Suterusu yellowpaper (V 0.2)

Dr. Lin

Co-founder and CTO of Suterusu project

Abstract. This work introduces the main technical modules of Suterusu project. We provide a detailed description of a confidential payment scheme compatible with the existing account-based smart contract platform such as Ethereum. Our proposed scheme does not require a trusted setup and both its communication and computational overhead are constant. In addition, we present a hybrid PoW/PoS mechanism to serve as Suterusu's consensus protocol.

1 Introduction

This yellowpaper will introduce technical details of the Suterusu project, including the cryptographic modules and consensus protocol we intend to use in our testnet. Our testnet will adopt a similar framework to the substrate library as the bedrock of our custom blockchain. It will implement and integrate our proposed confidential payment scheme for the smart contract platform. The confidential payment scheme will be based on groups of unknown order, more specifically class groups of imaginary quadratic order. Our proposed scheme does not require a trusted setup step and its communication and computational overhead is constant.

2 Overview of Suterusu confidential payment framework for smart contract platforms

Zether was recently proposed as a decentralized confidential payment framework tailored for the account-based model. It can be directly applied to smart contract platforms such as Ethereum. The Suterusu confidential payment framework can be viewed as Zether 2.0 with an improved zero-knowledge proof scheme as the underlying cryptographic module. Therefore, the general Suterusu confidential payment mechanism is close to that of Zether and the following exposition also borrows several notations from the Zether work.

Ethereum aims to build a decentralized financial platform based on smart contract. There are two types of accounts in the Ethereum blockchain network: externally-owned accounts (EOAs) and contract accounts. The ownership of an EOA account is controlled by the account's private key. An EOA can initiate a transaction, the content of which consists of the destination account address, a signature σ, the transferred amount, an optional data field representing inputs to a contract, a gas Limit value, and a gas Price value. The signature σ signs

the transaction. The Ethereum blockchain tracks the state of every account. During transaction processing, σ is verified against the transaction. A more sophisticated transaction such as confidential transfer usually comes with its own zero-knowledge proof. The Ethereum blockchain also needs to verify the respective zero-knowledge proof.

A transaction can either transfer unit token between accounts or trigger the execution of smart contract code. Since all the full nodes in the network is supposed to replicate each transaction and code execution, to avoid the abuse of the blockchain network, all the programmable computation including creating contracts, utilizing account storage and executing operations on the virtual machine requires is subject to gas fees, which can be purchased from the sender's account balance. Therefore, it is always desirable to minimize the amount of gas consumed by a transaction and this is what motivates our work.

The Suterusu confidential payment aims to provide a mechanism to convert any token provided by the smart contract platform, such as ETH to Suter token used in the Suterusu network. It will also allow Suter to be converted back to the token in the smart contract platform. Suterusu blockchain provides a confidential transfer module to anonymize any token that has been converted to Suter.

Suter contracts

The Suter payment framework mainly runs the following three smart contracts to realize the confidential payment for smart contract platforms.

- **Funding**. Anybody can fund an account by specifying the public key y and transferring some ETH. Fund converts ETH into Suter. The ETH gets stored in the smart contract and the Suter are homomorphically added to ys balance. If the account does not exist yet, a new one is created. The transfer operation in this contract is also attached with a range proof ensuring the deposit amount does not exceed the upper limit of Suter token.
- **Burn**. Burn converts Suter back to ETH. It verifies the proof of burn against the statement of burn to ensure that the sender knows the right private key and is claiming the right amount. It also checks a signature on the transaction data and the current value of counter, which prevents replay attacks.
- **Transfer**. This operation enables the confidential transfer of Suter tokens in the Suter blockchain. The proof of transfer makes sure that the ciphertext has the right form and the sender has enough money.

User algorithms

A user can run one of the following algorithms to interact with the smart contract. The output of these algorithms are raw transactions. They will be signed using the public key of the Ethereum account from which they are sent and destined to the Suter smart contract.

- **CreateAddress**$(1^\lambda)(sk, pk)$ provides a way for a user to uniquely identify itself to the smart contract. It takes the security parameter λ as input and outputs a secret key sk and the respective public key pk.

- **CreateFundTx**(pk, amt) adds funds to an account. It takes a public key pk and an amount amt as inputs.
- **CreateTransferTx**(sk_{from}, pk_{to}, amt, st) transfers money from one account to another. It takes a secret key sk_{from}, a destination public key pk_{to}, an amount amt, and the state of the smart contract st at a certain block height h as inputs. It outputs tx_{trans}. (For anonymous transfers, this algorithm would also take a set AnonSet as input, which would contain the public keys of both the senders and receivers. AnonSet would be a part of the output too.)
- **CreateBurnTx**(sk, st) withdraws the entire balance from an account. It takes a secret key sk and a state st as inputs.
- **ReadBalance**(sk, st) reads the balance of an account. It takes a secret key sk and state st as inputs, and outputs an integer b.

More details on the aforementioned contracts and algorithms can be found in the original Zether paper. The main difference between the Suter and Zether framework is the underlying confidential payment scheme, and more specifically the respective zero-knowledge proof scheme, which would be our focus in the following sections.

2.1 Roadmap

We will start with a description of our novel Suterusu confidential payment scheme. The hybrid proof of work/proof of stake (PoW/PoS) mining mechanism will be introduced in the following section. We will conclude with future work at the end of this paper.

3 Suterusu confidential payment scheme

Our basic confidential payment scheme is conceptually close to the Zether [1] confidential transfer scheme, which can be viewed as an adaption of confidential payment in the UTXO model with the underlying commitment scheme in the UTXO model replaced by the Elgamal encryption, and the respective zero-knowledge proof scheme modified accordingly.

To transfer an amount b^* from a public key y to a public key $\bar{y}$, a user first generates the encryption of the balance associated with the public key y, i.e., $(C_L, C_R)=(g^b y^r, g^r)$, where b is the balance. In addition, the user also generates $(C, D)=(g^{b^*} y^r, g^r)$ and $(\bar{C}, \bar{D})=(g^{b'} y^r, g^r)$, where b^* is the deducted amount from the user's balance while b' is the remaining balance and r is the randomness used in the encryption. We use zero-knowledge proof here to prove the correctness of confidential transfer, i.e., the balance encryption is well-formed and all the aforementioned amounts are consistent and belong to the correct range. In the following sections, we will separately introduce the Σ-protocol for these two statements and then briefly introduce how they can be combined together to form the final non-interactive zero-knowledge proof for our confidential payment scheme.

Our cryptographic modules are built upon class groups of imaginary quadratic order and the interested readers are referred to [5, 6] for details on class groups. The public parameters of the following protocols include the description of the class group $\mathcal{G}$ and the generators g, h and a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$, where λ is the security parameter.

3.1 Basic setup-free Σ-protocol for consistent balance encryption

The statement for consistent balance encryption for the case of class group is:

$$\left\{ \begin{pmatrix} C, \bar{C}, C_{L,n}, C_{R,n}, R, y, \bar{y}, \\ (sk, b^*, b', r) \end{pmatrix} : \begin{matrix} C = g^{b^*} \cdot y^r \wedge \bar{C} = g^{b'} \cdot \bar{y}^r \wedge C_{L,n} = g^{b'} \cdot C_{R,n}^{sk} \\ \wedge R = g^r \wedge y = g^{sk} \end{matrix} \right\}$$

Here $C_{L,n} = C_L/C$ and $C_{R,n} = C_R/C$. This statement ensures all these ciphertexts are well-formed in the sense that the sender knows the secret key sk and randomness r used in the encryption and the encrypted balance amounts before and after deduction are consistent. The concrete Σ-protocol for consistent balance encryption can be found in Fig. 1.

Theorem 1. *The proposed Σ-protocol in Fig. 1 is honest-verifier zero-knowledge and computationally sound in the generic group model for groups of unknown order.*

The proof of this theorem can be found in the academic version of this yellowpaper [2].

3.2 Setup-free Σ-protocol with a short transcript for consistent balance encryption

The basic scheme proposed in the previous section has a relatively fast verification due to the fact that all the modular exponentiation operations only involve exponents of length smaller than the security parameter. This is achieved mainly through the usage of the prime challenge ℓ. The tradeoff for fast verification is large communication overhead as demonstrated in Fig. 1. By removing the prime challenge, this section introduces a Σ-protocol with a short transcript as shown in Fig. 2.

Theorem 2. *The proposed Σ-protocol in Fig. 2 is honest-verifier zero-knowledge and computationally sound in the generic group model for groups of unknown order.*

The proof of this theorem can be found in the academic version of this yellowpaper [2].

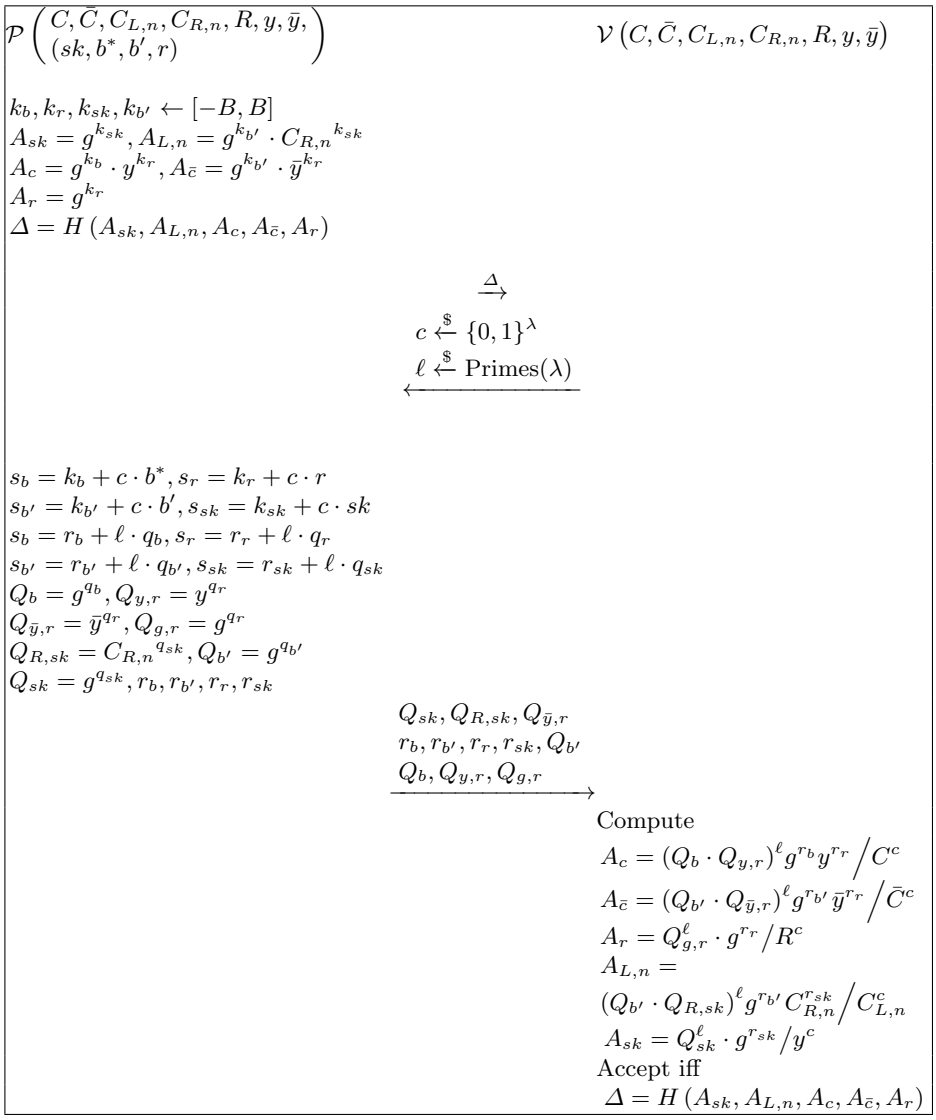

Fig. 1. Basic Σ-protocol for consistent balance encryption

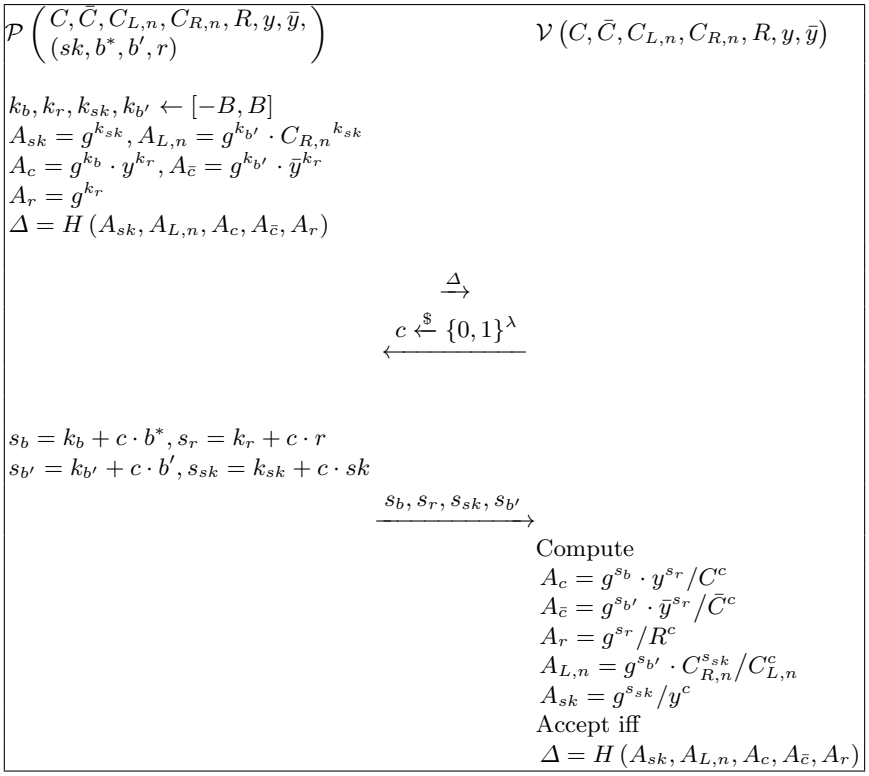

Fig. 2. Σ-protocol with a short transcript for consistent balance encryption

3.3 Basic Σ-protocol for setup-free range proof with free base

Zero-knowledge Range proof is used for ensuring there is no overflow attack in confidential payment. This section provides a zero-knowledge range proof for a secret committed in a Pedersen commitment over Class groups. Our proposed scheme can be viewed as a variant of the scheme introduced in [3] but over class groups and with free base chosen by the prover. The original scheme requires a trusted setup while our proposed construction removes this requirement.

Similar to that of [3, 4], to prove $x \in [a, b]$ the prover needs to prove $4(x - a)(b - x) + 1$ is a sum of three squares. The prover will run a variant of *Rabin-Shallit* algorithm [4] as a subroutine to compute the integer representations of $4(x - a)(b - x) + 1$, i.e., $\{x_i\}_{i=1}^{3}$ satisfying $4(x - a)(b - x) + 1 = \sum_{i=1}^{3} x_i^2$. The protocol presented in Fig. 3 is similar to the one proposed in [3] except the bases of the commitment are chosen freely by the prover. The public parameters of the following protocol include the description of the class group $\mathcal{G}$, two random generators of the group g and h and a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^{2\kappa}$. In the following description, both u and v are two public random bases chosen by the prover $\mathcal{P}$. B is an integer larger than $2^{2\lambda}|\mathcal{G}|$.

The soundness of the original zero-knowledge argument of range proof relies on the five facts as specified in Proposition 1 of [3]. All these five facts can be derived from the generic group model for groups of unknown order. Therefore, one can prove the following theorems on the security of the above protocol:

Theorem 3. *The proposed Σ-protocol in Fig. 3 is honest-verifier zero-knowledge assuming Decisional DH assumption holds in the underlying group.*

Theorem 4. *The proposed Σ-protocol in Fig. 3 is sound in the generic group model for groups of unknown order.*

The proof of these theorems can be found in the academic version of this paper [2].

3.4 Σ-protocol for short range proof with free base

One could also further shrink the range proof size by removing the prime challenge. The improved Σ-protocol can be found in Fig. 4.

one can prove the following theorems on the security of this protocol:

Theorem 5. *The proposed Σ-protocol in Fig. 4 is honest-verifier zero-knowledge assuming Decisional DH assumption holds in the underlying group.*

Theorem 6. *The proposed Σ-protocol in Fig. 4 is sound assuming both Discrete Logarithm problem and RSA problem are hard in the underlying group.*

The proof of these theorems can be found in the academic version of this paper [2].

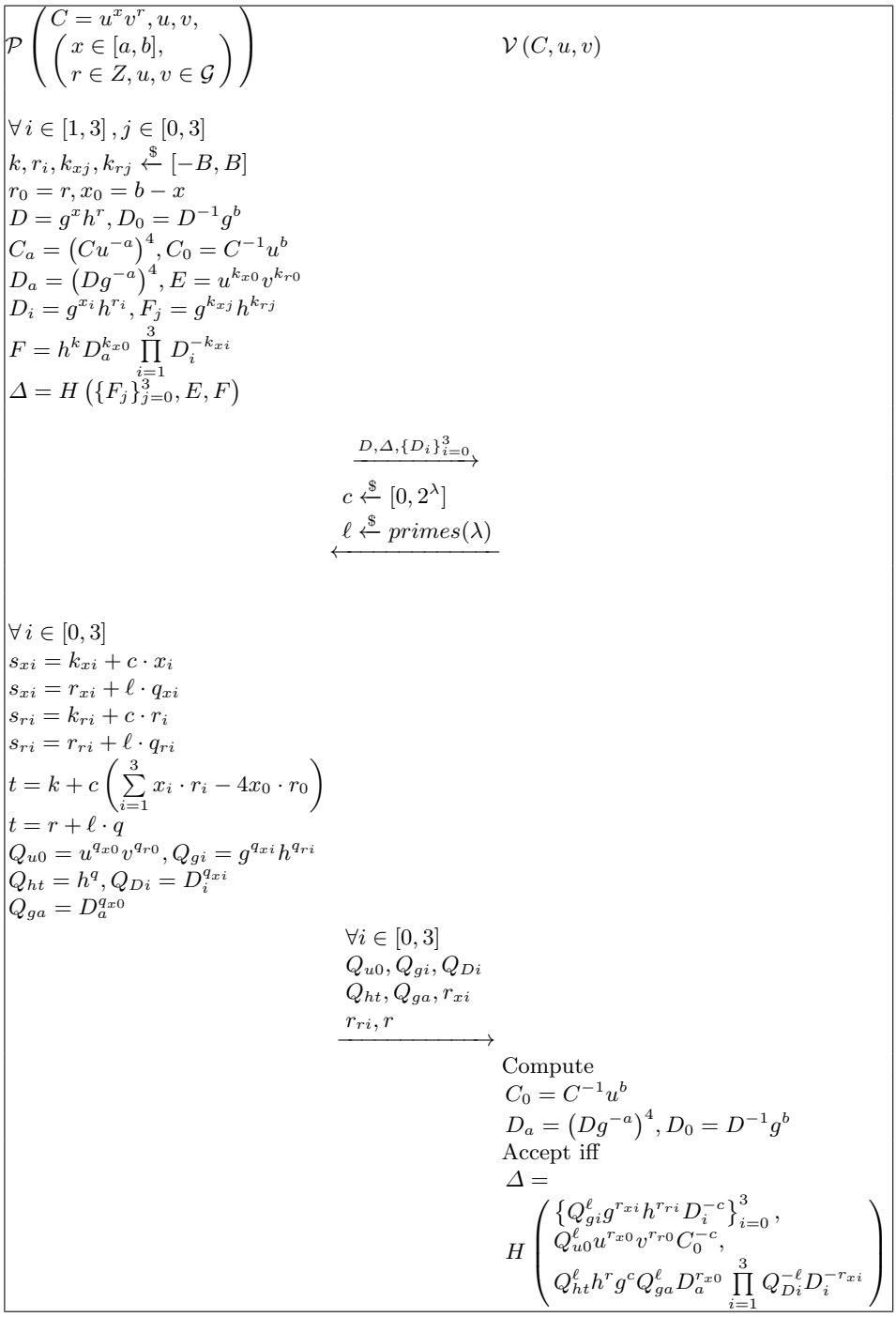

Fig. 3. Basic Σ-protocol for range proof with free base

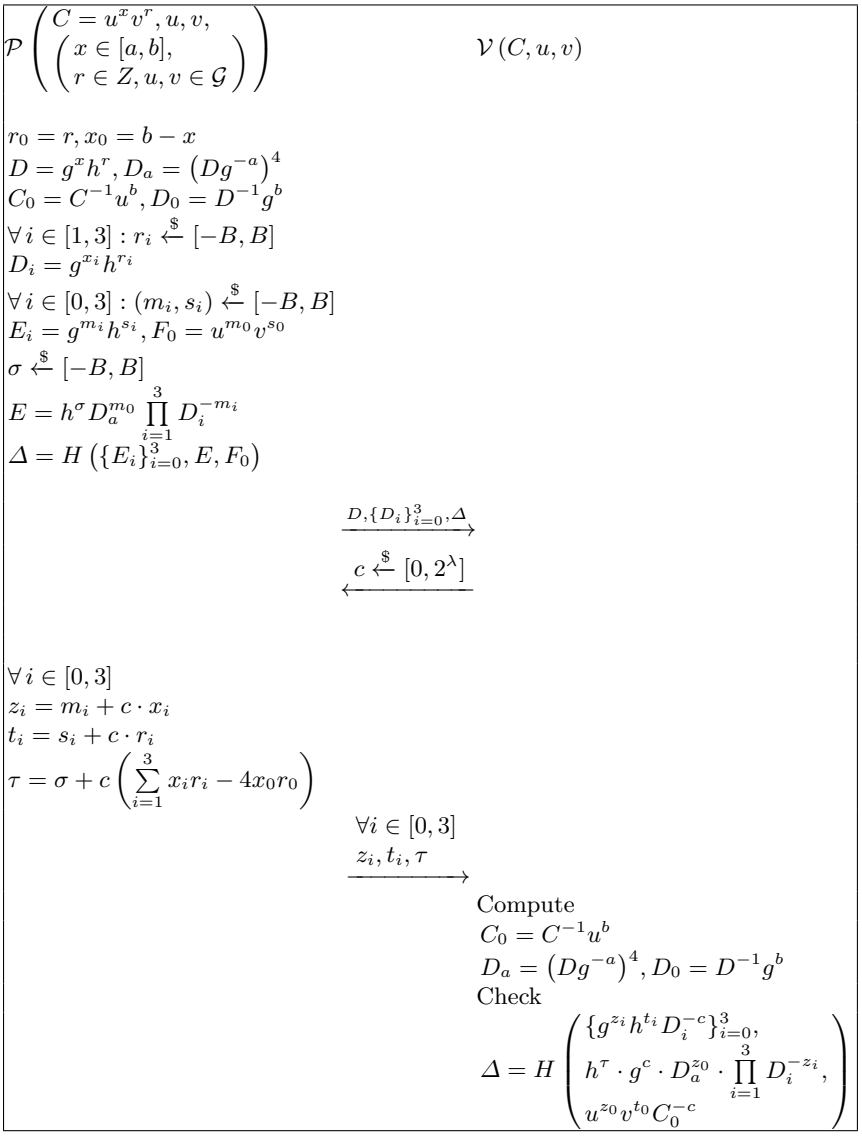

Fig. 4. Σ-protocol for short range proof

3.5 ZK-ConSNARK for confidential transfer in smart contract

The two modules presented in the previous sections can be combined into a single non-interactive zero-knowledge argument scheme with constant communication overhead, i.e., ZK-ConSNARK mentioned in our whitepaper. The respective statement is:

$$\left\{ \left(\begin{array}{c} C, \bar{C}, C_{L,n}, \\ C_{R,n}, R, y, \bar{y}, \\ (sk, b^*, b', r) \end{array} \right) \middle| \begin{array}{l} C = g^{b^*} \cdot y^r \wedge \bar{C} = g^{b^*} \cdot \bar{y}^r \wedge C_{L,n} = g^{b'} \cdot C_{R,n}^{sk} \\ \wedge R = g^r \wedge y = g^{sk} \wedge \\ b^* \in [0, MAX] \wedge b' \in [0, MAX] \end{array} \right\}$$

This statement is essentially a conjunction of the statements proposed in the previous two sections. By applying the Fiat-Shamir transformation, one could easily transform the proposed schemes into their non-interactive versions. The concrete scheme and proof can be found in the academic version of this work.

4 Implementation

Our testnet is a fork of substrate framework, which is a library created by Parity Technologies to facilitate the fast and easy development of a custom blockchain. There are mainly two benefits to base our testnet on a fork of substrate: first, we can develop the smart contract module, proof-of-stake and more sophisticated liquid decentralized meritocracy protocol by modifying technical modules provided by the substrate runtime module library; Secondly, through compiling the code to WebAssembly and deploying it as a message on the network, forkless client updates can be ensured due to the WebAssembly fallback. Other benefits include lightweight client, interoperability etc.

4.1 Discussion

There are mainly two kinds of potential attacks against a confidential payment scheme for the smart contract platforms as noted in the Zether paper [1]: front-running and replay attacks. The proposed protection mechanism in Zether other than the confidential transfer scheme, such as proof of burn, locking accounts to other smart contracts, etc. can be easily transferred to our setting albeit the security of the modified scheme will be reduced to assumptions over groups of unknown order. We will provide a more detailed description of the revised Zether framework over class group in the future version of this yellowpaper.

5 Consensus protocol: hybrid PoW/PoS

Our testnet will employ a hybrid PoW/PoS protocol as the consensus protocol. The validator nodes are required to invest a certain amount of hardware equipment and pay for the maintenance cost to maintain its infrastructure to ensure the security of the Suter network. There are mainly two responsibilities

of validators: 1. Transaction validation and mining; 2. Participating in the Suter network on-chain governance.

More specifically, the validator nodes need to invest in hardware in order to join the network and listen for new blocks. When a new block is proposed, it will validate the block. The block validation of a Suter payment transaction mainly consists of verifying zero-knowledge proof to make sure there is no double-spending attack. The validator nodes are also responsible for assembling new blocks and solving our proof-of-work puzzle to collect the mining reward. Our preliminary choice of PoW hash function will be memory-hard, similar to Ethash used by the Ethereum network. Ethash is ASIC-resistant, which would guarantee the decentralization degree of the Suter network.

In the future, the Suter network will adopt a hybrid PoW/PoS system. A random beacon will be applied to select a voter from the stakeholders. The voter will be responsible for generating a signature to approve the blocks mined by the PoW miners. This provides additional checks and balances mechanism on the PoW miners.

The miners are also required to participate in voting whenever there is a necessity for protocol change. For instance, there will be a transition period from our current PoS governance to the hybrid PoW/PoS mechanism. The initial block rewards will goes to the PoW miners and stakeholders according to the amount of contributed mining power. The Suter community will decide how to adjust this proportion by running our on-chain governing mechanism in the later development stage.

6 Future work

This yellowpaper describes the technical modules used in the Suterusu project, including the confidential payment scheme and hybrid PoW/PoS consensus protocol. Note that the proposed cryptographic modules is tentative and might be subject to change during our future development. We will implement our proposed schemes and compare them to the schemes with logarithmic complexity to choose the one with more practical performance parameters as the underlying cryptographic modules for our final implementation.

References

1. B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world.
2. G. Couteau and H. Lin. Confidential balance transfer for smart contract platform. Cryptology ePrint Archive, Report 2019, 2019.
3. G. Couteau, T. Peters, and D. Pointcheval. Removing the strong rsa assumption from arguments over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 321–350. Springer, 2017.
4. J. Groth. Non-interactive zero-knowledge arguments for voting. In *International Conference on Applied Cryptography and Network Security*, pages 467–482. Springer, 2005.

5. Russell W.F. L. and Giulio M. Subvector commitments with application to succinct arguments, 2019. https://eprint.iacr.org/2018/705.
6. H. Lipmaa. Secure accumulators from euclidean rings without trusted setup. In *International Conference on Applied Cryptography and Network Security*, pages 224–240. Springer, 2012.