Analysing and enhancing the privacy, security and fault-tolerance of India's Unified Payments Interface (UPI)

Prashant Agrawal[1], A Mukundan[1], Subodh Vishnu Sharma[2], Subhashis Banerjee[1,3]

[1]Centre for Digitalisation, AI and Society, Ashoka University
[2]Department of Computer Science and Engineering, IIT Delhi
[3]Department of Computer Science, Ashoka University

Abstract

The UPI has revolutionised India's digital payments ecosystem, enabling seamless interoperability across banks and allowing users to instantly transfer money using mobile devices and QR codes. However, despite its widespread adoption and proven ease of use, UPI lacks a clearly articulated threat model and a thorough analysis of its fault-tolerance, privacy, and security properties. We identify a comprehensive set of threats and requirements relevant to UPI-like real-time bank-to-bank payment systems, provide an abstract operational model of UPI, evaluate UPI against our idealized requirements, and compare it with other existing payment systems. We find that while UPI demonstrates strong interoperability and convenience, it relies on a centralised trusted intermediary and offers weak privacy protection and dispute resolution, leaving users vulnerable to transaction failures and fraud. We find similar limitations in other popular payment systems too. Unlike previous studies that primarily address unauthorised transactions, our work presents the first comprehensive analysis of privacy, security, and fault-tolerance of UPI and other real-time payment systems. We also propose two novel and practical cryptographic protocols, uniquely designed to resolve transaction disputes and enhance privacy without increasing the cognitive overload for the end-users.

CCS Concepts

• **Security and privacy → Systems security**.

Keywords

Payment Systems, UPI, Dispute Resolution, Privacy

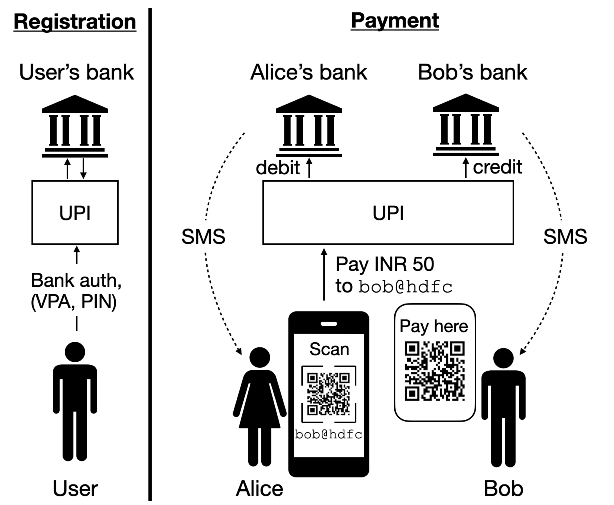

Figure 1: The UPI payment system at a high level. *Registration:* Users register on UPI by supplying their banking credentials and choosing a VPA and a 6-digit secret PIN. *Payment:* To pay INR 50 to Bob, Alice scans a QR code containing Bob's VPA `bob@hdfc` and sends a payment request to the UPI servers, supplying her own VPA and the secret PIN for authentication. The UPI servers resolve Alice and Bob's VPAs to their respective bank accounts and send matching debit and credit requests to their banks. The banks process the request immediately and notify Alice and Bob through SMS.

1 Introduction

The Unified Payments Interface (UPI) is the backbone of India's retail payments system. It offers a common infrastructure for simple mobile applications that can perform instant bank-to-bank transactions. To this end, it allows users to create a simple username, called a virtual payment address (VPA) or UPI ID, that is linked to their bank accounts. Users can then perform all payments using this simple unique ID, without having to sharing sensitive or cumbersome credentials. A typical UPI peer-to-merchant (P2M) payment goes as follows: the customer scans a printed QR code containing the UPI ID of the merchant and makes the payment through their mobile phone. This payment is immediately reflected in the bank accounts of both parties (see Figure 1).

The UPI is a substantial and critical payments system. It accounts for 46% of all digital payment transactions in the world [3]. As of March 2025, more than 18 billion transactions were made using UPI per month [45], with 661 Indian and international banks being live members [44] and over 40 mobile apps operating as UPI payment apps [43]. The UPI ecosystem has also evolved beyond the simple use-case of peer-to-merchant and in-person transactions, and many other modes of payment — peer-to-peer transactions, online transactions, recurring payments, refunds, etc. — are all supported via a coherent unified interface. In India, UPI is often considered to be a key enabler of financial inclusion [6, 50]. There are also numerous international collaborations, with 30 countries already engaged in some form of UPI integration [37].

Although the popularity and scale of the UPI are unprecedented, it does not have a well articulated and publicly available threat model, hence even its correctness, security and privacy objectives remain unclear. Much of UPI's design is proprietary and closed-source, which limits public scrutiny. On the other hand, well-founded concerns exist over both the security and the privacy aspects of UPI [1, 2, 32, 33, 54, 55, 57–60].

On the security front, Kumar et al. [32] noticed various vulnerabilities in an earlier version of the UPI (v1.0). In the worst case, this version could allow an adversary to empty the bank account of an individual who was not a UPI user at all. Although the main attack has been fixed in the current version of UPI (v2.0), other vulnerabilities remain [32]. Further, payment disputes and reliability issues are widely and frequently reported [1, 59]. UPI-based frauds utilising various social engineering tactics have also surged sharply in recent times [33, 54].

On the privacy front, a central issue is the usage of a unique VPA in all of a user's transactions, which makes multiple transactions by the same user linkable to each other. Such linking is well-known to have the potential of unlawful surveillance and profiling [16]. In fact, the VPA is often just the user's phone number, which significantly increases surveillance and cybersecurity concerns. The UPI architecture also distributes sensitive banking details and transaction information to several intermediate entities between the user and their bank, increasing the overall attack surface.

It is true that similar concerns exist with most other popular payments systems too. Most notably, the issue of clean dispute resolution (when payer and payee beliefs about a transaction diverge due to unreliable networks) remains largely unexamined, although prior research has focused on unauthorised transactions. In retail settings, where quick transaction finality guides decisions to pay or deliver goods, network uncertainty can cause financial loss with limited recourse.

In this paper, we analyse these privacy, security and fault-tolerance aspects of UPI and other payment systems and propose novel solutions. Specifically, we make the following main contributions:

- We first identify a comprehensive set of security, privacy and usability requirements for real-time bank-to-bank payment systems like the UPI (§3). Our threat model extensively addresses previously overlooked issues of tolerance to network failures, dispute resolution, and privacy.
- We present the first comprehensive overview of UPI, focusing on its user registration and payment transaction workflows (§4). Note that UPI's design is currently closed-source, and publicly available information is fragmented across various unofficial sources. To provide a structured protocol description suitable for academic analysis, we consolidate two key materials: 1) a now-archived version of UPI's procedural guidelines [42] previously published by the National Payments Corporation of India (NPCI), the governing body of UPI, and 2) an unofficial API specification document [41] of the UPI protocol published on a third-party file-sharing website. We acknowledge that our analysis applies to the real UPI system only insofar as it aligns with these sources.[1]
- We present a systematic evaluation of UPI and three other popular payment systems — credit/debit card systems, wallet-based systems, and digital cash — against our identified requirements (§5).[2] We find that UPI as well as the other systems suffer from poor fault-tolerance, privacy protection and trust management (Figure 4).
- We propose two novel payment system designs to fill the above gaps while balancing usability: one optimised for in-person transactions, and the other optimised for internet-based transactions (§6). We provide security proofs for our proposals in Appendices A and B. The proposals add minimal cognitive overload on the users and are efficient, as shown by our implementation of their critical components [10].

[1]We have reached out to the NPCI, multiple times, for confirming the authenticity of these API specifications. We have not heard back after an initial response in December 2024. We have also submitted a draft of this manuscript for their comments, but have not heard back as yet.

[2]Since we do not have access to the official specification of UPI, we limit our evaluation to only architectural issues and ignore implementation aspects.

2 Related work

Survey papers on payment systems. Many surveys of payment systems exist [5, 7, 29, 34, 56]. Typically, they taxonomise payment systems with respect to traditional security properties like confidentiality, integrity, authentication, etc. However, none of these analyse payment-related disputes or the issue of trust on the centralised payment system provider. In addition, none of them explicitly address the UPI's architecture.

Other payment systems and their security analyses. Card-based payment systems are classified into card-present and card-not-present variants, based on whether physical possession of the card is required. Various man-in-the-middle attacks, pre-play attacks and relay attacks enabling unauthorised transactions from the payer's account have been identified in EMV-based card-present systems [4, 13, 40]. Basin et al. [12] also point to an attack where the payee is tricked into accepting a payment that would later be declined by the banks. Similar issues, along with additional cardholder authentication issues, have been identified with the card-not-present systems like 3DSecure [31, 39], SET [31] and CAP [21], and with payment systems based on the near-field communication (NFC) technology [25, 26, 28, 38, 52, 53]. Overall, these works focus on unauthorised transactions, mis-authentications, and sometimes privacy, but they generally ignore issues of dispute resolution and centralised trust. Blockchain protocols like atomic swaps [30], payment channels [49], and zero-knowledge payment systems [36] provide strong decentralisation and dispute resolution properties, but they inherently operate in a blockchain-native setting. Thus, they do not directly address the requirements of real-time, bank-to-bank, sovereign currency payments like UPI, which demand legal tender settlement and instant confirmation for retail use cases.

Analysis of UPI. Kumar et al. [32] conducted the sole security analysis specifically directed at the UPI,[3] by reverse engineering UPI 1.0's security features. Through this, they identify attacks exploiting UPI's registration process that could allow an attacker to empty a victim's bank account. The main attack described by Kumar et al. has been fixed in UPI 2.0, but other vulnerabilities remain. For instance, attackers can still register UPI accounts against a victim's bank account without obtaining their linked SIM card [32]. Our work is broader in scope, and is based on an abstract model of UPI. It expands the range of threats under consideration, and anchors itself in a comparative exercise with other payments systems.

[3]Singh et al. [55] highlight scams and social engineering attacks in UPI, but their focus is on administrative issues for better management of dispute claims.

3 Idealised design requirements

We describe the design requirements for a real-time bank-to-bank digital payment system. We consider a system involving a payer P_0, a payee P_1, the payer's bank B_0, the payee's bank B_1, and a set of intermediaries facilitating the settlement of balances between P_0 and P_1's bank accounts. P_0 and P_1 interact with the payment system via a mobile app. We also refer to a trusted third party (TTP) or regulator R for identifying fraudulent parties and resolving disputes.

3.1 Trust assumptions

With respect to correctness, we assume that B_0 and B_1 are trustworthy, as banks are strictly regulated and audited entities. R is also trusted for correctly following protocol, fraud detection and dispute resolution. However, both P_0 and P_1 are untrusted as they may attempt to defraud each other or their banks. Further, since the payment-system intermediaries may not be as strictly regulated as the banks, neither they nor any of their sub-components are trusted for transaction correctness.

With regard to privacy, we assume all entities to be untrusted. Banks and the payment-system intermediaries may attempt to create detailed profiles of users based on their transaction patterns. Merchants acting as payees in payment transactions may want to extract sensitive personal information of their customers for marketing purposes. In addition, various internal or external adversaries may try to learn transaction details to launch future social engineering attacks on unsuspecting users.

The network is considered unreliable and fully observed and controlled by the adversary. Nevertheless, we assume that it is *partially synchronous* [22]: although there may be periods of network partitions where the parties may not be able to communicate with each other, each party must eventually get a chance to communicate after a finite (but unknown) amount of time. This is the standard model for internet-based systems and captures the temporary network glitches experienced by the parties at payment time.

3.1.1 Unmodelled threats. We now discuss some threats that we do not model in our analysis:

App compromises. We assume that the apps of honest users work as per the protocol and do not leak any sensitive data. In other words, we do not model risks posed to users due to malicious or compromised apps. These risks can be mitigated (albeit not eliminated) by employing a trusted core within the app and establishing a secure channel between this core and the user, by sending important information through alternate channels (like email), or by requiring secondary authentication factors outside the device.

Control-flow and side-channel information leakage. We do not model threats to user privacy due to leakage of control flow or side-channel information, e.g., IP addresses, DHCP logs, timing information, device details recorded during registration, etc. These threats are inherently difficult to eliminate due to their dependence on external infrastructure beyond the protocol's scope. Although anonymising networks [20] and strict access control at network providers' infrastructure can reduce exposure, complete mitigation of such side-channel privacy risks remains challenging.

3.2 Security requirements

Under the stipulated trust assumptions, we highlight our security requirements below. We indicate the party protected by the requirement as a subscript and explain the abbreviation using underlines.

(NU$_{P_0}$) No unauthorised transactions. Transactions must not take place from a user's bank account without their explicit authorisation, and the details of such transactions must correspond to the user's instructions. Attacks against this property can arise from malware on the user's device (even if the payment app itself remains uncompromised) or from compromised intermediaries.

Dispute resolution. With respect to disputes that may arise at the time of transactions, we require that, even on an unreliable network: 1) both P_0 and P_1 can arrive at a common belief about the transaction's particulars and its status within a reasonable period of time, and 2) this belief should be final: P_0 and P_1 should be able to ensure that their banks' eventual view about their account history matches this belief, irrespective of how the other party behaves. Thus, within a reasonable time period, the following finality and provability properties must hold:

(FF$_{P_0}$) Failure finality for P_0: If an honest P_0 does not receive the good from a potentially malicious P_1 after having initiated the transaction, there should be a way of guaranteeing that no funds are debited from P_0's account. Without this property, P_0 will have no recourse if P_1 withholds the relevant good after receiving the money. P_1 may actually encourage P_0 to make the payment again, citing transaction failure, and P_0 may believe it for the lack of a success receipt on their app due to network issues. In reality, though, the transaction might have succeeded, making P_0 lose funds through double payment.

Although this property may be unsatisfiable in general, the payment setting may render a reasonable assumption under which it could be satisfied. For example, in §6.1, we consider the in-person setting where P_0 and P_1 are physically together (e.g., in a store). The property is satisfied under the reasonable assumption that P_1 does not, after receiving the money, blatantly refuse service to P_0 without giving any reason. This assumption models P_1's desire to appear cooperative and exists in cash transactions too.

In internet-based transaction settings, such assumptions may not be reasonable. Because of the remote and anonymous nature of the transaction, P_0 is vulnerable to P_1 vanishing or refusing to honour a duly credited payment, and P_1 may have no incentive to cooperate. Thus, we also define the following weaker property.[4]

(***EDR$_{P_0}$***) ***External dispute resolution for P_0: If an honest P_0 successfully makes a payment to P_1 then P_0 should be able to provide a proof of completeness of the transaction to R and allow R to identify P_1's bank identity.*** With this guarantee, legal proceedings against P_1 may be initiated. This proof should be independently verifiable by R and not require trust on P_0 or P_1's apps or any intermediaries. P_0 is protected if R holds the power to

[4]Of course, the reverse threat of P_0 going missing after receiving a good is unavoidable and thus remote transactions must require the payer to initiate payment first, as is current widespread practice. We ignore this issue.

reverse fund transfers if P_1 is found guilty.

(SP$_{P_0}$) Success provability for P_0: If an honest P_0 believes that a transaction was successful, it should be able to convince an honest P_1 that the money would eventually be credited to P_1's account. Without this property, P_0 may obtain a transaction success notification, but this may not be enough to convince P_1 that the money would be eventually credited to their account irrespective of what P_0 does. This creates an impasse between P_0 and P_1 where P_0 does not wish to make another payment, and P_1 does not wish to deliver the agreed good.

(SF$_{P_1}$) Success finality for P_1: If an honest P_1 believes that a transaction is successful, there should be a guaranteed way of ensuring funds are credited to P_1's account. Without this property, P_1 may believe that the transaction succeeded and deliver the good to P_0, but have no way to get the money credited into their account. This may happen if P_0 tries to defraud P_1 by simultaneously reverting the transaction, or if B_1 makes additional checks that P_1 cannot make without a reliable connection.

(FP$_{P_1}$) Failure provability for P_1: If an honest P_1 believes that a transaction is unsuccessful, it should be able to convince an honest P_0 that funds would not later be debited from P_0's account. Without this property, there may obviously be an impasse.

Note that other combinations, i.e., $\mathbf{SF}_{P_0}$, $\mathbf{FP}_{P_0}$, $\mathbf{FF}_{P_1}$, and $\mathbf{SP}_{P_1}$, are uninteresting for a payment system.

Preventing frauds against banks. Now we discuss properties required to prevent frauds against banks by malicious P_0 and P_1.

(SD$_{B_0}$) B_0 must sufficiently debit P_0's account to cover its credit liability. B_0 must be guaranteed that the amount it may have to pay to other entities during a transaction is covered by the amount it debits from P_0's account. Otherwise, e.g., P_0 may, after making a payment to P_1, instruct B_0 to abort the transaction. This would cause B_0 to owe money to P_1's bank that it did not debit from P_0.

(LC$_{B_1}$) B_1 must limit the credit it makes to P_1's account to what it receives. B_1 should not be tricked into crediting a sum to P_1's account without some P_0 actually making a payment of the claimed amount to P_1, and thus without B_0 paying this sum to B_1. B_1 should also not be fooled into paying P_1 twice (a *double-spending attack*).

3.3 Privacy requirements

(PB) Privacy of users' sensitive banking details. Users' sensitive banking details (account numbers, bank-verified names, photographs, phone numbers, date-of-birth, residential addresses, etc.) must not leak to third-parties in the payment system [16]. Of course a user's bank is already privy to this information. But there is no need for intermediaries in the payment system, the counterparty in a payment transaction, their banks, and other external observers to learn this information. Note that this property protects information beyond authentication secrets like debit-card pin, one-time password (OTP), etc., which are already protected by property $\mathbf{NU}_{P_0}$.

(PI) Privacy of users' non-banking identity information. The payment system should also not leak users' non-banking identity information such as phone numbers, email addresses, digital identifiers, etc., to the payment system intermediaries and the payer/payee in a transaction. Given that these identifiers are often available on various other public databases, this leakage allows organisations to create detailed profiles of users by linking their financial transactions with these databases. Leaking phone numbers makes users particularly vulnerable, as it enables scammers to communicate with the users and manipulate them.

(PT) Privacy of users' transactions. Details of the transactions — payer and payee identitities, transaction amount, etc. — should not get leaked to parties that do not strictly need it. Such information enables surveillance and makes users vulnerable to fraud. Even if the identity of the users is hidden, partial information leaked from different transactions made by the same user may be correlated together to snoop into the user's financial activities [36].

3.4 Usability desiderata

We now mention some desirable usability features that considerably improve the system's adoptability for day-to-day retail transactions:

(BB) Direct bank-to-bank transfer. The payment system performs a direct bank-to-bank transfer from the payer's bank account to the payee's bank account at the end of a transaction. Systems where the money is held in a separate wallet do not provide the same kind of liquidity and trust as systems like UPI that directly put the money in a traditional, regulated bank account [61].

(WH) Usability without special hardware. It works without requiring the payer or the payee to own any special hardware, smartcards, card-reader machines, etc. Given the penetration of smartphones, though, it may be fine if users need to carry these phones for payment.

(RT) Real-time transaction finalisation. It is real-time and, assuming no network outages, communicates the final status of the transaction within a minute or so.

(WI) Usability without any internet. It does not require the payer or the payee to have *any* internet at the time of payment. It may, though, require eventual access to the internet to synchronise messages. Note that properties $\mathbf{FF}_{P_0}$-$\mathbf{FP}_{P_1}$ already imply that the payment system is safe to use without *reliable* internet, but this property goes one-step ahead and requires it to be usable without any internet at payment time.

4 The UPI operational model

We now give an abstract operational model of the UPI 2.0 infrastructure [41, 42]. We begin by introducing the parties involved in UPI in §4.1, and then describe the registration and payment phases of UPI in §4.2 and §4.3 respectively.

4.1 Parties

The UPI payment system comprises of a payer P_0, payee P_1, their banks B_0 and B_1, and the following intermediaries:

National Payments Corporation of India (NPCI). NPCI governs the UPI and facilitates all UPI transactions. It is a non-profit organisation owned by a consortium of public and private sector banks and operates under the oversight of the Reserve Bank of India (RBI), the central bank of India.

PSPs. PSPs are entities authorised to send and receive messages on behalf of the users to the NPCI. Since PSPs must collect and transmit users' sensitive financial information, only institutions regulated by the RBI, typically banks, are allowed to serve as PSPs.

TSPs. Although PSPs are capable of handling users' sensitive financial information, they may not always be best suited to maintaining consumer-facing applications. Thus, the NPCI allows PSPs to outsource certain functions to third-party Technology Service Providers (TSPs). TSPs (e.g., PayTM, GooglePay, PhonePe) provide mobile apps through which users perform UPI transactions.[5] TSPs are only allowed to communicate with the NPCI through their associated PSPs and are prohibited from accessing users' sensitive credentials. A user's PSP is the PSP associated with their app's TSP.

We denote P_0/P_1's PSP and TSP by $\mathsf{PSP}_0/\mathsf{PSP}_1$ and $\mathsf{TSP}_0/\mathsf{TSP}_1$. We present the general case where all the above parties are distinct and ignore otherwise simplified workflows [42].

4.2 Registration

The registration process of a new user P onto the UPI platform consists of the following main phases (see Figure 2).

4.2.1 Phase 1: Setting up device fingerprint and a VPA. To participate in UPI, P must download a TSP app. Usually a passcode or biometric is required to log in to the app, but the NPCI does not mandate a specific mechanism here.

The UPI protocol begins by linking a user's phone number with their physical mobile device. The user provides the TSP app their phone number ph, on which they receive an OTP via SMS. P must enter this OTP on the TSP app, which sends the OTP along with P's device details dev — device's IMEI number, device ID, App ID, etc. — to their backend server. The TSP server verifies the OTP and stores the tuple $(\mathsf{dev}, \mathsf{ph})$, aka the *device fingerprint.*

Next, the user's UPI ID (VPA) is chosen and linked to their registered phone number. The TSP server generates a unique VPA uid for P and sends it along with ph to its PSP. The VPA is in the format `name@psp`, where `name` is a unique username and `psp` identifies the PSP. The TSPs usually choose a default username from the users' phone number or email address. The PSP stores the tuple $(\mathsf{ph}, \mathsf{uid})$.

4.2.2 Phase 2: Identifying user's linked bank account. This phase is about identifying the bank account against which ph is registered.[6] For this, the TSP (via the PSP) raises a request to the NPCI to list all the UPI-compatible banks. The list is relayed back to the user, who selects the bank B where they maintain the account linked to ph.

To identify the specific account within this bank, the TSP (via the PSP) now shares (ph, B) with the NPCI. The NPCI requests B to fetch the details of the account linked with ph. If such an account exists, the bank shares the account details bid (account number, branch code, account owner's name and other details) to the NPCI. The NPCI relays it to the TSP and the user such that they only see a masked version of the account number (ma) — where only its last four digits are visible — whereas the PSP and the NPCI retain the full account details and store it against P's VPA. P must confirm, based on this masked number, if bid is their intended account.

4.2.3 Phase 3: Setting up a UPI pin. The final registration step is to verify whether P actually owns the account identified by bid and if so, to setup a UPI-specific secret PIN. The TSP app first asks P to enter authentication secrets bs associated with the account bid: last six digits of their debit card number, CVV, expiry date, etc., along with an OTP they receive from B. The app also asks P to set a UPI PIN us, which may be a 4 or 6-digit numeric credential.

The NPCI requires that TSP apps use the UI provided by an NPCI-developed common library to obtain sensitive data such as the bank credentials and the UPI PIN. The common library performs an encryption of the sensitive data on-device (against NPCI's public key $\mathsf{pk}_{\mathsf{NPCI}}$) and returns it to the app. The app then forwards these encryptions to the PSP servers.

The PSP requests the NPCI to associate this PIN with the user, forwarding it along with other encrypted banking information. The NPCI decrypts these ciphertexts and encrypts them again for B. B verifies the secrets bs and stores the UPI pin against the account bid, returning a confirmation. This pin acts as a second factor of authentication for authorising transactions on UPI (the first being the device fingerprint stored at the TSP servers).

4.3 Payment

We now describe the payment transaction between the payer P_0 and the payee P_1. Before the transaction, P_1 somehow needs to give their VPA uid_1 to P_0. This is typically done by P_0 scanning a QR code containing uid_1 and entering the payment amount m (a push transaction). Alternatively, P_1 can send a "collect request" to P_0 via P_0's TSP app, which displays uid_1 and m to P_0 (a pull transaction). We focus on the push transaction workflow (see Figure 3); the pull transaction works similarly with minor adjustments. Other workflows — payments against mobile number, Aadhaar number, account number, etc. — are also minor variations.

4.3.1 Phase 1: Payee validation. The first step is a validation of the recipient of the transaction by P_0. For this, the payee VPA uid_1 entered by P_0 is resolved to P_1's bank-verified name and relayed back to P_0 for verification. P_0 verifies whether the recipient's name matches their intent and enters the payment amount m if satisfied.

4.3.2 Phase 2: Transaction initiation. In this step, P_0 is authenticated and the final transaction details are resolved by the NPCI. P_0's TSP app computes P_0's first authentication factor — its device fingerprint — and the TSP server verifies it. If successful, the TSP server shares with PSP_0 the amount m, payee VPA uid_1 as confirmed by P_0, and P_0's verified device fingerprint. The app also uses the NPCI common library UI to obtain P_0's second authentication

[5]The NPCI does maintain a common library for these applications, but the TSPs are permitted a measure of innovation and variation.

[6]The UPI requires that a user's mobile number be linked to their bank account.

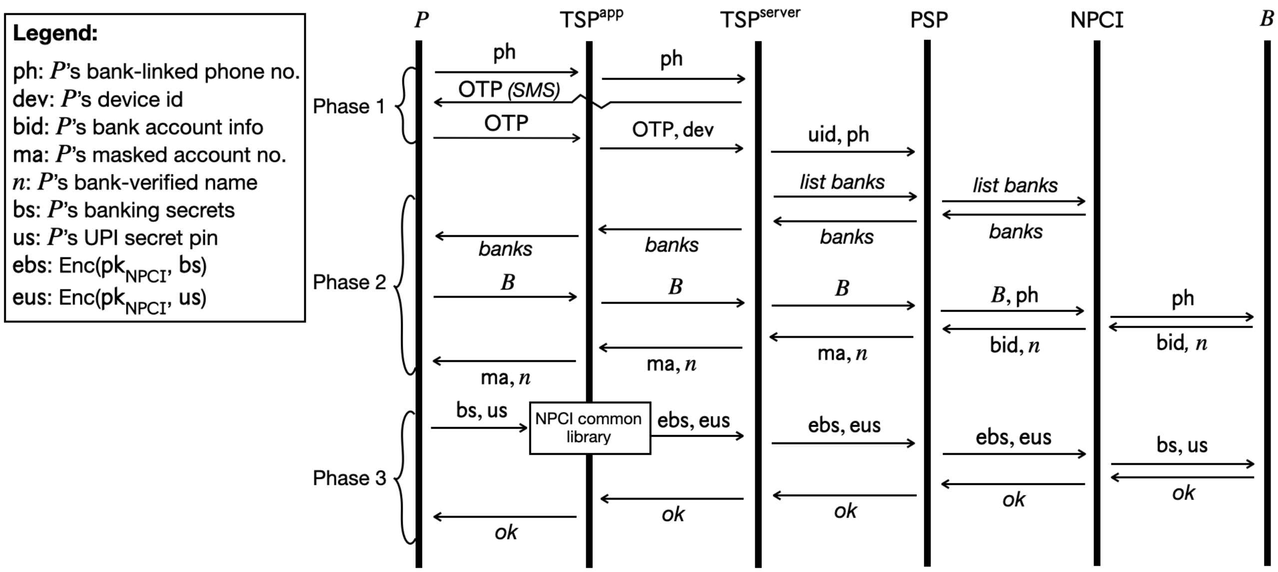

Figure 2: Registration of a fresh user P onto the UPI platform.

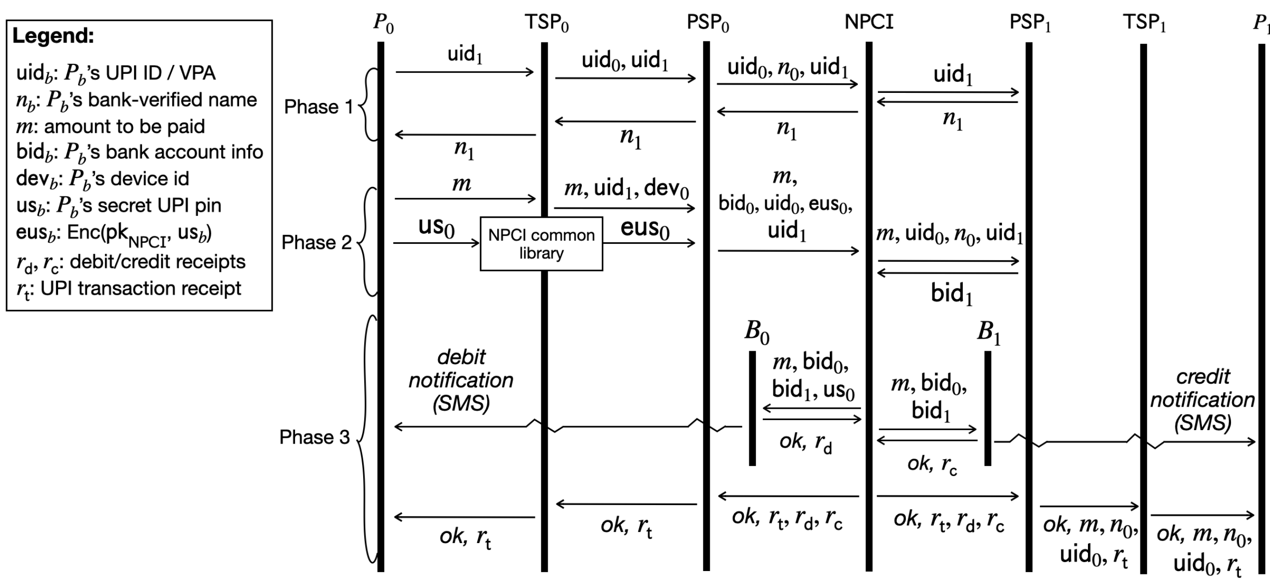

Figure 3: A UPI payment transaction (the so-called "push transaction").

factor: their secret UPI pin. The encrypted UPI pin (against $\mathsf{pk}_{\mathsf{NPCI}}$) obtained from the common library is sent to PSP_0.

With this information, PSP_0 sends a payment request to the NPCI, containing the amount m, P_0's identification details (stored during registration), P_0's encrypted UPI pin, and the payee's VPA uid_1. The NPCI first translates the payee's VPA into its bank account details by sending a request to PSP_1 that also contains P_0's identity information (to be used by PSP_1 during later confirmation to P_1). PSP_1 returns the account details bid_1 linked to uid_1.

4.3.3 Phase 3: Transaction finalisation. The NPCI can now formally attempt the transaction. It first sends a debit request to B_0 with the amount m, the account details of P_0 and P_1, and P_0's decrypted UPI pin. B_0 verifies that the given pin matches the UPI pin stored against bid_0 during registration. It then runs several regulatory checks, including whether P_0 maintains sufficient balance. If the checks pass, B_0 makes a debit entry in favour of bid_1 and sends a debit receipt r_d to the NPCI. B_0 may also send P_0 a direct SMS notification.

Given r_d, the NPCI sends a credit request to B_1 with the amount m and the account details of P_0 and P_1. B_1 performs the relevant credit operation towards bid_1 and returns a credit receipt r_c. B_1 may also notify P_1 through an SMS. B_0 and B_1 settle the aggregate balance among themselves during a separate clearinghouse step.

If NPCI does not receive both r_d and r_c within a timeout, it attempts to revert the transaction by repeatedly contacting both B_0 and B_1 till a consistent view is obtained. (Thus, the bank SMS notifications do not constitute a final confirmation for P_0 or P_1.) If both r_d and r_c have been received, the NPCI generates a final UPI transaction receipt r_t and sends a final confirmation $(r_\mathsf{d}, r_\mathsf{c}, r_\mathsf{t})$ to PSP_0, who relays r_t to P_0. The NPCI also sends $(r_\mathsf{c}, r_\mathsf{t})$ to PSP_1, who relays r_t along with (m, n_0, uid_0) to TSP_1 as P_1's final confirmation.

5 Analysis of UPI

We now analyse the aforementioned UPI architecture with respect to the threat model and usability desiderata highlighted in §3. We capture this analysis in Figure 4.

UPI's excellent usability is already well-established and proven by its widespread adoption [45]. Our analysis highlights its key reasons: UPI supports direct bank-to-bank transfer (**BB**), can be used without any special hardware beyond commodity smartphones (**WH**), and allows real-time transaction finalisation (**RT**). The lack of the payee's involvement during push transactions also makes day-to-day retail payments particularly convenient. It is an online payment system, though, and cannot work without the internet (no **WI**).[7] We now discuss the security and privacy threats in UPI.

Trust on NPCI for correctness and privacy: The first central issue is complete trust on NPCI both for correctness and privacy of transactions, making it the weakest link. Thus, under our threat model where no payment-system intermediaries are trusted, UPI does not satisfy any of the security or privacy requirements (row 1). However, since NPCI is a consortium of regulated banks and is itself under RBI's oversight, trusting it at least for transaction correctness may be a reasonable assumption in practice. We thus analyse UPI in this case too (row 2). We find that UPI suffers from the following weaknesses even in this relaxed model.

Potential for unauthorised transactions (partial $\mathbf{NU}_{P_0}$): Potential for unauthorised transactions exists during both registration and payment phases. Registration of a UPI account requires entering OTPs sent to the user's registered phone number, sending an SMS from this number, and entering the details printed on the user's debit card (but not any secret pin). As Kumar et al. [32] have shown, this can be easily broken as incoming SMS can be intercepted by malware on the user's phone, outgoing SMS can be spoofed [51], and debit card details are freely shared in India.

During transactions, the need to enter the secret UPI pin prevents such attacks. However, its secure and isolated input is equally important. The NPCI mandates the PSPs and TSPs to use the NPCI common library interface, but this puts a high level of trust in PSPs and TSPs as no technical safeguard seems to be in place. The NPCI common library also does not resist snooping by third-party malware via screen-overlaying, as shown by [32], and screen-capturing and key-logging attacks may also be possible. The NPCI decrypting users' bank-authentication secrets is also concerning.

Payment-related disputes and bank frauds ($\mathbf{FF}_{P_0}$ to $\mathbf{LC}_{B_1}$): Because of UPI's online nature, transaction failures and payment disputes are common. P_0 may not get a confirmation on transaction success within a reasonable time and thus believe that it failed, but money may have been debited from their account and they may not have any guaranteed way to revert the transaction (no $\mathbf{FF}_{P_0}$). Further, although the final NPCI confirmation on P_0's app confirms to P_0 that money has been debited from their account, this only serves as a weak proof for P_1 as screenshots could be easily manipulated (partial $\mathbf{SP}_{P_0}$). On the other hand, the final NPCI confirmation on P_1's app confirms to P_1 that money has been credited to their account and will not be reverted, since NPCI is trusted ($\mathbf{SF}_{P_1}$). But if the confirmation does not arrive in time due to network failures, P_1 cannot convince P_0 that the transaction failed (no $\mathbf{FP}_{P_1}$). The external dispute resolution property $\mathbf{EDR}_{P_0}$ holds because P_0 can complain to NPCI, who can verify the transaction status and identify the payee, but it comes with the cost of an always online TTP who learns about all of users' transaction details. Users cannot defraud the banks ($\mathbf{SD}_{B_0}$, $\mathbf{LC}_{B_1}$), because NPCI ensures eventual consistency between B_0 and B_1.

UPI also has poor privacy properties, as users' sensitive banking and non-banking details as well as transaction information is leaked to all intermediaries:

Leakage of sensitive banking details (no PB): Users' banking details like account numbers and bank-verified names are leaked to the PSPs and the NPCI, even though some information is hidden from the TSPs via the masked account numbers.

Leakage of identifiers like phone numbers, email addresses, and names (no PI): Phone numbers of users are shared with all intermediaries - the TSPs, PSPs and the NPCI. Further, since default VPAs of users is typically their phone number or email ID, it leaks their phone number/email username to the counterparty (P_1 for P_0; P_0 for P_1) in a payment transaction too. Both P_0 and P_1 also learn each other's bank-verified names. As these identifiers are often available from myriad other sources, leaking them make users particularly vulnerable to profiling and fraud. For example, the "collect request" fraud [57] exploits the fact that phone numbers are easily available and lead to valid VPAs. The attacker bombards fraudulent UPI collect requests against a VPA derived from a phone number, which appear as payment prompts in the VPA owner's UPI app. The attacker then calls the linked phone number to deceive the VPA owner into thinking that they are actually receiving money on UPI and need to provide their UPI PIN to receive it, when in fact this would deduct money from their account. Other frauds exploiting these identifiers also exist [58].

Leakage of all transaction details and linking of transactions via VPAs (no PT): Finally, transaction details like P_0 and P_1's banking identifiers, transaction amounts, etc., are leaked to both

[7] The NPCI has released an offline version of UPI called UPI Lite [46], which is an in-app wallet, but it does not provide any security features against tampering and is thus limited to very small value transactions upto INR 200.

Designs	**Usability**				**Security**								**Privacy**		
	BB	**WH**	**RT**	**WI**	$\mathbf{NU}_{P_0}$	$\mathbf{FF}_{P_0}$	$\mathbf{EDR}_{P_0}$	$\mathbf{SP}_{P_0}$	$\mathbf{SF}_{P_1}$	$\mathbf{FP}_{P_1}$	$\mathbf{SD}_{B_0}$	$\mathbf{LC}_{B_1}$	**PB**	**PI**	**PT**
UPI	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○
UPI (NPCI trusted)	●	●	●	○	◐	○	●	◐	●	○	●	●	○	○	○
Card-based (card network trusted)	●	○	●	○	●	○	●	●	●	○	●	●	○	●	○
Wallet-based (wallet service trusted)	○	●	●	○	●	○	●	◐	●	○	●	●	○	●	○
Online digital cash	●	●	●	○	◐	○	○	○	●	○	●	●	●	●	●
Offline digital cash (rational P_0)	●	●	●	●	◐	○	○	●	●	●	●	●	●	●	●
Our proposal in §6.1 (in-person; covert P_1)	●	●	●	○	●	●	○	●	●	●	●	●	●	●	●
Our proposal in §6.2 (offline TTP)	●	●	●	○	●	○	●	●	●	●	●	●	●	●	●

Figure 4: Evaluation of payment systems: ● denotes that the design satisfies the given requirement (under the assumptions given in parentheses), ○ denotes that it does not, and ◐ denotes that it does so only partially (see text for the specific interpretation).

the PSPs and the NPCI. All transactions are also labelled with the VPAs of the involved parties, which allows linking of transactions.

5.1 Comparison with other payment systems

We now turn to the literature to see if other existing payment systems provide reduced trust on intermediaries or better dispute resolution and privacy protection compared to UPI. We focus on major payment system designs that work in the setting of real-time bank-to-bank digital payments and ignore blockchain-based solutions, which are neither real-time nor bank-to-bank.

5.1.1 Card-based designs. Chip and PIN card-based systems like VISA and Mastercard rely on the EMV protocol for authenticating and authorising card payments on a point-of-sale (POS) terminal [56]. The typical flow involves a customer entering their physical card on a merchant-owned card machine, which communicates with the card chip and sends a payment request over the card network. The card network authenticates the cardholder by asking them to enter a secret PIN (sometimes even a traditional pen-and-paper signature or tap is enough). The card network then contacts the cardholder's bank to check if they have sufficient balance or credit. The payment is processed and the card machine is notified if the transaction was successful, which then prints a receipt.

Card-based systems offer direct bank-to-bank transfer and real-time transaction capabilities (**BB**, **RT**), but, unlike UPI, rely on card and card machines (no **WH**), hindering their widespread adoption in the informal economy in India.

The security and privacy guarantees are not better than UPI. Like UPI, one must trust the card network for both correctness and privacy of transactions. Dispute resolution is also weak: receipts for successful transactions may not get printed due to network or machine failures, but this leaves P_0 with no recourse (no $\mathbf{FF}_{P_0}$) and P_1 with no way to convince P_0 of transaction failure (no $\mathbf{FP}_{P_1}$). If the receipt does get printed, though, it convinces P_1 to give the good to P_0 ($\mathbf{SP}_{P_0}$) and also lets P_1 convince B_1 to credit the money to their account ($\mathbf{SF}_{P_1}$). Unauthorised transactions are prevented (in principle) because the EMV chip encrypts all sensitive information ($\mathbf{NU}_{P_0}$). The system satisfies other security properties just like UPI. The $\mathbf{EDR}_{P_0}$ also holds in internet-based settings as in UPI, but at the expense of an online TTP and the loss of users' privacy. The card network completely identifies the users and their transaction details (no **PB**, **PT**). Users' non-banking identity details are mostly protected, though, as card transactions typically do not require users to enter any non-banking information (**PI**).

5.1.2 Wallet-based designs. In payment systems like Paypal, Google Wallet, and Apple Pay, a digital wallet on the cloud is pre-loaded by the user with money from their bank account. During the payment transaction, users instruct the wallet service (generally, through a mobile app) to transfer cash from their wallet to the payee's wallet. All accounting of users' wallet balances is done by the wallet service. At the end of the transaction, the wallet service notifies both the payer and the payee independently. At any time later, users may encash their wallet balance to real money.

The main hurdle in the adoption of wallet-based designs is the lack of a direct bank-to-bank transfer facility (no **BB**), requiring users to pre-load the wallet and be on the same wallet service. They do avoid specialised hardware, though (**WH**).

Security-wise, a high level of trust is placed on the centralised wallet service, which can potentially compromise the accounting of users' wallet balances. The dispute resolution properties are also similar to UPI and card-based systems. Notably, P_0 may not get transaction confirmation in time; if so, they cannot revert the transaction (no $\mathbf{FF}_{P_0}$). If P_1 does not receive confirmation in time, they cannot convince P_0 of transaction failure (no $\mathbf{FP}_{P_1}$). Further, because of independent notifications to P_0 and P_1 as in UPI, their beliefs can differ. This limits P_0's ability to convince P_1 because of plausibly fake screenshots (partial $\mathbf{SP}_{P_0}$). Privacy guarantees are also weak, like card-based designs, with the wallet service learning banking details of the users and their transactions (no **PB**, **PT**).

5.1.3 Digital cash. Chaum conceptualised the notion of digital cash [16]. This system is unlike most other payment systems — it enables direct bank-to-bank transfer without trusting any central entity, and provides very strong privacy properties. In digital cash, P_0 pre-downloads digital coins from their bank account to their local device. To protect their privacy, P_0 can then transform these coins to another set of coins that are completely unlinkable from them. For the payment, P_0 simply gives the transformed coins to P_1, who can encash the digital coins for real money later by contacting B_1. Because of unlinkability of coins, the bank(s) cannot link a user's encashment request with another user's withdrawal request.

There is a caveat though: unlike a physical coin that once spent cannot be spent again, a digital coin can always be copied and spent multiple times. We analyse two representative approaches to combat this double-spending attack. The first one, proposed in

Chaum's original paper [16], actually requires P_1 to check with B_1 that the coin is unspent before approving the transaction. The solution unfortunately fails to achieve the unique offline property of physical cash (no **WI**). The other solution, *offline digital cash* [17] avoids this check and works in offline mode, preventing double-spending by identifying and punishing double-spending payers while protecting the anonymity of honest payers.

Digital cash avoids trusted intermediaries and offers excellent privacy, as no identity information of the users is leaked and all their transactions are unlinkable (**PB**, **PI** and **PT**). However, payment-related disputes remain. Because the knower of the coin information is the owner of the coin, P_0 gets no guarantee of reclaiming a coin if a malicious P_1 obtains it but then refuses to accept it citing verification failure (no $\mathbf{FF}_{P_0}$). In the online version, P_0 cannot convince P_1 about the validity of a given coin, because network failures may prevent P_1 from making the double-spending check (no $\mathbf{SP}_{P_0}$). On the other hand, if P_1 does not hear from B_1 within a timeout, it cannot convince P_0 that P_1 cannot later encash this coin (no $\mathbf{FP}_{P_1}$). In the offline version, locality helps: both P_0 and P_1 can locally convince themselves and each other about the coin's validity ($\mathbf{SP}_{P_0}$, $\mathbf{SF}_{P_1}$, $\mathbf{FP}_{P_1}$), but prevention against double spending holds only if P_0 is rational and fearful of punishment. Further, the perfect anonymity properties of digital cash hurt dispute resolution: unlike previous TTP-based systems, $\mathbf{EDR}_{P_0}$ does not hold because P_0 has no provision to provably identify the recipient of a coin. Coins living in the memory of users' phones are also risky targets, as anyone learning them effectively steals them (partial $\mathbf{NU}_{P_0}$).

Summary. In summary, while systems like UPI, cards and wallets suffer from the issues of centralised trust, poor privacy and poor dispute resolution, digital cash avoids centralised trust and provides excellent privacy, but it still suffers from poor dispute resolution and also presents new attack surfaces of coin theft. We attempt to address these limitations below.

6 Solution recommendations

We now present our solutions to fill the above gaps, especially around dispute resolution and privacy. The solution in §6.1 works for an in-person setting, whereas the one in §6.2 works for internet-based transactions, addressing the threat of P_1 disappearing ($\mathbf{EDR}_{P_0}$).

6.1 In-person transactions

We propose a solution that assumes a synchronous communication channel between P_0 and P_1. Communication between other entities is assumed unreliable (partially synchronous) but secure. The synchronicity assumption is justified in an in-person setting where P_0 and P_1 are physically co-present and can communicate by, e.g., scanning QR codes. Further, we assume that although P_1 may try to cheat, they do it only in a covert way that avoids them getting caught. This assumption is also natural in an in-person business setting where P_1 has a stake in protecting their business reputation.

Our solution has a rough analogy with a traditional paper-based credit note or cheque. P_1 initiates the transaction by asking P_0 to get a credit note signed by B_0 against P_1's ID. This can be done by P_0 scanning a QR code from P_1's device (just as in the current UPI system). P_0 then requests this credit note from B_0 via an unreliable network. If the signed note is received within a fixed amount of time, P_0 allows P_1 to scan the note from P_0's phone. This scanning process marks the payment and closely resembles the atomic process of physically handing in the credit note: the money is in P_0's hands before this step and in P_1's hands after it. P_1 verifies the signature on the note locally using B_0's public key and gives the good to P_0. Later, P_1 can encash the note with B_1 via an unreliable network, retrying until it gets reflected in their account balance.

Now we discuss the system's fault-tolerance and dispute-avoidance features. First, unreliable network while P_0 fetches the credit note from B_0 is tolerated by P_0 initiating a *reclaim request* if they do not receive the note within a timeout. This request reclaims any money that might have been deducted nonetheless due to network failures. P_0 can asynchronously retry the request until it is acknowledged to ensure that any deducted money is eventually reclaimed.

Second, note that P_1 must verify the signature before giving the good to P_0, otherwise they may be fooled into accepting an unencashable note. However, if the verification fails and P_1 refuses the note, P_0 needs assurance that P_1 is not bluffing and cannot encash it nonetheless. Towards this end, we make P_1 give a *cancellation credential* to P_0 (again, via a QR code) that guarantees P_0 that they can eventually cancel the transaction. A valid cancellation credential allows P_0 to confidently initiate repayment. Note that P_0 is protected only under the assumption that P_1 gives either the good or a valid cancellation credential to P_0. The assumption is justified because P_1 has to appear reasonable to P_0 and potentially other observers physically present at the transaction location.

Third, B_0 and B_1 resolve all disputes correctly and consistently by maintaining the following two important invariants: 1) an encashment request always overrrides any past or future reclaim requests, guaranteeing P_1 that they can encash the note even if P_0 tries to reclaim a note already spent; and 2) a cancellation request always overrides any past or future encashment requests, guaranteeing P_0 that they can get their money back even if P_1 tries to encash a note after pretending to cancel it.

The banks are also protected against fraud by malicious P_0 or P_1 and can detect any double spending. There is a caveat here, though. Although B_0 always debits sufficient amount from P_0's account balance, it is possible for P_0 to use a credit note for a payment, reclaim it before P_1 encashes it, and then immediately empty their account. When P_1 makes the encashment request, B_0 loses money because P_0's effective balance is now negative. We consider this risk a small one given that 1) P_0 is completely identified by B_0 who may ban P_0 or initiate legal proceedings against them, 2) the transaction amounts in retail transactions are typically far smaller than users' account balances, and 3) the banks can impose an encashment deadline and also limit the amount and number of transactions.

The privacy guarantees of the proposal are as follows. The solution naturally protects P_0's banking and identity information because the credit note does not include any such information. For P_1, a fresh virtual identity — a commitment of their payment-system identifier — is revealed, which remains completely unlinkable to the latter. Further, if P_0 and P_1 are both honest, it is hard to distinguish whether they are transacting with each other or someone else, even if B_0 and B_1 are colluding with each other (assuming the amount m, which the banks necessarily know in a bank-to-bank payment system, does not help them distinguish). This is mainly

Protocol $\Pi_{\text{pay-in-person}}$ between P_0 (Payer), P_1 (Payee), B_0 (Payer's Bank) and B_1 (Payee's Bank):

$\underline{\mathsf{Register}(P_b\langle \mathsf{bid}_b, \mathsf{bs}_b, \mathsf{uid}_b, \mathsf{us}_b\rangle, B_b\langle\ldots\rangle):}$

$P_b \dashrightarrow B_b$: $\mathsf{bid}_b, \mathsf{bs}_b, \mathsf{uid}_b, \mathsf{us}_b$
B_b: assert $\mathsf{uid}_b \notin \mathsf{UID}$, $\mathsf{BS}[\mathsf{bid}_b] = \mathsf{bs}_b$
B_b: $\mathsf{UID} := \mathsf{UID} \cup \{\mathsf{uid}_b\}$; $\mathsf{US}[\mathsf{uid}_b] := \mathsf{us}_b$
B_b: $\mathsf{BID}[\mathsf{uid}_b] := \mathsf{bid}_b$

$\underline{\mathsf{Transact}(P_0\langle m, \mathsf{uid}_0, \mathsf{us}_0\rangle, P_1\langle \mathsf{uid}_1, m\rangle, B_0\langle \mathsf{sk}_{B_0}, \ldots\rangle, B_1\langle \mathsf{pk}_{B_0}, \ldots\rangle):}$

P_1: $\mathsf{tid}_1 \xleftarrow{\$} \mathbb{Z}_q$
P_1: $\mathsf{vid}_1, r_{\mathsf{vid}_1} \leftarrow \mathsf{GenVID}(\mathsf{uid}_1)$
P_1: $\pi_{\mathsf{vid}_1} \leftarrow \mathsf{NIZKPK}\{(\mathsf{uid}_1, r_{\mathsf{vid}_1}) : \mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}\}$
$P_1 \rightarrow P_0$: $\mathsf{tid}_1, \mathsf{vid}_1, \pi_{\mathsf{vid}_1}$
P_0: $\mathsf{tid}_0 \xleftarrow{\$} \mathbb{Z}_q$; $\mathsf{tid} \leftarrow H(\mathsf{tid}_0, \mathsf{tid}_1)$; $r_{\mathsf{tid}} \xleftarrow{\$} \mathbb{Z}_q$; $\mathsf{ctid} \leftarrow g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}$
P_0: $\pi_{\mathsf{ctid}} \leftarrow \mathsf{NIZKPK}\{(\mathsf{tid}, r_{\mathsf{tid}}) : \mathsf{ctid} = g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}\}$
P_0: assert $\mathsf{NIZKVer}(\pi_{\mathsf{vid}_1}, \mathsf{vid}_1) = 1$
$P_0 \dashrightarrow B_0$: ("IssueReq", $\mathsf{uid}_0, \mathsf{us}_0, m, \mathsf{ctid}, \pi_{\mathsf{ctid}}, \mathsf{vid}_1, \pi_{\mathsf{vid}_1}$)
B_0: $\mathsf{bid}_0 := \mathsf{BID}[\mathsf{uid}_0]$
B_0: assert $\mathsf{us}_0 = \mathsf{US}[\mathsf{uid}_0]$, $\mathsf{NIZKVer}(\pi_{\mathsf{ctid}}, \mathsf{ctid}) = \mathsf{NIZKVer}(\pi_{\mathsf{vid}_1}, \mathsf{vid}_1) = 1$
B_0: run local transaction issue(ctid):
  assert $\mathsf{ctid} \notin \mathsf{CTID}_{\mathsf{requested}}$, $\mathsf{BAL}[\mathsf{bid}_0] \geq m$
  $\mathsf{TXN}[\mathsf{ctid}] := (\mathsf{bid}_0, m, \mathsf{vid}_1)$
  $\mathsf{BAL}[\mathsf{bid}_0] := \mathsf{BAL}[\mathsf{bid}_0] - m$
  $\mathsf{CTID}_{\mathsf{requested}} := \mathsf{CTID}_{\mathsf{requested}} \cup \{\mathsf{ctid}\}$
B_0: $c, \hat{r} \xleftarrow{\$} \mathbb{Z}_q$; $S \leftarrow (g_0 g_1^m \mathsf{ctid}\ \mathsf{vid}_1 g_4^{\hat{r}})^{\frac{1}{c+\mathsf{sk}_{B_0}}}$
$B_0 \dashrightarrow P_0$: $\hat{\sigma} := (S, c, \hat{r})$
P_0: if $\hat{\sigma}$ not received till timeout τ:
  $\mathsf{Reclaim}(P_0\langle \mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}\rangle, B_0\langle\ldots\rangle, B_1\langle\ldots\rangle)$
P_0: else:
  P_0: $\tilde{r} := \hat{r} + r_{\mathsf{tid}}$
  $P_0 \rightarrow P_1$: $\tilde{\sigma} := (S, c, \tilde{r})$, $\mathsf{tid}, \mathsf{tid}_0$ // the "payment"
  P_1: $r := \tilde{r} + r_{\mathsf{vid}_1}$
  P_1: $\sigma := (S, c, r)$
  P_1: if $\mathsf{tid} = H(\mathsf{tid}_0, \mathsf{tid}_1)$ and $e(S, \mathsf{pk}_{B_0} h^c) = e(g_0 g_1^m g_2^{\mathsf{tid}} g_3^{\mathsf{uid}_1} g_4^r, h)$:
    $P_1 \rightarrow P_0$: "success" // give the good to P_0
    P_1: $\pi_\sigma \leftarrow \mathsf{NIZKPK}\{(S, c, r) : e(S, \mathsf{pk}_{B_0} h^c) = e(g_0 g_1^m g_2^{\mathsf{tid}} g_3^{\mathsf{uid}_1} g_4^r, h)\}$
    $\mathsf{Encash}(P_1\langle \pi_\sigma, m, \mathsf{tid}, \mathsf{uid}_1\rangle, B_1\langle\ldots\rangle, B_0\langle\ldots\rangle)$
  P_1: else:
    P_1: $\sigma_c \leftarrow \mathsf{SPK}\{(\mathsf{uid}_1, r_{\mathsf{vid}_1}) : \mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}\}(\mathsf{tid})$
    $P_1 \rightarrow P_0$: σ_c
    P_0: assert $\mathsf{SPKVer}(\sigma_c, \mathsf{vid}_1, \mathsf{tid}) = 1$
    $P_0 \rightarrow P_1$: "ok" // agrees to cancellation; potentially repays
    $\mathsf{Cancel}(P_0\langle \mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}, \sigma_c\rangle, B_0\langle\ldots\rangle, B_1\langle\ldots\rangle)$

$\underline{\mathsf{GenVID}(\mathsf{uid}):}$

$r_{\mathsf{vid}} \xleftarrow{\$} \mathbb{Z}_q$; $\mathsf{vid} \leftarrow g_3^{\mathsf{uid}} g_4^{r_{\mathsf{vid}}}$; return $\mathsf{vid}, r_{\mathsf{vid}}$

$\underline{\mathsf{Reclaim}(P_0\langle \mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}\rangle, B_0\langle\ldots\rangle, B_1\langle\ldots\rangle):}$

repeat in background until P_0 receives 1:
 $P_0 \dashrightarrow B_0$: ("ReclaimReq", $\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}$)
 B_0: assert $\mathsf{ctid} = g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}$
 B_0: $(\mathsf{bid}_0, m, \mathsf{vid}_1) := \mathsf{TXN}[\mathsf{ctid}]$
 B_0, B_1: run distributed transaction reclaim(tid):
  assert $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{reclaimed}} \cup \mathsf{TID}_{\mathsf{encashed}} \cup \mathsf{TID}_{\mathsf{cancelled}}$
  B_0: $\mathsf{BAL}[\mathsf{bid}_0] := \mathsf{BAL}[\mathsf{bid}_0] + m$
  $\mathsf{TID}_{\mathsf{reclaimed}} := \mathsf{TID}_{\mathsf{reclaimed}} \cup \{\mathsf{tid}\}$
 $B_0 \dashrightarrow P_0$: 1 if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{reclaimed}}$ else 0

$\underline{\mathsf{Encash}(P_1\langle \pi_\sigma, m, \mathsf{tid}, \mathsf{uid}_1\rangle, B_1\langle\ldots\rangle, B_0\langle\ldots\rangle):}$

repeat in background until P_1 receives 1:
 $P_1 \dashrightarrow B_1$: ("EncashReq", $\pi_\sigma, m, \mathsf{tid}, \mathsf{uid}_1$)
 B_1: assert $\mathsf{NIZKVer}(\pi_\sigma, (m, \mathsf{tid}, \mathsf{uid}_1), \mathsf{pk}_{B_0}) = 1$
 B_0, B_1: run distributed transaction encash(tid):
  assert $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{encashed}} \cup \mathsf{TID}_{\mathsf{cancelled}}$
  if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{reclaimed}}$:
   revert reclaim(tid)
  B_1: $\mathsf{bid}_1 := \mathsf{BID}[\mathsf{uid}_1]$
  B_1: $\mathsf{BAL}[\mathsf{bid}_1] := \mathsf{BAL}[\mathsf{bid}_1] + m$
  B_0: $\mathsf{BAL}[B_1] := \mathsf{BAL}[B_1] + m$
  $\mathsf{TID}_{\mathsf{encashed}} := \mathsf{TID}_{\mathsf{encashed}} \cup \{\mathsf{tid}\}$
 $B_1 \dashrightarrow P_1$: 1 if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{encashed}}$ else 0

$\underline{\mathsf{Cancel}(P_0\langle \mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}, \sigma_c\rangle, B_0\langle\ldots\rangle, B_1\langle\ldots\rangle):}$

repeat in background until P_0 receives 1:
 $P_0 \dashrightarrow B_0$: ("CancelReq", $\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}, \sigma_c$)
 B_0: assert $\mathsf{ctid} = g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}$
 B_0: $(\mathsf{bid}_0, m, \mathsf{vid}_1) := \mathsf{TXN}[\mathsf{ctid}]$
 B_0: assert $\mathsf{SPKVer}(\sigma_c, \mathsf{vid}_1, \mathsf{tid}) = 1$
 B_0, B_1: run distributed transaction cancel(tid):
  assert $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{cancelled}} \cup \mathsf{TID}_{\mathsf{reclaimed}}$
  if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{encashed}}$:
   revert encash(tid)
  B_0: $\mathsf{BAL}[\mathsf{bid}_0] := \mathsf{BAL}[\mathsf{bid}_0] + m$
  $\mathsf{TID}_{\mathsf{cancelled}} := \mathsf{TID}_{\mathsf{cancelled}} \cup \{\mathsf{tid}\}$
 $B_0 \dashrightarrow P_0$: 1 if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{cancelled}}$ else 0

Figure 5: Our proposed payment protocol for in-person transactions. All algorithms have implicit access to q, generators $g_0, g_1, g_2, g_3, g_4 \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, and, for each $b \in \{0, 1\}$, public key $\mathsf{pk}_{B_b} := h^{\mathsf{sk}_{B_b}}$ of B_b against its secret key $\mathsf{sk}_{B_b} \in \mathbb{Z}_q$. Communication of P_0 and P_1 with the banks, denoted $\dashrightarrow$, is assumed secure but partially synchronous. Communication between P_0 and P_1, denoted $\rightarrow$, is assumed synchronous.

so because for any transaction between honest P_0 and P_1, either an encashment request or a reclaim/cancel request is made but never both. This keeps a credit note issuing request unlinkable from its corresponding encashment request.

Figure 5 shows our complete protocol. It allows real-time bank-to-bank transfers without any special hardware (**BB**, **RT** and **WH**), and achieves all our desired dispute-resolution, bank-fraud prevention and privacy properties for the in-person setting — see precise guarantees in §6.1.1 and proofs in Appendix A. The credit note also avoids the coin theft issue of digital cash, as only P_1 can encash it.

We now explain our main technical design choices:

Blind signatures for anonymity. We use a blind signature approach [16] based on BBS+ signatures [9] to maintain transaction privacy while receiving the credit note from B_0. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups of prime order q admitting a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, $g_i \in \mathbb{G}_1$ for all $i \in \mathbb{N}$, and $h \in \mathbb{G}_2$. A BBS+ signer's secret key is a random element $x \in \mathbb{Z}_q$ and public key is h^x; it creates a signature on a message tuple $(m_1, \ldots, m_k)$ by drawing $c, r \xleftarrow{\$} \mathbb{Z}_q$ and outputting $\sigma := ((g_0 g_1^{m_1} \ldots g_k^{m_k} g_{k+1}^r)^{\frac{1}{c+x}}, c, r)$. It can also issue a blind signature on $(m_1, \ldots, m_k)$ by receiving only a Pedersen commitment $\gamma = g_1^{m_1} \ldots g_k^{m_k} g_{k+1}^r$ from the signature requester, issuing a "quasi-signature" $\hat{\sigma} := (S, c, \hat{r}) = ((g_0 \gamma g_{k+1}^{\hat{r}})^{\frac{1}{c+x}}, c, \hat{r})$ for randomly drawn $c, \hat{r}$, and letting the requester obtains the BBS+ signature $\sigma := (S, c, r + \hat{r})$. (The requester also needs to give a non-interactive zero-knowledge (NIZK) proof of knowledge of $(m_1, \ldots, m_k, r)$ to guarantee that the signature is obtained only against a unique pre-determined value.)

We use this mechanism to let P_1 obtain a signature on the transaction details $(m, \mathsf{uid}, \mathsf{tid})$, where m denotes the transaction amount, uid denotes P_1's ID and tid denotes a fresh transaction ID, without B_0 learning uid and tid. During encashment, P_1 uses a NIZK proof of knowledge of σ, instead of directly supplying σ, to prevent linking with the corresponding issuing request.

Signatures of knowledge. Given that P_1 needs to be anonymous to P_0 and B_0, the cancellation credential issued by P_1 cannot be based on traditional PKI-based digital signatures. Instead, we use a *signature of knowledge* [15]. A signature of knowledge $\sigma := \mathsf{SPK}\{(x) : p(x, a)\}(m)$ (verified via an $\mathsf{SPKVer}(\sigma, a, m)$ algorithm) is a signature on a message m by the knower of a secret witness x such that the predicate $p(x, a)$ is true, where a is some public input. Here, we use it to convince P_0 that the cancellation credential was issued by the owner of the presented virtual ID.

Distributed transactions and rollback. We assume that the reclaim, encash and cancel transactions execute as distributed transactions between B_0 and B_1 with atomicity and strong consistency properties, i.e., either a transaction is fully executed or not at all, and when a transaction commits, all parties have a consistent view of the database. We also rely on standard transaction rollback mechanisms to revert the effects of a previously committed transaction.

Freshness of tids. Each transaction is tagged with a fresh transaction ID to prevent replay attacks and double spending. Since neither P_0 nor P_1 can be completely trusted to generate fresh tids, we let both contribute their randomnesses tid_0 and tid_1, and generate tid as the hash of the two. This prevents P_1 from getting fooled into accepting a previously spent note, and P_0 from getting fooled into deducting their balance for an already encashed transaction.

6.1.1 Security analysis. We now state the lemmas for our protocol's security and privacy properties. The proofs are in Appendix A. Let $\mathsf{bal}_b^i, \mathsf{bal}_b^f$ respectively denote P_b's balance at B_b (the value of $\mathsf{BAL}[\mathsf{bid}_b]$) at the beginning and end of the Transact protocol. Let $\mathsf{bal}_{B_1}^i, \mathsf{bal}_{B_1}^f$ denote the net balance that B_0 owes to B_1 at the beginning and end of the Transact protocol.

Lemma 1 ($\mathbf{NU}_{P_0}$). *If a* $\mathsf{TXN}[\mathsf{ctid}] = (\mathsf{bid}_0, m, \mathsf{vid}_1)$ *entry is recorded by* B_0 *then a user* P_0 *must have participated in a* Register *protocol with input* $(\mathsf{bid}_0, \mathsf{BS}[\mathsf{bid}_0], \mathsf{uid}_0, \mathsf{us}_0)$ *and a distinct* Transact *protocol with input* $(m, \mathsf{uid}_0, \mathsf{us}_0)$*, against a user* P_1 *who knows* $(\mathsf{uid}_1, r_{\mathsf{vid}_1})$ *satisfying* $\mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}$.

Lemma 2 ($\mathbf{FF}_{P_0}$). *If an honest* P_0 *does not receive the good from* P_1*, then* $\mathsf{bal}_0^f \geq \mathsf{bal}_0^i$.

Lemma 3 ($\mathbf{SP}_{P_0}$). *If* $\mathsf{bal}_0^f < \mathsf{bal}_0^i$*, then an honest* P_0 *can convince an honest* P_1 *to give the good to* P_0.

Lemma 4 ($\mathbf{SF}_{P_1}$). *If an honest* P_1 *gives the good to* P_0*, then* $\mathsf{bal}_1^f \geq \mathsf{bal}_1^i + m$.

Lemma 5 ($\mathbf{FP}_{P_1}$). *If an honest* P_1 *sends* σ_c *to* P_0 *at the end of the* Transact *protocol (does not give the good to* P_0*), then an honest* P_0 *sends "ok" to* P_1.

Lemma 6 ($\mathbf{SD}_{B_0}$). *For any given invocation of the* Transact *protocol, the amount that* B_0 *debits from* P_0*'s account is at least the credit liability of* B_0 *towards* B_1*, i.e.,* $(\mathsf{bal}_0^i - \mathsf{bal}_0^f) \geq (\mathsf{bal}_{B_1}^f - \mathsf{bal}_{B_1}^i)$.

Lemma 7 ($\mathbf{LC}_{B_1}$). *The amount that* B_1 *needs to credit to* P_1*'s account is at most the amount that* B_1 *receives from* B_0*, i.e.,* $(\mathsf{bal}_0^f - \mathsf{bal}_0^i) \leq (\mathsf{bal}_{B_1}^f - \mathsf{bal}_{B_1}^i)$.

Lemma 8 (**PB**, **PI**). *Neither* P_0 *nor* P_1*'s* bid *or* uid *leaks to anyone except their own banks* B_0 *and* B_1 *respectively. That is, if* P_0 *and* B_0 *are honest, then the view of all other parties can be simulated using just the transaction amount m and the information that* P_0 *holds a valid* bid_0 *and* uid_0*. If* P_1 *and* B_1 *are honest, then the view of all other parties can be simulated using the same information about* P_1.

Lemma 9 (**PT**). *For any invocation of the* Transact *protocol,* B_0 *and* B_1 *cannot distinguish between* P_0 *interacting with* P_1 *or* P_0 *interacting with* P_1'*, assuming the transaction amount m is the same and both payer and payee are honest in each interaction. Also, no external adversary identifies* P_0 *or* P_1 *or learns the amount m.*

6.2 Internet-based transactions

Internet-based transactions pose new risks that do not exist when the payment parties P_0 and P_1 are physically co-present. Specifically, P_1 may misuse a payment system's anonymity properties and disappear after receiving the funds, leaving P_0 with no recourse to raise disputes or identify P_1. To ensure fairness, the protocol must guarantee that (i) P_1 cannot receive funds unless P_0 simultaneously obtains a verifiable receipt, (ii) P_0 cannot obtain a payment receipt without making the payment, and (iii) P_1's identity remains private except in the event of a dispute, where a regulator can intervene. We map this problem to the *fair exchange* problem [8, 11, 18, 19, 24, 35, 47, 48], where two parties wish to exchange digital objects — funds and a receipt — without either party being able to cheat. In particular, we adapt the fair exchange protocol of Bao et al. [11] where disputes are resolved by a TTP T. A TTP is unavoidable for fair exchange [19], but unlike UPI, card-based and wallet-based approaches, our approach allows an *offline* TTP, who is not involved without a dispute. The TTP also does not learn a user's transaction details unless the user explicitly raises a dispute. We separate T, representing a trusted software, from the regulator R, who must manually resolve disputes about the transfer of goods.

The modified protocol (shown in Figure 6) proceeds as follows. First, during registration, the banks sign the uid given to the user for authentication, under an EUF-CMA secure signature scheme Sign/Ver. During the transaction, P_1 provides an encryption of the would-be payment receipt to P_0. This receipt is signed by P_1 using a signature of knowledge of their virtual ID and is encrypted under the public key of T (pk_T) to prevent access by P_0 before actually making the payment. Additionally, P_1 encrypts their identity (uid_1) under the regulator's public key (pk_R), preventing disclosure unless a dispute occurs. Both encryptions, along with appropriate NIZK proofs, are sent to P_0. After validating them, P_0 proceeds with the payment using the blind signature-based credit note mechanism

$\underline{\mathsf{Register}(P_b\langle \mathsf{bid}_b, \mathsf{bs}_b, \mathsf{uid}_b, \mathsf{us}_b\rangle, B_b\langle\ldots\rangle)\text{:}}$

$P_b \dashrightarrow B_b$: $\mathsf{bid}_b, \mathsf{bs}_b, \mathsf{uid}_b, \mathsf{us}_b$

B_b: (proceed as Fig. 5 to update the UID, US and BID databases)

$B_b \dashrightarrow P_b$: $\sigma_{\mathsf{uid}_b} \leftarrow \mathsf{Sign}(\mathsf{sk}_{B_b}, \mathsf{uid}_b)$

$\underline{\mathsf{Transact}(P_0\langle m, \mathsf{uid}_0, \mathsf{us}_0\rangle, P_1\langle \mathsf{uid}_1, m\rangle, B_0\langle\ldots\rangle, B_1\langle\ldots\rangle, T\langle\ldots\rangle)\text{:}}$

P_1, P_0: (obtain $\mathsf{tid} = H(\mathsf{tid}_0, \mathsf{tid}_1)$ as Fig. 5 by exchanging tid_0, tid_1)

P_1: (proceed as Fig. 5 to obtain $\mathsf{vid}_1, \pi_{\mathsf{vid}_1}, r_{\mathsf{vid}_1}$)

P_1: $\sigma_{\mathsf{rec}} \leftarrow \mathsf{SPK}\{(\mathsf{uid}_1, r_{\mathsf{vid}_1}) : \mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}\}((\mathsf{tid}, m))$

P_1: $c_{\sigma_{\mathsf{rec}}} \leftarrow \mathsf{Enc}(\mathsf{pk}_T, \sigma_{\mathsf{rec}})$

P_1: $\pi_{\sigma_{\mathsf{rec}}} \leftarrow \mathsf{NIZKPK}\{(\sigma_{\mathsf{rec}}) : c_{\sigma_{\mathsf{rec}}} = \mathsf{Enc}(\mathsf{pk}_T, \sigma_{\mathsf{rec}}) \wedge$
$\quad \mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1\}$

P_1: $c_{\mathsf{uid}_1} \leftarrow \mathsf{Enc}(\mathsf{pk}_R, \mathsf{uid}_1)$

P_1: $\pi_{\mathsf{uid}_1} \leftarrow \mathsf{NIZKPK}\{(\mathsf{uid}_1, r_{\mathsf{vid}_1}, \sigma_{\mathsf{uid}_1}) : c_{\mathsf{uid}_1} = \mathsf{Enc}(\mathsf{pk}_R, \mathsf{uid}_1) \wedge$
$\quad \mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}} \wedge \mathsf{Ver}(\sigma_{\mathsf{uid}_1}, \mathsf{pk}_{B_1}, \mathsf{uid}_1) = 1\}$

$P_1 \dashrightarrow P_0$: $\mathsf{tid}_1, \mathsf{vid}_1, \pi_{\mathsf{vid}_1}, c_{\sigma_{\mathsf{rec}}}, \pi_{\sigma_{\mathsf{rec}}}, c_{\mathsf{uid}_1}, \pi_{\mathsf{uid}_1}$

P_0: assert $\mathsf{NIZKVer}(\pi_{\sigma_{\mathsf{rec}}}, c_{\sigma_{\mathsf{rec}}}, \mathsf{vid}_1, \mathsf{tid}, m, \mathsf{pk}_T) = 1$

P_0: assert $\mathsf{NIZKVer}(\pi_{\mathsf{uid}_1}, c_{\mathsf{uid}_1}, \mathsf{vid}_1, \mathsf{pk}_R, \mathsf{pk}_{B_1}) = 1$

P_0: (proceed as Fig. 5 to obtain tid_0, tid and potentially $\tilde{\sigma} := (S, c, \tilde{r})$)

$P_0 \dashrightarrow P_1$: $\tilde{\sigma} := (S, c, \tilde{r})$, tid, tid_0

P_1: assert $\mathsf{tid} = H(\mathsf{tid}_0, \mathsf{tid}_1)$ and $e(S, \mathsf{pk}_{B_0} h^c) = e(g_0 g_1^m g_2^{\mathsf{tid}} \mathsf{vid}_1 g_4^{\tilde{r}}, h)$

$P_1 \dashrightarrow P_0$: σ_{rec} // the payment success receipt

P_1: $r \leftarrow \tilde{r} + r_{\mathsf{vid}_1}$; $\sigma := (S, c, r)$ **(A)**

P_1: $\pi_\sigma \leftarrow \mathsf{NIZKPK}\{(S, c, r) : e(S, \mathsf{pk}_{B_0} h^c) = e(g_0 g_1^m g_2^{\mathsf{tid}} g_3^{\mathsf{uid}_1} g_4^r, h)\}$

$\mathsf{Encash}(P_1\langle \pi_\sigma, m, \mathsf{tid}, \mathsf{uid}_1\rangle, B_1\langle\ldots\rangle, B_0\langle\ldots\rangle)$

P_0: if not received σ_{rec} from P_1 s.t. $\mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1$
within timeout τ:

$\quad P_0 \to T$: $c_{\sigma_{\mathsf{rec}}}, c_{\mathsf{uid}_1}, m, \mathsf{tid}, \mathsf{vid}_1, \tilde{\sigma} := (S, c, \tilde{r})$,

$\quad T$: $\sigma_{\mathsf{rec}} \leftarrow \mathsf{Dec}(\mathsf{sk}_T, c_{\sigma_{\mathsf{rec}}})$

$\quad T$: assert $\mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1$

$\quad T$: assert $e(S, \mathsf{pk}_{B_0} h^c) = e(g_0 g_1^m g_2^{\mathsf{tid}} \mathsf{vid}_1 g_4^{\tilde{r}}, h)$

$\quad T \to P_1$: $\tilde{\sigma} := (S, c, \tilde{r})$

$\quad T \to P_0$: σ_{rec}

$\quad P_1$: (proceed from step **(A)** above)

P_0: output $(\mathsf{vid}_1, \mathsf{tid}, m, \sigma_{\mathsf{rec}}, c_{\mathsf{uid}_1})$ // P_0 can take this to R

Figure 6: The modified Register **and** Transact **protocols for internet-based transactions, using the fair-exchange protocol of [11].**

from §6.1. Once the transaction is authorised, P_0 sends the signed credit note to P_1, proving that their account has been debited. If P_1 is honest, they reveal the cleartext payment receipt to P_0 right after verifying the note (P_1 encashes the note later).

If P_1 does not disclose the receipt within a predefined timeout, P_0 contacts T with the encrypted receipt and the note received from B_0. T verifies both of them and decrypts and releases the receipt to P_0. In this case, T also sends the note to P_1 to protect them from a malicious P_0. The exchange thus remains fair for both P_0 and P_1.

If P_1 violates the payment contract (disappears after receiving the money), P_0 can escalate the issue to R, who decrypts P_1's identity and facilitates redressal. The following lemma (proof in Appendix B) guarantees that R confirms the transaction details assumed by P_0 and identifies a valid uid linkable to P_1's long-term banking ID.

Lemma 10 (**EDR**$_{P_0}$). *If* $\mathsf{bal}_0^f < \mathsf{bal}_0^i$ *and* P_0 *is honest, then it outputs* $(\mathsf{vid}_1, \mathsf{tid}, m, \sigma_{\mathsf{rec}}, c_{\mathsf{uid}_1})$ *such that* $\mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1$, vid_1 *is a commitment of* $\mathsf{uid} := \mathsf{Dec}(\mathsf{sk}_R, c_{\mathsf{uid}_1})$, *where* sk_R *is* R*'s secret key, and* uid *was used in an invocation to the* Register *protocol.*

The approach also retains most of the security and privacy guarantees of §6.1. Fair exchange ensures that transactions are atomic, preventing disputes (**SP**$_{P_0}$, **SF**$_{P_1}$, **FP**$_{P_1}$). P_1's identity remains pseudonymous, ensuring privacy (**PB**, **PI**, **PT**) by default, while still allowing for selective disclosure in case of disputes. Note that R or T cannot obtain P_1's identity unless P_0 raises a dispute. This prevents large-scale profiling of both P_0 and P_1. Other properties remain similar to the original credit note version.

6.3 Practicalities and implementation

Both our protocols rely on BBS+ signatures and NIZK proofs. The protocol of §6.1 uses proofs of knowledge of BBS+ signatures and commitment openings, and signatures of knowledge of discrete log statements. Very efficient Σ-protocols for all these proofs are standard [9, 14]. The protocol of §6.2 uses additional, more complicated, NIZK proofs ($\pi_{\sigma_{\mathsf{rec}}}$ and π_{uid_1}). To understand their practicality, we implemented proof-of-concept zkSNARKs for them. Our implementation and the benchmarks are available at [10].

We used the ZoKrates library [23] with the default (Groth16, BN128) setting to model both $\pi_{\sigma_{\mathsf{rec}}}$ and π_{uid_1}. We used hybrid encryption with El Gamal KEM and symmetric DEM to encrypt uid_1. To encrypt the SPK σ_{rec}, a hash-based NIZK of a Σ-protocol, we used component-wise El Gamal encryption/hybrid-encryption. We used Poseidon [27] for computing hashes inside the ZK circuit.

All our experiments were run on an Apple Macbook Pro laptop with M4 Pro chip and 24 GB memory. We find that $\pi_{\sigma_{\mathsf{rec}}}$ and π_{uid_1} require 0.9s and 0.6s proving time respectively and < 10ms verification time each. The proof sizes are tiny, around 2-3 KB. Given that network latencies dominate internet-based payment systems, this is quite practical. Also, since the prover is the merchant in our setting, backend parallelisation opportunities exist too.

Usability-wise, although the in-person protocol adds extra user steps beyond a standard UPI push transaction (the payee needs to generate a virtual ID and verify the credit note), users' mental model is close to UPI pull transactions (also quite popular) and cash payments. The internet-based protocol requires multiple rounds between P_0 and P_1, but this is acceptable as they happen in the background and no user steps are required. Finally, while our protocols cleanly handle unreliable networks, they inherently require online communication with the banks to record the payment state (no **WI**). Avoiding this dependency is challenging, but UPI's widespread adoption [45] already shows that this is becoming less critical.

7 Conclusion

We comprehensively analyse the privacy, security, and fault-tolerance challenges in UPI and similar real-time payment systems. We proposed two new designs — one for in-person and another for internet-based transactions — addressing key limitations like dispute resolution and privacy. While our proposals are not immediately deployable to the current UPI, their usability features align well with UPI's goals and lay critical foundations for future sovereign payment systems. Note that we prioritise user privacy, but real deployments may also require conditional traceability (say, via a trapdoor) for law enforcement. We leave this and other details for future work.

References

[1] Abeer Ray. 2023. What should you do if your UPI transactions fail? A step-by-step guide. https://www.livemint.com/money/personal-finance/what-should-you-do-if-your-upi-transactions-fail-a-step-by-step-guide-151693831284290.html. Accessed: 2024-12-25.

[2] Abenaya Gunasekaran. 2024. UPI Fraud: Common Types and Methods to Prevent UPI Payment Frauds. https://razorpay.com/blog/upi-frauds-types-tactics/. Accessed: 2024-12-25.

[3] ACI Worldwide. 2023. World's Major Economies Playing Catch-Up as Widespread Adoption Drives Global Real-Time Payments Growth. https://www.businesswire.com/news/home/20230328005314/en/World%E2%80%99s-Major-Economies-Playing-Catch-Up-as-Widespread-Adoption-Drives-Global-Real-Time-Payments-Growth-%E2%80%93-ACI-Worldwide-Report. Accessed: 2024-11-03.

[4] Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven Murdoch, Ross Anderson, and Ron Rivest. 2009. Phish and chips: Traditional and new recipes for attacking EMV. In *Security Protocols: 14th International Workshop, Cambridge, UK, March 27-29, 2006, Revised Selected Papers 14*. Springer, 40–48.

[5] Waqas Ahmed, Aamir Rasool, Abdul Rehman Javed, Neeraj Kumar, Thippa Reddy Gadekallu, Zunera Jalil, and Natalia Kryvinska. 2021. Security in Next Generation Mobile Payment Systems: A Comprehensive Survey. *IEEE Access* 9 (2021), 115932–115950. https://doi.org/10.1109/ACCESS.2021.3105450

[6] Asian Development Bank. 2023. *Leveraging Lessons Learned from India's Unified Payments Interface for Digital Transformation in Asia and the Pacific*. Technical Report ADB Brief No. 299. Asian Development Bank. https://www.adb.org/sites/default/files/publication/964626/adb-brief-299-india-unified-payments-interface.pdf

[7] N. Asokan, P. Janson, M. Steiner, and M. Waidner. 2000. State of the art in electronic payment systems. In *Emphasizing Distributed Systems*, Marvin V. Zelkowits (Ed.). Advances in Computers, Vol. 53. Elsevier, 425–449. https://doi.org/10.1016/S0065-2458(00)80009-1

[8] Nadarajah Asokan, Victor Shoup, and Michael Waidner. 1998. Optimistic fair exchange of digital signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 591–606.

[9] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-size dynamic k-TAA. In *Security and Cryptography for Networks*. 111–125.

[10] Anonymous Authors. 2025. Benchmarks for NIZK proofs $\pi_{\sigma_{\text{rec}}}$ and π_{uid_1}. https://drive.google.com/file/d/16K5ie6g4wR8V3P_Q3YrtlpCnF9Ovf9EP/view?usp=sharing

[11] Feng Bao, Robert H Deng, and Wenbo Mao. 1998. Efficient and practical fair exchange protocols with off-line TTP. In *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186)*. IEEE, 77–85.

[12] David Basin, Ralf Sasse, and Jorge Toro-Pozo. 2021. The EMV standard: Break, fix, verify. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1766–1781.

[13] Mike Bond, Omar Choudary, Steven J Murdoch, Sergei Skorobogatov, and Ross Anderson. 2014. Chip and Skim: cloning EMV cards with the pre-play attack. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 49–64.

[14] Jan Camenisch. 1998. *Group signature schemes and payment systems based on the discrete logarithm problem.* Ph. D. Dissertation. ETH Zurich.

[15] Melissa Chase and Anna Lysyanskaya. 2006. On signatures of knowledge. In *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*. Springer, 78–96.

[16] David Chaum. 1983. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*. Springer, 199–203.

[17] David Chaum, Bert den Boer, Eugène van Heyst, Stig Mjølsnes, and Adri Steenbeek. 1990. Efficient Offline Electronic Checks. In *Advances in Cryptology — EUROCRYPT '89*, Jean-Jacques Quisquater and Joos Vandewalle (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 294–301.

[18] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. 2017. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 719–728.

[19] Richard Cleve. 1986. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. 364–369.

[20] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. 2004. Tor: The second-generation onion router.. In *USENIX Security*, Vol. 4. 303–320.

[21] Saar Drimer, Steven J Murdoch, and Ross Anderson. 2009. Optimised to fail: Card readers for online banking. In *Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers 13*. Springer, 184–200.

[22] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM* 35, 2 (April 1988), 288–323. https://doi.org/10.1145/42282.42283

[23] Jacob Eberhardt and Stefan Tai. 2018. ZoKrates - scalable privacy-preserving off-chain computations. In *IEEE iThings*. 1084–1091.

[24] Shimon Even, Oded Goldreich, and Abraham Lempel. 1985. A randomized protocol for signing contracts. *Commun. ACM* 28, 6 (1985), 637–647.

[25] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. 2010. On the security issues of NFC enabled mobile phones. *Int. J. Internet Technol. Secur. Syst.* 2, 3/4 (Dec. 2010), 336–356. https://doi.org/10.1504/IJITST.2010.037408

[26] Shirsha Ghosh, Joyeeta Goswami, Abhishek Kumar, and Alak Majumder. 2015. Issues in NFC as a form of contactless communication: A comprehensive survey. In *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*. IEEE, 245–252.

[27] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. 2021. Poseidon: a new hash function for zero-knowledge proof systems. In *USENIX Security*.

[28] Ernst Haselsteiner and Klemens Breitfuß. 2006. Security in near field communication (NFC). In *Workshop on RFID security*, Vol. 517. sn, 517.

[29] Paul JM Havinga, Gerardus Johannes Maria Smit, and Arne Helme. 1996. Survey of electronic payment methods and systems. In *Euromedia 1996, A scientific conference on Web technology, new media, communications and telematics theory, methods, tools and applications*. Society for Computer Simulation, 180–187.

[30] Maurice Herlihy. 2018. Atomic Cross-Chain Swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing* (Egham, United Kingdom) *(PODC '18)*. Association for Computing Machinery, New York, NY, USA, 245–254. https://doi.org/10.1145/3212734.3212736

[31] Pita Jarupunphol and Chris J Mitchell. 2003. Measuring 3-D Secure and 3D SET against e-commerce end-user requirements. In *Proceedings of the 8th Collaborative electronic commerce technology and research conference*. 51–64.

[32] Renuka Kumar, Sreesh Kishore, Hao Lu, and Atul Prakash. 2020. Security analysis of unified payments interface and payment apps in India. In *29th usenix security symposium (usenix security 20)*. 1499–1516.

[33] Manoj Sharma. 2024. UPI Frauds: 6.3 lakh cases worth Rs. 485 cr reported in FY25 so far. https://www.fortuneindia.com/macro/upi-frauds-63-lakh-cases-worth-485-cr-reported-in-fy25-so-far/119275. Accessed: 2024-12-25.

[34] Muddassir Masihuddin, Burhan Ul Islam Khan, MMUI Mattoo, and Rashidah F Olanrewaju. 2017. A survey on e-payment systems: elements, adoption, architecture, challenges and security concepts. *Indian Journal of Science and Technology* 10, 20 (2017), 1–19.

[35] Silvio Micali. 2003. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. 12–19.

[36] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*. 397–411. https://doi.org/10.1109/SP.2013.34

[37] Moneycontrol. 2022. India in Talks with 30 Countries Regarding UPI: MeitY Minister Ashwini Vaishnaw. https://www.moneycontrol.com/news/business/india-in-talks-with-30-countries-regarding-upi-meity-minister-ashwini-vaishnaw-8778441.html. Accessed: 2024-11-03.

[38] Collin Mulliner. 2009. Vulnerability analysis and attacks on NFC-enabled mobile phones. In *2009 International Conference on Availability, Reliability and Security*. IEEE, 695–700.

[39] Steven J Murdoch and Ross Anderson. 2010. Verified by Visa and MasterCard SecureCode: Or, How Not to Design Authentication: (Short Paper). In *International Conference on Financial Cryptography and Data Security*. Springer, 336–342.

[40] Steven J Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. 2010. Chip and PIN is Broken. In *2010 IEEE Symposium on Security and Privacy*. IEEE, 433–446.

[41] National Payments Corporation of India. 2016. Unified Payments Interface - API and Technical Specifications Document. https://pdfcoffee.com/product-doc-pdf-free.html. Archived. Accessed: 2024-11-03.

[42] National Payments Corporation of India. 2016. Unified Payments Interface - Procedural Guidelines. https://web.archive.org/web/20200413132629/https://www.npci.org.in/sites/default/files/UPI-PG-RBI_Final.pdf. Archived. Accessed: 2024-11-03.

[43] National Payments Corporation of India. 2024. UPI 3rd Party Apps. https://www.npci.org.in/what-we-do/upi/3rd-party-apps. Accessed: 2024-12-24.

[44] National Payments Corporation of India. 2024. UPI Live Members. https://www.npci.org.in/what-we-do/upi/live-members. Accessed: 2024-12-24.

[45] National Payments Corporation of India. 2024. UPI Product Statistics. https://www.npci.org.in/what-we-do/upi/product-statistics. Accessed: 2024-12-24.

[46] National Payments Corporation of India. 2025. UPI Lite: Make instant small value payments at lightning speed. https://www.npci.org.in/PDF/npci/upi-lite/Product-Booklet.pdf. Accessed: 2025-02-07.

[47] Tatsuaki Okamoto and Kazuo Ohta. 1994. How to simultaneously exchange secrets by general assumptions. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*. 184–192.

[48] Henning Pagnia, Felix C Gärtner, et al. 1999. *On the impossibility of fair exchange without a trusted third party*. Technical Report. Citeseer.

[49] Joseph Poon and Thaddeus Dryja. 2016. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf Accessed: 2025-04-13.

[50] Shailesh Rastogi, Chetan Panse, Arpita Sharma, and Venkata Mrudula Bhimavarapu. 2021. Unified Payment Interface (UPI): A digital innovation and its impact on financial inclusion and economic development. *Universal Journal of Accounting and Finance* 9, 3 (2021), 518–530.

[51] Bradley Reaves, Nolen Scaife, Dave Tian, Logan Blue, Patrick Traynor, and Kevin R. B. Butler. 2016. Sending Out an SMS: Characterizing the Security of the SMS Ecosystem with Public Gateways. In *2016 IEEE Symposium on Security and Privacy (SP)*. 339–356. https://doi.org/10.1109/SP.2016.28

[52] Michael Roland, Josef Langer, and Josef Scharinger. 2011. Security Vulnerabilities of the NDEF Signature Record Type. 65 – 70. https://doi.org/10.1109/NFC.2011.9

[53] Michael Roland, Josef Langer, and Josef Scharinger. 2013. Applying relay attacks to Google Wallet. In *2013 5th International Workshop on Near Field Communication (NFC)*. 1–6. https://doi.org/10.1109/NFC.2013.6482441

[54] Shivani Bazaz. 2024. UPI fraud cases surge by 85% in FY24: Key insights and data. https://www.cnbctv18.com/business/finance/upi-fraud-cases-rise-85-pc-in-fy24-increase-parliament-reply-data-19514295.htm. Accessed: 2024-12-25.

[55] Anubhutie Singh, Beni Chugh, Deepti George, and Lakshay Narang. 2024. *Curbing Scams in Unified Payments Interface: A White Paper on Facilitating Real-Time Reporting and Management*. Technical Report. Dvara Research and Data Security Council of India. Accessed: 2024-12-25.

[56] Siamak Solat. 2017. Security of electronic payment systems: A comprehensive survey. *arXiv preprint arXiv:1701.04556* (2017).

[57] The Economic Times. 2024. New UPI fraud trend: Fraudsters spamming UPI IDs with multiple collect requests; one careless approval means money gone from bank a/c. https://economictimes.indiatimes.com/wealth/save/new-upi-scam-alert-upi-autopay-set-up-request-could-swindle-money-out-of-your-account-if-you-are-not-careful/articleshow/112584562.cms?from=mdr. Accessed: 2024-11-03.

[58] The Economic Times. 2024. New UPI scam: Fraudsters send you 'Rs 200.00' and ask to return Rs 20,000; how to protect yourself from UPI overpayment scam. https://economictimes.indiatimes.com/wealth/save/new-upi-scam-fraudsters-send-you-rs-200-00-and-asks-to-return-rs-20000-00-how-to-protect-yourself-from-upi-overpayment-scam/articleshow/110815938.cms?from=mdr. Accessed: 2024-11-03.

[59] The Times of India. 2024. Stuck or failed UPI transactions: Common issues and how to deal with them. https://timesofindia.indiatimes.com/business/financial-literacy/banking/common-issues-and-solutions-for-stuck-or-failed-upi-transactions/articleshow/108112007.cms. Accessed: 2024-12-25.

[60] Umang Poddar. 2023. Are your UPI payments farming your personal data without your consent? https://scroll.in/article/1045536/are-your-upi-payments-farming-your-personal-data-without-your-consent. Accessed: 2024-12-25.

[61] Vivek Belgavi and Mihir Gandhi. 2019. Changing preferences: UPI's dominance over digital wallets in the payments market. https://www.pwc.in/assets/pdfs/consulting/financial-services/fintech/point-of-view/pov-downloads/changing-preferences-upis-dominance-over-digital-wallets-in-the-payments-market.pdf. Accessed: 2024-12-26.

A Proofs for Lemmas in Section 6.1.1

LEMMA 1 ($\mathbf{NU}_{P_0}$). *If a* $\mathsf{TXN}[\mathsf{ctid}] = (\mathsf{bid}_0, m, \mathsf{vid}_1)$ *entry is recorded by* B_0 *then a user* P_0 *must have participated in a* Register *protocol with input* $(\mathsf{bid}_0, \mathsf{BS}[\mathsf{bid}_0], \mathsf{uid}_0, \mathsf{us}_0)$ *and a distinct* Transact *protocol with input* $(m, \mathsf{uid}_0, \mathsf{us}_0)$*, against a user* P_1 *who knows* $(\mathsf{uid}_1, r_{\mathsf{vid}_1})$ *satisfying* $\mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}$.

PROOF. Since B_0 records a $\mathsf{TXN}[\mathsf{ctid}] = (\mathsf{bid}_0, m, \mathsf{vid}_1)$ entry only during the handling of an ("IssueReq", uid_0, us_0, m, ctid, π_{ctid}, vid_1, π_{vid_1}) message, it must have checked that $\mathsf{bid}_0 = \mathsf{BID}[\mathsf{uid}_0]$ and $\mathsf{us}_0 = \mathsf{US}[\mathsf{uid}_0]$. Thus, B_0 must have earlier received a Register protocol request with input $(\mathsf{bid}_0, \mathsf{BS}[\mathsf{bid}_0], \mathsf{uid}_0, \mathsf{us}_0)$. Since the channel between P_0, the owner of bid_0, and B_0 is secure and assuming that secret $\mathsf{BS}[\mathsf{bid}_0]$ is hard to guess, P_0 must have participated in the Register protocol with input $(\mathsf{bid}_0, \mathsf{BS}[\mathsf{bid}_0], \mathsf{uid}_0, \mathsf{us}_0)$.

Next, assuming that secret us_0 is hard to guess, the secure channel between P_0 and B_0 also implies that P_0 must have sent a message ("IssueReq", uid_0, us_0, m, ctid, π_{ctid}, vid_1, π_{vid_1}) to B_0. Thus, P_0 must have participated in a Transact protocol with input $(m, \mathsf{uid}_0, \mathsf{us}_0)$. Further, since P_0 verifies π_{vid_1} in this protocol, by the knowledge soundness of π_{vid_1}, an extractor must be able to extract $(\mathsf{uid}_1, r_{\mathsf{vid}_1})$ such that $\mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}$. This extraction must be unique by the computational binding property of Pedersen commitments.

Now suppose there exist two distinct entries $\mathsf{TXN}[\mathsf{ctid}] = \mathsf{TXN}[\mathsf{ctid}'] = (\mathsf{bid}_0, m, \mathsf{vid}_1)$, but P_0 participates only once in a Transact protocol with input $(m, \mathsf{uid}_0, \mathsf{us}_0)$. Let the extractor output $(\mathsf{uid}_1, r_{\mathsf{vid}_1})$ satisfying $\mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}$ in this invocation. Since P_0 only interacts in one invocation of the Transact protocol with input (m, uid_0, us_0), it must have sent only one message of the form ("IssueReq", uid_0, us_0, m, ..., ..., vid_1, ...). Since B_0 ensures that $\mathsf{ctid} \neq \mathsf{ctid}'$, P_0 does not send a "IssueReq" message against either ctid or ctid'. Thus, by the secure channel property, one of the messages received by B_0 must be rejected, leading to a contradiction. □

LEMMA 2 ($\mathbf{FF}_{P_0}$). *If an honest* P_0 *does not receive the good from* P_1*, then* $\mathsf{bal}_0^f \geq \mathsf{bal}_0^i$.

PROOF. Suppose P_0 does not receive the good from P_1 (does not obtain "success") at the end of the Transact protocol. The following three cases arise (the case that P_0 does not receive any response from P_1 or receives an invalid σ_c does not arise in our considered in-person setting):

Case I: P_0 *failed to verify* π_{vid_1}*:* In this case, P_0 does not send a ("IssueReq", ..., ctid, ...) message. Thus, B_0 does not execute an $\mathsf{issue}(\mathsf{tid})$ transaction. Thus, if $\mathsf{bal}_0^f < \mathsf{bal}_0^i$ then the adversary must have either broken the secure channel between P_0 and B_0 or learnt P_0's authentication secrets bs_0 or us_0. Both lead to a contradiction under our assumptions.

Case II: P_0 *verified* π_{vid_1} *but did not receive* $\hat{\sigma}$ *from* B_0 *till timeout* τ*:* In this case, P_0 initiates reclaim requests with B_0 and does not give anything to P_1. Under the partial synchrony model, and because of retries by P_0, some reclaim request is eventually received by B_0. Consider the first such request. Note first that the assertion $\mathsf{ctid} = g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}$ made by B_0 passes, because an honest P_0 sends the correct opening of ctid. Next, note that the $\mathsf{issue}(\mathsf{tid})$ transaction is executed at most once, because of the $\mathsf{ctid} \notin \mathsf{CTID}_{\mathsf{requested}}$ check made by B_0.

If $\mathsf{issue}(\mathsf{tid})$ has not executed at all, the reclaim request fails when accessing $\mathsf{TXN}[\mathsf{ctid}]$, but this keeps P_0's balance $\mathsf{BAL}[\mathsf{bid}_0]$ intact and the claim holds.

If $\mathsf{issue}(\mathsf{tid})$ has been executed, then $\mathsf{BAL}[\mathsf{bid}_0] = \mathsf{bal}_0^i - m$ right before the execution of the $\mathsf{reclaim}(\mathsf{tid})$ transaction, because of secret channels and secrecy of bs_0 and us_0. We show below that the $\mathsf{reclaim}(\mathsf{tid})$ transaction executes and restores $\mathsf{BAL}[\mathsf{bid}_0] = \mathsf{bal}_0^i$. Since no further transactions are executed against tid, this is also P_0's final balance.

We now claim that $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{encashed}}$. Note that P_0 did not send anything to P_1. Thus, if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{encashed}}$, then the assert condition made by B_1 just before executing the $\mathsf{encash}(\mathsf{tid})$ transaction must have passed. However, this is the signature verification equation of a BBS+ signature against the message $(m, \mathsf{tid}, \mathsf{uid}_1)$ under B_0's public key, but no signature against any message containing tid was given to the adversary. Thus, the adversary must have broken the EUF-CMA security of the BBS+ signature scheme, which leads to a contradiction.

Further, $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{reclaimed}} \cup \mathsf{TID}_{\mathsf{cancelled}}$, because this is the first reclaim request received against tid and no cancel request against tid was sent by P_0. Note that the probability of tid clashing with any previous transaction's tid is negligible, because $\mathsf{tid} = H(\mathsf{tid}_0, \mathsf{tid}_1)$, tid_0 was sampled uniformly from a large space $\mathbb{Z}_q$, and H is collision-resistant.

Thus, the validation conditions of the $\mathsf{reclaim}(\mathsf{tid})$ transaction are satisfied, and $\mathsf{BAL}[\mathsf{bid}_0] = \mathsf{bal}_0^f = \mathsf{bal}_0^i$.

Case III: P_0 received a valid cancel credential σ_c from P_1: In this case, P_0 verifies that $\mathsf{SPKVer}(\sigma_c, \mathsf{vid}_1, \mathsf{tid}) = 1$ and initiates cancellation requests. Under the partial synchrony model and because of retries by P_0, some cancellation request is eventually accepted by B_0. As above, the assertion $\mathsf{ctid} = g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}$ made by B_0 holds, as does the SPK verification assertion. Further, accessing $\mathsf{TXN}[\mathsf{ctid}]$ gives the correct variables $(\mathsf{bid}_0, m, \mathsf{vid}_1)$ set during the $\mathsf{IssueReq}(\mathsf{ctid})$ request accepted by B_0. Thus, the $\mathsf{cancel}(\mathsf{tid})$ transaction must have executed.

Note that $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{cancelled}} \cup \mathsf{TID}_{\mathsf{reclaimed}}$ as this is the first cancellation request against tid received by B_0 and an honest P_0 does not initiate reclaim requests in this case. Further, since $\mathsf{IssueReq}(\mathsf{ctid})$ executes only once, $\mathsf{BAL}[\mathsf{bid}_0] = \mathsf{bal}_0^i - m$ at the beginning of the $\mathsf{cancel}(\mathsf{tid})$ transaction. Now, if $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{encashed}}$, this straightaway means that $\mathsf{BAL}[\mathsf{bid}_0] = \mathsf{bal}_0^i$ at the end of the $\mathsf{cancel}(\mathsf{tid})$ transaction. Even if $\mathsf{tid} \in \mathsf{TID}_{\mathsf{encashed}}$, the $\mathsf{encash}(\mathsf{tid})$ transaction could not have executed the revert $\mathsf{reclaim}(\mathsf{tid})$ step (because $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{reclaimed}}$), and thus reversal of $\mathsf{encash}(\mathsf{tid})$ does not further reduce $\mathsf{BAL}[\mathsf{bid}_0]$ and the claim holds. □

Lemma 3 ($\mathbf{SP}_{P_0}$). *If $\mathsf{bal}_0^f < \mathsf{bal}_0^i$, then an honest P_0 can convince an honest P_1 to give the good to P_0.*

Proof. If P_0 sends $\tilde{\sigma}$ to P_1 then $\tilde{\sigma} = (S, c, \tilde{r}) = (g_0 g_1^m g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}} \mathsf{vid}_1 g_4^{\hat{r}}, c, \hat{r} + r_{\mathsf{tid}})$ and $\mathsf{tid} = H(\mathsf{tid}_0, \mathsf{tid}_1)$. P_1 thus evaluates $\sigma = (S, c, r) = (g_0 g_1^m g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}} g_3^{\mathsf{uid}} g_4^{r_{\mathsf{vid}_1}} g_4^{\hat{r}}, c, \hat{r} + r_{\mathsf{tid}} + r_{\mathsf{vid}_1})$. Thus, the signature verification equation checked by P_1, as well as the hash verification equation of tid, passes and P_1 gives the good to P_0. If P_0 does not send $\tilde{\sigma}$ to P_1, then by cases 1 and 2 of Lemma 2, $\mathsf{bal}_0^f = \mathsf{bal}_0^i$ and thus the claim holds vaccuously. □

Lemma 4 ($\mathbf{SF}_{P_1}$). *If an honest P_1 gives the good to P_0, then $\mathsf{bal}_1^f \geq \mathsf{bal}_1^i + m$.*

Proof. Suppose P_1 gives the good (sends "success") to P_0. In this case, P_1 initiates encashment requests against tid (retrying on failure), so one such request must be eventually received by B_1 under the partial synchrony model. The assert condition on σ checked by B_1 at this point is exactly the same as the verification done by P_1 before it gives the good to P_0, so $\mathsf{encash}(\mathsf{tid})$ must be executed. Further, since P_1 had verified that $\mathsf{tid} = H(\mathsf{tid}_0, \mathsf{tid}_1)$, tid must be fresh by the collision-resistance property of H, as at least one of its inputs tid_1 is fresh with a high probability. Thus, $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{encashed}}$.

Now suppose for contradiction that $\mathsf{tid} \in \mathsf{TID}_{\mathsf{cancelled}}$. Thus, the $\mathsf{cancel}(\mathsf{tid})$ transaction must have executed. Thus, B_0 must have received a cancellation request ("CancelReq", ctid, tid, r_{tid}, σ_c) such that $\mathsf{ctid} = g_2^{\mathsf{tid}} g_4^{r_{\mathsf{tid}}}$ and $\mathsf{SPKVer}(\sigma_c, \mathsf{vid}_1', \mathsf{tid}) = 1$, where vid_1' was received by B_0 in a previous ("IssueReq", uid_0', us_0', m', ctid, $\pi_{\mathsf{ctid}'}$, vid_1', $\pi_{\mathsf{vid}_1'}$) request.

First, note that if vid_1' equals vid_1 created by P_1 then passing the check $\mathsf{SPKVer}(\sigma_c, \mathsf{vid}_1, \mathsf{tid}) = 1$ means that the adversary has broken the EUF-CMA security of the signature of knowledge, since it neither knows the opening $(\mathsf{uid}_1, r_{\mathsf{vid}_1})$ of vid_1 nor any signature of knowledge was given to it by P_1.

Now we consider the case that $\mathsf{vid}_1' \neq \mathsf{vid}_1$. By the knowledge soundness of $\mathsf{NIZKVer}$ for $\pi_{\mathsf{ctid}'}$ and $\pi_{\mathsf{vid}_1'}$, the adversary must know openings $(\mathsf{tid}', r_{\mathsf{tid}'})$ and $(\mathsf{uid}_1', r_{\mathsf{vid}_1'})$ such that $\mathsf{ctid} = g_2^{\mathsf{tid}'} g_4^{r_{\mathsf{tid}'}}$ and $\mathsf{vid}_1' = g_3^{\mathsf{uid}_1'} g_4^{r_{\mathsf{vid}_1'}}$. By the computational binding of Pedersen commitments, $(\mathsf{tid}', r_{\mathsf{tid}'})$ must equal $(\mathsf{tid}, r_{\mathsf{tid}})$ supplied during the cancel request. Thus, the adversary receives a BBS+ quasi-signature $\hat{\sigma}$ on the message $(m', \mathsf{tid}, \mathsf{uid}_1')$ under public key pk_{B_0}. Now, if $\mathsf{uid}_1' \neq \mathsf{uid}_1$ or $m' \neq m$, then making P_1 pass the signature verification condition against message $(m, \mathsf{tid}, \mathsf{uid}_1)$ implies that the adversary breaks the EUF-CMA security of the BBS+ signature scheme. Else, if $\mathsf{uid}_1' = \mathsf{uid}_1$ and $m' = m$ but $r_{\mathsf{vid}}' \neq r_{\mathsf{vid}}$, then the probability that $\tilde{\sigma}$ leads to P_1 passing the verification is negligible. Thus, $\mathsf{vid}_1' = g_3^{\mathsf{uid}_1'} g_4^{r_{\mathsf{vid}_1'}} = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}} = \mathsf{vid}_1$, which is a contradiction. Thus, $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{cancelled}}$.

Thus, the assert condition $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{encashed}} \cup \mathsf{TID}_{\mathsf{cancelled}}$ inside $\mathsf{encash}(\mathsf{tid})$ passes. Thus, $\mathsf{BAL}[\mathsf{bid}_1]$ at the end of the $\mathsf{encash}(\mathsf{tid})$ transaction is $\mathsf{bal}^i + m$ (the "revert $\mathsf{reclaim}(\mathsf{tid})$" step does not modify $\mathsf{BAL}[\mathsf{bid}_1]$). Since no further transactions involving bid_1 are executed, $\mathsf{bal}_1^f = \mathsf{bal}_1^i + m$. □

Lemma 5 ($\mathbf{FP}_{P_1}$). *If an honest P_1 sends σ_c to P_0 at the end of the* Transact *protocol (does not give the good to P_0), then an honest P_0 sends "ok" to P_1.*

Proof. This follows directly from the completeness of the signature of knowledge. □

Lemma 6 ($\mathbf{SD}_{B_0}$). *For any given invocation of the* Transact *protocol, the amount that B_0 debits from P_0's account is at least the credit liability of B_0 towards B_1, i.e.,* $(\mathsf{bal}_0^i - \mathsf{bal}_0^f) \geq (\mathsf{bal}_{B_1}^f - \mathsf{bal}_{B_1}^i)$.

Proof. Suppose for contradiction that there exists some invocation of the Transact protocol where the amount that B_0 effectively debits from P_0's account is less than the amount B_0 needs to credit against B_1. We consider the following cases:

Case I: No encash *transaction was executed.* Note that B_0's credit liability towards B_1 increases only during an encash transaction, so in this case this liability is zero. Thus, B_0 must effectively debit negative amount from P_0's account, i.e., effectively increase P_0's balance at the end of the protocol. Since P_0's balance can potentially increase only via frivolous ReclaimReq or CancelReq messages, we analyse them as below:

(1) A ("ReclaimReq", ctid, tid, r_{tid}) (resp. ("CancelReq", ctid, tid, r_{tid})) message is rejected if another ("ReclaimReq", ctid, tid′, r_{tid}') (resp. ("CancelReq", ctid, tid′, r_{tid}')) message resulted in a successful reclaim (resp. cancel) transaction earlier. Note that if $\mathsf{tid} = \mathsf{tid}'$, then the message is explicitly rejected inside the first assertion inside the $\mathsf{reclaim}(\mathsf{tid})$ (resp. $\mathsf{cancel}(\mathsf{tid})$)

transaction. The case $\mathsf{tid} \neq \mathsf{tid}'$ is prevented by the computational binding property of the Pedersen commitment ctid.

(2) If a $\mathsf{ReclaimReq}(\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}})$ request and a $\mathsf{CancelReq}(\mathsf{ctid}, \mathsf{tid}', r'_{\mathsf{tid}})$ are both sent, then at least one of them is rejected. As above, the case $\mathsf{tid} = \mathsf{tid}'$ is rejected by the first assertion inside the reclaim or cancel transaction initiated by the second such request. The case $\mathsf{tid} \neq \mathsf{tid}'$ is prevented by the computational binding of ctid.

(3) At most one of $\mathsf{ReclaimReq}(\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}})$ and $\mathsf{CancelReq}(\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}})$ requests can result in a reclaim or cancel transaction. This happens only if there exists a prior ("IssueReq$''$, ...) message that had successfully set $\mathsf{TXN}[\mathsf{ctid}] = (\mathsf{bid}_0, m, \dots)$, otherwise the $\mathsf{TXN}[\mathsf{ctid}]$ memory access before the transaction fails. In these cases, B_0 has to increase P_0's balance by amount m, but it also had debited amount m during the IssueReq request. Thus, the effective debit from P_0's account is zero (non-negative), leading to a contradiction.

Case II: Some $\mathsf{encash}(\mathsf{tid})$ *transaction was executed.* We first argue that for each such executed $\mathsf{encash}(\mathsf{tid})$ transaction, there must exist a distinct matching $\mathsf{issue}(\mathsf{ctid})$ transaction that debits amount m from P_0. We say that the $\mathsf{issue}(\mathsf{ctid})$ transaction *matches* an $\mathsf{encash}(\mathsf{tid})$ transaction if $m' = m$ and $\mathsf{tid}' = \mathsf{tid}$, where m' denotes the amount supplied in the "IssueReq" message resulting in the $\mathsf{issue}(\mathsf{ctid})$ transaction and tid' (along with some randomness r'_{tid}) is the opening of ctid extracted by the extractor of the supplied NIZK proof π_{ctid}. The argument goes as follows:

Suppose there does not exist any issue transaction matching with a given $\mathsf{encash}(\mathsf{tid})$ transaction. Then, the adversary would not have obtained any signature on a message $(m, \mathsf{tid}, \cdot)$ but must pass the BBS+ signature verification for a message of the form $(m, \mathsf{tid}, \cdot)$ right before executing the $\mathsf{encash}(\mathsf{tid})$ transaction. This leads to a contradiction under the EUF-CMA security of BBS+ signatures. Next, suppose there exist two distinct $\mathsf{encash}(\mathsf{tid})$ and $\mathsf{encash}(\mathsf{tid}')$ transactions matching a single issue transaction. The case $\mathsf{tid} = \mathsf{tid}'$ is explicitly disallowed by the assert condition inside the second encash transaction. The case $\mathsf{tid} \neq \mathsf{tid}'$ is prevented by the computational binding property of Pedersen commitment ctid.

Thus, let $\mathsf{issue}(\mathsf{ctid})$ be the unique payment transaction matching a given $\mathsf{encash}(\mathsf{tid})$ transaction. Since amount m is deducted from P_0's account during this transaction, it covers for B_0's credit liability towards B_1 as long as this money cannot be reclaimed or cancelled. As argued in case I above, at most one of a $\mathsf{ReclaimReq}(\mathsf{ctid}, \mathsf{tid}', r'_{\mathsf{tid}})$ and a $\mathsf{CancelReq}(\mathsf{ctid}, \mathsf{tid}', r'_{\mathsf{tid}})$ request against a given ctid can ever be accepted. Further, by the computational binding property of ctid, it must be that $\mathsf{tid}' = \mathsf{tid}$. We thus consider the following cases:

(1) *A* $\mathsf{cancel}(\mathsf{tid})$ *transaction got executed before the* $\mathsf{encash}(\mathsf{tid})$ *transaction.* This case is impossible because of the assertion $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{cancelled}}$ at the beginning of the $\mathsf{encash}(\mathsf{tid})$ transaction.

(2) *A* $\mathsf{reclaim}(\mathsf{tid})$ *transaction got executed before the* $\mathsf{encash}(\mathsf{tid})$ *transaction.* The reclaim transaction must have reclaimed exactly amount m debited during the $\mathsf{issue}(\mathsf{ctid})$ transaction. However, this reclaim is reverted during the $\mathsf{encash}(\mathsf{tid})$ transaction, resulting in an effective debit of amount m from P_0's account.

(3) *A* $\mathsf{reclaim}(\mathsf{tid})$ *transaction got executed after the* $\mathsf{encash}(\mathsf{tid})$ *transaction.* This case is impossible because of the assertion $\mathsf{tid} \notin \mathsf{TID}_{\mathsf{encashed}}$ at the beginning of the $\mathsf{reclaim}(\mathsf{tid})$ transaction.

(4) *A* $\mathsf{cancel}(\mathsf{tid})$ *transaction got executed after the* $\mathsf{encash}(\mathsf{tid})$ *transaction.* The cancel transaction in this case must have put back the amount m debited during the $\mathsf{issue}(\mathsf{ctid})$ transaction into P_0's account. However, it also reverts the $\mathsf{encash}(tid)$ transaction, leading to zero effective credit liability of B_0 towards B_1.

In all the cases, the credit liability of B_0 towards B_1 does not exceed the amount it effectively debits from P_0's account and thus we have arrived at a contradiction. □

LEMMA 7 ($\mathbf{LC}_{B_1}$). *The amount that B_1 needs to credit to P_1's account is at most the amount that B_0 credits to B_1, i.e.,* $(\mathsf{bal}_0^f - \mathsf{bal}_0^i) \leq (\mathsf{bal}_{B_1}^f - \mathsf{bal}_{B_1}^i)$.

PROOF. Note that B_1's credit liability towards P_1 is exactly the same as B_0's credit liability towards B_1, as set during a single encash transaction. Thus, irrespective of whether the encash transaction executes, does not get executed, or gets reverted, the claim holds trivially. □

LEMMA 8 (**PB**, **PI**). *Neither P_0 nor P_1's* bid *or* uid *leaks to anyone except their own banks B_0 and B_1 respectively. That is, if P_0 and B_0 are honest, then the view of all other parties can be simulated using just the transaction amount m and the information that P_0 holds a valid* bid_0 *and* uid_0. *If P_1 and B_1 are honest, then the view of all other parties can be simulated using the same information.*

PROOF. We first prove this for honest P_0 and B_0. First, because of secure channels between the banks, any information that P_0 shares with B_0 is strictly confidential. Next, the only piece of information that P_0 shares with either P_1 or B_1 is the tuple $(\tilde{\sigma}, \mathsf{tid}, \mathsf{tid}_0)$, none of which depends on uid_0 or bid_0. Thus, P_1 and B_1's view can be trivially simulated by a simulator that only obtains the transaction amount m and not any information about uid_0 or bid_0 (except that they are valid registered ids).

Now we prove it for an honest P_1 and B_1. As above, any information that P_1 shares with B_1 remains confidential due to secret channels. The only information that P_1 shares with P_0 or B_0 is $(\mathsf{tid}_0, \mathsf{vid}_1, \pi_{\mathsf{vid}_1}, m)$. By the perfect hiding property of Pedersen commitments, vid_1 can be simulated by a random sample from $\mathbb{G}_1$ and π_{vid_1} can be simulated by its corresponding ZK simulator. Further, tid_0 is uniformly random. Thus, P_0 and B_0's view can be simulated with simply the transaction amount m. □

LEMMA 9 (**PT**). *For any invocation of the* Transact *protocol, B_0 and B_1 cannot distinguish between P_0 interacting with P_1 or P_0 interacting with P_1', assuming the transaction amount m is the same and both payer and payee are honest in each interaction. Also, no external adversary identifies P_0 or P_1 or learns the amount m.*

Proof. Note that B_0 learns the tuple (uid_0, us_0, m, ctid, π_{ctid}, vid_1, π_{vid_1}) from the IssueReq message sent by P_0. Since P_0 and P_1/P_1' are both honest, we have the following cases:

Case I: A ("ReclaimReq", $\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}$) message is sent by P_0 but no ("EncashReq", ...) message is sent by P_1/P_1': By the perfect hiding property of Pedersen commitments, vid_1 is simulatable by sampling a random element from $\mathbb{G}_1$, and π_{vid_1} could be simulated using its ZK simulator. Since no further information about uid_1 is revealed as the amount was never encashed, the world where P_1 with identity uid_1 interacts with P_0 and the world where P_1' with identity uid_1' interacts with P_0 are indistinguishable.

Case II: A ("CancelReq", $\mathsf{ctid}, \mathsf{tid}, r_{\mathsf{tid}}, \sigma_c$) message is sent by P_0 but no ("EncashReq", ...) message is sent by P_1/P_1': This case follows similarly to the above, with the additional argument that σ_c can be simulated by the zero-knowledge property of the signature of knowledge.

Case III: An ("EncashReq", $\pi_\sigma, m, \mathsf{tid}, \mathsf{uid}_1$) message is sent by P_1/P_1' but no ("ReclaimReq", ...) or ("CancelReq", ...) message is sent by P_0: In this case, B_1 additionally learns $(\pi_\sigma, m, \mathsf{tid}, \mathsf{uid}_1)$. Here, π_σ can be simulated by its ZK simulator. Next, tid is indistinguishable to a random element from $\mathbb{Z}_q$ as ctid obtained above is perfectly hiding. Lastly, uid_1, obtained in the world where P_0 interacts with P_1, is indistinguishable from uid_1', obtained in the world where P_0 interacts with P_1', because vid_1 obtained above is perfectly hiding.

The last sentence of the lemma holds directly because of secure channels between the users and their banks. □

B Proofs for the Lemma in Section 6.2

Lemma 10 ($\mathbf{EDR}_{P_0}$). *If* $\mathsf{bal}_0^f < \mathsf{bal}_0^i$ *and P_0 is honest, then R obtains* $(\mathsf{vid}_1, \mathsf{tid}, m, \sigma_{\mathsf{rec}}, c_{\mathsf{uid}_1})$ *such that* $\mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1$*,* vid_1 *is a commitment of* $\mathsf{uid}_1 := \mathsf{Dec}(\mathsf{sk}_R, c_{\mathsf{uid}_1})$*, and* uid_1 *was used in an invocation to the* Register *protocol.*

Proof. By arguing similarly to the proof of Lemma 2, we can show that the condition $\mathsf{bal}_0^f < \mathsf{bal}_0^i$ is not satisfied for the following two cases and thus the claim holds vaccuously for these cases:

- *P_0 failed to verify π_{uid_1} or $\pi_{\sigma_{\mathsf{rec}}}$:* In this case, P_0 sends no "IssueReq" message. Thus, $\mathsf{bal}_0^f = \mathsf{bal}_0^i$ by the secure channel assumption and the confidentiality of secrets us_0 and bs_0.
- *P_0 did not receive $\hat{\sigma}$ from B_0 till timeout τ:* In this case, P_0 initiates a reclaim request. As argued in Lemma 2, eventually one such request is accepted, which successfully restores P_0's account balance.

Now, we argue the main case, where P_0 passes the above steps and gives $\tilde{\sigma}$ to P_1. In this case, P_0's final balance can potentially be less than their initial balance due to a potential encashment request by P_1. Note that both $\pi_{\sigma_{\mathsf{rec}}}$ and π_{vid_0} are verified by P_0 in this case. By the knowledge soundness of $\pi_{\sigma_{\mathsf{rec}}}$, there must exist an extractor $\mathcal{E}_{\sigma_{\mathsf{rec}}}$ that outputs σ_{rec} such that $c_{\sigma_{\mathsf{rec}}} = \mathsf{Enc}(\mathsf{pk}_T, \sigma_{\mathsf{rec}})$ and $\mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1$. By the knowledge soundness of π_{uid_1}, there exists a PPT extractor $\mathcal{E}_{\mathsf{uid}_1}$ that outputs $(\mathsf{uid}_1, r_{\mathsf{vid}_1}, \sigma_{\mathsf{uid}_1})$ such that $c_{\mathsf{uid}_1} = \mathsf{Enc}(\mathsf{pk}_R, \mathsf{uid}_1)$, $\mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}$ and $\mathsf{Ver}(\sigma_{\mathsf{uid}_1}, \mathsf{pk}_{B_1}, \mathsf{uid}_1) = 1$. We argue by the following two cases that P_0 outputs a valid σ_{rec}, i.e., one satisfying $\mathsf{SPKVer}(\sigma_{\mathsf{rec}}, \mathsf{vid}_1, (\mathsf{tid}, m)) = 1$, proving the first clause of the claim.

- *P_0 received a valid σ_{rec} from P_1 within timeout τ:* The claim holds trivially by the case assumption.
- *P_0 did not receive a valid σ_{rec} from P_1 within timeout τ:* In this case, P_0 contacts T with the information received so far. By the correctness of decryption, T obtains the same σ_{rec} as output by $\mathcal{E}_{\sigma_{\mathsf{rec}}}$. Thus, it is valid. Further, since P_0 (and B_0) are honest, the second assertion made by T also passes. Thus, T actually sends a valid σ_{rec} to P_0.

Further, since $c_{\mathsf{uid}_1} = \mathsf{Enc}(\mathsf{pk}_R, \mathsf{uid}_1)$, by the correctness of decryption, $\mathsf{uid} := \mathsf{Dec}(\mathsf{sk}_R, c_{\mathsf{uid}_1}) = \mathsf{uid}_1$. Thus, since $\mathsf{vid}_1 = g_3^{\mathsf{uid}_1} g_4^{r_{\mathsf{vid}_1}}$, the second clause of the claim holds. Now, if uid was never supplied during an invocation of the Register protocol, then the adversary breaks the EUF-CMA security of the signature scheme Sign/Ver, since $\mathsf{Ver}(\sigma_{\mathsf{uid}_1}, \mathsf{pk}_{B_1}, \mathsf{uid}_1) = 1$. Thus, the third clause of the claim holds too.

□