliminaries and Definitions

It is necessary to define some terms before presenting our blockchain model. These definitions are consistent within this document and serve as a reference for disambiguation.

2.1 Public Parameters

Let $\mathbb{G}$ be a cyclic group, such that the discrete logarithm problem (DLP) is hard, and define the scalar field of $\mathbb{G}$ as $\mathbb{F}_p$, with prime p. Let G, U, and H be distinct, with unknown DLP relationship, generators of the group $\mathbb{G}$. Define two hash functions: $\mathcal{H}^s : \{0,1\}^* \to \mathbb{F}_p$ and $\mathcal{H}^p : \{0,1\}^* \to \mathbb{G}$, modeled as random oracles and independent of one another. Assume all subscripted or indexed G and H define independent generators of $\mathbb{G}$ and are public. Additional parameters which are public will be explicitly stated to be so, and will be defined consistently. These definitions are included in their respective sections for the sake of context.

2.2 Blockchain Definitions

Discreet's blockchain model allows two kinds of payments. One of these methods will be *transparent transactions*, which define a means of value transfer without anonymity and confidentiality. These will be of immediate use for centralized exchange listings and lightweight transactions. Unfortunately most payment networks and exchanges will require some form of KYC or traceability, and by implementing transparency, Discreet's chances for adoption increase dramatically. Note that this does not at all diminish nor compromise the security and privacy of our blockchain model. The second model, *private transactions*, implement the full privacy standard presented.

The following definitions will explicitly state which transaction model it falls under *only* for transparent transactions. If the transaction type is not specified, assume it applies to private transactions.

To further disambiguate, *Discreet* will refer to the project, network, and technology presented. $DIS or DIS will refer to the coin.

2.2.1 UTXO

UTXOs, or unspent transaction outputs, are the atomic unit of a blockchain's current state. They contain information regarding the transaction which created it, as well as the wallet in which it resides. Most importantly, it contains a Pedersen Commitment encoding how much coin the unspent transaction output holds. In the case of transparent transactions, this value is instead an unsigned integer representing the amount of droplets the UTXO contains.

These have the benefit of encapsulating the state of a blockchain without requiring every historical transaction to be stored. For most nodes, this allows for pruning sufficiently old transactions, thus freeing up a large amount of non-volatile storage space. It is important to note that large-scale pruning can allow for certain vectors for attack and is not encouraged for full nodes. However for lightweight nodes the storage of key images and the UTXOs will be sufficient in validating transactions and maintaining the security of the Discreet network.

2.2.2 Wallets

Wallets will adopt a dual-key stealth address protocol (DKSAP), where users will have a private view key and a private send key (pv, ps) and corresponding public keys (PV, PS). Transparent wallets will only have a private and public key (pt, PT). Discreet adopts the concept of "shielding" and "deshielding" from Zerocash [30]. These define, respectively, transactions from transparent addresses to stealth addresses, and stealth addresses to transparent addresses.

All key pairs will use an elliptic curve for generation and rely on a sufficiently random oracle, i.e. a random number generator, to create these key pairs. There are many valid elliptic curves proposed for blockchain's public key cryptography, with the most common one being SECP256k1. This is the elliptic curve that Bitcoin chose to use, and no known elliptic curve attacks have been published for it.

Wallets will also have a public identifier called an *address*, which will be an alphanumeric representation of a possibly truncated public key. All binary representations of addresses will have a prefix byte representing the version of Discreet the address was created with, along with the public spend and view key, totaling 65 bytes. The text representation of addresses will be the concatenation of the base-52 encoded public spend and view keys, prefixed with a version number and suffixed with the first four bytes of a hash checksum of the address string. For transparent addresses, the RIPEMD-160 hash of the public key is used, prefixed with a version number and also suffixed with a hash checksum.

2.2.3 Transactions

Transactions abstract discrete atomic changes to a blockchain's state. For the case of Bitcoin, these are value transfers; i.e., they are the spending and minting of new Bitcoin denominations. In the case of Ethereum, they represent changes to the many data structures used by the Ethereum Virtual Machine for smart contract states [31]. In the case of most distributed databases, transactions are the fundamental operations performed on the data structure used for storage. These are recorded on some form of ledger; for blockchain, this is a distributed ledger of transactions (DLT).

Discreet models transactions as both value transfers of DIS and as smart contract evaluations. Essentially, there will be multiple classes of transactions which occur on the Discreet network, and for each, the transaction primitive will provide all necessary fields.

Primarily, we adopt Hawk's approach [32] to differentiating between public and private portions of a smart contract. Public contract code can be executed similarly to smart contracts on Ethereum's Virtual Machine (EVM). Private contract code will compile to an arithmetic circuit for evaluation and proof by a node in the Discreet network. The details of contract evaluation will be explained in detail in a later section.

Transactions will have a low fee and scale linearly with transaction size. Additionally, for private contract code, compute nodes which verify the code in a transaction will receive a fee based on the complexity of the proof. The address of the compute node is included in a transaction suffix, and will be included and paid out after the transaction is put into a confirmed block.

The anatomy of a transaction is as follows:

- Transaction class: this is an identifier for the type of transaction.
- Transaction public key: this is used for generating the one-time address destinations of the outputs.
- Transaction code: this is a length-prefixed array of bytes that contains the smart contract bytecode, if one is being published.
- Contract hash: if a contract is being called, the hash identifier of the transaction is put in this field.
- Contract inputs: if a contract is being called, the arguments passed into the contract being called are put in this field.
- Inputs: this is a list of all UTXOs being spent in this transaction. Included in each input is the key image, Pedersen commitment, and any additional information needed for the signature and obscuring the identity of the spender.
- Outputs: this is a list of UTXOs being generated by the transaction. Included with this is the public key used for the recovery of funds by the receiver and the generated Pedersen commitment.
- Signatures: this is a list of all ring signatures and proofs for the authorization of each input spent in the transaction.
- Proof: this is simply the range proof for validating the transaction; i.e. no funds were created or destroyed and no overflows occurred.

Note that depending on the class of transaction, some fields might not be present. Additionally, the most common classes of transactions are transparent and private transactions which do not call or create any smart contracts.

2.2.4 Nodes

Nodes in the Discreet come in four flavors:

1. **Validation Nodes** - these are responsible for validating the bulletproofs, ring signatures, and other proofs included in Discreet transactions. They maintain a list of key images for each transaction and the full set of UTXOs.
2. **Full Nodes** - these maintain a full transactional history of the Discreet network on a blockchain DLT.
3. **Consensus Nodes** - these perform the consensus algorithm used to select the correct future state of the blockchain. They maintain a full history of consensus for validation of each process performed.
4. **Compute Nodes** - these evaluate the expensive smart contract calls in for Discreet. Since the calculations require computationally powerful systems, it is necessary to separate compute nodes from the other three types of nodes. These are fully defined in section 5.

Note that every node is a validation node to some degree, and consensus nodes and compute nodes are full nodes. Only consensus nodes receive block rewards and transaction fees, with computation fees rewarded to compute nodes. However, validation nodes need not store the entire state. This differentiation is necessary when discussing light nodes for mobile and small desktop systems.

2.2.5 Blocks

Blocks aggregate individual transactions into a Merkle tree so consensus can be done efficiently on multiple transactions. Additionally, by aggregating transactions into a single unit, propagation of data and bootstrapping is made simpler and faster. Each block header contains a Merkle root, a timestamp, and information regarding consensus for the block. The body of the block contains a full Merkle tree of transactions. It also has a suffix with information regarding computation fee payouts to compute nodes.

2.2.6 Consensus

While Proof-of-Work consensus methods based on computational power harden a network against double spend, among other things, these become inefficient and are unable to scale. Proof-of-Stake solutions are another possibility. However, staking solutions to consensus are coupled heavily to human interaction, which does not guarantee a trustless model for consensus (rather, one resistant to immediate corruption). There are many other variants of consensus that propose security, many of which fail this guarantee [33] [34].

Discreet requires a topological sort on blocks and consensus on their total ordering at its core. This is performed using an asynchronous Byzantine Fault Tolerant (aBFT) consensus model, Aleph [35]. This significantly extends the base work presented by HoneyBadgerBFT [36] and reduces the asymptotic latency for transaction validation while optimizing the communication complexity. In addition, Aleph provides a trustless setup for randomness beacons, making the consensus model fully decentralized. The requirement for using randomness in consensus is due to the FLP-impossibility [37], which states that consensus cannot be achieved in a finite number of rounds using a deterministic algorithm. These randomness beacons operate an extension of threshold signatures [38] by employing ring signatures for sharing secrets [22]. The details of their implementation can be found in their paper.

Additionally, the protocol logic is decoupled from the consensus network's communication layer. Thus transaction propagation, block minting and distribution, and the consensus protocol itself can occur in parallel. This also requires that the underlying data structure for the blockchain be a "communication history DAG" (ch-DAG), which is made common through the communication layer, and is made consistent through the protocol.

A breakdown of the speed and transaction throughput of this consensus protocol demonstrates it can scale to 100,000 TPS with 20 second confirmation times, but can handle 100-1000 TPS with latency around 5-10 seconds [39].

Block rewards and transaction fees are paid out to the minter of the head block. Computation fees are paid out accordingly to each compute node after transaction confirmation. These encourage adoption of consensus nodes and compute nodes to strengthen the network's security and computational power.

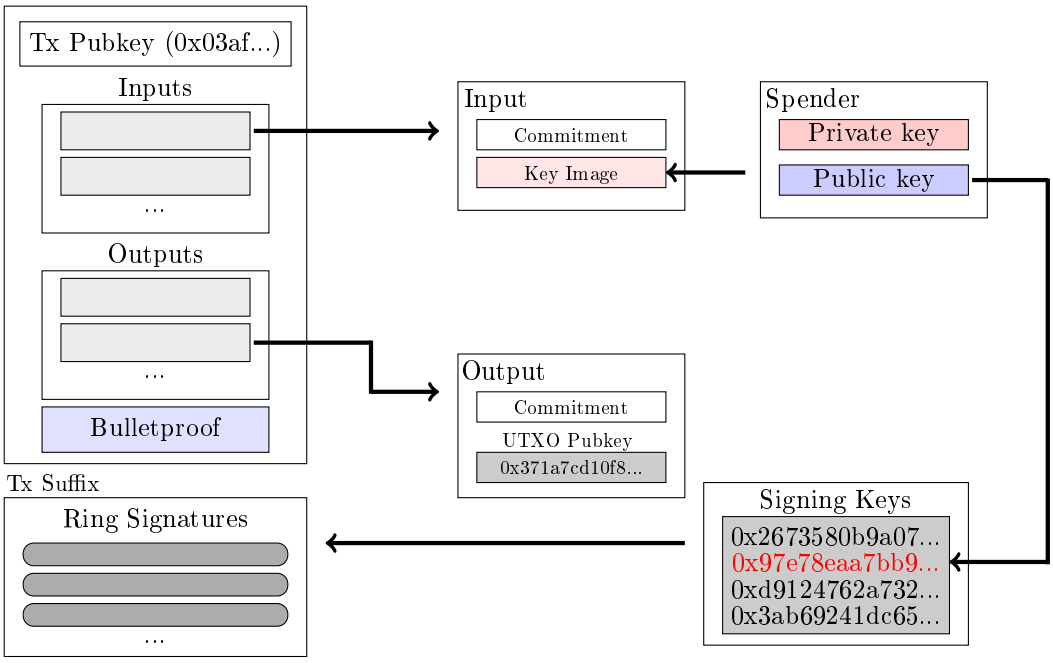

Figure 1: Basic value transfer transaction

3 Transactional Model

For the sake of simplicity, the transaction model described will focus solely on value transfer. Specifics to smart contract deployment and execution will be explained in their own section. Additionally, as said previously, focus will be on defining the private transaction model.

3.1 Security Definitions

The specific elliptic curve that will be implemented for Discreet's elliptic curve cryptography (ECC) is Curve25519. It is computationally efficient for key pair generation, multiplication, and Diffie-Hellman key exchange, while also upholding the 256-bit key length security promise of approximately 128 bits. Point compression will be used to compress public keys.

As for the hashing algorithm, Discreet will make use of 256-bit SHA256 from the SHA2 suite of cryptographic hashing algorithms. This was chosen over SHA3 primarily due to optimization and overhead trade-offs. Discreet hashes will also be double-hashed as is the case with Bitcoin.

3.2 DKSAP

Anonymous transactions will use a dual-key stealth address protocol (DKSAP) similar to Zcash and Monero [30] [40]. Wallets will consist of two keypairs, known as the view keys (k^v_*, K^v_*) and spend keys (k^s_*, K^s_*), typically subscripted with a letter representing the owner of the wallet (Alice will have view keys (k^v_A, K^v_A), etc). For the sake of example, the description of DKSAP will describe

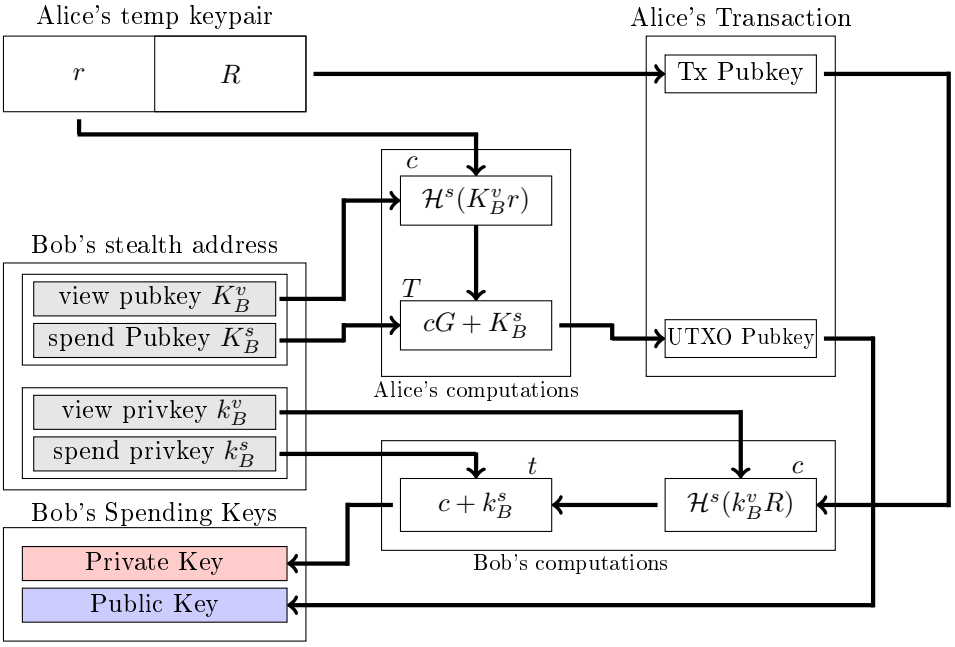

Figure 2: Dual Key stealth address protocol used in private transactions

how a typical interaction would work between two Discreet users, Alice and Bob.

Alice has view keys (k_A^v, K_A^v) and spend keys (k_A^s, K_A^s), while Bob has view keys (k_B^v, K_B^v) and spend keys (k_B^s, K_B^s). Alice and Bob both make their public spend and view keys public to each other through their wallet addresses. For a typical payment from Alice to Bob, the dual key stealth address protocol works as follows:

- Alice generates a security parameter $r \in \mathbb{F}_p$ and calculates using the generator G as a one-way function $R = rG$. This key pair (r, R) is known as the temporary key pair.
- Alice then calculates a shared secret using Diffie-Hellman $c = \mathcal{H}^s(K_B^v r)$.
- Alice can generate a public key using this parameter again with the group generator and Bob's public spend key: $T = cG + K_B^s$.
- Alice then creates a transaction with transaction public key being R and the UTXO for Bob having UTXO public key T.
- Bob can recover the private key for spending this UTXO with his private view and spend keys, first recomputing $c = \mathcal{H}^s(k_B^v R)$, and then $t = c + k_B^s$. With the key pair (t, T), Bob can spend the UTXO. This key pair will be referred to as his spending keys.

Notice that this was made non-interactive through the Fiat-Shamir transformation, and is computationally hard by the intractability of computational Diffie-Hellman (CDH). A visualization of this interation is presented in Figure 2.

3.3 Commitments

Commitments for Discreet transactions will be Pedersen Commitments which have the property of additive homomorphism, i.e. for commitments to values a and b, $C(a+b) = C(a) + C(b)$.

For a given output in a transaction, the Pedersen Commitment (used interchangeably with the word "commitment") is calculated, for transaction security parameter r, public key K_B^v, and amount b_B, using the equations:

$$C(y, b) = yG + bH \tag{1}$$

$$y = \mathcal{H}^s(\lambda_y, \mathcal{H}^p(K_B^v r)) \tag{2}$$

$$b = b_B \oplus_8 \mathcal{H}^s(\lambda_b, \mathcal{H}^p(K_B^v r)) \tag{3}$$

Where λ_y and λ_b correspond to known salts for each hash computation and $\oplus_8$ refers to the 8-byte truncated XOR operation on the first 8 bytes of the hash and the full 8 byte value of b_B. The salts can be a string of characters or random data, but are consistently used for all commitments in the Discreet network. This is primarily adopted from Monero, which produces a Keccak digest with $\lambda_y = "commitment\_mask"$ and $\lambda_b = "amount"$.

Additionally, computations for commitments can also be done on a tensor of values. This will be useful in the following section, which requires the following definition:

$$C(f, r) = rG + \sum_{i,j,k} f_{i,j,k} G_{i,j,k} \tag{4}$$

With $f \equiv (f_{i,j,k}) \subset \mathbb{F}$, blinding factor $r \in \mathbb{F}$, and group generators $(G_{i,j,k}) \subset \mathbb{G}$. Operations on tensors are performed component-wise (i.e. for tensors $f \equiv (f_{i,j,k})$, $g \equiv (g_{i,j,k})$, $f + g \equiv (f_{i,j,k} + g_{i,j,k})$).

3.4 Signatures

The signatures which Discreet will use are sub-linear in size and build on traditional ring signatures. Triptych ring signatures [41], and their extension based on the hardness of the dual-target discrete logarithm problem, Arcturus [42], form the basis of the signature scheme Discreet uses. Discreet will implement Arcturus as the primary signature scheme. The reason for choosing this over RingCT3.0 [43] or another sub-linear ring signature method is motivated by the need for flexibility and speed in verification. Arcturus is more flexible than RingCT3.0 and more efficient, which can be seen in a comparative analysis included in the Arcturus paper [42].

Before presenting Arcturus, it is necessary to define notation used by the original authors. The Kronecker delta function is defined as $\delta(i, j) = 1$ iff $i = j$, otherwise 0. Notation for digit-wise decomposition of a number i in base n and with length m is given by:

$$\sum_{j=0}^{m-1} i_j n^j = i \tag{5}$$

3.4.1 $\sum$-Protocols

A $\sum$-protocol is an interactive challenge-response protocol between a prover and a verifier which collaborate to create and validate a proof for a given relation. These are complete, special sound, and special honest verifier zero-knowledge, and are best defined in [44]. While they are interactive by definition, they can be made non-interactive using the Fiat-Shamir transformation [23]. In this specific instance of the transformation, random verifier challenges are replaced with challenges from a random oracle model, typically by a hash function on the proof transcript.

3.4.2 Definitions

The anonymity set size is given by $N = n^w$, where n represents a base and m a scale. n is chosen to be 2, and for Discreet, m is chosen to be 6 or higher, for a minimum requirement of the anonymity set size $N \geq 64$.

A given transaction will contain w inputs and o outputs, with signing keys $\{t^{(u)}\}_{u=0}^{w-1}$ and anonymity set keys $\{T_i\}_{i=0}^{N-1}$ and their corresponding commitments $\{P_i\}_{i=0}^{N-1}$, with output commitments $\{Q_j\}_{j=0}^{o-1}$. The spender has knowledge of the indices of the w UTXOs they are spending, this set being $\{l^{(u)}\}_{u=0}^{w-1}$. Furthermore, as defined, $t^{(u)}G = M_{l^{(u)}}$. It is also important to decompose the commitments before defining the relation; $\{P_i \equiv s_i G + a_i H\}_{i=0}^{N-1}$ and $\{Q_j \equiv q_j G + b_j H\}_{j=0}^{T-1}$.

One more parameter necessary for the signature scheme are *linking tags*, which are used synonymously with key images, and are defined by the mapping $t^{(u)} \to J^{(u)} \equiv (t^{(u)})^{-1} U$. Similar to key images, linking tags are used to detect double spend. If two key images are the same for a given UTXO, then that transaction is an attempt of double spend, which will be rejected by validator nodes.

The relation defined for the $\sum$-protocol Arcturus uses is:

$$\mathcal{R} \equiv \Bigg\{ \{T_i\}_{i=0}^{N-1}, \{P_i\}_{i=0}^{N-1}, \{J^{(u)}\}_{u=0}^{w-1}, \{Q_j\}_{j=0}^{o-1} \subset \mathbb{G}; (\{l^{(u)}\}_{u=0}^{w-1}, \{t^{(u)}\}_{u=0}^{w-1}, y) : M_{l^{(u)}} = t^{(u)} G \text{ and } t^{(u)} J^{(u)} = U \forall u \in [0, w-1] \text{ and } \sum_{u=0}^{w-1} P_{l^{(u)}} - \sum_{j=0}^{o-1} Q_j = yG \Bigg\}$$

The input parameters for the relation $\mathcal{R}$ are the same as the ones defined previously for the transaction model. The following section will present the prover-verifier interactions which compose the $\sum$-protocol for Arcturus.

3.4.3 Protocol

The prover $\mathcal{P}$ initiates the protocol with the following actions:

- A random value $r_A \in \mathbb{F}$ and random $\{a_{j,i}^{(u)}\}_{i=1,j=0,u=0}^{n-1,m-1,w-1} \subset \mathbb{F}$ are selected. Additionally, the prover sets

$$\{a_{j,0}^{(u)}\}_{j=0,u=0}^{m-1,w-1} = -\sum_{i=1}^{n-1} a_{j,0}^{(u)} \tag{6}$$

- The prover defines the tensor commitment $A \equiv C(a, r_A)$.
- The prover defines $\{\sigma_{j,i}^{(u)}\}_{i=0,j=0,u=0}^{n-1,m-1,w-1} \subset \mathbb{F}$ such that $\sigma_{j,i}^{(u)} \equiv \delta(l_j^{(u)}, i)$, the Kronecker delta between the decomposed index $l^{(u)}$ and index i.
- The prover then chooses random $r_B, r_C, r_D \in \mathbb{F}$ and defines $B \equiv C(\sigma, r_B)$, $C \equiv C(a(1-2\sigma), r_C)$, and $D \equiv C(-a^2, r_D)$.
- The prover defines coefficients $\{p_{k,j}^{(u)}\}_{k=0,j=0}^{N-1,m-1}$ such that:

$$p_k^{(u)}(x) \equiv \prod_{j=0}^{m-1} (\sigma_{j,k}^{(u)} x + a_{j,k}^{(u)}) = \delta(l^{(u)}, k) x^m + \sum_{j=0}^{m-1} p_{k,j}^u x^j \tag{7}$$

  implicitly using the index decomposition. Furthermore, the prover defines $p_{k,j} \equiv \sigma_{u=0}^{w-1} p_{k,j}^{(u)}$ and $p_k(x) \equiv \sigma_{u=0}^{w-1} p_k^{(u)}(x)$.
- The prover now selects random $\{\rho_j^{(u)}\}_{j=0,u=0}^{m-1,w-1} \subset \mathbb{F}$ and $\{\bar{\rho}_j^{(u)}\}_{j=0,u=0}^{m-1,w-1} \subset \mathbb{F}$.
- The prover produces $\mu \equiv \mathcal{H}^s(\{T_i\}, \{P_i\}, \{J^{(u)}\})$.
- Next, the prover defines $\{X_j\}_{j=0}^{m-1} \subset \mathbb{G}$ such that the following holds:

$$X_j \equiv \sum_{k=0}^{N-1} p_{k,j} \mu^k T_k + \sum_{u=0}^{w-1} \rho_j^{(u)} G \tag{8}$$

  And defines $\{Y_j\}_{j=0}^{m-1} \subset \mathbb{G}$ such that the following holds:

$$Y_j \equiv U \sum_{k=0}^{N-1} p_{k,j} \mu^k + \sum_{u=0}^{w-1} \rho_j^{(u)} J^{(u)} \tag{9}$$

  And finally defines $\{Z_j\}_{j=0}^{m-1} \subset \mathbb{G}$ such that the following holds:

$$Z_j \equiv \sum_{k=0}^{N-1} p_{k,j} P^k + \sum_{u=0}^{w-1} \bar{\rho}_j^{(u)} G \tag{10}$$

The prover finishes by sending to the verifier $A, B, C, D, \{X_j\}, \{Y_j\}, \{Z_j\}$. The verifier receives this information and generates a challenge $\xi \subset 0, 1^*$. This is then sent back to the prover, where the following occurs:

- The prover defines $\{f_{j,i}^{(u)}\}_{i=1,j=0,u=0}^{n-1,m-1,w-1}$ such that $f_{j,i}^{(u)} \equiv \sigma_{j,i}^{(u)} \xi + a_{j,i}^{(u)}$.
- The prover also defines $z_A \equiv r_A + \xi r_B$ and $z_C \equiv \xi r_C + r_D$. The prover defines $\{z_R^{(u)}\} \subset \mathbb{F}$ such that:

$$z_R^{(u)} \equiv \mu^{l^{(u)}} t^{(u)} \xi^m - \sum_{j=0}^{m-1} \rho_j^{(u)} \xi^j \tag{11}$$

- Finally, the prover defines:

$$z_S \equiv \xi^m \left(\sum_{u=0}^{w-1} s^{(u)} - \sum_{j=0}^{o-1} q_j \right) - \sum_{j=0}^{m-1} \left(\xi^j \sum_{u=0}^{w-1} \bar{\rho}_j^{(u)} \right) \tag{12}$$

The prover then sends $\{f_{j,i}^{(u)}\}_{j=0,i=1,u=0}^{m-1,n-1,w-1}, z_A, z_C, \{z_R^{(u)}\}, z_S$ to the verifier.
The verifier now performs the final calculations before full verification. First, the verifier sets:

$$f_{j,0}^{(u)} \equiv \xi - \sum_{i=1}^{n-1} f_{j,i}^{(u)} \tag{13}$$

And there are five conditions for the verifier to accept, which are:

$$A + \xi B = C(f, z_A) \tag{14}$$

$$\xi C + D = C(f(\xi - f), z_C) \tag{15}$$

$$\sum_{k=0}^{N-1} \mu^k T_k \Big[\sum_{u=0}^{w-1} \Big(\prod_{j=0}^{m-1} f_{j,k_j}^{(u)} \Big) \Big] - \sum_{j=0}^{m-1} \xi^j X_j - \sum_{u=0}^{w-1} z_R^{(u)} G = 0 \tag{16}$$

$$U \sum_{k=0}^{N-1} \mu^k \Big[\sum_{u=0}^{w-1} \Big(\prod_{j=0}^{m-1} f_{j,k_j}^{(u)} \Big) \Big] - \sum_{j=0}^{m-1} \xi^j Y_j - \sum_{u=0}^{w-1} z_R^{(u)} J^{(u)} = 0 \tag{17}$$

$$\sum_{k=0}^{N-1} P_k \Big[\sum_{u=0}^{w-1} \Big(\prod_{j=0}^{m-1} f_{j,k_j}^{(u)} \Big) \Big] - \sum_{j=0}^{m-1} \xi^j Z_j - \sum_{j=0}^{o-1} Q_j - z_S G = 0 \tag{18}$$

It is encouraged to look at the original specifications of this algorithm for a better overview and for proofs of the $\sum$-protocol, written by Dr. Sarang Noether [42].

For Discreet transactions, proofs will be $(w+3)\lg N + w + 7$ units in size, where each unit is 32 bytes. Additionally, assuming an anonymity set size of 64, proofs will include a compressed representation of the outputs they use in the anonymity set. This technique is adopted from Monero. Unlike Monero, we use 6 bytes to represent the compressed reference to previous outputs, to account for potential collision over time. This means, for an average transaction with two inputs and two outputs, a proof will be 1632 bytes— a reasonable size for a sub-linear ring signature scheme, with the added benefit of significant increase in anonymity.

NOTE: at the time of writing, the hardness assumption for this technology seems not to hold [45]. We will be implementing the sub-linear ring signature technology, Triptych, instead. There are negligible differences in terms of computational efficiency for this algorithm. However, the proof size will be larger as a consequence. We will reproduce the necessary formulae to describe Triptych (which uses the traditional Computational Diffie-Hellman hardness assumption) in a coming update.

While it is unfortunate the Dual-Target Discreet Logarithm Problem does not have a valid hardness assumption, we will continue to review all the privacy technology our technical specification uses. It is important that we produce a working product with the highest level of security.

3.5 Range Proofs

Range proofs provide the necessary tools for ensuring a commitment to a value lies within a valid range. These are necessary to prevent negative value commitments, overflows, and many other ways attackers could arbitrarily create new coins or erase existing coins. The following is the standard relation for a range proof:

$$\Bigg\{(\{G_j\}_{j=0}^{mn-1}, \{H_j\}_{j=0}^{mn-1} \subset \mathbb{G}, G, H \in \mathbb{G}, \mathbf{V} \in \mathbb{G}^m; \mathbf{v}, \gamma \in \mathbb{Z}_p^m) : \tag{19}$$

$$V_j = v_j G + \gamma_j H \wedge v_j \in [0, 2^n - 1] \text{ for } j \in [1, m]\Bigg\} \tag{20}$$

Where m represents the number of commitments to make a statement on; n is typically the number of bits in a value, which for Discreet is 64; $\mathbf{g}, \mathbf{h}$ are distinct and independent generators of $\mathbb{G}$; and $\mathbf{V}$ is the vector of commitments to make the statement on, where $V_j = v_j G + \gamma_j H$, with blinding key γ_j.

The validity of a range proof relies on the ability to calculate what is known as a zero-knowledge weighted inner product. Additionally, the nature of range proofs allows them to be aggregated; indeed, the relation for the range proof Discreet uses, i.e. the one presented above, is an aggregation of range proofs. Specific details as to their soundness and calculation can be found in [29]. This specific style of range proof is known as a *bulletproof*; specifically, it is an extension of them which accelerates verification speed and reduces the number of terms needed to store the proof. Also, much like the ring signature scheme Discreet uses, bulletproofs can be easily batched for faster verification times and higher throughput.

Their true power lies in their ability to make arbitrary statements on arithmetic circuits. This is important for Discreet since it can allow proofs to be generated for private smart contracts. This forms the basis of fully anonymous and confidential smart contracts on the Discreet network. Discreet will make use of bulletproofs and this ability extensively.

4 Consensus

The consensus algorithm Discreet will use is based on Aleph [35], which builds on atomic broadcasts and randomness beacons. Communication history, i.e. the creation and propagation of blocks, is stored in a directed acyclic graph (DAG). What makes Discreet's consensus algorithm conceptually efficient is the complete separation of communication logic (i.e. networking, propagation of blocks, etc) and the protocol logic responsible for achieving consensus on the correct DAG state.

The consensus network assumes the number of nodes at any given time is $N = 3f + 1$, where f is the number of faulty or malicious nodes. Each node has a public identifier pk_i and secret key sk_i for use of signing and verifying messages. Blocks created by consensus nodes will have a number of references called *parents*, which require each new block to have valid links to other blocks created by honest nodes. This forms the basis of the DAG which the consensus mechanism uses.

Specifically, Discreet's DAG will have the following properties. Firstly, for every block minted by an honest node, that block will eventually reach all other honest nodes in the network. Secondly, all honest nodes and their local copies of the DAG will continue to grow unbounded. Finally, for every two honest nodes holding two blocks from the same creator and same round number, these blocks will be equal; i.e., the DAG is *fork-free*. The advantage of having a DAG with these properties is that it will be reliable, and asymptotically resilient to malicious actors.

Discreet's consensus network will ensure that no incorrect blocks are broadcast successfully by making use of reliable broadcast channels. The specifications of which demand all blocks minted by any honest node in the network form a chain with other blocks from that same node. Furthermore, each block must have at least $2f + 1$ parents from the previous round of consensus. Finally, each block must have parents created by pairwise-distinct nodes. Essentially, this requires blocks to be minted only after sufficient advancement and gathering of knowledge from other nodes in the network, while upholding theoretical minima for communication complexity.

A general overview of the consensus algorithm is that, for a given node and its local DAG, it returns a result (which could be a vote or a block) or returns something signifying no result is available yet. For the sake of the consensus protocol, this means each node waits for at least $2f + 1$ honest results from the previous round before making a decision on the next round.

The next problem is how each node chooses the ordering for blocks received. Since this is quintessentially the consensus problem, a specific statement must be made about the general ordering of blocks within all local copies of each local DAG for each node. While a deterministic algorithm can be made for each node on how to order blocks within their DAG for a given round of consensus, by the FLP impossibility, no deterministic protocol will ever achieve consensus. Hence, the decision to choose a *head* block for a consensus round is equivalent to the consensus problem. Choosing a head must further be random, yet be possible for $2f + 1$ nodes to do asynchronously and without a trusted setup. Simply put: common randomness is required for the consensus problem to be solved.

The solution to providing common randomness is the creation of *randomness beacons*, which operate in a distributed manner and without a trusted setup.

Discreet's randomness beacons utilize threshold signatures, removing the requirement full distributed key generation and only dealing to a subset within the consensus network for a given round. This can be done safely since only $2f + 1$ nodes are required to achieve consensus for a given round, under the standard assumption for asynchronous BFT.

This is done, without a trusted dealer, through *key boxes*. Each honest node for a given round constructs one of these and generates their own tossing and verification keys for the threshold signatures required for the randomness beacons. They are constructed by a node k as follows:

- A random polynomial of degree f is constructed:

$$A_k(x) = \sum_{j=0}^{f} a_{k,j} x^j \tag{21}$$

- Then a commitment to the polynomial is computed (this is not a pedersen commitment):

$$C_k = (a_{k,0}, ..., a_{k,f}) \tag{22}$$

- Tossing keys and verification keys (TK_k, VK_k) are given by:

$$tk_{k,i} \equiv A_k(i) \text{ for } i = 1, 2, ..., N \tag{23}$$

$$vk_{k,i} \equiv tk_{k,i} G = \prod_{j=0}^{f} C_{k,j}^{i^j} \tag{24}$$

- The node k now encrypts tossing keys for every node i using their public keys pk_i:

$$e_{k,i} \equiv \text{Enc}(pk_i, tkk, i) \tag{25}$$

  And $E_k \equiv (e_{k,1}, ..., e_{k,N})$.

- The key box for node k is then defined as $KB_k \equiv (C_k, E_k)$. The key set $KS_k \equiv (TK_k, VK_k)$. VK_k can be reconstructed by (24) without leaking the tossing keys for other nodes, and each node can decrypt their tossing key from E_k.

Since it is impossible to determine if all nodes in a given round honestly produced their key sets, each round must also vote on whether or not their given tossing keys are valid. Votes cannot be falsified since an honest dealer for a tossing key will always produce matching verification keys, which relies on the hardness of computational Diffie-Hellman. Thus deceit on an honest node is impossible in this scheme.

A node i votes whether or not it received a valid tossing key from node k if $tk_{k,i}G = vk_{k,i}$, where $tk_{k,i} \equiv \text{Dec}(sk_i, e_{k,i})$. The verification function returns 1 if the relation holds, and returns $\text{Dec}(sk_i, e_{k,i})$, the result of the decryption, otherwise. Note that this is the vote used to determine if a given key box from node k for a specific round, KB_k, is valid. This cannot be falsified since any node can verify if the plaintext given agrees with their ciphertext. Additionally, if a node transmits a falsely positive vote, it cannot properly compute a share of the common secret since it does not have a valid tossing key. This is essentially

stating byzantine fault tolerance for the given protocol.

At a later round each vote is collected by a node i, and a set of keys to trust, $T_i \subset [N]$ can be determined. These must be keys that the node i is familiar with KB_k and that enough votes on KS_k being correct have been seen, such that the node is confident generating secrets from this key set will be successful. Additionally, $\|T_i\| \geq f + 1$, meaning at least one honest node is included in the set. The secrets are aggregated and used to generate a common random secret using a nonce m after choosing a head l:

$$A(x) \equiv \left(\sum_{k \in T_l} A_k(x) \right) \tag{26}$$

$$MC_i \equiv \mathcal{H}^s(\sigma_{m,i}) = \mathcal{H}^s(m^{A(0)}) \tag{27}$$

Minting a block V for round $r + 6$ for a node i corresponds to generating a set T_i as previously defined, with the following conditions: (1) if the block from node k and round r is visible; (2) if the block from round $r + 3$ from node j is visible; and (3) if the key box from the jth node can be verified. All consensus information KB_i, T_i, for the round are included in V.

To generate a secret used for picking the head, first defined is the nonce $m \equiv$ "i‖r", the string concatenation of index i and round number r. Each node produces a share, i.e. the computed common random secret from (27), in their $r + 9$ block, and is not obligated for inclusion if, for a given node k, KB_j was voted to be incorrect or if no vote was collected in round $r + 3$. This generated shared secret is used by each node to pick a head, and thus create more blocks and key sets, ad infinitum. The proofs for this protocol and a more formal definition of each algorithm can be found in [35]. The specific parameters which the consensus mechanism uses and the pseudocode for each algorithm is not included to provide a general overview of the scheme used in the original paper. Additional optimizations against potential attacks and for achieving minimal communication will be researched during implementation of Discreet's consensus.

5 Programming

Smart contracts provide a means of general purpose execution of programs on a blockchain, effectively turning it into a much more general state machine. The idea was originally proposed by Szabo [3], but found widespread use on the Ethereum platform by Vitalik Buterin [31]. Smart contracts provide a means for combining multi-party computation (MPC), financial services, and general-purpose programming. They can take a number of explicit parameters, including information about the blockchain and metainformation on the contract itself, and perform useful computations. For instance, Ethereum's basic scripting environment still allows for blockchain games like CryptoKitties, decentralized autonomous organizations (DAOs), creation of additional token environments, and much more. There is no shortage of examples demonstrating what is possible within the smart contract paradigm. Thus, Discreet would be incomplete without supporting some form of blockchain programming. This essentially is the primary motivation behind Discreet's computational framework.

Ethereum and others took inspiration from viewing databases as finite state machines, where a single call to a smart contract performed mutations on the blockchain's state through transactions which acted as atomic functions [31]. This approach relied on a total ordering of the transactions, which is canonically the consensus problem solved for in blockchains and is therefore preserved. However, consensus communication, the consensus protocol itself, and the computational network are all one and the same. The problem with coupling these actions together lies in scalability— the entire blockchain is limited by the worst bottleneck in any of the network layers. While there are layer-2 solutions for this scalability issue, none fully tackle the entire problem.

5.1 Paranet

Polkadot gives a solution to blockchain scalability with their "Parachain" concept [46]. It is essentially a layer-1 scalability solution which offers blockchain-as-a-service by providing a bare-bones global network rather than a specific data structure. It can be seen as a homogenous "layer zero" solution that can support a number of parallel decentralized databases and protocols, hence the name Parachain. Interestingly, their Parachain technology can be seen as a primitive means to offering a generic for instantiating blockchains. However, there is a way to improve this concept significantly.

Instead of offering a generic for blockchain, what if one could provide a generic for any decentralized network? This is the heart of Discreet's approach to blockchain programming and services. Instead of focusing on layer-1 as a solution, Discreet will provide a layer-0 solution for scalability. This concept is appropriately dubbed **Paranet**, since it specifies a number of parallel networks which operate together to achieve their holistic goal.

The main Discreet network is seen as a central hub performing a total ordering on all transactions. It must store all data necessary for performing computations, which includes contract and user data. Additionally, some form of transcript for each computation, alongside the history of all transactions in the network, must be stored. However, the method for which computation is achieved is decoupled from the main Discreet network altogether.

The Discreet compute network, or DCN, is a Paranet running alongside the

main Discreet network. It consists of a special variant of consensus nodes which operate to achieve consensus on a fair distribution of computational problems. When a call to a contract is received, the network makes use of Aleph's trustless randomness beacons [35] to pick a single node in the DCN to perform the computation. This is proven to be fair since the randomness bits given by Aleph's randomness beacons are sampled from a uniform distribution. Regardless of the type of call (private or transparent), the DCN can ensure proper behavior across nodes and take swift action to mitigate bad actors.

5.2 TACNs

The specific variant of consensus nodes the DCN is built from are known as Trustless Autonomous Compute Nodes, or TACNs. In order to operate a TACN, they must first register their node in the DCN by declaring a static public key as their node's identity and offer a stake of 100 $DIS. This stake can be lost if malicious or byzantine behavior is observed, and will be paid out to the minter of the head block in the DCN's communication history DAG.

Each node in the DCN, after completing a computation, must submit a compute-complete transaction to the main Discreet network containing a hash transcript of the call and a signature. For private smart contracts, a bulletproof of the computation is also included. A TACN is incentivized through transaction fees incurred when computing a call to a contract. Similar to Ethereum's GAS, contracts have a fee based on the number and kind of computations performed during a call. The TACN will receive these fees after their compute-complete transaction is confirmed in the main Discreet network.

The consensus problem TACNs work on solving is two-fold. First, they must achieve consensus on a fair distribution of calls to each TACN in the DCN. Secondly, they must ensure the communication history DAG is consistent among all honest nodes— the exact same as the main Discreet network. These messages can range from a number of things, but primarily consist of verification-related broadcasts.

When a TACN is suspected to be malicious or byzantine, or if any part of verification yields falsehood (e.g. invalid proof, invalid signature, incorrect outputs, timeout, etc), a challenge may be issued to the suspected node by any other node known as the auditor. In reality the auditor is the entire subset of honest nodes in the network, since all honest nodes will detect this error. Depending on the type of error, a penalty is issued to the suspected TACN. For byzantine failures, which can be caused by network dropout and timeout, the TACN is removed from the DCN until such a time when the node can provide proof they are online and active. This change is reflected in the communication history DAG such that all honest nodes are aware of it. This can be guaranteed by the reliability of Aleph in maintaining this local DAG across all honest nodes. For malicious failures, the suspected TACN not only loses their initial stake of 100 $DIS, but is also blacklisted from the DCN permanently. The effectiveness of these punishments relies on the hardness of the consensus protocol, which, as seen in section 4, is strong. After the communication history DAG is updated, the main Discreet network is made aware of the issue and a new TACN is picked to compute the call in the next round.

Note that regardless of the kind of smart contract call received, the DCN can reliably achieve consensus and function alongside the main Discreet network. It

is also important to note that the DCN receives calls to contracts and other communications in the form of ingress communication from the main Discreet network rather than directly from users. This abstraction is necessary since Paranets are not *fully separate* networks but a *heterogeneous system* of networks.

5.3 DVM

When a call to a contract is received by the DCN and a TACN is picked to run the call, the node instantiates what is known as a Discreet Virtual Machine (DVM). This is the low-level abstraction of the protocol used for running smart contracts in the DCN. Each operation performed in a DVM has an associated fee, which depends on the complexity and overhead the operation creates. Additionally, all operations are atomic and may vary greatly in granularity, hence the variation in fees is based on the kind of operation performed.

The DVM is given access to the contract being called and is instantiated with the appropriate inputs given to the TACN when receiving the call. During the computation, a hash transcript is generated which is updated after each atomic operation performed. At the end of the computations, this hash transcript is published in the DCN as a compute-complete transaction, which is then forwarded to the main Discreet network. After this transaction is confirmed, the TACN is paid out the fees for this call.

As for the structure of the DVM, a persistent data structure mimics a heap, which can have any form depending on implementation. This heap is generated from the contract's data field. A DVM comes equipped with an instruction and data stack, the former of which is loaded with the endpoint in the contract called, and the latter of which is loaded with the arguments given. There is also a structure called the transaction stack, which caches all events emitted by the call. If additional calls are generated by the original call, these are performed solely by the original TACN and create their own DVMs as well. All transactions created by additional calls are stored in the main transaction stack.

After the compute-complete transaction is generated, all transactions from the DVM's transaction stack are published to the main Discreet network or the DCN, depending on the type of transaction generated. These will be processed accordingly by each network. Additionally, the new contract state is published in the DCN, which updates the contract's data field with the persistent heap used by the DVM.

It is crucial to take notice that concurrent calls may be made to the same contract, which can lead to race conditions and faulty behavior. There are a number of solutions to this problem, with the most attractive one being a "continuation passing style" approach. In the contract, endpoints which when called may create race conditions will explicitly state this. When a call is made to an endpoint in a contract with this condition, the DVM prior to execution will emit a request for a lock on the contract's data field. This is added to a simple fair queue which relinquishes the lock to the TACN's DVM once available, and ensures that the correct contract state is given as an input. This lock is relinquished by the TACN's DVM after generating the compute-complete transaction.

This is by no means a permanent solution to this problem, and future development into this is guaranteed to take place. However this should be sufficient for Discreet for now. Note that another solution would be to keep locks for

all fields in a contract, and for DVMs to make explicit requests for locks for each data field in the contract needed. However this is easily dismissed since it will increase communication complexity by an unecessary degree. One further solution could be to delegate all contract operations to a specific TACN in the network rather than for each call to a contract. This approach has been explored as a viable solution to this problem but might lead to unfairness in the DCN, since some contracts will be more popular (and therefore generate more revenue for some TACNs through fees) than others. Additionally, calls to external contracts will need to be executed by the appropriate TACNs and may lead to more overhead than necessary. More research will be done into this, but the solution provided currently is sufficient for Discreet and is capable of scaling well.

Finally, it is worthy of clarification to differentiate between communication between DVMs in the form of calls and communication between TACNs. DVM communication is not serialized nor logged and exists ephemerally while a root call is being performed. In contrast, TACN communication is certainly serialized and stored by all honest nodes in the DCN as this information is necessary for the network to function.

5.4 Smart Contracts

Discreet smart contracts will come in two flavors: transparent and private. Transparent smart contracts will operate the same as Ethereum smart contracts. For private smart contracts, all inputs and outputs are kept confidential between the DVM computing the call and the individual who performed the call (either another DVM or a user in the main Discreet network). Additionally, the entire computation is expressed as an arithmetic circuit in such a way that a bulletproof can be generated to prove the correctness of the computation. This proof is included in the compute-complete transaction.

Smart contracts are deployed by a user in a special transaction, with a field in the transaction's data section storing the bytecode for the contract, among other things. Deployment of a contract has a baseline fixed fee and a fee which depends on the size of the contract published, in kilobytes.

When a call to a contract is created, users will specify the maximum amount of fees they are willing to pay. If the fees incurred during execution are greater than this number, the transaction is dropped and the call is not performed. It might be worth exploring penalties for users when not enough fees are specified consistently and repeatedly, although introducing such penalties weakens the computational protocol's byzantine fault tolerance.

5.5 Discript

Discript is the name of the scripting language Discreet will provide users with. It will be designed with simplicity and ease-of-use first and foremost. As more institutions adopt blockchain solutions for technology, payment systems, organizational structuring, financial services, and governance, the audience for blockchain programming is widened. Regardless of the kind of background a developer has, even if they have no prior experience with programming, it should be possible for them to make effective smart contracts as quickly as possible. Principles of development, syntax, and concepts surrounding Discreet smart

contracts must not be lost on any individual. Additionally, code written in Discript will be easy to read and quick to parse. The language must be flexible for both newcomers and experts, and be forgiving in strictness. This is not an easy task, and will take some time to develop; however this is not by any means impossible.

Discript will be a high-level scripting language for general purpose development and blockchain programming. Troubleshooting issues in contract scripts will be swift. Additionally, primary features of the language will reflect the powerful technologies Discreet supports and makes use of. Writing advanced transparent and private contracts will not be out of reach for inexperienced developers, and the syntax will target a wide demographic such that anyone could easily learn how to develop with Discreet. Language features will be modular in nature, thus opening the possibility to decentralized application developers to mod their DVMs to support new features.

The language will initially be pythonic since such languages are easy for scripting and simple enough for most individuals to learn quickly. Concepts from Solidity and other blockchain programming languages will be explored and implemented in Discript, adhering to their specification yet simplifying it enough for any developer to understand how to use it properly and effectively.

5.6 Decentralized Applications

Decentralized Applications, or DApps, differ from smart contracts in that instead of operating *on* the blockchain, they *interact* with it. As such they can be seen as abstract objects which introduce their own network and protocol alongside main Discreet network. However, this is a concept already introduced: these are Paranets. Thus, DApps are understood as Paranets which interact with the main Discreet network, the DCN, and other DApps. They can freely modify the behavior of their consensus protocol, execution environments, resources like storage and processing, and operational logic, while still making use of the main Discreet network and DCN.

While one could view Discreet DApps as their own blockchain, the looser requirement that they implement or instantiate their own consensus protocol allows more generic data structures to be used. This immediately opens the doors for opportunities to create very complex and powerful DApps, ranging from Chainlink-like oracles to blockchain video games to distributed social media to IPFS to decentralized exchanges. Additionally, there is absolutely no requirement these services are built on Discreet, and simple relay networks, much like the relay chains of Polkadot, can wrap external services in such a way that Discreet smart contracts and DApps can use them. It is important to note that these relay networks must also perform their own consensus.

Discreet will provide a flexible and intuitive toolkit for developers to create their own DApps as Discreet Paranets. Services such as decentralized file storage, data oracles, and decentralized exchanges will be trivial to implement and integrate in the Discreet network, thus extending the capabilities of Discreet to cover the entire domain of decentralized technology. This is an important shift in focus, as blockchain will not always be the only model for decentralized networking. However, the problems in creating a trustless, decentralized network are always the same, regardless of the solution provided. This is Discreet's way of future-proofing against obsoletion and maximizing interoperability.

6 Discreet

Discreet will have a total supply of 270 million $DIS. 60 million $DIS will be distributed during an IDO taking place June 5th, 2021. 20 million will be paired with the raised funds on a decentralized exchange. 10 million $DIS will be used for faucets, bounties, payments, and many other things necessary for Discreet community involvement and development. The remaining supply will be block rewards over many years, 5 $DIS per block minted given for each reward, assuming an average block time of 10 seconds. This means the last $DIS mined from a block will be a little over 17 years after network launch, assuming no changes to the rewards. Incentives for running Discreet consensus nodes and TACNs however will continue to exist in the form of fees.

The droplet size for $DIS will be one trillionth. The motivation behind providing such divisibility is simple: in the future, if any fee associated with a certain transaction must be adjusted to reflect the current pricing of $DIS, these can be implemented without creating any fork. Additionally, this maximizes the utilization of our underlying privacy technology. The full network is expected to be completed around the end of Q2 in 2022. Details regarding the ICO will be updated in the future.

Initial transaction fees will be as low as possible, standardized at launch to be one millionth of a coin per kilobyte. Smart contract deployments will be equally minimized on launch of main net. Calls to smart contracts will have both a standard transaction fee and a fee for the number of computations performed. For most operations this is very small, and a full breakdown of pricing will be available when Discript is fully specified.

Discreet will supply both an IDE for Discript, a testground for running smart contracts prior to publishing, and wallet integrations for calling smart contracts. Additionally, setting up and running any kind of node will be as simple as an installation. APIs for an explorer will be provided by the Discreet team, as well as APIs and tools for DApp development. The user interface for the wallet will include all necessary functions for both new and advanced users, and will be universal. The wallet application will rely on a slimmer version of validation node for mobile and offer a variety of versions for Desktop.

In conclusion, we have presented a formal specification and definition of Discreet. Future work may be done with Discreet and will be published only by the Discreet Foundation or official team members.

6.1 Acknowledgements

Many thanks to the developers and proof-readers of this technical paper. In particular, a special thank you to the Monero Research Foundation, Global Decentralized Technology Research Institute, and the teams responsible for all algorithms presented. While the Discreet Foundation is the sole contributor, the monumental amount of research into the technologies necessary for Discreet are decades in the making.

References

[1] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gun Sirer. *KARMA : A Secure Economic Framework for Peer-to-Peer Resource Sharing.* Department of Computer Science, Cornell University, Ithaca, NY 14853. 2005.

[2] Wei Dai. *b-money.* http://www.weidai.com/bmoney.txt. 1998.

[3] Nick Szabo. http://unenumerated.blogspot.com/2005/12/bit-gold.html. "Unenumerated," Jan 1970.

[4] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (2008).

[5] Kyle Croman and Ittay Eyal. *On Scaling Decentralized Blockchains (A Position Paper).* Initiative for CryptoCurrencies and Contracts (IC3). 2016.

[6] *Bitcoin Cash.* https://www.bitcoincash.org. 2017.

[7] Craig Steven Wright. *Bitoin Satoshi's Vision.* Paper not included.

[8] Joseph Poon and Thaddeus Dryja. *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments.* 2016.

[9] Oliver Gugger. *Private Altruist Watchtowers.* https://github.com/lightningnetwork/lnd/blob/master/docs/watchtower.md. 2019.

[10] Gregory Maxwell. *Simple Schnorr Multi-Signatures with Applications to Bitcoin.* https://allquantor.at/blockchainbib/pdf/maxwell2018simple.pdf. 2018.

[11] Tom Elvis Jesudor. *MIMBLEWIMBLE.* https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt. 2016.

[12] Mauro Conti. *A Survey on Security and Privacy Issues of Bitcoin.* https://arxiv.org/pdf/1706.00916.pdf. "GRS – an excellent overview of security problems present in established cryptocurrencies".

[13] Adam Back. *Hashcash - A Denial of Service Counter-Measure.* 2002. URL: http://www.hashcash.org/hashcash.pdf.

[14] *Bitcoin Mining Map.* University of Cambridge. 2019. URL: https://cbeci.org/mining_map.

[15] K.J. O'Dwyer. *Bitcoin Mining and its Energy Footprint.* 25th IET Irish Signals and Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communities Technologies (ISSC 2014/CIICT 2014). 2014.

[16] Paul A. Gosar 116th Congress. *H.R.6154 - Crypto-Currency Act of 2020.* 2020. URL: https://www.congress.gov/bill/116th-congress/house-bill/6154.

[17] *REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on Markets in Crypto-assets, and amending Directive (EU).* 2020. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52020PC0593.

[18] FinCEN. *Requirements for Certain Transactions Involving Convertible Virtual Currency or Digital Assets.* 2021. URL: https://www.govinfo.gov/content/pkg/FR-2021-01-15/pdf/2021-01016.pdf.

[19] *Anti-money laundering (AMLD V) - Directive (EU) 2018/843*. 2018. URL: https://ec.europa.eu/info/law/anti-money-laundering-amld-v-directive-eu-2018-843_en.

[20] *Cryptocurrency and Regulation of Official Digital Currency Bill*. 2021.

[21] Rui Zhang, Rui Xue, and Ling Liu. *Security and Privacy on Blockchain*. 2019. URL: https://arxiv.org/pdf/1903.07602.pdf.

[22] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to Leak a Secret". In: *Advances in Cryptology — ASIACRYPT 2001*. Ed. by Colin Boyd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 552–565.

[23] Fiat A. and Shamir A. *How To Prove Yourself: Practical Solutions to Identification and Signature Problems*. Odlyzko A.M. (eds) Advances in Cryptology — CRYPTO' 86. CRYPTO 1986. 1987. DOI: https://doi.org/10.1007/3-540-47721-7_12.

[24] Joseph K. Liu and Duncan S. Wong. "Linkable Ring Signatures: Security Models and New Schemes". In: *Computational Science and Its Applications – ICCSA 2005*. Ed. by Osvaldo Gervasi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 614–623. ISBN: 978-3-540-32044-9.

[25] Eiichiro Fujisaki and Koutarou Suzuki. *Traceable Ring Signatures*. NTT Information Sharing Platform Laboratories, NTT Corporation. 2006.

[26] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups". In: *Information Security and Privacy*. Springer Berlin Heidelberg, 2004, pp. 325–335.

[27] Shen Noether. *RING CONFIDENTIAL TRANSACTIONS*. 2015.

[28] Jonathan Bootle, Andrew Poelstra, and Greg Maxwell. *Bulletproofs: Short Proofs for Confidential Transactions and More*. 2018.

[29] Heewon Chung, Kyoohyung Han, and Chanyang Ju. *Bulletproofs: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. 2020. URL: https://eprint.iacr.org/2020/735.pdf.

[30] Eli Ben-Sasson and Matthew Green. *Zerocash: Decentralized Anonymous Payments from Bitcoin*. 2014.

[31] Dr. Gavin Wood. *ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER*. 2020.

[32] Andrew Miller. *Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts*. 2015. URL: https://eprint.iacr.org/2015/675.pdf.

[33] Christian Cachin and Marko Vukolic. "Blockchain Consensus Protocols in the Wild". In: *CoRR* abs/1707.01873 (2017). arXiv: 1707.01873. URL: http://arxiv.org/abs/1707.01873.

[34] Ittai Abraham et al. "Revisiting Fast Practical Byzantine Fault Tolerance". In: *CoRR* abs/1712.01367 (2017). arXiv: 1712.01367. URL: http://arxiv.org/abs/1712.01367.

[35] Adam Gągol et al. "Aleph: Efficient Atomic Broadcast in Asynchronous Networks with Byzantine Nodes". In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. AFT '19. Zurich, Switzerland: Association for Computing Machinery, 2019, 214–228. ISBN: 9781450367325. DOI: 10.1145/3318041.3355467. URL: https://doi.org/10.1145/3318041.3355467.

[36] Andrew Miller. *The Honey Badger of BFT Protocols*. 2016. URL: https://eprint.iacr.org/2016/199.pdf.

[37] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". In: *J. ACM* 32.2 (Apr. 1985), 374–382. ISSN: 0004-5411. DOI: 10.1145/3149.214121. URL: https://doi.org/10.1145/3149.214121.

[38] Dan Boneh, Benn Lynn, and Hovav Shacham. *Short Signatures from the Weil Pairing*. 2004.

[39] Damian Straszak and Michal Handzlik. *Aleph Zero: Aleph Protocol Proof-of-Concept*. 2019. URL: https://github.com/alephzerofoundation/Proof-of-Concept.

[40] Nicolas van Saberhagen. *CryptoNote v 2.0*. 2013.

[41] Sarang Noether and Brandon Goodell. *Triptych: logarithmic-sized linkable ring signatures with applications*. Cryptology ePrint Archive, Report 2020/018. https://eprint.iacr.org/2020/018.pdf. 2020.

[42] Sarang Noether. *Arcturus: efficient proofs for confidential transactions*. Cryptology ePrint Archive, Report 2020/312. https://eprint.iacr.org/2020/312. 2020.

[43] Tsz Hon Yuen et al. *RingCT 3.0 for Blockchain Confidential Transaction: Shorter Size and Stronger Security*. 2019. URL: https://eprint.iacr.org/2019/508.pdf.

[44] Jens Groth and Markulf Kohlweiss. *One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin*. 2014. URL: https://eprint.iacr.org/2014/764.pdf.

[45] ukoe@protonmail.com. *Analysis: Dual-Target Discrete Logarithm Assumption*. https://github.com/UkoeHB/break-dual-target-dl. 2021.

[46] Dr. Gavin Wood. *Polkadot: Vision for a Heterogenous Multi-Chain Framework*. URL: https://polkadot.network/PolkaDotPaper.pdf.