

PRCash: Fast, Private and Regulated Transactions for Digital Currencies

Karl Wüst¹, Kari Kostiainen¹, Vedran Čapkun², and Srdjan Čapkun¹

¹ Department of Computer Science, ETH Zurich

² HEC Paris

Abstract. Fiat currency implemented as a blockchain can enable multiple benefits such as reduced cost compared to expensive handling of cash and better transparency for increased public trust. However, such deployments have conflicting requirements including **fast payments, strong user privacy and regulatory oversight**. None of the existing blockchain transaction techniques supports all of these three requirements. In this paper we design a new blockchain currency, called PRCash, that addresses the above challenge. The primary technical contribution of our work is a novel regulation mechanism for transactions that use cryptographic commitments. We enable regulation of spending limits using zero-knowledge proofs. PRCash is the first blockchain currency that provides fast payments, good level of user privacy and regulatory control at the same time.

1 Introduction

Over the last ten years, decentralized cryptocurrencies based on blockchains have gained significant attention. The primary technical primitives of blockchains are consensus and transactions. Currencies like Bitcoin [1] leverage permissionless consensus schemes and therefore operate without any trusted authority. The main drawback of permissionless consensus is low performance. Permissioned blockchains, such as ones based on Byzantine agreement, achieve better performance, but require pre-assigned validators. Regardless of the chosen consensus model, most blockchains use transactions that offer some level of anonymity. Additionally, blockchains provide transparency of money creation and transaction correctness.

While blockchains were originally envisioned to operate without any trusted parties, recently the idea of central banks issuing a fiat currency on a blockchain has gained popularity [2,3,4,5,6,7,8,9]. A fiat currency implemented as a blockchain could provide multiple benefits to the society, including reduced cost compared to expensive handling of cash, improved privacy over current non-anonymous digital payments like credit card payments, and transparency for increased public trust.

Fiat currencies have critical requirements. The first is high performance, as the such systems must be able to handle high transactions loads fast (e.g., process thousands of payment transactions per second overall and confirm individual

payments within seconds). The second requirement is user privacy. The third is regulation, as without any regulatory oversight, criminal activities such as money laundering are difficult to prevent. The lack of regulatory support is a major obstacle for the adoption of cryptocurrencies as fiat money.

High performance, strong anonymity and regulatory oversight are conflicting requirements and current blockchain transaction techniques provide only some of them. For example, transactions that use plaintext identities and amounts are fast to process and easy to regulate but provide no privacy. Usage of pseudonyms, similar to Bitcoin transactions, improves user privacy, but makes regulation ineffective. Novel transaction techniques like Confidential Transactions [10] and Mimblewimble [11] leverage cryptographic commitments for increased privacy protection. Such transaction enable hidden payment identities and values and easy transaction mixing but no regulation. More sophisticated cryptographic schemes like Zerocash [12] provide full transaction unlinkability which is often considered the strongest notion of privacy for blockchain currencies. Recent research has also shown how regulatory oversight can be added to such payments [13]. However, such techniques suffer from poor performance. For example, creation of Zerocash transactions takes up to minutes and requires downloading the entire ledger which may be infeasible on resource-constrained mobile devices. Therefore, such solutions cannot easily replace cash or card payments.

In this paper, we design a new blockchain currency, called **PRCash**, that addresses the above conflict between performance, privacy and regulation. The main use case for our solution is to enable deployment of fiat money on a blockchain by a trusted authority like a central bank. We focus on the permissioned blockchain model where transactions are confirmed by a set of appointed validators, because permissioned consensus provides significantly better performance. We assume that money is issued by a central authority. However, we emphasize that our solution is orthogonal to how consensus is achieved or how money is issued.

The primary technical contribution of our work is a novel regulation mechanism. We use commitment-based Mimblewimble transactions [11] as a starting point for our solution, because such transactions provide attractive hiding properties and sufficient performance. We add regulatory support to such transactions using a novel zero-knowledge proof construction and improve the privacy of Mimblewimble with small modifications to the transaction creation protocol.

In our regulation scheme, we limit the total amount of money that any user can receive anonymously within an epoch. Such limits are implemented using verifiable pseudorandom identifiers and range proofs. We choose to control receiving of money, to mimic existing laws in many countries (e.g., in the US, received cash transactions exceeding \$10,000 must be reported to the IRS), but our solution can be easily modified to limit spending as well. The user can choose for each payment if it should be made anonymous as long as he stays within the allowed limit, chosen by a regulatory authority. Anonymous transactions preserve the privacy properties of Mimblewimble, i.e. they hide payer identity, recipient identity and the transaction value. While validators of the blockchain system

have limited ability to link transactions with the same recipient issued within a short period of time, privacy towards third parties is even improved compared to Mimblewimble due to validators mixing transactions which removes the link between transaction inputs and outputs.

We implemented a prototype of **PRCash** and evaluated its performance. Transaction creation and verification is fast. For example, creation of a typical transaction and associated proofs takes less than 0.1 seconds and verification of 1000 transactions per second is possible with modest computing infrastructure (e.g., 4 validators with 25 quad-core servers each). When standard Byzantine agreement is used for consensus, transactions can also be confirmed quickly (e.g. within a second), which makes **PRCash** suitable for real-time payments.

Our regulation mechanism maintains the core privacy properties of Mimblewimble transactions, namely hidden sender and recipient identities and transaction amounts and easy mixing. Similar to Mimblewimble, our solution does provide full unlinkability of transactions. To the best of our knowledge, **PRCash** is the first blockchain currency that provides high performance, significantly improved privacy and regulation support at the same time.

Regulation based on zero-knowledge proofs has been previously proposed for coin-based currencies by Camenisch et al. [14]. In contrast to our solution, coin-based currencies used in [14] do not hide the recipient identity or provide transparency. Regulation extensions have also been designed for Zerocash [13]. While such schemes provide stronger anonymity guarantees and more expressive regulatory policies than our solution, their performance is significantly inferior which prevents usage in many practical scenarios. Finally, centrally-issued cryptocurrencies, like RSCoin [7], have been proposed prior to us. The main focus of such works is on consensus performance while our work focuses on transaction privacy and regulation.

To summarize, in this paper we make the following contributions:

- *Novel regulation mechanism.* We propose **PRCash**, a new blockchain currency. The primary technical contribution of this solution is a novel regulation mechanism that leverages zero-knowledge proofs for commitment-based transactions.
- *Implementation and evaluation.* We show that our transactions and regulation mechanism enable fast, fault-tolerant, large-scale deployments.

The rest of this paper is organized as follows. Section 2 gives an overview of our solution. Section 3 describes our currency in detail. We analyze the security in Section 4 and explain our implementation and evaluation in Section 5. Section 6 reviews related work and Section 7 concludes the paper.

2 PRCash Overview

Our goal in this paper is to design a new blockchain currency that enables fast payments at large scale, strong user privacy and regulatory support. The primary deployment model we consider is one where our solution is used by a central bank to implement fiat money on a blockchain. In this section we give an overview of our solution, **PRCash**.

2.1 System and Trust Model

Figure 1 shows our system model. We consider a standard permissioned blockchain model that is complemented with a regulatory authority and a central issuer of money. Here, we describe the involved entities:

Issuer. In our currency new money is created by a central entity called the *issuer*. For simplicity the primary model we consider in this paper is one where the issuer is a single entity like a central bank. In Appendix B.5 we explain how this role can be distributed if needed.

Users. Users in our system can act in two roles: as *payers* and as payment *recipients*. Users of the currency can be private individuals or organizations.

Validators. We assume a set of permissioned *validators* that maintain the ledger. The role of the validators could be taken, e.g., by commercial banks or other institutions appointed by the central bank.

Regulator. The flow of money is regulated by a central entity called the *regulator*. For simplicity, we assume that the role of the regulator is taken by a single entity, e.g., by a public authority like the IRS. In Appendix B.6 we explain how this role can be distributed among multiple parties.

If PRCash is used for a privately-issued currency, these roles can be assigned differently.

We consider an adversary that controls all networking between the users and from users to validators. The validators and the regulator are connected with secure links. Users are in possession of the public keys of the validators and the regulator and can establish secure connections to them. We otherwise rely on the standard assumptions of permissioned consensus (i.e., honest two-thirds majority of validators).

2.2 High-Level Operation and Regulation Main Idea

In many countries, the law requires reporting of large financial transactions. For example, in the US companies and individuals are mandated to report any received cash transaction that exceeds \$10,000 [15]. To enable enforcement of such laws, we design a regulation mechanism that limits the *total amount* of anonymous payments any user can *receive* within a time period (epoch). By adjusting the amount and the period, authorities can control the flow of anonymous money, e.g., reception of anonymous payments up to \$10,000 could be allowed within a month. With small changes, limits can also be put on spending instead of receiving.

Figure 1 illustrates the high-level operation of PRCash. To supply new money, the issuer creates signed issuance transactions that it sends to the validators, who verify them and publish them to the ledger. Each user enrolls in the system by obtaining a payment credential (certificate) from the regulator. As the user may lose his certificate, or the corresponding private key, we limit their validity to I_Δ epochs.

Payments involve two parties: the payer (Alice) and the recipient (Bob). To initiate a payment, Alice and Bob first agree on the transaction value. Each payment transaction consists of inputs and outputs (where the inputs are outputs

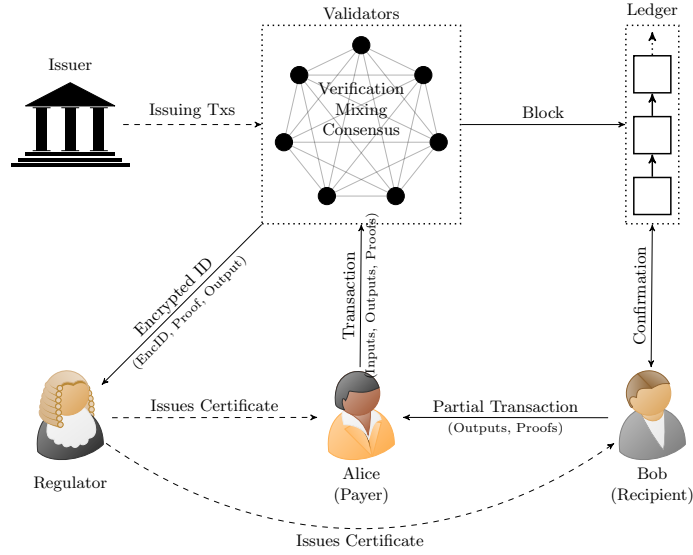


Fig. 1. System model and operation. In PRCash, new money is created centrally by the issuer. Users enroll in the system by obtaining certificates from the regulator. In each payment, the payer (Alice) and the recipient (Bob) prepare a transaction that is sent to permissioned validators who verify its correctness and add it to the next block in the public ledger. If the transaction exceeds the allowed amount of anonymous payments for Bob, he has to reveal his identity to the regulator by encrypting it with the regulator’s public key.

from previous transactions) that are cryptographic commitments that hide payer and recipient identities and transferred amounts, similar to Mimblewimble [11]. The blinding factors for the output commitments are chosen such that the sum of the input commitments is equal to the sum of the output commitments, if the sum of the input values is equal to the sum of the output values. This allows verifying the correctness of a transaction without knowledge of the transferred values. One of the outputs is a special non-spendable output to which no value is attached. This allows the recipient of a transaction to create output commitments without the payer knowing the blinding factor, i.e., the blinding factor of the commitment is only known to the recipient of a payment, and can thus be used to authenticate a following payment.

To realize regulation for such transactions, for each payment the user has two choices. First, if the user wants that the transaction remains anonymous, he must prove without disclosing his identity that he does not exceed the limit v_a in the current epoch e . Second, if the user wants to exceed his anonymous receiving limit, he must connect his identity encrypted with the regulator’s public key to the transaction.

For anonymous transactions within the limit, each user computes a pseudo-random ID per epoch (PID_e) that he attaches to his transaction outputs. He

additionally attaches a zero-knowledge proof that the ID was computed correctly and a range proof over the sum of all transaction outputs from this PID. These values are sent together with the transaction outputs to the validators. The proofs are checked by the validators and after verifying their correctness, the PIDs and the corresponding proofs are not published with the transactions for efficiency and to preserve unlinkability towards third parties.

Note that, if they choose to use non-anonymous outputs, the attached proof contains their identity encrypted with the regulator’s public key, i.e. towards any other entity, they remain anonymous. Bob prepares his part of the transaction (that includes value outputs and proofs) and sends it to Alice, who completes the transaction (by adding inputs, change outputs, proofs, and an encrypted identifier in case of a non-anonymous transaction). Alice sends the complete transaction to the validators.

The validators work in rounds. In each round, the validators collect incoming transactions, verify their correctness, mix the order of transaction inputs and outputs for increased privacy, and agree on the set of transaction that should be published. Consensus among validators is achieved through standard (Byzantine fault tolerant) protocols. At the end of the round, the validators publish a set of verified transactions as a new block on the ledger. Once the recipient (Bob) verifies the presence of the transaction in the ledger, he considers the payment confirmed. Bob can then use the value outputs from this transaction as inputs in the next payment.

If a transaction does not pass the verification (e.g., Alice or Bob attempts to create a transaction that exceeds the allowed anonymity limit, transaction inputs and outputs do not match, or one of the attached proofs is invalid), the transaction is rejected by the validators and not included in the next block. If the transaction contains any non-anonymous outputs, the validators first verify its correctness, and then forward the encrypted identifier to the regulator, who can recover the identity of Alice or Bob, depending on which transaction output was made non-anonymous.

Since anonymous change outputs are indistinguishable from anonymous value transferring outputs, they count towards the receiving limit. However, as users are in control of the size of the outputs they receive, they can mitigate this issue by using smaller received outputs, by splitting larger outputs in non-anonymous transactions, or by creating large change outputs non-anonymously.

3 PRCash Details

In this section, we describe PRCash in further detail. Our solution uses a number of cryptographic techniques as building blocks. We provide background on them in Appendix A.

3.1 System Initialization

Our system uses two groups $G = \langle g \rangle$ and $\mathbf{G} = \langle \mathbf{g}_1 \rangle = \langle \mathbf{g}_2 \rangle = \langle \mathbf{h} \rangle$ of the same order, where the discrete logarithms of \mathbf{g}_1 , \mathbf{g}_2 , and \mathbf{h} with respect to each other are unknown. The involved entities perform the following initialization steps:

Regulator. The regulator generates a keypair $(pk_{R,S}, sk_{R,S})$ for randomizable signatures (cf. Appendix A.4), an encryption keypair $(pk_{R,E}, sk_{R,E})$ for El-gamal encryption, and publishes the public keys as part of the system setup.

Validators. Each validator creates a keypair and publishes the public key as part of the system setup. Validators can use the private keys for signing new blocks. Users use the validator public keys to send transactions securely to the validators. We assume the typical permissioned blockchain model where a trusted authority dynamically assigns a set of validators, i.e. the set of validators can be updated.

Issuer. The issuer also creates a keypair that he uses for transactions that create and delete money. The issuer publishes his public key as part of the system setup.

3.2 User Enrollment

Every new user obtains the system setup that includes the public keys of the regulator, issuer, and validators. To enroll in the system, the user generates a keypair $(pk_U, sk_U) = (\mathbf{g}_1^{sk_U}, sk_U)$ for regulation proofs and sends the public key to the regulator while proving knowledge of the secret key (cf. Appendix A.2). To ensure that a user cannot enroll multiple identities, and thus circumvent the regulation, the regulator has to verify the identity of the user. If a PKI is already in place, this can be used for identification, otherwise users could, e.g., be required to visit a registration office in person.

The regulator then creates a certificate consisting of a randomizable signature σ on (sk_U, I_V) based on the user's public key pk_U and I_V , the index of the first epoch in which the certificate is valid, and sends the signature σ to the user. Recall that a randomizable signature is a signature on a list of committed values (cf. Appendix A.4). Using values pk_U and I_V , the regulator creates and signs the commitment $pk_U \cdot \mathbf{g}_2^{I_V} \mathbf{h}^r = \mathbf{g}_1^{sk_U} \mathbf{g}_2^{I_V} \mathbf{h}^r$ where r is chosen at random.

3.3 Transaction Creation

Blockchain transactions based on cryptographic commitments, such as Confidential Transactions [10] and MimbleWimble [11], have attractive features. They hide payer and recipient identities and transaction amounts, provide public verifiability and easy mixing. However, such transaction have also the undesirable property that the payment recipient necessarily sees the change outputs created by the payer. This means that, e.g., a merchant can link two independent sales if a client uses a change output from a previous transaction with the same merchant. For these reasons, we use MimbleWimble (cf. Appendix A.6) as our starting point, but modify transaction creation slightly for improved privacy.

Similar to [10,11], our transactions are based on a group G in which the discrete logarithm problem is hard, with generators g and h for which the discrete logarithm to each others base is unknown. These generators are used to represent transaction inputs and outputs as homomorphic commitments to the associated value (we use Pedersen commitments [16]), thereby hiding their values from other parties. The homomorphic commitments have the property that one can easily add and subtract committed values without opening the commitments, e.g. for

two output commitments $\text{Out}_1 = g^{r_1} h^{v_1}$ and $\text{Out}_2 = g^{r_2} h^{v_2}$ to the values v_1 and v_2 , one can easily compute a commitment to their sum $v_1 + v_2$ by multiplying the commitments: $g^{r_1} h^{v_1} \cdot g^{r_2} h^{v_2} = g^{r_1+r_2} h^{v_1+v_2}$. If the blinding factors are chosen such that the sum of the blinding factors of the inputs is equal to the sum of the blinding factors of the outputs, this property can be used to check that the sum of the input values of a transaction is equal to the sum of the output values, and the knowledge of the blinding factors can be used to authenticate and authorize payments [11] by creating an additional *excess output* $\text{Ex}_0 = g^{r_0}$ such that the product of the output commitments (including Ex_0) is equal to the product of all input commitments.

In our modified version, the exponent in Ex_0 is simply another random value, but we add an additional output value r_Δ which facilitates mixing transactions and which has to be chosen such that the product of all output commitments and g^{r_Δ} is equal to the product of the inputs. We provide the details of our modified Mimblewimble construction in Appendix B and show in Appendix B.2 that the knowledge of the blinding factor of an output is a secure method for payment authorization.

3.4 Regulation Proof Creation

In each epoch e , the user computes a pseudorandom ID as $\text{PID}_e = f_{sk_U}(e)$ (cf. Appendix A.1) and initializes the value of anonymously spent transaction outputs to $v_e = 0$. Regulation proofs are created either when Bob creates value outputs during transaction preparation or when Alice creates change outputs during transaction completion. For each output, the user can choose if it should be made anonymous or non-anonymous. For each new output, the user creates a *regulation proof*. Depending on whether the output should be anonymous or not, he does one of the following to construct the proof:

Anonymous Output. If the user wants to create an output anonymously and the value v_o of the transaction output plus the previously (in epoch e) received amount v_e is below the limit v_a , the user adds PID_e and a zero-knowledge proof of knowledge (cf. Appendix A.2) of (sk_U, I_V, σ) to the transaction such that:

- (i) The certificate is valid in the current epoch, i.e., a range proof that $I_{\text{current}} - I_\Delta < I_V \leq I_{\text{current}}$.
- (ii) The value PID_e is equal to the output of the pseudorandom function based on the secret key sk_U on input e , i.e., $\text{PID}_e = f_{sk_U}(e)$.
- (iii) The certificate is valid, i.e. $\text{verify}(pk_{R,S}, (sk_U, I_V), \sigma) = \text{true}$

In detail, the regulation proof consists of the following steps:

- (i) The user creates two commitments $\mathbf{A} = \mathbf{g}_1^{sk_U} \mathbf{h}^{r_1}$ and $\mathbf{B} = \mathbf{g}_2^{I_V} \mathbf{h}^{r_2}$ with two fresh random values r_1 and r_2 and proves knowledge of a signature on the openings of these commitments.
- (ii) Prove that \mathbf{B} is a commitment to an integer in the range $[I_{\text{current}} - I_\Delta + 1, I_{\text{current}}]$.
- (iii) Given the commitment \mathbf{A} to the value sk_U , prove that

$$\text{PID}_e = f_{sk_U}(e) = g^{1/(e+sk_U)}$$

i.e., this is the following proof of knowledge:

$$PK\{(\alpha, \gamma) : \mathbf{A} = \mathbf{g}_1^{\alpha} \mathbf{h}^{\gamma} \wedge g \cdot \text{PID}_e^{-e} = \text{PID}_e^{\alpha}\}$$

We use the common notation where greek letters correspond to values of which knowledge is being proven (cf. Appendix A.2). In the proof above, α corresponds to sk_U and γ corresponds to the blinding value of the commitment. The second term proves that the ID was computed correctly since

$$\begin{aligned} g \cdot \text{PID}_e^{-e} &= \text{PID}_e^{\alpha} \\ \Rightarrow g &= \text{PID}_e^{e+\alpha} \\ \Rightarrow g^{\frac{1}{e+sk_U}} &= (\text{PID}_e^{e+\alpha})^{\frac{1}{e+\alpha}} = \text{PID}_e \end{aligned}$$

The interactive protocol can be easily converted to a non-interactive signature on the message $M = H(o)$ using the Fiat-Shamir heuristic [17], where o is the transaction output. Including this message in the zero-knowledge proof binds the proof to the transaction output.

- (iv) The user additionally creates a range proof over the product of all anonymous outputs that share the same identifier PID_e , proving that their combined value $v_e + v_0$ is below the allowed limit v_a .

The user then updates $v_e := v_e + v_o$ after completing the transaction.

Non-anonymous Output. If the user does not want to create the output anonymously or the value v_o of the output plus v_e is above the transaction amount limit v_a , the user adds his public key encrypted with the public key of the regulator to the transaction, together with a proof that the encryption was created correctly. The user completes the following steps to create the regulation proof:

- (i) The user creates two commitments $\mathbf{A} = \mathbf{g}_1^{sk_U} \mathbf{h}^{r_1}$ and $\mathbf{B} = \mathbf{g}_2^{I_v} \mathbf{h}^{r_2}$ with two fresh random values r_1 and r_2 and proves knowledge of a signature on the openings of these commitments.
- (ii) Prove that \mathbf{B} is a commitment to an integer in the range $[I_{current} - I_{\Delta} + 1, I_{current}]$.
- (iii) Compute $C = \text{ENC}(pk_U, pk_{R,E}) = (g^{y_1}, pk_{R,E}^{y_1} \cdot pk_U)$
- (iv) Given the commitment \mathbf{A} to the value sk_U , prove that

$$C = \text{ENC}(pk_U, pk_{R,E}) = (g^{y_1}, pk_{R,E}^{y_1} \cdot pk_U)$$

i.e., this is the following proof of knowledge:

$$PK\{(\alpha, \gamma_1, \gamma_2) : \mathbf{A} = \mathbf{g}_1^{\alpha} \mathbf{h}^{\gamma_1} \wedge C[0] = g^{\gamma_2} \wedge C[1] = pk_{R,E}^{\gamma_2} g^{\alpha}\}$$

Here, α again corresponds to sk_U and γ_1 corresponds to the blinding value of the commitment, while γ_2 corresponds to the random value used for the Elgamal encryption of the users public key. The interactive protocol can again be converted to a non-interactive signature on the message $M = H(o)$ using the Fiat-Shamir heuristic [17], where o is the transaction output, to bind the proof to the transaction output.

3.5 Transaction Verification

The validators work in rounds and verify every received transaction. A transaction is correct, if

- (i) all inputs are unspent outputs of previous transactions,
- (ii) the range proofs for all outputs are correct,
- (iii) the zero-knowledge proof for excess outputs is correct, and
- (iv) the total amount of transaction inputs matches the outputs: $\prod_{i=1}^n \text{In}_i = g^{r_\Delta} \cdot \text{Ex}_0 \cdot \prod_{i=1}^{k+m} \text{Out}_i$

In addition to verifying the correctness of the transaction itself, the validators verify the regulation proofs. First, the validators verify the randomized certificate, i.e., they verify the signature on the provided commitments and check if the range proof for I_V is correct. If the verification fails, the transaction is discarded.

Otherwise, for anonymous transaction outputs, the validators verify that PID_e has been computed correctly and that the proof is bound to the associated output. If this check succeeds, they compute the product of all outputs from epoch e that share the pseudorandom identifier PID_e and check if the provided range proof holds for this product. If this is the case, the total associated value is below the allowed limit and the transaction can be included in the next block. Otherwise, the transaction is discarded.

For non-anonymous transaction outputs, the validators verify the corresponding regulation proof, i.e., that the public key of the user has been encrypted correctly with the public encryption key of the regulator and that this proof is bound to the associated transaction output. If these verifications are successful, the validators include the transaction in the next block and forward the output and the proof to the regulator, otherwise the transaction is discarded.

When the regulator receives transaction outputs with their corresponding proofs, he can decrypt the encrypted public key which serves as identifier for the user. The regulator also checks the proofs to ensure that the output was indeed created by the owner of the corresponding public key. Since the regulator knows the real-world identities associated with each public key, he can then take action as required.

In Appendix B, we provide details on how transactions in a block can be mixed by the validators, how blocks can be structured and on how the issuer can create and destroy currency.

4 Security Analysis

In this section, we provide an informal security analysis of **PRCash**. We first discuss the integrity guarantees of the system. Then we discuss the provided privacy properties, in particular, how our modifications of Mimblewimble [11] (which provides value and identity hiding, but not full unlinkability) and the added regulation impact privacy.

Payment authorization. We first consider an attacker that tries to spend an output belonging to another user without the knowledge of the corresponding blinding factor. We show in Appendix B.2 that if an adversary capable of such an

attack exists, our assumptions are violated, namely either the discrete logarithm problem can be solved efficiently in the used group or the adversary knows the discrete logarithm of h to base g , where g and h are the generators used for the commitments. The intuition behind this is that, to create a valid transaction, the outputs require range proofs for which knowledge of the blinding factor is needed and the outputs have to be chosen such that their product is equal to that of the inputs.

Double-spending protection. During each round, each non-compromised validator discards transactions with previously used or otherwise invalid inputs (cf. Section 3.5), and then all validators run a Byzantine fault tolerant consensus protocol. Thus, compromised validators cannot produce a block that would contain conflicting transactions and will be accepted by the network.

Creation of money. Only the issuer can create new money. Creation of money using normal transactions is prevented as the validators verify (i) the range proofs of all outputs for overflow and (ii) that the sum of inputs values matches the sum of output values, and only include compliant transactions in the next block. The underlying consensus protocol guarantees that each block contains only compliant transactions.

Regulation enforcement. The security of our regulation system relies on the security of the underlying zero-knowledge proofs and the pseudorandom function. The pseudorandom function (cf. Appendix A.1) is secure under the decisional Diffie-Hellman inversion assumption (DDHI). The zero-knowledge proofs rely on the hardness of the discrete logarithm problem (which is implied by DDHI) and they are secure as non-interactive proofs in the random oracle model using the Fiat-Shamir heuristic [17,18].

Privacy towards third parties. Transaction values are completely hidden and can therefore not leak any information about a transaction. Additionally, all transactions are mixed by the validators, and since the delta outputs of all transactions are summed up (cf. Section 3.3) and not published individually, it becomes impossible for third parties examining the ledger to determine which outputs belong to which inputs, even for a merchant receiving a transaction. PRCash therefore provides k -anonymity [19] against third parties, where k is the number of transactions in a block. For example, even if an adversary knows that Alice paid Bob in a transaction with output Out_1 contained in a block with 500 transaction inputs, he can only guess Alice' input with probability of at most $\frac{1}{500}$. If more privacy is desired, blocks can be made larger and validators could even add dummy transactions (with a tradeoff in efficiency).

Privacy between users. As the payer finalizes the transaction, the recipient only sees his own outputs, i.e. he is in the same position as the third party entity with partial information as described above. The payer additionally sees output commitments from the recipient which allows him to see when the output is spent. However, once the output has been used, no more information is leaked to the user.

Privacy towards validators. Recall that we assume the standard trust model for permissioned consensus where up to one third of the validators may

be malicious or get compromised by the adversary. Malicious validators do not learn transaction amounts or user identities, as our transactions are based on cryptographic commitments. Malicious validators can link transaction inputs to the corresponding outputs for all the transactions that they receive, but they cannot link inputs to their outputs for transactions that are mixed by other validators. Additionally, malicious validators are able to link multiple outputs from the same epoch that share the same pseudorandom ID. Therefore our solution does not provide full unlinkability towards validators. If combined with additional out-of-band information, this could potentially lead to some loss of privacy towards validators. The expected number of outputs sharing the same PID can be controlled by adjusting the length of the epoch (shorter epochs means fewer transactions with the same PID). Transaction linking can be addressed by using third party mixing services.

5 Evaluation

We implemented a prototype of **PRCash** to evaluate its performance. In this section, we describe our implementation, transaction verification models, verification overhead, and overall performance in terms of throughput and latency. We concentrate on performance in terms of verification time as opposed to proof generation time here, since verification is the limiting factor in our system. Note, however, that proof generations times are similar to verification times for all proofs, i.e. transactions can be created efficiently, even on devices with restricted performance such as mobile phones. On a standard PC, creation of a typical transaction takes less than 0.1 seconds.

5.1 Implementation

We implemented a prototype that covers the generation and verification of transactions, including regulation proofs. Our implementation uses the randomizable signature from Pointcheval and Sanders [20] for the generation of certificates. Other signatures with efficient protocols, such as CL-Signatures [21,22], could be used as well. We use the RELIC toolkit [23] for the elliptic curve and bilinear map operations. Our implementation makes use of the 256-bit elliptic curve BN-P256 as the base curve of a type-3 pairing that we use for the randomizable signatures. Our range proofs use commitments to digits in base 4 (cf. Appendix A.3) as this is in practice the most efficient base for the size and computation of bit-commitment based proofs. Size and computation required for the proofs could be optimized using bulletproofs from Bünz et al. [24].

5.2 Verification Models

The throughput and latency of **PRCash** depends on the used transaction verification model. For our evaluation, we consider the following two verification models, to give examples of performance under different assumptions and requirements.

VM1: Full replication. In this model, all validators verify all transactions, including the regulation proofs, and consensus is needed on the validity of all transactions and proofs. This model guarantees transaction correctness,

double-spending protection, and enforcement regulation at all times, assuming our standard permissioned consensus trust model (at most one third malicious or compromised validators).

VM2: Partitioned regulation, replicated verification. In this model, all validators verify correctness of all transactions including their range proofs, but excluding the regulation proofs. Verification of regulation proofs is instead partitioned evenly among the validators. If one validator attests to the validity of a regulation proof, it is accepted by the other validators. If a validator gets compromised, users can transact anonymously above the regulatory limit. This model may be used, if it is acceptable to lose the ability to enforce regulation momentarily. Transaction correctness (i.e., no new money is created and no double-spending occurs) is guaranteed regardless of the compromise. This model may be suitable, if e.g. regulation is delegated to commercial banks that act as validators and check the regulation proofs for their customers.

5.3 Transaction Verification Overhead

We measured the verification overhead (shown in Table 1), averaged over 1000 runs on a single core of an Intel Core i7-4770 CPU, for the following proof types:

ZKPoK of discrete log. This is a zero-knowledge proof of knowledge (ZKPoK) of the discrete logarithm and is required to verify that an excess output has no value attached.

PIDProof. This is the proof that the pseudo-random ID was constructed correctly, i.e., the user who created the proof is in possession of a valid certificate on his key and that the PID was derived correctly from this key. Depending on the number of epochs for which the signature is valid, the computation time differs, due to the included range proof. In Table 1, the measurements for epoch ranges between 2^6 and 2^{10} are shown.

EncIDProof. This is the proof that the user who created the proof is in possession of a valid certificate on his key and that his corresponding public key was correctly encrypted with the public key of the regulator. Again, the verification time differs depending on the number of epochs for which the certificate is valid.

RangeProof. The range proof by itself is used to show that an output is in the correct range, which is necessary to show that no overflow occurs, and to prove that the sum of anonymous outputs with the same PID are below the allowed threshold. The size and verification time of the range proof depend on the size of the range. For example, with a granularity of cents, a range of 2^{32} would allow transaction outputs of up to 43 million dollars.

Most commonly, transactions will have one value-transferring output, one change output, one or more inputs, plus an excess and a delta output. Since inputs do not require range proofs, and the time required to compute the commitment to the sum of their values is negligible compared to the proof verification time, we can estimate the time required to validate a standard transaction independently

Table 1. The average time for proof verification for different proof types and their sizes.

Proof Type	Time [s]	Size [bytes]
ZKPoK of discrete log (DLProof)	0.00038	64
PIDProof (epoch range = 2^6)	0.01067	1033
PIDProof (epoch range = 2^8)	0.01235	1226
PIDProof (epoch range = 2^{10})	0.01404	1419
EncIDProof (epoch range = 2^6)	0.01115	968
EncIDProof (epoch range = 2^8)	0.01284	1161
EncIDProof (epoch range = 2^{10})	0.01452	1354
RangeProof (range = 2^8)	0.00665	722
RangeProof (range = 2^{16})	0.01345	1544
RangeProof (range = 2^{20})	0.01678	1930
RangeProof (range = 2^{32})	0.02722	3088

of the number of inputs. In the case of a transaction with two anonymous outputs (different PIDs each), a full verification of the transaction requires verifying one ZKPoK of a discrete logarithm, two PID proofs, and four range proofs (one for each individual output and one per PID).

Since the maximum amount for anonymous transactions is limited, one can use a smaller range proof than for non-anonymous transactions. For example, the US requires reporting for transactions above \$10,000 [15]. An equivalent regulatory rule with a granularity of cents would approximately correspond to a range of 2^{20} . Assuming a certificate validity of 2^{10} epochs, this leads to a total verification time of 0.096 seconds.

For transactions with non-anonymous outputs, we can allow a much larger range (e.g., 2^{32}), since in this case the goal is not to limit transaction size but to prevent overflows. Such a transaction requires two range proofs, giving, in the same setting as before, a verification time of 0.084 seconds. Combinations, where one output is anonymous and one is not, are, of course, also possible. Given this transaction verification overhead, within one second, roughly ten transactions can be fully verified on a single core. From this value we can in turn estimate the required computing resources to handle the expected transaction load.

In verification model **VM1**, each validator checks all transactions and proofs. To verify 1000 tps, each validator would require approximately 25 quad-core servers. In **VM2**, transactions and range proofs are verified by all validators to protect against overflows in outputs, but verification of regulation proofs can be partitioned across the validators. Assuming 16 validators, each of them would require 15 quad-core servers to process 1000 tps.

In Appendix C, we estimate figures for latency and throughput given a standard consensus protocol (PBFT [25]) using measurements from Croman et al. [26]. The numbers show that using 16 validators, a throughput of 480 transactions per second can be achieved. Since the nodes in the experiment by Croman et al. were globally distributed and only had limited bandwidth, it is reasonable to assume that higher throughputs can be achieved in the setting we consider, if validators are geographically close and may even be connected through dedicated lines.

6 Related Work

Regulation in coin-based currencies. Camenisch et al. introduced an e-cash system where a trusted authority can control the total amount of anonymously spent money [14]. We use similar zero-knowledge proof techniques for PRCash. However, these two solutions have noteworthy differences. In their scheme, it suffices to limit the number of transactions, since the system is coin-based, i.e., the number of spent coins is equal to the amount. In our solution, we also need to take into account the values of the transactions, while keeping them secret. In a coin-based scheme, the size of the transaction and the computation required to verify the proofs grows with the transaction value. Additionally, such a system is not transferable and thus leaks the total amount received by the merchant to the bank once it is deposited. Partial value secrecy is possible when offline payments are allowed, but this option ensures only double-spending detection (no prevention). In comparison, PRCash provides better privacy, constant payment overhead, and more transparency.

Regulation in blockchain currencies. Zerocash [12] is a sophisticated decentralized anonymous payment scheme that leverages a blockchain. Zerocash provides what is commonly considered the strongest level of anonymity, i.e., it hides transaction identities and values and makes transactions unlinkable. Garman et al. [13] have proposed a solution for regulation for Zerocash payments. However, as with regular Zerocash transactions, while verification is efficient, transaction creation is prohibitively expensive in terms of computation, which makes it unusable for replacement of cash or card payments, where transaction should be finalized within seconds. Additionally, Zerocash-style transactions requires full nodes, as a client has to download the entire ledger and decrypt every transaction to determine whether it is the recipient of the transaction. These requirements make anonymous transactions unpractical for resource constrained devices and causes most participants to use unshielded transactions in practice (i.e. in Zcash [27]), which decreases anonymity overall [28].

Centrally-issued currencies. RSCoin [7] is a centrally-issued cryptocurrency solution. The main technical contribution of their work is scalability of consensus, while the primary contribution of our work is a novel regulation mechanism that address the conflict between performance, privacy and regulation.

7 Conclusion

Despite more than three decades of research on digital currencies, their adoption as fiat money issued by a central bank has not become a reality. While the reasons for this may be numerous, and not always purely technical, a major obstacle for their adoption is the fact that such deployments have conflicting technical requirements. In this paper, we have presented PRCash that is the first blockchain currency with transactions that are fast, private and regulated at the same time.

References

1. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
2. Morten L Bech and Rodney Garratt. Central bank cryptocurrencies. 2017.
3. David Mills, Kathy Wang, Brendan Malone, Anjana Ravi, Jeff Marquardt, Clinton Chen, Anton Badev, Timothy Brezinski, Linda Fahy, Kimberley Liao, Vanessa Kargenian, Max Ellithorpe, Wendy Ng, and Maria Baird. Distributed ledger technology in payments, clearing, and settlement. Washington, 2016. Board of Governors of the Federal Reserve System. <https://doi.org/10.17016/FEDS.2016.095>.
4. Carolyn A. Wilkins. Fintech and the financial ecosystem: Evolution or revolution?, 2016. <http://www.bankofcanada.ca/wp-content/uploads/2016/06/remarks-170616.pdf>.
5. Mas working with industry to apply distributed ledger technology in securities settlement and cross border payments, 2017. <http://www.mas.gov.sg/News-and-Publications/Media-Releases/2017/MAS-working-with-industry-to-apply-Distributed-Ledger-Technology.aspx>.
6. JP Koning. Fedcoin: a central bank-issued cryptocurrency. *R3 Report*, 15, 2016.
7. George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.
8. Stefan Ingves. The e-krona and the payments of the future, 2018. <https://www.riksbank.se/globalassets/media/tal/engelska/ingves/2018/the-e-krona-and-the-payments-of-the-future.pdf>.
9. Amanda Billner. Now there are plans for ‘e-krona’ in cash-shy sweden, 2018. <https://www.bloomberg.com/news/articles/2018-10-26/riksbank-to-develop-pilot-electronic-currency-amid-cash-decline>.
10. Gregory Maxwell. Confidential transactions. https://people.xiph.org/~greg/confidential_values.txt, 2015.
11. Tom Elvis Jedusor. Mumblewimble. <http://mumblewimble.org/mumblewimble.txt>.
12. Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014.
13. Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In *International Conference on Financial Cryptography and Data Security*, pages 81–98. Springer, 2016.
14. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash. In *International Conference on Security and Cryptography for Networks*, pages 141–155. Springer, 2006.
15. 31 CFR 1010.330 - Reports relating to currency in excess of \$10,000 received in a trade or business. <https://www.law.cornell.edu/cfr/text/31/1010.330>, 2012.
16. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO ’91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.
17. Amos Fiat and Adi Shamir. *How To Prove Yourself: Practical Solutions to Identification and Signature Problems*, pages 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.
18. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Eurocrypt*, volume 96, pages 387–398. Springer, 1996.

19. Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Technical report, SRI International, 1998.
20. David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Cryptographers' Track at the RSA Conference*, pages 111–126. Springer, 2016.
21. Jan Camenisch and Anna Lysyanskaya. *A Signature Scheme with Efficient Protocols*, pages 268–289. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
22. Jan Camenisch and Anna Lysyanskaya. *Signature Schemes and Anonymous Credentials from Bilinear Maps*, pages 56–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
23. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIBrary for Cryptography. <https://github.com/relic-toolkit/relic>.
24. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Efficient range proofs for confidential transactions. Technical report, Cryptology ePrint Archive, Report 2017/1066, 2017. <https://eprint.iacr.org/2017/1066>, 2017.
25. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.
26. Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.
27. Zcash. <https://z.cash/>.
28. George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 463–477, Baltimore, MD, 2018. USENIX Association.
29. Yevgeniy Dodis and Aleksandr Yampolskiy. *A Verifiable Random Function with Short Proofs and Keys*, pages 416–431. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
30. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.
31. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
32. Ueli M Maurer. Unifying zero-knowledge proofs of knowledge. *AFRICACRYPT*, 9:272–286, 2009.
33. Jan Camenisch and Markus Stadler. *Efficient group signature schemes for large groups*, pages 410–424. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
34. Jan Camenisch, Rafik Chaabouni, et al. Efficient protocols for set membership and range proofs. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 234–252. Springer, 2008.
35. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology—EUROCRYPT 2000*, pages 431–444. Springer, 2000.
36. Wenbo Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Public Key Cryptography*, pages 60–71. Springer, 1998.
37. T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.
38. Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 522–526. Springer, 1991.

39. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 307–315, New York, NY, 1990. Springer New York.

A Background

In this appendix, we provide background information on the cryptographic primitives that we use as building blocks for our currency.

A.1 Dodis-Yampolskiy Pseudorandom Functions

Dodis and Yampolskiy introduced a pseudorandom function [29] which, for a secret key sk and a generator g of a group G is defined as

$$f_{sk}(x) = g^{1/(x+sk)}$$

This construction is secure if the Decisional Diffie-Hellman Inversion assumption holds in group G .

A.2 Zero-Knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge (ZKPOK) [30] allows one party (the prover) to prove to another party (the verifier) that a certain statement is true, without revealing any other information. Well-known techniques for proving knowledge of a discrete logarithm in zero-knowledge exist, such as [31,32]. We use the notation introduced in [33] for proofs of knowledge. For example

$$PK\{(\alpha, \beta) : x = g_1^\alpha h^\beta \wedge y = g_2^\alpha\}$$

is a zero-knowledge proof of integers α and β , s.t. $x = g_1^\alpha h^\beta$ and $y = g_2^\alpha$ where g_1, h, x are elements of a group $G_1 = \langle g_1 \rangle = \langle h \rangle$ and g_2, y are elements of a group $G_2 = \langle g_2 \rangle$. In this notation, the values of which knowledge is proven are denoted by greek letters and all other values are known to the verifier. Using the Fiat-Shamir heuristic [17], such proofs can be converted to non-interactive proofs of knowledge.

A.3 Range Proofs

A range proof [34,35] is a specific type of zero knowledge proof that allows to prove to a verifier that a committed value lies within a given range. A simple range proof can consist of proving that the committed value is one of the values in the interval, e.g., to prove that a commitment C is a commitment $g^r h^x$ where $x \in [0, 7]$ and r is a random blinding factor, one can prove this with the following proof of knowledge:

$$PK\{(\alpha) : C = g^\alpha \vee C \cdot h^{-1} = g^\alpha \vee C \cdot h^{-2} = g^\alpha \vee \dots \vee C \cdot h^{-7} = g^\alpha\}$$

Clearly, this becomes inefficient with larger ranges. Instead one can decompose the value into multiple commitments to the powers of two (bit commitments), for example, such that the product of these commitments is equal to the original commitment. It then suffices to prove for every commitment that it either commits to zero or the correct power of two [36]. The proof above then becomes the following proof by splitting the commitment into three commitments

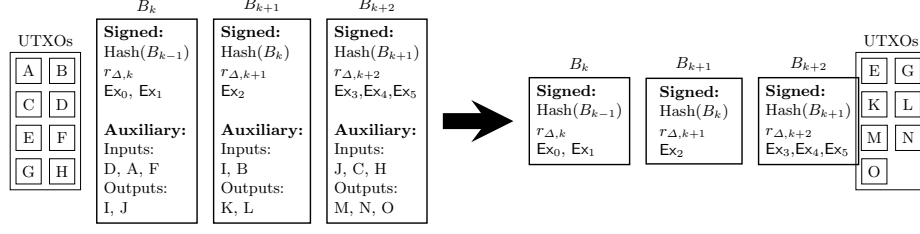


Fig. 2. Blockchain compression. Given a set of unspent transaction outputs (UTXOs) the blockchain can be easily compressed when adding new blocks (here B_k, B_{k+1}, B_{k+2}). All inputs of the new blocks are removed from and new outputs are added the UTXO set. The inputs and outputs can then be removed from the individual blocks, thus compressing the chain.

C_0, C_1, C_2 :

$$\begin{aligned}
 PK\{(\alpha, \beta, \gamma) : & (C_0 = g^\alpha \vee C_0 \cdot h^{-1} = g^\alpha) \\
 & \wedge (C_1 = g^\beta \vee C_1 \cdot h^{-2} = g^\beta) \\
 & \wedge (C_2 = g^\gamma \vee C_2 \cdot h^{-4} = g^\gamma)\}
 \end{aligned}$$

This approach can be generalized to proofs in any base [34].

A.4 Randomizable Signatures

A randomizable signature scheme provides the ability to prove the possession of a signature on committed values. The signature schemes by Camenisch and Lysyanskaya [21,22] or the scheme by Pointcheval and Sanders [20] allow to obtain a signature on a list of committed values without disclosing the values to the signer. The recipient of the signature can then prove efficiently that he is in possession of a signature on these values using fresh commitments on the same values.

A.5 Elgamal Encryption

Elgamal encryption [37] is an encryption scheme based on the difficulty of the discrete logarithm and thus compatible with standard techniques for zero-knowledge proofs. Elgamal encryption is secure in a group G if the Decisional Diffie-Hellman assumption holds in that group. To encrypt a message $m \in G$ with the public key $pk \in G$, a value r is chosen at random and the ciphertext is then computed as $(g^r, m \cdot pk^r)$. The recipient of a ciphertext (c_1, c_2) can then decrypt the message using the secret key sk (corresponding to the public key pk) as $m = c_2 \cdot (c_1^{sk})^{-1}$.

A.6 Confidential Transactions and MimbleWimble

In a transaction-based digital currency, Confidential Transactions [10] represent the transaction input and output values as homomorphic commitments. As the commitments are homomorphic, one can choose the blinding factors for the outputs such that the sum of the input commitments is equal to the sum of

the output commitments, if the sum of the input values is equal to the sum of the output values. This allows verifying the correctness of a transaction without knowledge of the transferred values.

MimbleWimble [11] uses the same approach, but adds an additional non-spendable output to which no value is attached. This allows the recipient of a transaction to create output commitments without the payer knowing the blinding factor, i.e., the blinding factor of the commitment is only known to the recipient of a payment, and can thus be used to authenticate a following payment.

MimbleWimble has the property that the blockchain can be easily compressed. Our solution inherits this property from MimbleWimble. The signed part of each block contains a hash of the previous block, all excess outputs of the block, and the sum of the delta outputs. The auxiliary part of the block contains all transaction inputs and all transaction outputs. Outputs of previous transactions that are used as inputs in the new block can be removed from the set of unspent transaction outputs (UTXOs) while new outputs are added. All spent outputs and inputs can be completely removed from storage once the UTXO set has been updated. Using the UTXOs, excess and delta outputs of all blocks, and the values of issuance and deletion transactions, the full chain can still be verified. All of this combined can be interpreted as one large transaction that, if valid, implies the validity of the whole blockchain. An example for how this compression works in PRCash is shown in Figure 2.

B Transaction Details & Block Creation

In this Appendix, we provide the details of the modifications made to Mimblewimble [11] transactions, prove that knowledge of the blinding factors can be used for payment authorization, and we give an overview of how transactions can be mixed and blocks can be created.

B.1 Transaction Creation

To prevent the transaction tracking of Mimblewimble [11] transactions, mentioned in Section 3, we modify the transaction creation such that the payer finalizes the transaction. To increase payment anonymity further, we also include another output (r_Δ) that does not have a value attached. This additional output is submitted to the validators as a scalar such that multiple transactions can be merged. Inclusion of such additional output makes it impossible to later match transaction inputs to corresponding outputs.³

Our transaction creation protocol, that includes the regulation proofs explained above, is shown in Figure 3. The protocol proceeds as follows:

- (i) The recipient, Bob, creates k value outputs $\text{Out}_i = g^{r'_i} h^{v'_i}$ ($1 \leq i \leq k$), for the payment value $v_T = \sum_{i=1}^k v'_i$. For each of the value outputs, he also

³ Matching transaction inputs to outputs after reordering is in general already an NP-complete problem (subset sum). However, most transactions will only have few inputs and outputs, which can make linking feasible in practice without this additional measure.

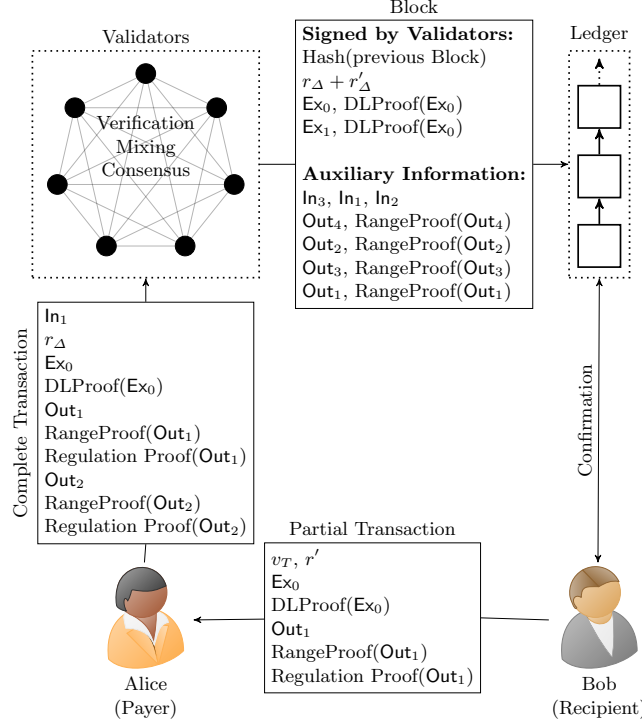


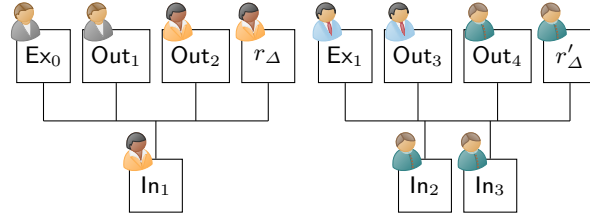
Fig. 3. Transaction and block creation. In this example transaction, Alice pays an amount v_T to Bob. First, Bob creates a partial transaction that he sends to Alice, who completes it by adding her inputs, outputs and proofs. Alice then sends the complete transaction over a secure connection to a validator. The validators verify and mix the transactions and reach consensus on a block that they then sign and publish as part of the ledger. The block in this example consists of the two transactions shown in Figure 4.

creates a range proof to prove that the value is in a valid range (i.e., that no overflow occurs where money is created out of nothing). He additionally attaches a regulation proof to each output as described above in Section 3.4. He then creates an *excess output* $\text{Ex}_0 = g^{r'_0}$ that has no value attached, proves knowledge of r'_0 by proving knowledge of the discrete log of Ex_0 to base g ($\text{DLProof}(\text{Ex}_0)$) and sends his outputs (including range proofs, proof of knowledge of r'_0 and regulation proofs), v_T and $r' = r'_0 + \sum_{i=1}^k r'_i$ to Alice. The additional excess output Ex_0 is required to ensure that only Bob can spend his newly created outputs. Otherwise Alice would know the sum of the blinding factors of his outputs and could thus spend them. An example for such a *partial transaction* is shown in Figure 3, where Bob creates one value output (Out_1).

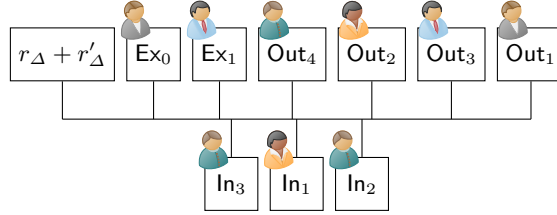
- (ii) If Alice agrees with the transaction value v_T , with her inputs $\text{In}_i = g^{r_i} h^{v_i}$ ($1 \leq i \leq n$), s.t. $v = \sum_{i=1}^n v_i$ and $r = \sum_{i=1}^n r_i$, she creates m change outputs

$\text{Out}_i = g^{r'_i} h^{v'_i}$ ($k < i \leq k + m$), s.t. $v - \sum_{i=k+1}^{k+m} v'_i = v_T$ and range proofs and regulation proofs for these outputs. She then computes a *delta output* $r_\Delta = r - \sum_{i=k+1}^{k+m} r'_i - r'$ and combines all of her inputs, Bob's and her outputs (including all proofs) and r_Δ into a *complete transaction*. Alice's inputs are outputs of previous transactions that can be money issuing transactions as described in Appendix B.5. In the example in Figure 3, Alice uses one input (In_1) and one change output (Out_2) in the transaction.





- (iii) Finally, Alice sends the complete transaction to one or more validators, as shown in Figure 3 encrypted under their public keys. The number of validators depends on the used transaction validation strategy (see Section 5).



(a) A transaction from Alice to Bob (on the left) and a transaction from Charlie to Dave (on the right). The transactions fulfill the conditions $\text{In}_1 = g^{r_\Delta} \text{Ex}_0 \text{Out}_1 \text{Out}_2$ and $\text{In}_2 \text{In}_3 = g^{r'_\Delta} \text{Ex}_1 \text{Out}_3 \text{Out}_4$, respectively.



(b) The two transactions from above can be merged as shown here. The merged transaction fulfills the condition $\prod_{i=1}^3 \text{In}_i = g^{r_\Delta + r'_\Delta} \text{Ex}_0 \text{Ex}_1 \prod_{i=1}^4 \text{Out}_i$ and is thus still a valid transaction. Since the order of inputs and outputs is irrelevant for the validity condition, inputs and outputs can be reordered arbitrarily (e.g. ordered in binary order).

Fig. 4. Transaction combining. Shown above are two transactions before and after combining. The first transaction (on the left) goes from Alice to Bob, the second transaction (on the right) from Charlie to Dave. Outputs created by Alice are marked , outputs created by Bob , outputs created by Charlie  and outputs created by Dave .

The validators then verify the transaction as described in Section 3.5. Two example transactions are shown in Figure 4a. For the transaction from Alice to Bob, the validators check if $\text{In}_1 = g^{r_\Delta} \text{Ex}_0 \text{Out}_1 \text{Out}_2$ and if the proof of knowledge of the discrete logarithm of Ex_0 ($\text{DLProof}(\text{Ex}_0)$), as well as the range proofs for Out_1 and Out_2 are correct.

B.2 Payment Authorization Proof

In this Appendix, we show that, given a group $G = \langle g \rangle = \langle h \rangle$, a transaction output $\text{Out} = g^r h^v$ can only be spent by an authorized entity, i.e. an entity who knows the secret blinding factor r .

Theorem 1. *If the discrete logarithm problem is hard in G and the discrete logarithm of h to base g is unknown, the probability that an output $\text{Out} = g^r h^v$ can be spent by an adversary without knowing r is negligible.*

Proof. For our proof, we distinguish the following two cases:

Case 1: W.l.g. assume that an attacker is able to create three outputs $\text{Out}' = g^{r'} h^v$, $\text{Ex}' = g^{r''}$ and r_Δ s.t. $\text{Out} = g^{r_\Delta} \text{Ex}' \text{Out}'$, i.e. $g^r h^v = g^{r_\Delta} g^{r''} g^{r'} h^v$ with non-negligible probability. For the transaction to be valid, the adversary needs to attach a proof of knowledge of r'' and a range proof for v , i.e. a proof of knowledge of r', v s.t. v is in the allowed range. As the zero knowledge proofs are sound if the discrete logarithm problem is hard, the adversary knows r', r'', v (and of course r_Δ). Consider a game where the adversary wins, if on input $(g^r h^v, v)$ the adversary outputs values r', r'', r_Δ s.t. $g^r h^v = g^{r_\Delta} g^{r''} g^{r'} h^v$. Clearly, the adversary considered above can win this game with non-negligible advantage.

We now show how to construct a solver for the discrete logarithm problem given this adversary. On input $X = g^x$, the solver creates a randomized instance $(X \cdot g^{r_1} h^{r_2}, r_2) = (g^{x+r_1} h^{r_2}, r_2)$ with r_1, r_2 chosen uniformly at random. On output (y, z, w) from the adversary, the solver computes and outputs $x' = y + z + w - r_1$. If the adversary wins the game, clearly $x = x'$, i.e. the solver correctly computes the discrete logarithm of the input X and thus, if the adversary has a non-negligible probability to be able to create a valid transaction, the discrete logarithm problem can be solved efficiently in group G . As this violates our assumption, it follows that an attacker cannot create such a valid transaction with non-negligible probability.

Case 2: W.l.g. assume that an attacker is able to create three outputs $\text{Out}' = g^{r'} h^{v'}$, $\text{Ex}' = g^{r''}$ and r_Δ s.t. $v \neq v'$ and $\text{Out} = g^{r_\Delta} \text{Ex}' \text{Out}'$, i.e. $g^r h^v = g^{r_\Delta} g^{r''} g^{r'} h^{v'}$, with non-negligible probability. For the transaction to be valid, the adversary needs to attach a proof of knowledge of r'' and a range proof for v' , i.e. a proof of knowledge of r', v' s.t. v' is in the allowed range. As the zero knowledge proofs are sound if the discrete logarithm problem is hard, the adversary knows r', r'', v' (and of course r_Δ). We consider the game where the adversary wins, if on input $(g^r h^v, v)$ the adversary outputs values (r', r'', r_Δ, v') s.t. $g^r h^v = g^{r_\Delta} g^{r''} g^{r'} h^{v'}$ and $v \neq v'$.

We now show how this adversary can be used to compute $\text{DL}_g(h)$. The solver creates a randomized instance $(g^{r_1} h^{r_2}, r_2)$ with r_1, r_2 chosen uniformly at random and gives this as input to the adversary. On output (y, z, w, u) from the adversary, the solver computes and outputs $x = (y + z + w - r_1)(r_2 - u)^{-1}$. If the adversary wins the game, this corresponds to $\text{DL}_g(h)$, which violates our assumption, i.e. an attacker cannot create such a valid transaction.

B.3 Mixing and Consensus

The validators collect a set of verified transactions and in the end of the round mix them by using two merging properties of our transactions. The first merging option is to *combine* two valid transactions together which creates another valid transaction. Combining several transactions into one large transaction breaks the direct correlation between inputs and outputs in the original transactions. The more transactions are combined in one round, the harder it is for third parties to link inputs and outputs based on published, combined transactions. An example for this process is shown in Figure 4 where two transactions are combined into one and the inputs and outputs are reordered. Since the order of inputs and outputs is irrelevant for the correctness of a transaction, they can be reordered arbitrarily (e.g. ordered in binary order). Additionally, by only publishing the sum of the delta outputs instead of the individual values, deciding which set of transaction outputs belong to which set of inputs becomes impossible.

The second merging option is *compacting*. If an output of one transaction appears as an input in another transaction, the matching input-output pair can be simply be removed, resulting in a smaller but still valid transaction. Compacting makes transaction linking more difficult and improves storage efficiency. Once the validator has verified and merged (mixed) all received transactions in the current round, the remaining inputs and outputs can be simply sorted as a list for publishing.

The validators then need to achieve consensus over the content of the next block depending and we assume that they run a Byzantine fault tolerant consensus protocol to protect against double spending. Validators can cache unspent transaction outputs from all previous blocks to speed up verification of new transactions (needed for double-spending protection). After achieving consensus over a block, validators can remove all inputs of the block from their cached set and add all new outputs to it.

B.4 Block Structure

Each block consists of a first part signed by the validators and a second part containing auxiliary information. The first signed part contains the sum of all delta outputs, all excess outputs including the zero-knowledge proofs of their exponents, and the hash of the previous block. Additionally, if the block contains an issuance or a deletion transaction, the signed part also contains the explicit amounts of money that are added or removed. As auxiliary information, the block contains a list of inputs and a list of outputs including their range proofs.

An example block that consists of two transactions is shown in Figure 3. The signed part of the block only contains the excess outputs and the sum of the delta outputs of all transactions (Ex_0 , Ex_1 and $r_\Delta + r'_\Delta$ in the example). The transaction inputs and transaction outputs with a value do not need to be included in the signed part, but they still need to be published including the range proofs of the outputs, so that other parties can verify the correctness of the blockchain.

This block structure allows compression of the blockchain by compacting transactions across blocks. Outputs of previous transactions that are used as inputs in the new block can be removed from storage without losing the ability to verify the complete chain. All that is required for the verification is the set of unspent transaction outputs, excess and delta outputs of all blocks, and the values of issuance and deletion transactions. All of this combined can be interpreted as one large transaction that, if valid, implies the validity of the whole blockchain. This makes the storage required to verify the full chain very small and slowly growing for third parties that do not want to store all transactions. An example for this is shown in Figure 2 (Appendix A).

B.5 Issuance

Our currency provides an explicit mechanism for the issuer to increase, or decrease, the amount of currency in circulation. This can be done with a special transaction type that requires a signature from the issuer.

Specifically, the issuer can publish an *issuance transaction* with an explicitly stated amount v . The issuer creates k transaction outputs $\text{Out}_i = g^{r'_i} h^{v'_i}$ ($1 \leq i \leq k$), such that $v = \sum_{i=1}^k v'_i$, and which all have a range proof attached. The issuer then additionally creates an excess output $\text{Ex}_0 = g^{r'_0}$, s.t. $r'_0 + \sum_{i=1}^k r'_i = 0$ and proves knowledge of r'_0 . The transaction is valid, if h^v is equal to the sum of the outputs. The outputs created by such an issuing transaction could, e.g., be transferred to commercial banks who can then further distribute the newly created money. The issued amount v is published in plaintext to the next block with the issuance transaction.

The role of the issuer can easily be distributed among multiple parties by requiring signatures from multiple parties for issuance transactions. This may be particularly interesting for private deployments, where there is no central bank that can be assumed to be trusted.

B.6 Distributing Regulation

The role of the regulator can be distributed between multiple parties without changes to the rest of the system by using a threshold cryptosystem. In such a scheme, a set of n parties would be responsible for regulation, of which at least a threshold number k must cooperate to decrypt an encrypted identity. To set up the system, the regulator parties would run a key generation protocol that creates a public key and distributes shares of the corresponding secret key to the parties. The created public key is then used as the regulator public key in our system.

Since we use Elgamal encryption in our system, which can be used for threshold encryption (e.g. [38]), the process of encrypting identities and creating proofs does not differ from the system described in Section 3.4. In order to decrypt the ciphertexts without reconstructing the shared secret key, the regulator parties then again need to run a decryption protocol (e.g. [39]).

Table 2. The latency given the number of validators and batch size (from [26]). Estimated values for the throughput given the verification model. Batch size would correspond to the block size in our system.

#	Validators	Batch	Latency	VM1	VM2
4 (1 region)	6.2MB	0.288s	2000 tx/s	3400 tx/s	
8	1.6MB	0.58s	250 tx/s	420 tx/s	
8	6.2MB	1.48s	390 tx/s	670 tx/s	
16	1.6MB	0.69s	210 tx/s	360 tx/s	
16	3.1MB	1.04s	280 tx/s	480 tx/s	
32	0.4MB	0.48s	80 tx/s	130 tx/s	
32	1.6MB	0.925s	160 tx/s	270 tx/s	
64	0.4MB	0.824s	40 tx/s	70 tx/s	
64	1.6MB	1.79s	80 tx/s	140 tx/s	

C Consensus Performance

The validators also need to run a consensus protocol to agree on the set of transactions that are published to the ledger. We use the measurements from Croman et al. [26] to estimate the performance of a standard consensus protocol (PBFT [25]). As their measurements become bandwidth bound with larger batches, we can use their numbers for latency and throughput to estimate the throughput in our system if the batch size in bytes remains the same.

The sizes of our proofs are summarized in Table 1. Note that the transaction outputs can be reconstructed from their range proofs, i.e., the size for range proofs includes the transaction output. Since a normal transaction, consisting of multiple inputs and two outputs with an attached value, has a size of approximately 10.8kB (anonymous outputs) or 9.0kB (non-anonymous outputs) including all proofs, and the numbers from [26] use 190 byte transactions, the throughput in terms of transactions per second (tx/s) has to be adjusted conservatively by a factor of 0.018 if we require consensus on all proofs (**VM1**). In a deployment where no consensus on the validity of the regulation proofs is required (**VM2**), the transaction size reduces to approx. 4.0kB (anonymous) or 6.3kB (non-anonymous), i.e., we conservatively adjust by a factor of 0.030.

The achieved values for throughput and latency for the two verification models are shown in Table 2. For example, with 16 validators and a batch size of 3.1MB, the latency of consensus is 1.04s and the throughput is 480 tps (**VM2**) and 280 tps (**VM1**), respectively. Note that the measurements on which our estimations are based, were conducted with nodes that were globally distributed in 8 regions (except for the 4 node experiment). Croman et al. [26] used t2.medium Amazon EC2 instances which have limited bandwidth. If the validators are geographically close and have a higher bandwidth or dedicated lines between them (which would be reasonable for a digital fiat currency), the throughput could be increased and the latency could be reduced further.