Received: 30 October 2018 | Revised: 22 March 2019 | Accepted: 13 April 2019
DOI: 10.1002/cpe.5333

WILEY

SPECIAL ISSUE PAPER

BitFlow: Enabling real-time cash-flow evaluations through blockchain

Lasse Herskind[1] | Alberto Giaretta[2] | Michele De Donno[1] | Nicola Dragoni[1,2]

[1]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
[2]Centre for Applied Autonomous Sensor Systems, Örebro University, Örebro, Sweden

Correspondence
Nicola Dragoni, DTU Compute, Technical University of Denmark, 2800 Kongens Lyngby, Denmark; or Centre for Applied Autonomous Sensor Systems, Örebro University, 702 81 Örebro, Sweden.
Email: ndra@dtu.dk

Summary

Disbursement registration has always been a cumbersome, opaque, and inefficient process, up to the point that most businesses perform cash-flow evaluations only on a quarterly basis. We believe that automatic cash-flow evaluations can actively mitigate these issues. In this paper, we present BitFlow, a blockchain-based architecture that provides complete cash-flow transparency and diminishes the probability of undetected frauds through the BitKrone, a non-volatile cryptocurrency that maps to the Danish Krone (DKK). We show that confidentiality can be effectively achieved on a permissionless blockchain using Zero-Knowledge proofs, ensuring verifiable transfers and automatic evaluations. Furthermore, we discuss several experiments to evaluate our proposal, in particular, the impact that confidential transactions have on the whole system, in terms of responsiveness and from an economical expenditure perspective.

KEYWORDS

BitFlow, blockchain, bulletproofs, cash-flow, Ethereum, zero-knowledge

1 | INTRODUCTION

Companies invest significant resources in estimating the value allocation and cash-flow thoroughly. Still, the spending is often assessed only quarterly, as the process is cumbersome and inefficient using the available tools of today. On account of this, it can be challenging for companies to assess if their cash-flow is as expected, and harmonizes with reality.

In the area of business, this status quo provides an opaque picture of whether the companies are having a positive flow of funds or if they are unable to pay the bills. Furthermore, this opaque nature makes it difficult for philanthropic organizations to sufficiently prove to their donors that the expenditure is following the budget and meeting the expectations.

Blockchain technology addresses these issues as it provides the essential mechanisms to streamline the process of cash-flow evaluations. Moreover, by combining value and registration, blockchain technology can ensure greater clearness than traditional systems.

However, the use of public verification mechanisms, as currently found in smart contracts capable blockchains (eg, Ethereum), poses an obstacle in terms of broader adoption, since confidential transactions are not possible. For example, Company A cannot properly conceal the amount it transfers to Company B, enabling Company C to deduce eventual confidential (thus, critical) alliances between the two parties, as well as favourable discounted prices. Needless to say, this leakage of information is highly undesirable for businesses.

Conversely, this level of transparency would be very welcomed for charities, as it would enable the honest ones to prove their supporters that their donations are spent properly. Consequently, honest charities would attract more sponsors than the shady ones. The same applies to governments, since the capability of holding the government accountable for squandering public money, or other similar misconducts, fuels citizens' trust.

Current blockchain-based solutions transfer value in the form of tokens not directly bound to a real-world value and this produces extreme volatility. Therefore, using existing cryptocurrencies (eg, Bitcoin[1] and Ethereum[2]) would introduce undesired high fluctuations into company finances. Until the volatility issue is not addressed, the use of existent cryptocurrencies-based solutions to monitor cash-flow cannot be considered.

1.1 | Contribution of this paper

In this paper, we present BitFlow, an ecosystem based on the Ethereum-blockchain[2] that enables automatic (and almost real-time) evaluation of liquidity,[*] budgeting, and cash-flow. BitFlow permits to model companies through smart-contracts, allowing automatic linking of employees cash-flow with their department.

BitFlow builds upon the idea of utilizing stable-coins and treasuries to remove the separation between financial operations and subsequent listing. By combining value and registration, we achieve transparency on expenditure within companies. In particular, to overcome the volatility problem and to provide companies with a powerful evaluation tool, BitFlow is based on the concept of BitKrone (BKK), a token directly linked to the Danish Krone (DKK). Since complete transparency might not be beneficial for every company, we deem necessary to support confidential transfers to achieve widespread adoption of the system. Accordingly, we propose a confidential version of the BKK, namely, Private Krone (PKK). PKK can be used, for instance, to hide salaries and alike from the public, while still allowing public verification of the transfers. We propose two different approaches to achieve publicly-verifiable confidential transfers, through zk-SNARKS (ZoKrates)[3-5] and Bulletproofs,[6] and we show the respective performance results.

The BitFlow proof-of-concept has been developed for showcasing the use of blockchain technologies to the Danish Ministry of Development Cooperation, and its source code is available on GitHub.[7] It is indeed believed that charities and humanitarian aids would greatly benefit from the use of systems like BitFlow. For instance, in Denmark, almost half of non-supporters do not support charities because of a lack of transparency.[8] With BitFlow, not only an organization would be able to fight corruption more efficiently than now, but it would also be able to prove that the money is spent correctly.

1.2 | Outline of this paper

This paper is structured as follows. Section 2 introduces fundamental concepts. Section 3 and Section 4 describe BitFlow main idea and implementation, respectively. Section 5 focuses on Private Krone (PKK), discussing how confidential transfers can be achieved while maintaining the public verifiability. Section 6 evaluates and discusses BitFlow performance. Section 7 and Section 8 discuss competing solutions and future developments, respectively. Finally, Section 9 recaps the manuscript contribution.

2 | PRELIMINARIES

In this section, we introduce some fundamental concepts to understand BitFlow.

Token.	A digital object stored on the blockchain. It can be purely digital or a placeholder for a physical object, and it can be *fungible* or *non-fungible*. Like regular currencies, fungible tokens are freely interchangeable, if their face value is equivalent. Non-fungible tokens are unique and distinct from another. In BitFlow, tokens are fungible.
Smart Contract.	A segment of immutable code deployed on the blockchain. Smart contracts can be executed only as implemented by their creator, without any modification. This immutability ensures the execution of predefined actions, only when specific properties are met. As an example, Ethereum supports such contract to be modelled by utilizing, a Turing-complete language.[2]
Oracle.	An external source of information, which allows data to flow from the outside world into the blockchain.[9] Oracles, essential to trigger smart contracts and provide critical information for their functions, can be software inputs given from a website (eg, through an API) or hardware inputs from a physical sensor.

3 | BITFLOW: THE CONCEPT

The idea behind BitFlow is to transfer money through a cryptocurrency, simultaneously and immutably registering such transfer on a blockchain. In this way, we facilitate clarity, as expenses are publicly audible.

As already mentioned, since volatility is undesirable within company funds, BitFlow uses a stable coin, the BKK, which maps 1:1 to the DKK. Locking the BKK at a fixed value protects it from any fluctuation concerning the real economy, ensuring reliable accounting. Moreover, as there will always be as many DKK in the treasury as BKK in circulation, users will be able to swap back to DKK at any time. This availability guarantees the ability to trade with clients outside BitFlow ecosystem seamlessly.

From a high-level perspective, BitFlow exposes the following functionalities:

- ease to exchange DKK to BKK and PKK, and vice versa;
- users can transfer tokens to other users;

[*] Only funds, not the liquidity of goods within the company.

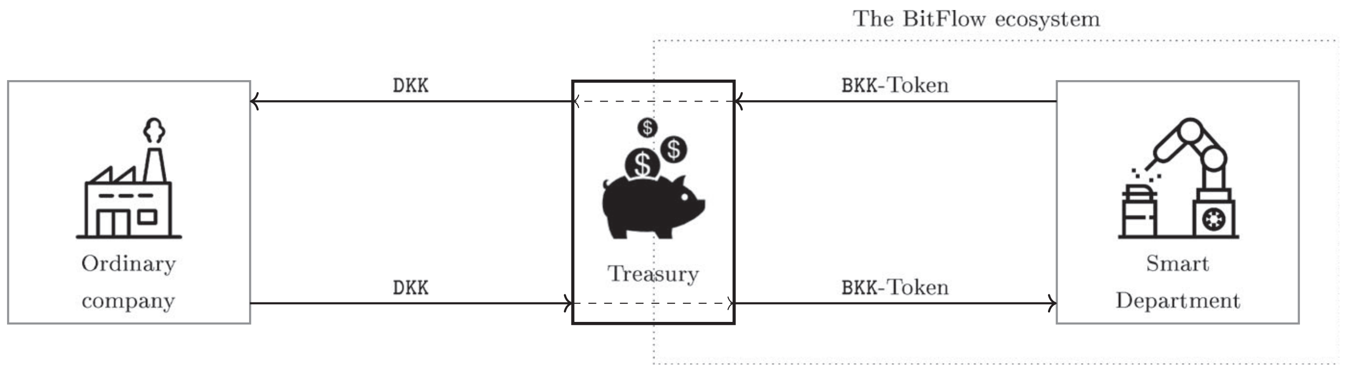


FIGURE 1 The BitFlow architecture utilizes blockchain tokens (eg, the BKK) together with smart departments to enable automatic cash-flow evaluation. Through the use of the treasury, it enforces seamless interactions with peers outside the ecosystem by facilitating exchanges between blockchain value (eg, the BKK) and real-world value (eg, the DKK)

- supervisors of departments can manage employees and sub-departments;
- employees can manage department budget and transfer funds;
- public evaluation of budgets, spending, and liquidity.

There is a minimum set of data associated with every transfer: *sender*, *recipient*, *amount*, and *time*. This clearness makes it manageable for anyone to assess expenditures and to hold employees accountable for any fraud or misuse of funds. Furthermore, this data could be used by employees to perform fine-grained adaptive budgets, up to single-day precision. Thanks to the blockchain, budget changes and transfer of funds happen in a totally transparent, and nearly real-time, way.

In the rest of this section, we present the three critical elements of BitFlow, depicted in Figure 1, ie, tokens, treasuries, and smart departments.

3.1 | Tokens

As previously stated, BitFlow uses BKK, a fungible token that builds on the ERC-20[†] Token Standard, and PKK, a fungible token that builds upon BKK but does not enforce the complete ERC-20 standard. Given that BitFlow requires to change DKK in a perfectly matching amount of BKK, the latter has to meet a BKK precision of 100th of a coin, in order to achieve a DKK two-decimals precision.

In addition to the actions enforced by ERC-20, we must allow minting[‡] as needed, just as a central bank can create new coins (such as DKK). In order to avoid volatility, mining is not desirable or allowed. Only the owner of the coin can mint, similar to how central banks control the minting of new coins.

3.2 | Treasuries

A treasury is a blockchain entity modelled as a smart contract and connected to an oracle, which links DKK transfers to the blockchain itself. Through this setup, once a certain amount of DKK is transferred to the oracle, the treasury automatically transfers the equal amount of BKK/PKK to the specified recipient.

Since treasuries act as a real-time currency exchange (between fiat currencies and crypto ones), we must allow them to mint BKK/PKK. Given that someone has to exchange DKK to receive BKK/PKK, the condition *amount*(DKK) ≥ *amount*(BKK)[§] will always be satisfied.

Since the treasury-oracle will be storing real-world value (DKK), it cannot act fully autonomously, it must have an owner. Furthermore, since we have to entrust this oracle with our funds in order to use this ecosystem, we have to trust the owner and controller of the oracle.

There are two possible types of owners, namely, *Private* and *Consortium*.

Private.	A user can decide to run his own BitFlow currency and implement his own internal treasury, which entails that the owner will have unrestricted power over the oracle and the exchange rate used to represent currencies. The downside of this choice is that, since there is no third trusted party to vouch for the exchange rates, a privately owned system cannot be used for taxation purposes. As an example, who ensures that 1 DKK corresponds to 1 BKK? A private owner might decide to represent 1 DKK with 10 BKK.
Consortium.	If the owners act in a consortium of organizations, no sole actor will have unrestricted power. Therefore, funds and control will be less centralized, while providing the same possibilities as above. Furthermore, if the consortium has members from

[†] https://theethereum.wiki/w/index.php/ERC20_Token_Standard
[‡] Minting is the act of creating new tokens.
[§] Amount(<coin>) is the circulating amount of tokens of the given coin.

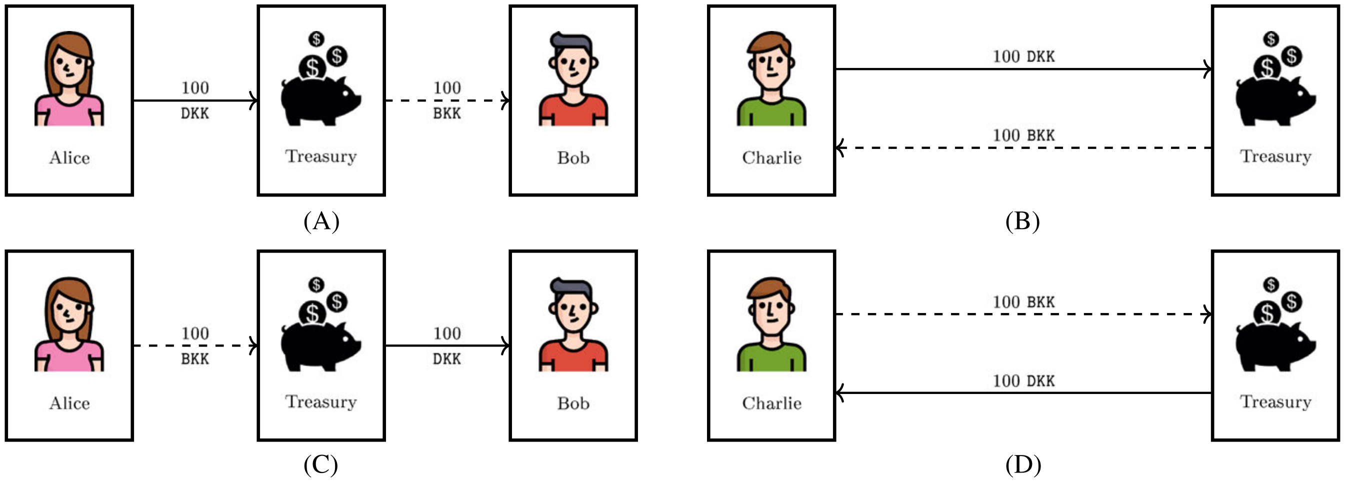


FIGURE 2 Treasury usage: set of actions available for BitFlow users. Dashed lines denote transactions of BKK, while solid lines refer to DKK. Icons from Flaticon. A, Alice exchanges and transfers 100 DKK to Bob; B, Charlie exchanges his DKK to BKK; C, Alice exchanges and transfers 100 BKK to Bob ; D, Charlie exchanges his BKK to DKK

governmental institutions (eg, the department of tax regulation), the cash-flow evaluation could be an enabler for automatic tax filing and fraud detection.

It is important to note that, regardless of what choice is made, the treasury might (or might not) suffer from centralization problems. A treasury could be implemented on a single machine, which would introduce a single point of failure in the whole structure, or it might be distributed. For sure, small businesses are more likely to make the wrong choice of implementing a treasury on a single machine, but even ill-advised consortiums might fall in the same trap. Therefore, the centralization problem is strictly related to the technical implementation choice made by the owner, and it does not depend on the nature of the treasury itself.

That being said, we suggest that there should only be one treasury for the pair, DKK/BKK, controlled by a consortium of banks, auditing firms and the government. Such consortium is to be preferred over single controllers since the distribution of power across competitors should lead to a zero-sum game and thereby an honest oracle. Furthermore, letting the banks act as DKK-holder would be an obvious choice in Denmark because of the high trust in society. In the case of a stable-coin backed 1:1 by a collaboration between banks, the following statement by Chaum et al fits quite well:

"Generating an electronic cash should be difficult for anyone, unless it is done in cooperation with the bank." - Chaum et al[10]

The set of available actions for BitFlow users is outlined in Figure 2.

3.3 | Smart departments

A smart department is a department modelled through a smart-contract. As smart contracts can own other smart contracts, we can create graphs where vertices consist of departments and edges represent relations, with the flow of funds as weights. This gives us enough flexibility to model the structure of large companies, where single "owning" departments coordinate sub-departments and distribute funds.

In such a view, a sub-department is only allowed to receive funds from its owner, which is ideal to control the cash-flow of complex companies. If a sub-department, supposed to receive money only from its department, has other sources of income, the system can trigger a warning for errors or fraudulent behaviours.

We can model departments with multiple sources of income with a particular vertex, the `funnel`, which sums up multiple inputs and provides a single output to the department. As shown in Figure 3A, the funnel enables multiple external incomes to flow into the top-most department. In turn, departments are divided into sub-departments, which are allowed by-design to receive funds only from their parent. Note how no edge is connected to a same-level vertex and how each vertex has a single input edge.

3.4 | Linking the pillars

To fully allow the performing of cash-flow evaluation, we need to link BKK, treasuries, and smart departments. As shown in Figure 3B, we can use a treasury to act as a funnel and regroup external multiple incomes in a single flow.

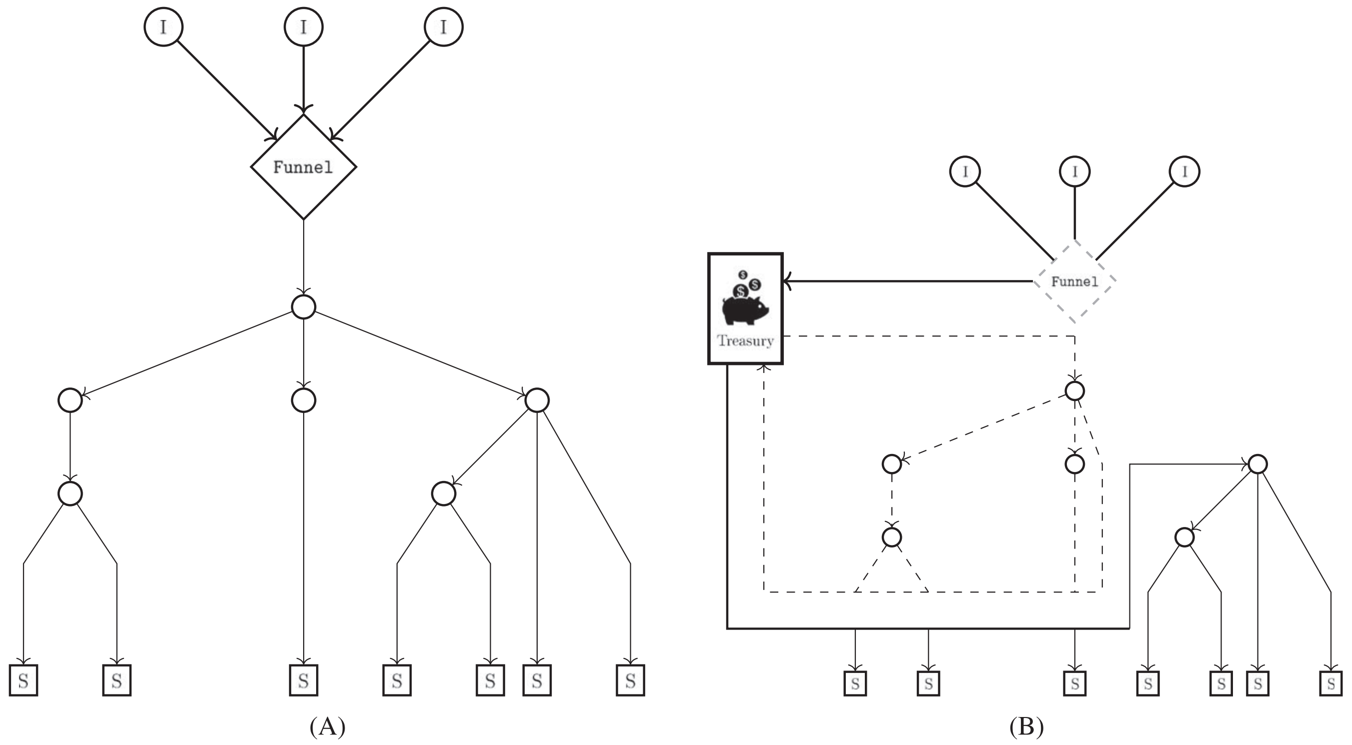


FIGURE 3 The departments marked with *I* are from other companies, marking an income to the company. Vertices marked with *S* are spendings of the company. A, Modeling the structure of a company, its income, and spending, through the use of department-vertices; B, The BitFlow ecosystem modeled on a company with departments in rural regions. Dashed lines are transfers in `BKK`, and solid lines are `DKK`

To enable companies to pay suppliers and employees in a currency used for everyday expenses, BitFlow must allow the exchange of `BKK` to `DKK` before issuing the payment, using the treasury as a currency exchange point. As previously envisioned, if the coin owner is a governmental institution and not a private one, the treasury also provides all the necessary information for taxation purposes.

4 | BITFLOW: IMPLEMENTATION

The current BitFlow proof-of-concept (*PoC*) is based on the Ethereum blockchain, following the infrastructure shown in Figure 4. The choice of Ethereum lies in its widespread adoption, which will hopefully ensure a broad usage of the proposed system. Solidity[2] has been used as

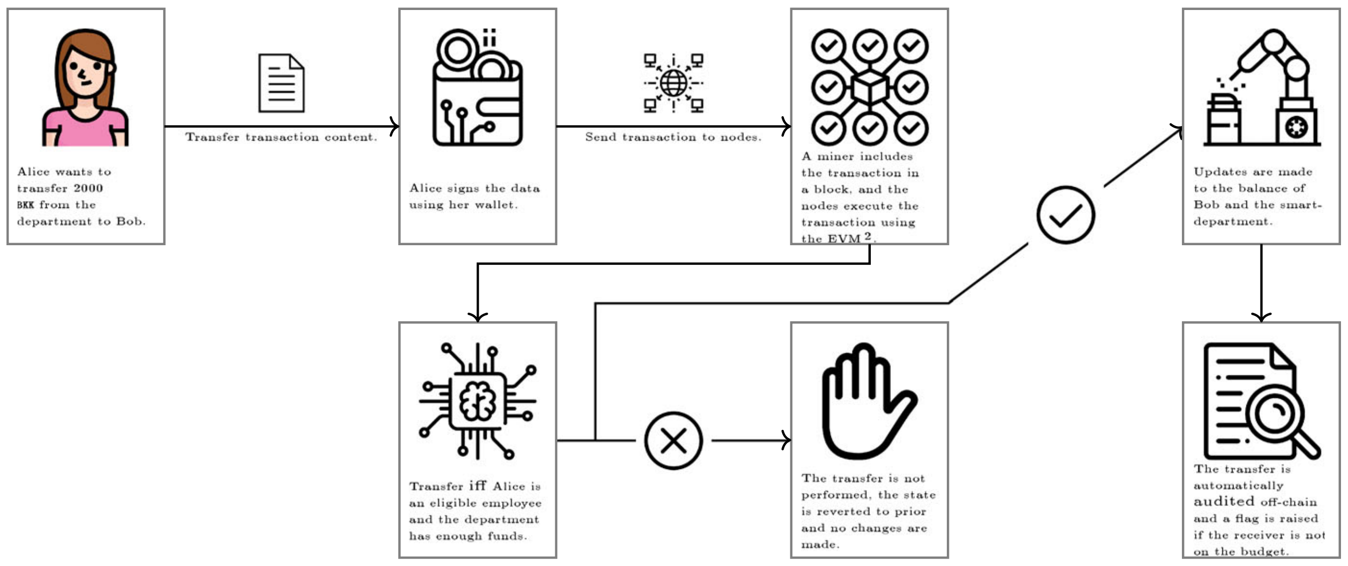

Alice wants to transfer 2000 BKK from the department to Bob.
Transfer transaction content.
Alice signs the data using her wallet.
Send transaction to nodes.
A miner includes the transaction in a block, and the nodes execute the transaction using the EVM 2.
Updates are made to the balance of Bob and the smart-department.
Transfer iff Alice is an eligible employee and the department has enough funds.
The transfer is not performed, the state is reverted to prior and no changes are made.
The transfer is automatically audited off-chain and a flag is raised if the receiver is not on the budget.

FIGURE 4 An overview of the infrastructure enabling the BitFlow system to function as intended

TABLE 1 BitFlow actions and who is allowed to execute them

Action	Who
Constitute supervisor	Owner & supervisor of parent department
Change treasury address	Owner & supervisor
Hire employee	Owner & supervisor
Fire employee	Owner & supervisor
Create child department	Owner & supervisor
Remove child department	Owner & supervisor
Add child department	Owner & supervisor
Transfer child department	Owner & supervisor
Manage budgets	Employee
Transfer internal funds	Employee
Transfer funds to the outside	Employee
Evaluation of liquidity	Everyone
Evaluation of budget	Everyone
Evaluation of spending	Everyone

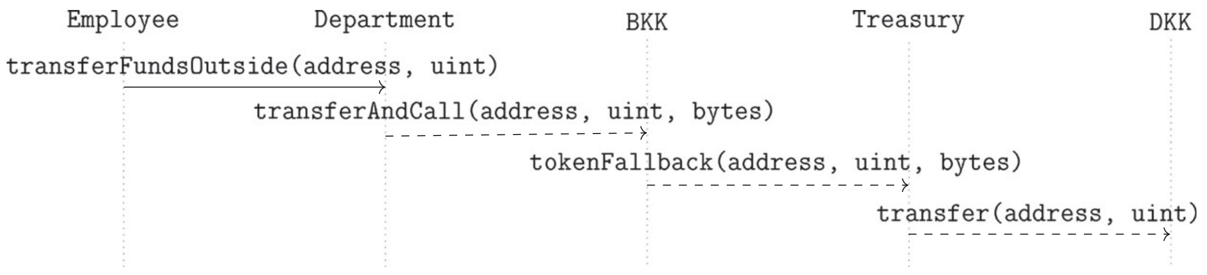

Employee
Department
BKK
Treasury
DKK
transferFundsOutside(address, uint)
transferAndCall(address, uint, bytes)
tokenFallback(address, uint, bytes)
transfer(address, uint)

FIGURE 5 The `transferAndCall` function, using the Treasury as `TokenRecipient`, enables exchanges in a single interaction

smart contract language, enabling us to define contracts in a high-level language with extensive documentation. Last but not least, the SafeMath library[††] has been used to avoid over- and under-flows in the extended ERC-20 tokens (eg, the `BKK`).

In order to meet the desired BitFlow functionalities (see Figure 3), a minimum set of necessary actions has to be implemented. Furthermore, access control routines are needed to ensure that only eligible peers can execute certain actions. The access control list outlined in Figure 1 represents the foundation of an abstract department contract.

To easily allow liquidity, budget, and spending evaluation specified in Table 1, as well as the day-grained budget managing, knowing the transfers (through ERC-20 events) is not sufficient. Therefore, we propose to store a data-object, `budgetElement`, for each actor the company has business with. Departments set into every `budgetElement` the amount budgeted for each date and map the `budgetElements` to the respective actors addresses. In turn, this enables anyone to evaluate expenditures and employees to budgeting in detail.

Due to space constraints, in this section, we do not discuss the implementation of all the actions listed in Table 1, but we will focus on one of those as an example, namely, `Transfer funds, to the outside`. In Section 5, implementation aspects of the addition of confidentiality are briefly outlined.

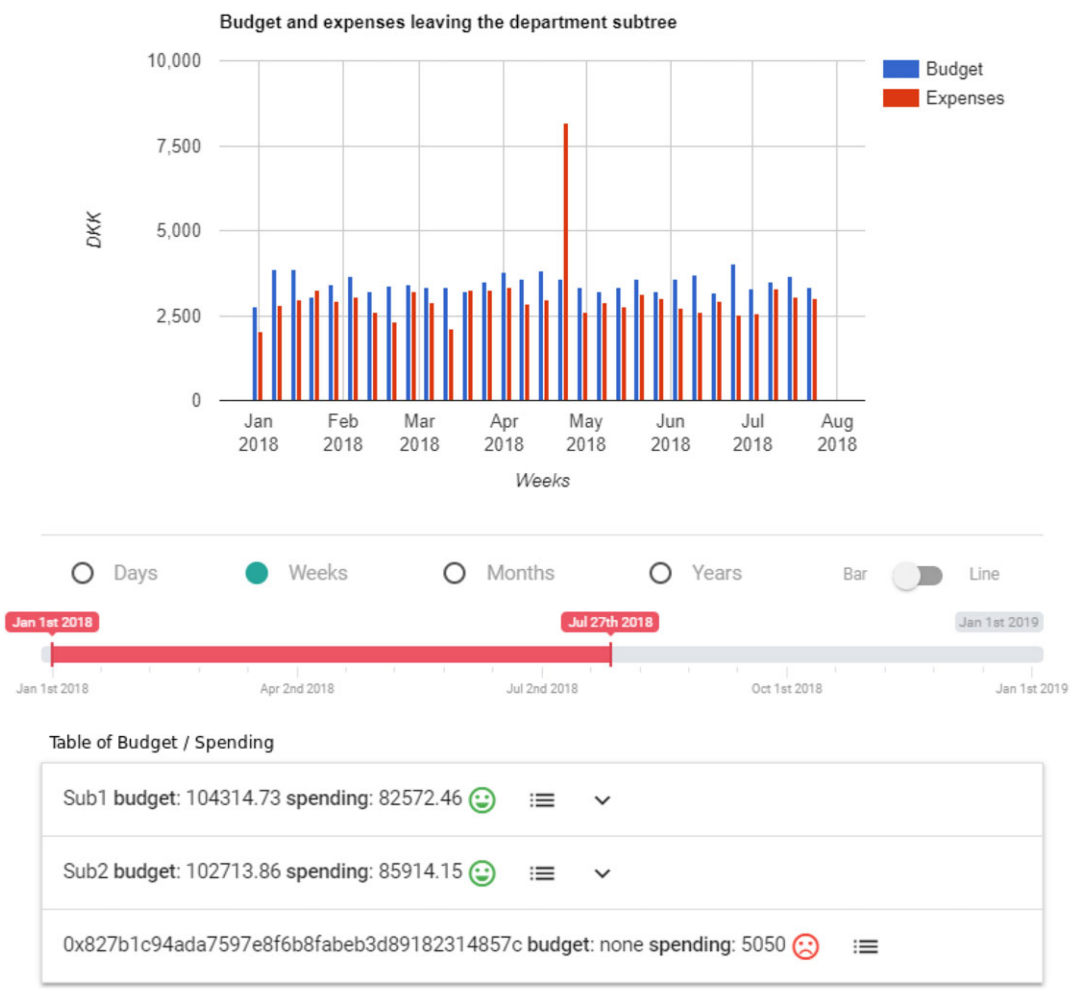
For all the technical details about the design and implementation of BitFlow, please refer to the work of Herskind[11] and GitHub.[7]

4.1 | Single interaction transfers with exchange

The ability of performing a single interaction and exchange can be enforced by implementing the token standard TransferAndCall.[‡‡§§] In BitFlow, we enforced this standard through an abstract contract, TokenRecipient (defined within the `BKK`).

As shown in Figure 5, once an employee requests a transfer and exchange, the department will transfer the funds in `BKK` and forward the recipient address to the treasury. The treasury will then convert to `DKK` and finally transfer the correct amount to the forwarded recipient. Since departments contain the treasury address and perform transfers on behalf of employees, they provide a minimal interface for the user, simply requiring the recipient address and the desired amount.

[††]https://openzeppelin.org/api/docs/math_SafeMath.html
[‡‡]https://github.com/ethereum/EIPs/issues/677
[§§]https://medium.com/@jgm.orinoco/ethereum-smart-service-payment-with-tokens-60894a79f75c



○ Days ● Weeks ○ Months ○ Years Bar Line

Sub1 **budget**: 104314.73 **spending**: 82572.46

Sub2 **budget**: 102713.86 **spending**: 85914.15

0x827b1c94ada7597e8f6b8fabeb3d89182314857c **budget**: none **spending**: 5050

FIGURE 6 Screenshot from the GUI. In this example, an employee is trying to transfer funds to himself, and the system raises a flag (a sad face in red) to inform the verifier about the fraudulent behaviour

4.2 | Graphical user interface

In order to make it fast and easy for non-technical verifiers to asses expenditure and identify fraudulent behaviour, we have implemented a simple web interface using web3.js to let our private Ethereum test-network act as a data provider. The extracted data is then fed to Google-charts[¶¶] to generate the interactive budget/expenses plot as seen at the top of Figure 6.

Monitoring the events, the GUI is updated to reflect all changes in the blockchain. This enables us to showcase more fine-grained expenditure reports from the table beneath the plot, allowing everyone to view accumulated expenditure as well as an extensive list of all transactions performed by the company and its sub-departments.

5 | ENFORCING CONFIDENTIALITY: FROM BKK TO PKK

Privacy is vital for most businesses that need to conceal transactions information to competitors and clients. Furthermore, confidential balances will hide wealth from prying eyes. Therefore, we need to implement "private" transactions within BitFlow, in order for it to be widely adopted.

A transaction is genuinely private if it enforces *confidentiality* (hiding the amount) and *anonymity* (hiding sender and recipient). Even though it is technically possible to enforce both, for the sake of simplicity, the current BitFlow version deals only with confidentiality. To do so, we propose a confidential version of BKK, namely, Private Krone (PKK), storing commitments to values on the blockchain instead of the balances, enabling publicly verification of confidential information without revealing it.

Since only the commitments, not the amounts, are stored on-chain (we will further explain this concept in Section 5.2), the blockchain alone can no longer provide sufficient knowledge to evaluate cash-flow. Consequently, in order to track its cash-flow, an actor has to store off-chain commitment openings (eg, the hidden values and blinding factors), in such a way that this information is accessible to the company. A global

[¶¶] https://developers.google.com/chart/

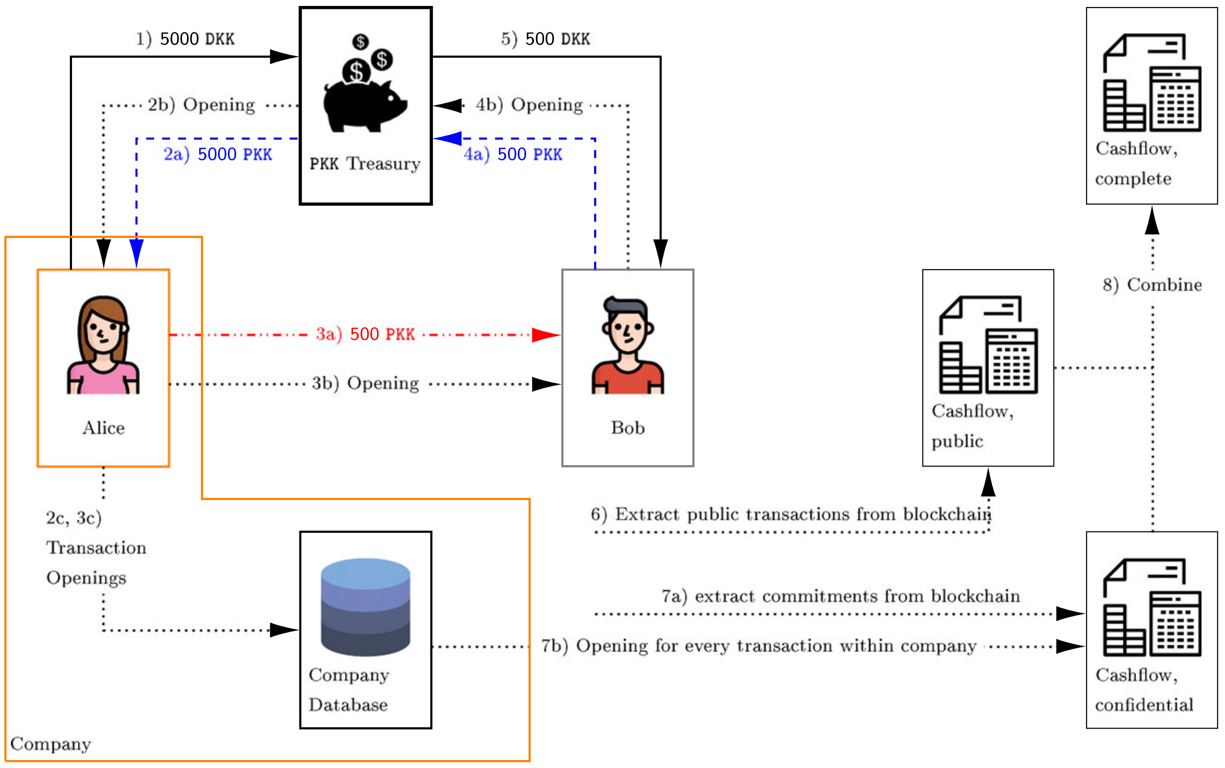


FIGURE 7 The interaction of entities supports cash-flow evaluation on hidden values. Solid black lines are off-chain DKK bank transfers. Blue dashed lines are confidential PKK blockchain transfers (where the PKK-treasury has knowledge of value). Red dash-dot-dot lines are truly confidential PKK blockchain transactions, where only the interacting parties know the value. Dotted lines are off-chain actions, eg, data transfer, extraction, or data-processing

BitFlow centralized off-chain could be an option, but it would introduce a considerable single point of failure, counteracting all the decentralization efforts of our proposal. Therefore, to address this point, we deem a set of private off-chain databases the best choice. Each company would own and govern its private off-chain database and store, only, the commitments of transactions in which the company is directly involved.

To minimize the risk of leakage, the PKK-treasury enables trading between the pair DKK and PKK only, not between BKK and PKK. The rationale behind this is that BKK amounts are publicly stored on the blockchain; therefore, anyone could observe that an X amount of BKK has been exchanged for Y PKK, and deduce that X=Y, even though for PKK only commitments are stored on-chain. Instead, exchanging directly DKK for PKK does not leak amounts, since transactions from DKK to PKK are stored off-chain and only commitments are publicly observable. As seen from Figure 7, the treasury only gains knowledge of the trades it interacts directly with, and does not have any knowledge of amounts transferred between other actors (eg, the PKK-treasury does not know how much money Alice and Bob exchange).

More explicitly, Figure 7 provides an example of a chain of interactions between a company employee Alice and the outsider Bob. Initially, Alice transfers 5000 DKK to the PKK-treasury (interaction 1 in Figure 7). In return, (interactions 2a and 2b), the PKK-treasury mints on-chain 5000 PKK for Alice and transfers the opening of the commitment, allowing her to spend the PKK. Since Alice is an employee, she stores the opening in the company database (interaction 2c). Then, in interaction 3a, she buys some goods for the company from Bob, and pays him 500 PKK. To ensure that Bob can use the funds, the opening to the transfer commitment is sent to him off-chain (interaction 3b). At the same time, Alice stores this exchange opening in the company database (interaction 3c). If Bob wishes to, he can spend the PKK as he desires but, for the sake of example, Bob decides to get DKK out of the PKK sum. He transfers on-chain 500 PKK to the PKK-treasury and sends off-chain the opening (interactions 4a and 4b). The treasury will transfer 500 DKK to Bob's bank account. Everyone can generate the public cash flow (interaction 6), but only with the private data from the company, one can generate the confidential cash flow (interactions 7a and 7b) and thereby the complete overview (interaction 8). By giving the tax authorities read-access to the company private database, tax evaluations can be performed seamlessly.

5.1 | Ensuring confidentiality

To enforce the property of confidentiality, we have to enable transactions without disclosing the balance nor the amount transferred.

Doing so with public verifiability, with a low communication-complexity, and without the need for a trusted third-party, is possible through the use of Zero-Knowledge proofs.

`"The goal of Zero-Knowledge proofs is for a verifier to be able to convince herself that a prover possesses knowledge of a secret parameter, called a witness, satisfying some relation, without revealing the witness to the verifier or anyone else" - Christian Lundkvist`[##]

More specifically, the unique family of Non-interactive Zero-Knowledge (NiZK) proofs can be used (due to space limitations, we do not introduce the theory behind such proofs, and we remind the reader not familiar with it to consult the vast literature, for instance, the works of Balasubramanian and Mala[12] and Blum et al[13]).

In our specific case, we want to prove to a verifier (a miner) that `balance ≥ transfer_amount`, without leaking information about the transfer. In this context, the idea of self-governing proof generation and public verification without the need for a trusted third-party makes the non-interactive Zero-Knowledge proof family a clearcut solution.

In Section 5.2, we propose a transfer protocol and two distinct implementations of such. The two implementations build, respectively, on zk-SNARKS[3] (Section 5.3) and on Bulletproof[6] (Section 5.4).

5.2 | Transfer protocol

The transfer protocol is the procedure to follow in order to achieve confidential transfers. As a matter of fact, this is what distinguishes `BKK` from `PKK`.

Through the Zero-Knowledge approach, we use public commitments to hide balances. In order to implement the public commitments used in this paper, we chose to utilize the Pedersen commitment scheme, meaning that BitFlow hiding (and binding) commitments exhibit homomorphic capabilities.[14] Homomorphism enables the critical capability of performing calculations and verifications over encrypted values, without revealing them. The commitment scheme used is utilizing Elliptic Curves[15] and is shown in Equation (1)

$$\chi(x, \lambda) = xg + \lambda h, \qquad \chi(x + y, \lambda_1 + \lambda_2) = \chi(x, \lambda_1) + \chi(y, \lambda_2). \tag{1}$$

This means that a single `transfer` function is defined as follows:

$$c_b = \chi(\text{balance}, \lambda_1), \qquad c_t = \chi(\text{amount}, \lambda_2), \tag{2}$$

$$\overline{c_b} = c_b + c_t = \chi(\text{balance} + \text{amount}, \lambda_1 + \lambda_2), \tag{3}$$

where χ is the public homomorphic commitment that hides a value, and a random secret (the commitment opening; x and λ). When an amount, with the commitment c_t, is sent to a recipient, the recipient has to update her c_b by adding the commitments and prove knowledge of the $\overline{c_b}$. In order to do so, she has to know the openings, eg, λ_1, λ_2, balance, and amount.

However, with a single transfer, a malicious user is free to not provide to the recipient the amount or the λ. The consequence would be a lock on the recipient, as she would be unable to prove knowledge of her new updated balance, thus, unable to perform new transfers. Therefore, we need to enrich the single `transfer` function used for `BKK` and use a `sending` and a `receiving` function for `PKK`.

Sending `PKK`

1. Calculate and prove knowledge of commitment for the remaining balance;
2. Prove that the transaction amount and the remaining balance are non-negative;
3. Share knowledge of transaction amount to the receiver, in a confidential confidential manner.

Receiving `PKK`

1. Calculate and prove knowledge of commitment for the new balance.

5.3 | ZoKrates

Among the various NiZK proofs, zk-SNARKs is a short and generic proof based on the Fiat-Shamir heuristic. Through zk-SNARKs, one can potentially prove anything built on the Ethereum Virtual Machine.[‖]

To implement zk-SNARKs in BitFlow, we use the ZoKrates toolbox.[5] The use of zk-SNARKs requires a trusted setup but provides proofs that are constant in size. At the time of writing, ZoKrates is in an early stage of development and only a limited set of its high-level language has been

[##] https://media.consensys.net/introduction-to-zksnarks-with-examples-3283b554fc3b
[‖] https://www.ingwb.com/media/2122048/zero-knowledge-range-proof-whitepaper.pdf

- $\mathcal{P}, \mathcal{V}$ know $\mathsf{X}, \mathrm{g}, \mathrm{h}$ and the same hashfunction $\mathcal{H}$;
- $\mathcal{P}$ uses x and λ as inputs and computes:

$$\mathsf{T} = \chi(\mathsf{t}_1, \mathsf{t}_2), \qquad \mathsf{c} = \mathcal{H}(\mathrm{g}, \mathsf{X}, \mathsf{T}), \qquad \mathsf{t}_1, \mathsf{t}_2 \overset{\$}{\leftarrow} \mathbb{Z}, \tag{4}$$

$$\mathsf{S} = \chi(\mathsf{s}_1, \mathsf{s}_2), \qquad \mathsf{s}_1 = \mathsf{x} * \mathsf{c} + \mathsf{t}_1, \qquad \mathsf{s}_2 = \mathsf{x} * \mathsf{c} + \mathsf{t}_2, \tag{5}$$

$$\mathcal{P} \rightarrow \mathcal{V}: \ \mathsf{T}, \mathsf{S}; \tag{6}$$

- $\mathcal{V}$ verifies:

$$\mathsf{S} \overset{?}{=} \mathsf{X} * \mathsf{c} + \mathsf{T}, \qquad \mathsf{c} = \mathcal{H}(\mathrm{g}, \mathsf{X}, \mathsf{T}). \tag{7}$$

FIGURE 8 NiZK proof used by $\mathcal{P}$ to prove knowledge of x

released. This means that the toolbox is not ready to be used in mission-critical systems, but it is complete enough to be used (and to show its potentialities) in our Proof-of-Concept (PoC).

We use hashes as commitments to balances and amounts. Since ZoKrates still does not provide a hash-function, we use a simple polynomial function, $\mathcal{H}(x) = x^3$, inadequate for a production-grade solution, but good enough for our PoC. The structure of the proofs follows the model provided in Section 5.2.

5.4 | Bulletproofs

This proposal builds on the homomorphic Pedersen commitment scheme[14] and the Fiat-Shamir heuristic.[16] For the implementation, we used Elliptic Curves[15] and we based the full verification contract on BANKEX[***] work (see BitFlow's GitHub repository[17]). The proof-generation of both range-proofs[†††] and knowledge-proofs derive from the Java-Bulletproof implementation,[6] and it is visible on GitHub.[17]

In Figure 8, we provide a NiZK proof usable by a prover $\mathcal{P}$ to demonstrate its knowledge of a secret (ie, x and λ of $X = \chi(x, \lambda)$). From an implementation point of view, the verification can be performed on-chain through a verification contract, as shown in `proof_knowledge_x.sol`, on our Github repository.[17] A more in-depth theoretical background can be found in the works of Camenisch and Stadler[18] and Hohenberger.[19] Notation: $\mathbf{k}^n = \begin{bmatrix} k^0 & .. & k^n \end{bmatrix}$ $\quad \mathbf{a} \odot \mathbf{b} = \begin{bmatrix} a_1 * b_1 & .. & a_n * b_n \end{bmatrix}$ $\quad \mathbf{a} \cdot \mathbf{b} = \sum_{i=0}^{n-1} a_i * b_i$.

5.4.1 | Implementing bulletproofs

Bulletproofs are batchable non-interactive zero-knowledge protocol that achieve efficient range-proofs without needing a trusted setup, and that scale logarithmically in size.[6] Bulletproofs achieve their efficiency by halving the length of two commitment vectors for $log_2(n)$ times iterations. Therefore, the upper limit n is a power of 2

$$v \in [0, 2^n - 1] \qquad n \in \mathbf{2}^k. \tag{8}$$

To convince a verifier $\mathcal{V}$ that $v \in [0, 2^n - 1]$, the prover $\mathcal{P}$ has to prove the knowledge of a vector $\mathbf{a}_L$, for which we have a vector commitment such that

$$\mathbf{a}_L \cdot \mathbf{2}^n = v \qquad \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \qquad \mathbf{a}_L \odot \mathbf{a}_R = \mathbf{0}^n \qquad V = \chi(v, \lambda). \tag{9}$$

In case that $v \notin [0, 2^n - 1]$, then $\mathbf{a}_L \cdot \mathbf{2}^n \neq v$, which causes a rejection of the proof. To prove knowledge of such vector, we can rewrite the first three properties of Equation 9 as described by Bünz et al[6] into a single inner product

$$(\mathbf{a}_L - z * \mathbf{1}^n) \cdot (\mathbf{y}^n \odot (\mathbf{a}_R + z * \mathbf{1}^n) + z^2 * \mathbf{2}^n) = z^2 * v + \delta(y, z), \tag{10}$$

$$\delta(y, z) = (z - z^2)(\mathbf{1}^n \cdot \mathbf{y}^n) - z^3 * (\mathbf{1}^n \cdot \mathbf{2}^n). \tag{11}$$

The range-proof can now be verified using a single inner product proof. An uncompressed protocol, as defined by Bulletproofs,[6] is outlined in the BitFlow full specification.[11]

6 | EVALUATION

Aiming at assessing the efficiency of our Proof-of-Concept (PoC), we examine the throughput in Transactions Per Second (TPS) of our implementation, and we compare it to a traditional payment solution, such as VISA. For the sake of clarity, the purpose of our PoC is to showcase

[***] https://github.com/BANKEX/ETHDenver_ConfidentialTransactions
[†††] The proof that a secret lies in a specific interval. It is one way to prove the knowledge of personal data, without revealing it.

TABLE 2 Throughput, in terms of Transactions Per Second (TPS) of each analyzed solution, compared with a traditional payment solution (VISA)

	Transactions Per Second (TPS)			
Action	**Public**	**zk-SNARK**	**Range proof**[a]	**Visa**
Send funds	14.67	0.26	0.13	56000
Receive funds	×	0.26	4.53	×
Transfers	14.67	0.13	0.13	56000

[a] The Bulletproofs are *not* batched, enabling future efficiency improvements.

TABLE 3 Truncated public commitments of ZoKrates and Bulletproofs

		ZoKrates	Bulletproof	
User-id	**Balance**	**Public**	**λ**	**Public**
2	5	125	27	$\begin{pmatrix} \text{0x16f059} \ldots \text{d7} \\ \text{0x2365a7} \ldots \text{c2} \end{pmatrix}$
1	7	343	225	$\begin{pmatrix} \text{0x182c70} \ldots \text{80} \\ \text{0x26c73e} \ldots \text{85} \end{pmatrix}$
3	7	343	201	$\begin{pmatrix} \text{0x1701d5} \ldots \text{b7} \\ \text{0x29b419} \ldots \text{14f} \end{pmatrix}$

the technical feasibility of such a system, not a final and financially viable product. Consequently, even though we use Ethereum as a basis for our implementation, we do not take into account the financial aspect of Ethereum smart-contracts execution (ie, the transaction fees paid by the issuers).

Since the following calculations are highly dependent on varying variables, such as gas limit and blocktime,[2] we arbitrarily set the gas limit at 7986194 (derived from block #5303130[‡‡‡]) and the average blocktime at 15 seconds.[§§§]

The calculations are performed as Equation (12) and the comparable results are shown in Table 2

$$\text{TPS} = \frac{\left\lfloor \frac{\text{block\_size}}{\text{tx\_cost}} \right\rfloor}{\text{blocktime}}. \tag{12}$$

It is clear that widespread adoption of our solution is not currently possible because of the low throughput. However, our proposal could be useful for some organizations, such as humanitarian aid organizations, that value transparency and accountability with respect to their donors.[8]

6.1 | What about security and confidentiality?

First of all, despite the fact that zk-SNARKs and Bulletproofs are too expensive to be widely deployed on top of existing blockchains, they constitute an opportunity to verify confidential information in an inherently open and transparent environment. An ability that, if harnessed, could propel beyond financial transaction, and could be used within other crucial activities (eg, commerce-agreements, credit-scoring, and proving liquidity before acquisition).

That being said, while both strategies strive towards a mutual goal, they enforce varying degrees of security. As previously stated, at the time of writing the ZoKrates toolbox does not provide a hash-function fit for mission-critical purposes. Moreover, zk-SNARKs require a trusted setup,[3] which further complicates the implementation. On the other hand, Bulletproofs do not require a trusted setup and can enforce a security level of 128 bits.[6] As shown in Table 3, Bulletproofs enable equivalent balances to correspond to different commitments, enforcing stronger confidentiality in comparison to the hash-function method suggested with zk-SNARKs. On account of this and the equivalent performance, we suggest the use of Bulletproofs.

We should spend some words on the very nature of transactions. Transactions are inherently not equal, ie, it is clear to see that purchasing a pizza does not necessarily entail the same level of required confidentiality, with respect to acquiring a company. Taken into account that `BKK` and `PKK` can coexist by-design, BitFlow enables its users to choose for each transaction the level of confidentiality (consequently, promptness, and price). The capability of deciding whether a certain amount should be cheap but public, or more expensive but confidential, is an invaluable property for various companies. Equally important to consider, such a hybrid approach would still support taxation evaluation. Indeed, even though only part of the transactions would be public, the cash-flow in its entirety would be still visible to the relevant authorities, as well as to the company itself.

[‡‡‡]https://etherscan.io/block/5303130
[§§§]https://etherscan.io/chart/blocktime

To give an example about the usefulness of this elastic approach, let us speak about charities. On the one hand, charities massively benefit from public transactions, which enable them to show goodwill and appropriate expenditure of donor funds. On the other hand, it could endanger its employees in corrupt regions, if criminals were to know that the local charity office recently received a large amount of money. By utilizing confidential transactions and public zero-sum proofs, together with tax-evaluations capabilities, honest charities could mitigate employees' exposure while proving their donors that no fund accumulation occurs in nefarious regions.

A zero-sum proof proves that the sum of commitments is a commitment to zero (0). It can therefore be used to show that the sum of inputs is equal to the sum of outputs (eg, that a department does not accumulate wealth). Using the notation introduced in Equation (1), we can verify n commitments as $\sum_{i=1}^{n} \chi(x_i, \lambda_i) = g^{\sum_{i=1}^{n} x_i} h^{\sum_{i=1}^{n} \lambda_i} = g^0 h^0 = \chi(0, 0)$, if and only if, the sum of the hidden values x is zero (0), $\sum_{i=1}^{n} x_i = 0$, and the holder has thoughtfully been tracking and picking her blinding factors, λ's, to satisfy $\sum_{i=1}^{n} \lambda_i = 0$. Such proofs on top of tax evaluations should convince donors about the honesty of the charity.

If confidential transactions have a number of merits, they also introduce some inherent flaws. For example, public verification of range-proofs is computationally expensive, up to the point that it could overload the underlying blockchain (eg, Ethereum).

Furthermore, to perform proper tax evaluation, the authorities would require knowledge of x and λ for every public commitment. This necessity would force companies to securely store the secret values for every confidential transaction they have ever performed. In turn, the system might fail its purpose to simplify tax evaluations and become too cumbersome for the citizens.

7 | RELATED WORK

Besides BitFlow, other works proposed to use blockchain solutions to automatically evaluate businesses finances. Furthermore, projects seeking to provide cryptocurrencies without volatility, the so-called stable coins, have been around for quite some time.

Lipton et al[20] proposed to develop a stable-coin backed by assets (FIAT), which utilizes a consortium of sponsors (equivalent to banks in BitFlow), in order to enable swaps between real-world value and blockchain value. Beside the scientific literature, companies have also been looking into utilizing stable-coins, ie, TrustToken,[¶¶¶] MakerDao,[###] and Tether[‖‖‖] are some of those organizations. Even though such companies try to establish a stable coin to improve online economy, the same concept of non-volatile currency is a foundation pillar for BitFlow.

A number of studies also proposed different ways to enforce confidential transactions. For example, Chaum et al[10] built Untraceable Electronic Cash, a system where the users have to collaborate with the bank, in order to create new coins (a similar concept found in Bitflow). While Untraeable Electonic Cash and BitFlow follow similar flows to create their coins, spending processes differ quite a bit. On the one hand, with Untraceable Electronic Cash the receiver has to interact with the bank in order to ensure that no double-spending occurs. On the other hand, BitFlow enforces this feature through the use of a distributed ledger technology (eg, Ethereum).

Aragon is a startup that proposes to increase organizations transparency and efficiency through more efficient internal elections and funds management.[****] From the public accounting perspective, significant similarities can be found with BitFlow cash-flow evaluations. Yet, Aragon proposes to utilize existing cryptocurrencies instead of stable coins, which could potentially expose businesses to undesirable volatility, as already mentioned.

Last, the Fortune 500 insurance company Allianz announced that they are considering to introduce a stable token in their organization, in order to efficiently model funds movement.[††††] Allianz believes that transparently exposing funds cash-flows would empower customers' knowledge, thus, their trust in the management. Even though Allianz and BitFlow goals seem similar, Allianz does not provide in-depth technical information about how they plan to implement such solution.

8 | FUTURE WORK

The current BitFlow version does not sufficiently scale for widespread adoption. Hence, in this section, we discuss some possible enhancements that constitute our future work.

Proof Mechanism.	The Bulletproofs is batchable, meaning that multiple proofs can be grouped and executed as one, improving the performance over individual proofs. Considering the fact that transactions are already batched into blocks, proofs within the same block could be grouped to enhance NiZK proofs performance further.
Improvements to Ethereum.	As we cannot fix the scalability issues just by improving the proof mechanism, we need to consider some alternatives. Various scaling solutions have been proposed, such as Sharding, Raiden, and Plasma.[‡‡‡‡]

[¶¶¶] https://blog.trusttoken.com
[###] https://makerdao.com/
[‖‖‖] https://tether.to/
[****] https://aragon.one/core
[††††] https://www.forbes.com/sites/tomgroenfeldt/2017/11/14/allianz-prototypes-blockchain-for-global-self-insurance-client
[‡‡‡‡] https://medium.com/imbrexblog/sharding-raiden-plasma-the-scaling-solutions-that-will-unchain-ethereum-c590e994523b

TABLE 4 Transactions Per Second (TPS), assuming 100-fold throughput. The last column shows the number of needed transactions for our sample country, Denmark

	Transactions Per Second (TPS)			
Action	Public	zk-SNARK	Range proof[a]	Needed in Denmark[b]
Send funds	1467	26	13	650
Receive funds	×	260	453	×
Transfers	1467	13	13	650

[a](14.25 + 17.86) million monthly bank transfers in Denmark at spike, May 2017. [b]The Bulletproofs are *not* batched, enabling future efficiency improvements.http://finansdanmark.dk/toerre-tal/institutter-filialer-ansatte/betalinger/transaktioner/

According to Ethereum evaluations:

> "the security threshold does decrease from 50% to ≈ 30-40%, but this is still a surprisingly low loss of security for what may be a 100-1000x gain in scalability with no loss of decentralization" - Ethereum Wiki[§§§§]

In the hypothesis of a 100-fold scaling improvement, under the assumption that `gas_limit` is fixed, the throughput would go from the values in Table 2 to the ones in Table 4.

Comparing the throughput capabilities (Table 2), it is clear that VISA is still far beyond the capabilities of a blockchain-based solution. However, comparing the number of public transactions per second with the number of required transactions per second in Denmark (ie, our sample country), as in Table 4, we can affirm that the scaling would be enough to allow a national-spread adoption. On the other hand, at the time of writing, both the zk-SNARKs and the Bulletproofs solutions cannot scale enough to face whole nation requirements.

9 | CONCLUSION

In this paper, we have proposed the design, implementation, and evaluation of BITFLOW, an Ethereum-based solution that facilitates automatic and fast cash-flow evaluations through a fluctuation-free token, namely, BKK.

As Ethereum does not implement privacy by design, widespread adoptation in the industry is highly unlikely. To address this, we have proposed a confidentiality-focused model of BKK, the PKK. We have suggested two different implementations of a NiZK protocol, one based on zk-SNARKs and the other one on Bulletproofs.

Scalability-wise, none of the two privacy-compliant solutions can scale enough to achieve widespread adoption, at the current state of the Ethereum blockchain. Nonetheless, Ethereum anticipated some scaling solutions that would increase the throughputs, somewhere between 100-fold and 1000-fold. Even though these improvements would not be enough to achieve widespread adoption of confidential transactions, they would be sufficient to achieve nation-wide adoption of the public transactions. Indeed, in this paper, we have showed how BITFLOW would double the necessary throughput required by Denmark, which has been used as sample country.

ORCID

Alberto Giaretta https://orcid.org/0000-0001-9293-7711

REFERENCES

1. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf
2. Wood G. Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Proj Yellow Pap.* 2014;151:1-32.
3. Ben-Sasson E, Chiesa A, Genkin D, Tromer E, Virza M. SNARKs for C: verifying program executions succinctly and in zero knowledge. In: 2013 Advances in Cryptology (CRYPTO 2013); 2013; Santa Barbara, CA.
4. Sasson EB, Chiesa A, Garman C, et al. Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy; 2014; San Jose, CA.
5. Eberhardt J, Tai S. ZoKrates-scalable privacy-preserving off-chain computations. Paper presented at: 2018 IEEE International Conference on Blockchain; 2018; Halifax, Canada.
6. Bünz B, Bootle J, Boneh D, Poelstra A, Wuille P, Maxwell G. Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy; 2018; San Francisco, CA.
7. Herskind L. Bitflow sourcecode. 2018. https://github.com/LHerskind/BitFlow_Narrow
8. Østergaard Jacobsen P, Bo Jensen J. *Danskerne og Velgørenhed Anno 2013.* Danmark: Copenhagen Business School; 2013.

[§§§§]https://github.com/ethereum/wiki/wiki/Sharding-FAQ

9. Xu X, Pautasso C, Zhu L, et al. The blockchain as a software connector. Paper presented at: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA); 2016; Venice, Italy.
10. Chaum D, Fiat A, Naor M. Untraceable electronic cash. In: 1990 Advances in Cryptology; 1990; Santa Barbara, CA.
11. Herskind L. Bitflow - a blockchain solution making real-time cash-flow evaluations a reality. 2018. https://findit.dtu.dk/en/catalog/2434703292
12. Balasubramanian K, Mala K. Zero knowledge proofs: A survey. In: Balasubramanian K, Rajakani M, eds. *Algorithmic strategies for solving complex problems in cryptography*. Hershey, PA: IGI Global; 2018:111-123.
13. Blum M, Feldman P, Micali S. Non-interactive zero-knowledge and its applications. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88); 1988; Chicago, IL.
14. Pedersen TP. A threshold cryptosystem without a trusted party. In: 1991 Advances in Cryptology; 1991; Brighton, UK.
15. Koblitz N. Elliptic curve cryptosystems. *Math Comput*. 1987;48(177):203-209.
16. Fiat A, Shamir A. How to prove yourself: Practical solutions to identification and signature problems. In: 1986 Advances in Cryptology; 1986; Santa Barbara, CA.
17. Herskind L. Proof generation. 2018. https://github.com/LHerskind/ConfidentialTransactions
18. Camenisch J, Stadler M. Proof systems for general statements about discrete logarithms. 1997. ftp://ftp.inf.ethz.ch/pub/crypto/publications/CamSta97b.pdf
19. Hohenberger S. 600.641 special topics in theoretical cryptography. lecture 10: More on proofs of knowledge. 2007. https://www.cs.jhu.edu/~susan/600.641/scribes/lecture10.pdf
20. Lipton A, Hardjono T, Pentland A. Digital trade coin: towards a more stable digital currency. *Royal Soc Open Sci*. 2018;5(7). Article ID 180155.

How to cite this article: Herskind L, Giaretta A, De Donno M, Dragoni N. BitFlow: Enabling real-time cash-flow evaluations through blockchain. *Concurrency Computat Pract Exper*. 2019;e5333. https://doi.org/10.1002/cpe.5333