An Innovative Electronic Payment Framework for Secure Peer-to-Peer Transactions

Abhigyan Mukherjee
Email: abhigyan.mukherjee@yahoo.com

***Abstract*—With the growing reliance on peer-to-peer (P2P) networks for digital transactions, traditional electronic payment systems require enhancements to ensure security, efficiency, and trust. This study introduces an innovative digital payment framework enabling currency-based exchanges between consumers and vendors within a peer-to-peer environment. The outlined approach is inspired by Millicent's scrip methodology and leverages digital envelope encryption to bolster protection. Unlike conventional payment methods that heavily rely on financial institutions, the protocol minimizes their involvement, restricting their role to trust establishment and transaction finalization. The system introduces a distributed allocation model, where merchants locally authorize payments, reducing transaction overhead and enhancing scalability. Additionally, the protocol is optimized for repeated payments, making it particularly efficient for recurring transactions between the same buyer and merchant. By integrating cryptographic techniques and decentralizing payment authorization, this protocol presents a secure, efficient, and scalable solution for digital payments in P2P environments.**

I. Introduction

The swift growth of the web has driven the transformation of online trade, creating a virtual platform that enables the transfer of digital payments and exchange-related information across networks. The success of electronic commerce can be attributed to key characteristics of the Internet, such as its openness, high-speed connectivity, anonymity, digitization, and global reach (1; 2).

As of 2024, the Internet connects over 5.35 billion users worldwide, representing approximately 66.2% of the global population (3). Visionary online businesses such as Amazon.com (4) and eBay (5) recognized the immense commercial potential of this growing user base, offering global services for buying and selling goods through web-based platforms. These platforms operate on a centralized infrastructure, which ensures a certain level of security for users. The primary advantage of such a model lies in its ability to enforce rules effectively. However, centralized architectures also present critical challenges: they create a single point of failure and introduce scalability constraints due to bandwidth and computing resource limitations, leading to high infrastructure demands (6).

Moreover, this centralized approach is often impractical for small businesses or independent merchants that lack the financial capacity to support significant infrastructure costs. This challenge has paved the way for peer-to-peer (P2P) (7; 8; 9) systems as an alternative solution. The P2P paradigm is gaining traction as a distributed computing framework capable of leveraging the resources of edge devices, including personal computers and mobile devices, to create a decentralized and efficient network. P2P infrastructures inherently provide scalability and fault tolerance, as demonstrated by successful modern systems such as Venmo (10), which facilitates direct mobile payments between individuals while maintaining security and user convenience. Furthermore, P2P networks have become a foundational component of blockchain technology, where decentralized architectures facilitate secure and transparent transactions without reliance on a central governing entity (11; 12).

This paper introduces a novel electronic payment protocol that harnesses the potential of peer-to-peer (P2P) networks. Unlike traditional electronic commerce models, which depend heavily on financial institutions (13), the peer-to-peer model enables individuals to switch between buyer and seller roles. This distributed digital economy is ideal for virtual marketplaces where pre-owned goods are traded. Within this framework, every user can function as both a vendor and a purchaser utilizing only their own device (14). The proposed protocol ensures a fully anonymous, secure, and practical transaction framework where all peers can function as both sellers and buyers. Additionally, it integrates a comprehensive security mechanism that protects both personal and order-related information from unauthorized access. The core of this protocol builds on modern encryption principles, specifically leveraging the concept of a "digital envelope"—where a symmetric session key encrypts the payload and is itself encrypted with the recipient's public key—used in contemporary implementations of Transport Layer Security (TLS) (15).

The digital envelope ensures confidentiality while reducing the computational overhead typically associated with public key encryption. The recipient uses their private key to decrypt the symmetric session key and then decrypts the message efficiently. This technique enhances both performance and privacy, making it suitable for peer-to-peer payments in resource-constrained devices. Additionally, mechanisms for session-key encryption improve resistance to cryptographic attacks and provide forward secrecy when paired with ephemeral keys (16; 17).

To support anonymity and prevent double-spending, the protocol also draws conceptual inspiration from lightweight token systems such as Millicent's scrip mechanism (18), which includes serial numbers and merchant-specific credentials. Though Millicent is now outdated, its architectural concepts—like on-demand token issuance and merchant-specific certificates—remain influential. The proposed protocol inte-

grates modern equivalents of these principles through anonymous credentials and digital bearer tokens embedded with validity constraints.

Modern platforms such as Venmo, Cash App, and various blockchain-based apps showcase how peer-to-peer payments can function seamlessly across decentralized networks. These applications often employ TLS 1.3 for secure communications, JSON Web Tokens (JWTs) for authentication, and encrypted metadata for privacy. Unlike legacy Secure Socket Layer (SSL) systems—which are now deprecated—TLS ensures end-to-end encryption with improved handshake performance and stronger ciphers.

The role of TLS in secure transactions is essential but has its limitations:

1) TLS encrypts communication channels but does not digitally sign payment messages, limiting non-repudiation guarantees.
2) TLS does not prevent merchants from accessing sensitive buyer data such as bank account numbers unless complemented by tokenization or privacy-preserving protocols.
3) Unlike the proposed P2P model, TLS-based systems often rely on centralized certificate authorities, introducing trust dependencies.
4) Secure financial details and authentication credentials require additional application-level protections beyond what TLS offers.

An alternative peer-to-peer transaction mechanism of interest is PPay (19), which operates as an autonomous, lightweight payment framework utilizing dynamic, self-regulated electronic tokens in an offline setting. While PPay reduces the broker's involvement and operational load, it does so at the expense of security. Although this design enhances system performance significantly, it renders the protocol inadequate for medium-to-large transactions, unlike the proposed P2P payment model.

A more secure version of PPay, known as WhoPay (20), provides a structured infrastructure for secure electronic commerce transactions while maintaining anonymity between transacting parties. However, WhoPay necessitates a substantial database for storing transaction scripts and does not fully account for the inherent instability of P2P networks (9). The scalability of both PPay and WhoPay hinges on frequent transactions related to the transfer and renewal of digital scrips. These transactions necessitate the involvement of a third party acting as a substitute for the broker. If this third party is offline, the broker must intervene, thereby increasing its workload.

Consider, for instance, an electronic marketplace where customers sporadically enter the system to make purchases. In such a scenario, the intermittent participation of peer customers renders these protocols impractical. The proposed P2P payment model addresses these limitations by employing robust encryption, adaptive session management, and efficient consensus mechanisms.

Unlike traditional payment systems such as Venmo and CashApp that depend on centralized control and account-based identity verification, our protocol promotes pseudonymity, minimizes broker involvement post-initialization, and decentralizes trust in a scalable way.

A. Motivation and Problem Statement

Existing peer-to-peer (P2P) payment systems such as Venmo and CashApp rely heavily on centralized servers, account-bound identities, and third-party trust models. These systems often compromise user privacy, incur high computational and storage overheads on central entities, and are vulnerable to single points of failure.

Our proposed protocol addresses these issues by introducing a decentralized, token-based transaction model where users maintain control over their identity via pseudonymous credentials. Trust is established through cryptographic tokens, while the broker—typically a financial institution—is involved only in essential operations such as issuing and redeeming digital cash. This design not only enhances privacy and scalability but also reduces the broker's computational burden during peer-to-peer exchanges.

II. Proposed Electronic-Payment Protocol

This paper introduces a novel electronic-payment protocol involving three principal entities:

- **Customer**: The party initiating the payment.
- **Merchant**: The entity receiving the payment.
- **Acquirer Gateway**: An intermediary bridging electronic payments with traditional financial infrastructures. It is responsible for transaction authorization (21). This entity will henceforth be referred to as **the broker**.

The broker's role is essential for establishing trust among transacting entities. However, its presence introduces the risk of a *single point of failure* (22), a common issue in client/server-based payment systems. Although the broker cannot be entirely eliminated due to its financial and security significance, its participation in transactions has been minimized within the proposed protocol to mitigate associated risks (23).

The remainder of this paper is structured as follows: Section 2 introduces the involved entities, key definitions, and notation (24). Section 3 elaborates on the user registration mechanism and public key exchange process. Section 4 outlines potential security threats, while Section 5 details the security requirements for each entity. The payment process is discussed in Section 6. In Section 7, the computational burden on the broker is analyzed. Finally, Section 8 presents concluding remarks.

A. Protocol Terminology

The following terms define the components utilized in the protocol (25):

1) ***AcctNum***: Represents the customer's bank account number.
2) ***AcctSecret***: A composite secret comprising two elements: the broker's confidential key and the peer customer's or peer merchant's private key. Both the broker and the respective peer possess this shared secret (26).

3) ***PID***: A distinct identifier associated with the peer customer or merchant, used for authentication. It is computed as: Hash(AcctSecret—Hash(AcctSecret)—AcctNum) (16).
4) ***PeerID***: A unique identifier assigned to each participant (peer) in the protocol, ensuring anonymity and preventing the disclosure of personal identity information (27).
5) ***BankToken***: A form of electronic currency issued by the broker (bank) (28).
6) ***MerchantToken***: A digital currency issued by a vendor, which can only be utilized within that merchant's system (29).
7) ***TokenStructure***: Comprises multiple fields, as illustrated in Figure 1.

B. Symbols and Notations

Table I presents the notation used for cryptographic operations within the protocol. Additionally, Table I provides an overview of the fundamental message elements used in the payment framework (30).

TABLE I
NOTATION OF CORE MESSAGE ELEMENTS

Symbol	Definition
M_i	Message label
P_{ID_i}	Unique identifier of the peer participant
V_t	Value assigned to the BankToken, MerchantToken, or transaction item
Rnd	Randomly generated nonce
PID_i	Unique identifier of the customer's or merchant's financial account
BT_j	BankToken
MT_j	MerchantToken
Sec_t	Corresponding security key associated with BankToken or MerchantToken
$Auth$	Authorization status, where $Auth = OK$ or NOK
OI	Order details (product name, price, quantity, unique transaction ID)
Msg	Generic information message

III. PUBLIC KEY INFRASTRUCTURE

The proposed decentralized payment protocol is fundamentally built upon public key cryptography, requiring a robust and reliable mechanism for verifying users' public keys. In this system, a central certification authority (CA)—referred to as the broker—is introduced, possessing a private key for signing and encryption. All participating entities (customers and merchants) possess the corresponding public key of the broker for signature verification and secure communication.

While this design facilitates streamlined trust management, it also introduces significant risks due to its reliance on a single point of trust. If the broker's private key is compromised, it could lead to mass impersonation, forgery of digital certificates, and data breaches across the system. Furthermore, the secure distribution and verification of the broker's public key are paramount—any tampering with this process can expose the system to man-in-the-middle (MITM) attacks. Additionally, without a clearly defined key revocation mechanism, the system lacks the agility to respond to threats in real-time.

Beyond cryptographic risks, operational concerns such as performance bottlenecks, scalability limitations, and insider threats also challenge the broker's role. As user adoption increases, relying on a single entity to verify, store, and manage all key-pair credentials creates overhead and reduces fault tolerance.

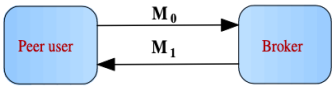
To address these challenges, modern trust frameworks advocate for more decentralized approaches such as Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs). These technologies enable users to generate and control their identifiers and associated public keys independently, reducing reliance on centralized authorities. Trust is established through cryptographic proofs and verifiable presentations, which can be selectively disclosed and cryptographically verified without requiring a central broker to mediate all transactions.

A. Peer Enrollment

When a peer entity (customer or merchant) initiates a request to create an account, their confidential details (e.g., account number and PANSecret) are securely stored in the broker's database. Additionally, a cryptographic key pair (public/private key) is generated and saved locally in the user's secure device environment. The peer then submits their public key, along with a unique user ID, to the broker to complete the registration process.

This process—illustrated in Figure 1—includes:

- Generation of the enrollment request R_0;
- Confidential data such as PAN and identity information (ID_C);
- Asymmetric encryption using a randomly generated session key K_0;
- Digital signatures to ensure authenticity.



C_0	*Registration request*
X_0	PAN
M_0	**C_0, UID_P, $Enc_{K0}(Sign_{KP}(X_0))$, $Enc_{K0}(SignOnly_{KP}(ID_C), K_P)$, $PKEnc_{KB}(K_0)$**
C_1	*Registration response*
X_1	C_1, UID_B, I
M_1	**X_1**

Fig. 1. Peer Enrollment

The user constructs:

$$\begin{aligned} R_0 &= \text{Enrollment request} \\ U_{ID_P} &= \text{Peer user's unique identifier} \\ PAN &= \text{Peer's bank account number} \\ ID_C &= \text{Peer's identity data} \\ K_{P_u} &= \text{Peer's public key} \\ K_{P_r} &= \text{Peer's private key} \\ K_B &= \text{Broker's public key} \\ K_0 &= \text{Randomly generated symmetric key} \\ Y_0 &= PAN \end{aligned} \tag{1}$$

and transmits to the broker:

$$M_0 = \big(R_0, U_{ID_P}, \text{Enc}_{K_0}(\text{Sign}_{K_{P_r}}(Y_0)), \text{Enc}_{K_0}(\text{SignOnly}_{K_{P_r}}(ID_C)), \text{Enc}_{K_0}(K_{P_u}), \text{PKEnc}_{K_B}(K_0)\big) \quad (2)$$

If the broker confirms the user's uniqueness and data validity, it responds with:

R_1 = Enrollment Confirmation,
U_{ID_B} = Broker's unique identifier

$$Y_1 = \{R_1, U_{ID_B}, I\}, \quad M_1 = Y_1 \quad (3)$$

B. Public Key Retrieval

Secure peer-to-peer communication mandates the retrieval of a counterparty's public key (e.g., customer requesting merchant's public key). Since the broker maintains a public key directory, it also acts as a lookup service during key discovery—although this centralization adds additional trust and latency risks.

Figure 2 illustrates this interaction:

- The requester signs and encrypts their query (M_0) to ensure authenticity and privacy.
- The broker validates the signature and retrieves the target public key from its database.
- The broker returns M_1, containing the target peer's public key, digitally signed to prevent tampering.

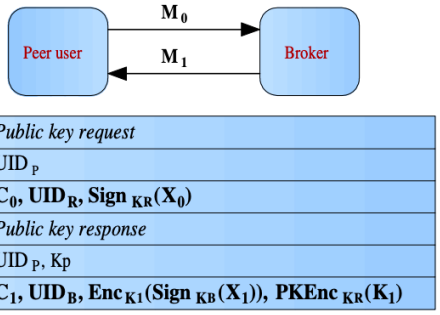


Fig. 2. Public Key Retrieval

While this method ensures confidentiality and authenticity, it reiterates the dependency on the broker's availability and trustworthiness. In future iterations, replacing the broker's key directory with DID-based registries can significantly reduce such dependencies. These registries allow peers to publish their identifiers and public keys in decentralized, verifiable, and immutable environments (e.g., blockchain or distributed ledgers), thereby improving transparency and fault tolerance in key management.

IV. Adversaries and Threats

In this context, I consider three distinct adversaries:

1) **Eavesdropper:** An entity that intercepts communications with the intention of extracting sensitive data, such as financial credentials, PAN information, and unique identifiers.
2) **Active Attacker:** A malicious entity that injects fraudulent messages into the system to manipulate its behavior.
3) **Insider:** A legitimate user or an entity that has gained access to a legitimate user's confidential data. For example, a dishonest vendor attempting to receive unauthorized payments from a customer.

The Internet is an open and decentralized network, where no single entity has complete control over network resources and functionalities. Consequently, there is always a risk that communication between trusted entities could traverse systems controlled by adversaries. Additionally, existing routing mechanisms lack inherent safeguards against cyber threats. Thus, message confidentiality and authentication cannot be presumed unless appropriate cryptographic protocols are implemented.

Another crucial concern is the reliability of merchants engaged in online transactions. The digital marketplace includes numerous small-scale vendors, often referred to as the cottage industry. It is relatively simple for an adversary to establish a fraudulent online storefront to harvest customers' sensitive data ((31)). To mitigate this risk, customer identifiers must be securely transmitted to the broker without exposure to merchants. The merchant, in turn, only requires an authorization token from the broker to complete the transaction.

Furthermore, I identify three primary forms of attacks that could be initiated by customers or external adversaries: double-spending, faulty scrip generation, and scrip forgery. These attacks are described as follows:

- **Double Spending:** A scrip is generated using two confidential values: the *MasterScripSecret* and the *MasterCustomerSecret*, which remain exclusively known to the scrip issuer. To prevent reuse, each time a scrip is redeemed, its associated secrets are deleted from the issuer's records, ensuring that it cannot be utilized for another transaction.
- **Faulty Scrip Generation:** In the payment protocol, participants can function as both customers and merchants, and they have the ability to create scrips. However, any generated scrip remains valid only for transactions with its originating entity, as it embeds the unique identifier of the producer (Figure 1).
- **Scrip Forgery:** A scrip consists of two components: a scrip body that holds transaction details and a cryptographic certificate that serves as its signature. Unauthorized modifications to the scrip body can be detected by verifying the integrity of the scrip's signature.

V. Security Requirements

A. Issuer/Acquirer Security Measures

It is assumed that the issuer and acquirer share a baseline level of trust. Furthermore, a secure communication infrastructure is already in place between these entities, enabling them to collaborate securely. As a result, their roles and associated security requirements are unified.

- *Verifiable Proof of Customer Authorization:* Before the broker registers a debit transaction from a customer's

bank account, it must possess irrefutable proof that the account holder has authorized the payment. This proof should be non-replayable, preventing reuse for any other transaction. Additionally, since merchants are considered potential adversaries, they must not be able to fabricate unauthorized transactions.

- *Verifiable Proof of Merchant Legitimacy:* When a broker approves a payment to a specific merchant, it must retain a tamper-proof record indicating that the customer initiated the payment request and that the merchant is a verified entity.

B. Merchant Requirements

- *Validation of Broker's Transaction Authorization*: The merchant must secure indisputable proof that the broker has sanctioned the transaction.
- *Validation of Customer's Transaction Authentication*: Before seeking authorization from the broker, the merchant must ensure that the customer has explicitly verified the transaction request. Additionally, the merchant is responsible for confirming the authenticity of the customer's intent prior to relaying any payment-related message to the broker.

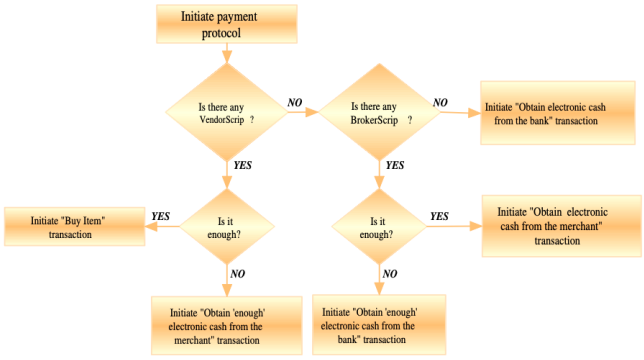


Fig. 3. Flowchart illustrating the payment protocol

C. Customer Requirements

- *Anonymity*: Unlike centralized payment systems such as Venmo or Cash App, where user identities are tied to real-world accounts and all transactions are logged on a central server, our protocol uses pseudonymous tokens and cryptographic identifiers to preserve user anonymity. The customer is identified only via a one-time digital signature and token-specific secrets during a transaction, which are invalidated post-verification. Moreover, trust is evaluated through the secure exchange of digital scrip and not through static identity checks. This ensures that customers can transact securely without revealing their long-term identity to the merchant or broker. Each transaction's integrity is cryptographically enforced, and since the script is non-reusable and time-stamped, it is immune to replay attacks.
- *Privacy*: The protocol ensures that customers' order details and payment information remain confidential. For instance, an investor acquiring stock-related data may not wish for competitors to gain insights into their interests. Encryption mechanisms safeguard this information, preserving customer privacy. However, it should be noted that the protocol does not guarantee unlinkability between customers and merchants concerning the broker.
- *Prevention of Unauthorized Charges*: The system must prevent unauthorized debits from a customer's bank account. Transactions should only be processed with valid credentials, including the bank account number, PAN-Secret, and the corresponding private key. Malicious entities, whether rogue actors on the internet or dishonest merchants, must be unable to fabricate fraudulent transactions that could gain broker approval. This security measure must hold even if the customer has participated in numerous legitimate transactions in the past. Specifically, sensitive customer identifiers should not be transmitted in plaintext and must be protected against brute-force attacks.
- *Verification of Transaction Authorization by Broker*: Customers may require proof that the broker has authorized a given transaction.
- *Merchant Authentication*: Customers should be able to verify that the merchant is a legitimate participant in the payment system.
- *Purchase Receipt*: Since the broker maintains a complete log of all transactions, issuing a receipt is optional.

VI. Payment Processing

The subsequent sections outline the foundational steps of the proposed decentralized payment framework, illustrated through a use case involving a virtual marketplace for pre-owned items. These preliminary actions are essential for fostering trust between buyers and sellers.

In the first stage, termed "Acquire digital funds from the financial institution," the purchaser secures virtual money by performing a singular large-scale transaction. Following this, during the "Exchange digital funds with the vendor" phase, a fraction of the obtained assets is traded for an equivalent sum in the seller's digital currency, once again executing a high-value transfer.

A detailed flowchart depicting the payment process is illustrated in Figure 3.

A. Acquiring BrokerScrip from the Bank

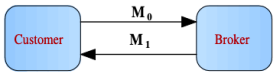
A customer intending to make purchases in the electronic marketplace must first obtain BrokerScrip, which is later exchanged for VendorScrip to complete the transaction. This process begins when the customer connects with the broker and purchases the required amount of BrokerScrip using real currency (Figure 4). Once the transaction is validated, the broker provides the customer with a unique BrokerScrip. Notably, a user can possess only a single active BrokerScrip, which must be fully utilized before requesting another. This

BrokerScrip functions as an instrument for acquiring digital currency from a seller.

In transaction message M_0, the combination of the customer's digital signature and unique identification provides strong verification for the broker, ensuring that the transaction request is legitimate. Additionally, a nonce is included to prevent replay attacks, while encryption safeguards the customer's identity and ensures message integrity.

Once the transaction request is validated, the broker generates the corresponding BrokerScrip and logs the transaction details for future reference, especially in the event of a dispute. The broker then sends the issued BrokerScrip in a new transaction message, M_1. This message includes the broker's digital signature, confirming the authenticity of the issued BrokerScrip. Moreover, the inclusion of a nonce ensures that the message is not a replayed transaction. Encryption mechanisms provide an added layer of confidentiality.

As depicted in Figure 4, the process illustrates...



C_0	*BrokerScrip request*
X_0	W_{Br}, N
M_0	**C_0, UID_C, Enc_{K0}(SignOnly $_{KC}$(ID_C),Sign $_{KC}$(X_0)), PKEnc $_{KB}$(K_0)**
C_1	*BrokerScrip response*
X_1	B_0, CS_{B0}, N
M_1	**C_1, UID_B, Enc_{K1}(Sign $_{KB}$(X_1)), PKEnc $_{KC}$(K_1)**

Fig. 4. Acquiring BrokerScrip from the Bank

Upon receiving M_1, the customer decrypts the message, verifies the broker's signature, and ensures that the value of the received BrokerScrip aligns with the requested amount. If all validations succeed, the customer securely stores the following encrypted values:

$$PKEnc_{K_C}(B_0) \tag{4}$$

$$PKEnc_{K_C}(CSB_0) \tag{5}$$

B. Converting BrokerScrip into VendorScrip

Each merchant exclusively accepts VendorScrip issued by them. Therefore, a customer who possesses BrokerScrip but lacks VendorScrip must initiate a conversion process before making a purchase. If the BrokerScrip's value meets or exceeds the desired VendorScrip amount, the exchange transaction proceeds as illustrated in Figure 5.

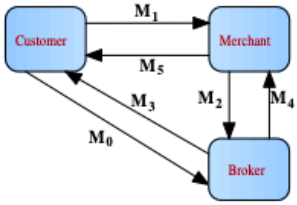
In transaction message M_0, the customer provides their digital signature, identification details, and the associated CustomerSecret of the BrokerScrip. This information enables the broker to verify the customer's authorization of the transaction. If valid, the broker logs the details for record-keeping.

Subsequently, the customer sends another message, M_1, to the merchant. This message, containing the customer's digital signature, identification, and BrokerScrip details, serves as proof of authorization. The broker ensures that the message is not a replayed attempt by verifying the uniqueness of the BrokerScrip. Encryption ensures the confidentiality of transaction details, and only the broker can decrypt the customer's ID. The merchant processes only their portion of the message:

$$(C_1, U_{ID_C}, \text{Enc}_{K_2}(\text{Sign}_{K_C}(X_2)), \text{PKEnc}_{K_M}(K_2)). \tag{6}$$

Upon receiving the message, the merchant decrypts the content and validates the customer's signature. The signature provides proof that the customer has authorized the transaction. If the received data is verified successfully, the merchant constructs M_2 and sends it to the broker. This message contains the merchant's digital signature and identification details, verifying that the merchant approves the transaction. Additionally, encryption ensures the confidentiality of transaction details, particularly the merchant's identification, and guarantees that only the broker can access the relevant information.

When the broker receives M_2, they decrypt the message, authenticate the signatures, and validate the associated transaction details, including the BrokerScrip and the parties involved. They also confirm that the transaction values match.



C_0	*Initiate VendorScrip request*
X_0	UID_M
M_0	**C_0, UID_C, Enc_{K0}(SignOnly $_{KC}$(ID_C)), Enc_{K0} (Sign $_{KC}$(X_0)), PKEnc $_{KB}$(K_0)**
C_1	*VendorScrip request*
X_1	B_0, CS_{B0}, W_{V0}
X_2	W_{V0}
M_1	**C_1, UID_C, Enc_{K1}(SignOnly $_{KC}$(ID_C)), Enc_{K1}(Sign $_{KC}$(X_1)), PKEnc $_{K0}$(K_1), Enc_{K2}(Sign $_{KC}$(X_2)), PKEnc $_{KM}$(K_2)**
C_2	*Authorization request*
X_3	W_{V0}
M_2	**C_2, UID_M, Enc_{K3}(SignOnly $_{KM}$(ID_M), Sign $_{KM}$(X_3)), PKEnc $_{KB}$(K_3), Enc_{K1}(SignOnly $_{KC}$(ID_C),Sign $_{KC}$(X_1)), PKEnc $_{KB}$(K_1)**
C_3	*Change BrokerScrip*
X_4	B_1, CS_{B1}
M_3	**C_3, UID_B, Enc_{K4}(Sign $_{K3}$(X_4)), PKEnc $_{KC}$(K_4)**
C_4	*Authorization response*
X_5	R
M_4	**C_4, UID_B, Sign $_{K5}$(X_5)**
C_5	*VendorScrip response*
X_6	V_0, CS_{V0}
M_5	**C_5, UID_M, Enc_{K5}(Sign $_{KM}$(X_6)), PKEnc $_{KC}$(K_5)**

Fig. 5. Converting BrokerScrip into VendorScrip

A transaction is considered legitimate if the worth of the BrokerScrip meets the required VendorScrip amount. Upon successful validation, the broker formulates two distinct messages: one directed to the customer (M_3) and another intended for the merchant (M_4). Additionally, the broker's records are modified to reflect the transaction details.

In M_3, the broker's digital signature serves as proof to the customer that the message originates from an authorized source. Encryption mechanisms ensure that the transmitted details remain confidential.

Likewise, in M_4, the broker's digital endorsement confirms to the merchant that the exchange is authentic. Upon obtaining this communication, the merchant validates the broker's authentication. If the verification succeeds, the merchant formulates M_5 and transmits it to the customer; otherwise, the process is halted.

For enhanced security, message M_5 includes the merchant's digital signature, offering the customer verifiable proof that the transaction was genuinely authorized. Moreover, the use of a session key for encryption ensures the confidentiality of all transmitted data.

Upon receiving two messages—M_3 from the broker and M_5 from the merchant—the customer proceeds to decrypt both and validate their digital signatures. Once the data is verified and found to be correct, the customer securely archives the following encrypted items:

$$\text{PKEnc}_{K_{User}}(VT_1) \tag{7}$$

$$\text{PKEnc}_{K_{User}}(\text{VCert}_1) \tag{8}$$

$$\text{PKEnc}_{K_{User}}(BT_2) \tag{9}$$

$$\text{PKEnc}_{K_{User}}(\text{BCert}_2) \tag{10}$$

C. Purchase Transaction

When a customer possesses the appropriate *MerchantScrip* necessary for acquiring an item from a vendor, they initiate the transaction by transferring it to the merchant (Figure 6). Upon receiving the scrip, the merchant validates its authenticity, deducts the required amount, and generates a new scrip reflecting the remaining balance, which is sent back to the customer. This process signifies the successful completion of the payment.

In M_0, the inclusion of a digital signature serves as proof to the broker that the customer has indeed authorized the transaction and that the transmitted data remains unaltered. When the broker receives this message, it verifies the signature and logs the transaction details into a secure record.

In M_1, the presence of *CustomerKey*, alongside the customer's digital signature, assures the merchant of the transaction's legitimacy. Additionally, encryption mechanisms safeguard the confidentiality of the exchanged information, allowing the merchant to detect any unauthorized modifications. Upon receiving the encrypted message, the merchant decrypts the contents, validates the signature, and verifies the provided *MerchantScrip*. Once confirmed, the merchant issues the remaining balance as a new *MerchantScrip* in a subsequent message (M_2). Finally, an informational message is dispatched to the broker (M_3).

In M_2, the merchant's digital signature guarantees to the customer that the transaction has been properly authorized. Encryption further ensures confidentiality during transmission. When the customer receives this message, they decrypt its contents, verify the signature, and validate the amount of the newly issued *MerchantScrip*. The securely stored values, represented as $\text{PKEnc}_{K_U}(V^*)$ and $\text{PKEnc}_{K_U}(\text{CSV}^*)$, are retained locally for future reference.

For M_3, the broker is assured of the message's integrity and the merchant's authorization through digital signature verification. Upon successful validation, the broker retrieves the corresponding log entry and updates the transaction record accordingly.

D. Acquiring Sufficient Electronic Cash from the Bank

The customer always possesses a single instance of *BrokerScrip*, which serves as a medium for multiple transactions involving the acquisition of *VendorScrips*. Whenever the value of the desired *VendorScrip* exceeds the available *BrokerScrip*, a transaction is initiated (Figure 10).

In message M_0, the customer provides their identification, the scrip's *CustomerSecret*, and a digital signature, ensuring that the broker can verify the legitimacy of the request. Encryption mechanisms are employed to maintain the confidentiality of the transmitted data. Furthermore, as the scrip is designed to be non-reusable, the broker is protected against replay attacks. Upon successful validation of the transmitted data, the broker constructs a new message, denoted as M_1, transmits it back to the customer, and logs the transaction details for record-keeping.

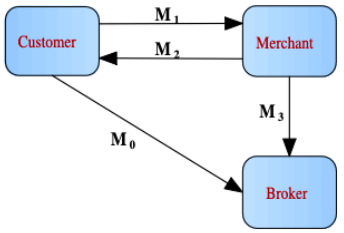
Within message M_1, the broker's digital signature acts as a robust proof that the transaction has been authorized. Additionally, encryption guarantees that the contents of the communication remain confidential.

Upon receiving M_1, the customer decrypts its elements, verifies the broker's signature, and confirms that the allocated amount of *BrokerScrip* corresponds to the requested value. Subsequently, the customer securely stores the encrypted components $PKEnc_{K_C}(B_1)$ and $PKEnc_{K_C}(CSB_1)$ locally for future use.

E. Acquiring Sufficient Electronic Cash from the Merchant

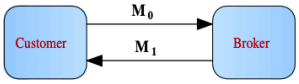
In a scenario where a customer has already engaged in transactions with a merchant and possesses VendorScrip issued by this specific merchant but requires additional funds because the item's value surpasses the available VendorScrip balance, a supplementary transaction is initiated (Figure 11).

The customer generates message N_0 and transmits it to the broker. The corresponding CustomerSecret associated with the scrip (BrokerScrip/VendorScrip), along with the digital



C_0	*Initiate Purchase request*
X_0	UID_M, W_P
M_0	**$C_0, UID_C, Sign_{KC}(X_0)$**
C_1	*Purchase request*
X_1	V_0, CS_{V0}, OI
M_1	**$C_1, UID_C, Enc_{K0}(Sign_{KC}(X_1)), PKEnc_{KM}(K_0)$**
C_2	*Purchase response*
X_2	V_1, CS_{V1}, OI
M_2	**$C_2, UID_M, Enc_{K1}(Sign_{KC}(X_2)), PKEnc_{KC}(K_1)$**
C_3	*Purchase request initiated*
X_3	UID_C, W_P
M_3	**$C_3, UID_M, Sign_{KM}(X_3)$**

Fig. 6. Buy Item



C_0	*"Enough" BrokerScrip request*
X_0	$W_{B1}, {}_{*0}, CS_{B0}$
M_0	**$C_0, UID_C, Enc_{K0}(SignOnly_{KC}(ID_C)), Sign(X_0)), PKEnc_{KB}(K_0)$**
C_1	*"Enough" BrokerScrip response*
X_1	B_1, CS_{B1}
M_1	**$C_1, UID_B, Enc_{K1}(Sign_{KB}(X_1)), PKEnc_{KC}(K_1)$**

Fig. 7. Obtain "enough" electronic cash from the bank

signature, serves as authentication proof, confirming that the customer has authorized the transaction. Furthermore, encryption mechanisms ensure both confidentiality and data integrity.

Upon receipt of the message, the broker processes the designated portion. The broker decrypts the relevant components, verifies the digital signature, and authenticates the provided BrokerScrip. Additionally, the broker evaluates whether the received BrokerScrip holds a value sufficient to cover the required difference $(X_1 - X_2)$ in VendorScrip. If all conditions are met, the broker constructs message N_1, forwards it to the merchant, and locally records the received BrokerScrip, including any remaining balance. The transaction details are subsequently stored in a log file.

In message N_1, the broker's digital signature functions as evidence to assure the merchant that the broker has authorized the transaction. Encryption techniques maintain the confidentiality of the transmitted information.

Upon receiving N_1, the merchant processes the concatenated message, which consists of two segments: the broker's newly generated message and the original message forwarded from the customer. The merchant first handles the broker's portion by decrypting its elements and verifying the signature. If the message elements are valid, the merchant then processes the customer's portion by decrypting its contents, verifying the digital signature, and authenticating the VendorScrip. If all elements are validated, the merchant ensures that X_1 and X_3 are equal.

If these values align, the merchant generates two response messages: N_3 for the customer and N_2 for the broker.

In message N_3, the merchant's digital signature serves as confirmation for the customer that the transaction has been approved. Additionally, encryption is applied to maintain data confidentiality.

Message N_2 contains the merchant's digital signature, ensuring the broker that the merchant has officially approved the transaction. The broker verifies the signature and, upon successful validation, deletes the temporarily stored BrokerScrip. If any residual balance exists in BrokerScrip, the broker constructs message N_4, transmits it to the customer, and subsequently removes the temporary record of the change. Finally, the broker updates the transaction log file accordingly.

Message N_4 reassures the customer that the broker has validated and approved the transaction, as demonstrated by the broker's digital signature. Encryption mechanisms continue to safeguard data integrity and confidentiality.

Upon successful processing, the customer receives the merchant's confirmation message (N_3).

The customer decrypts the received elements and validates the accompanying digital signature. Subsequently, they verify whether the value of the received *VendorToken* matches the requested one. Once confirmed, the customer securely stores $PKEnc_{K_C}(V_2')$ and $PKEnc_{K_C}(CSV_2')$ for future reference.

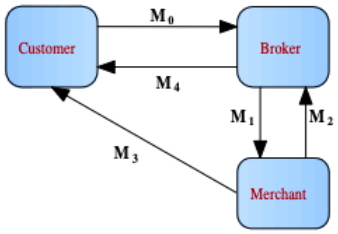
Additionally, the customer receives the broker's message (M_4'), decrypts its components, and verifies its digital signature. Further, they ensure that the received *BrokerToken* corresponds to the expected value. Upon successful validation, the customer securely stores $PKEnc_{K_C}(B_2')$ and $PKEnc_{K_C}(CSB_2')$ in their local system.

F. BrokerToken Withdrawal

When the customer no longer wishes to make purchases in the electronic marketplace, they have the option to withdraw their *BrokerToken* and transfer its value back to their associated bank account (Figure 12).

In message M_0', the *TokenSecret* alongside the customer's unique identifier and digital signature serves as proof of authorization for the transaction. Encryption mechanisms ensure both data confidentiality and integrity. Moreover, since the scrip is non-reusable, any replay attacks can be effectively identified and mitigated.

Upon receiving this message, the broker decrypts its contents and validates the digital signature. The broker then cross-checks the customer's identity and verifies the *BrokerToken*. If all details are authenticated, the transaction is recorded, and the associated *MasterTokenSecret* and *MasterCustomerSecret* are

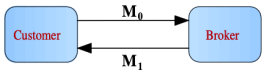


C_0	*"Enough" VendorScrip request*
X_0	$W_{V1}, W_V, UID_M, B_0, CS_{B0}$
X_1	W_{V1}, V_0, CS_{V0}
M_0	**$C_0, UID_C, Enc_{K0}(Sign_{KC}(X_0)), PKEnc_{KB}(K_0), Enc_{K1}(Sign_{KC}(X_1)), PKEnc_{KM}(K_1)$**
C_1	*Authorization request*
X_2	W_{V1}, UID_C
M_1	**$C_1, UID_B, Enc_{K2}(Sign_{KB}(X_2)), PKEnc_{KM}(K_2), Enc_{K1}(Sign_{KC}(X_1)), PKEnc_{KM}(K_1)$**
C_2	*Authorization response*
X_3	A, UID_C
M_2	**$C_2, UID_M, Sign_{KM}(X_3)$**
C_3	*"Enough" VendorScrip response*
X_4	V_1, CS_{V1}
M_3	**$C_3, UID_M, Enc_{K3}(Sign_{KM}(X_4)), PKEnc_{KC}(K_3)$**
C_4	*Change BrokerScrip*
X_5	B_1, CS_{B1}
M_4	**$C_4, UID_B, Enc_{K4}(Sign_{KB}(X_5)), PKEnc_{KC}(K_4)$**

Fig. 8. Obtain "enough" electronic cash from the merchant

securely erased. The broker then sends an acknowledgment message (M_1') to the customer, where the digital signature assures the customer that the transaction has been successfully authorized.

Finally, upon receiving this confirmation message, the customer verifies the signature's authenticity. Once verified, they permanently delete the withdrawn *BrokerToken* and its corresponding *TokenSecret* from their local storage.

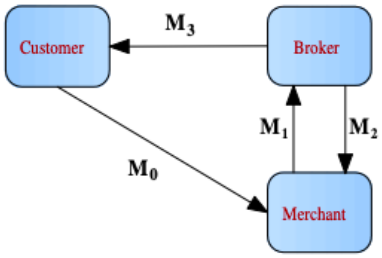


C_0	*BrokerScrip withdraw request*
X_0	B_0, CS_{B0}
M_0	**$C_0, UID_C, Enc_{K0}(SignOnly_{KC}(ID_C)), Enc_{K0}(Sign_{KC}(X_0)), PKEnc_{KB}(K_0)$**
C_1	*BrokerScrip withdraw response*
X_1	I
M_1	**$C_1, UID_B, Sign_{KB}(X_1)$**

Fig. 9. BrokerScrip withdraw

G. VendorScrip Withdrawal

Customers also have the option to withdraw their VendorScrip and transfer its equivalent value back into their bank accounts (Figure 10).



C_0	*VendorScrip withdraw request*
X_0	V_0, CS_{V0}
X_1	W_{V0}
M_0	**$C_0, UID_C, Enc_{K0}(Sign_{KC}(X_0)), PKEnc_{KM}(K_0), Enc_{K1}(SignOnly_{KC}(ID_C), Sign_{KC}(X_1)), PKEnc_{KB}(K_1)$**
C_1	*Authorization request*
X_2	W_{V0}, UID_C
M_1	**$C_1, UID_M, Enc_{K2}(Sign_{KM}(X_2)), PKEnc_{KB}(K_2), Enc_{K1}(SignOnly_{KC}(ID_C), Sign_{KC}(X_1)), PKEnc_{KB}(K_1)$**
C_2	*Authorization response*
X_3	R, UID_C
M_2	**$C_2, UID_M, Enc_{K3}(Sign_{KB}(X_3)), PKEnc_{KM}(K_3)$**
C_3	*VendorScrip withdraw response*
X_4	UID_M, I
M_3	**$C_3, UID_B, Enc_{K4}(Sign_{KM}(X_4)), PKEnc_{KC}(K_3)$**

Fig. 10. VendorScrip withdrawal process

In transaction message T_0, the presence of the buyer's distinct identifier and cryptographic signature provides assurance to the broker that the payment request is valid. Moreover, embedding the scrip's *CustomerSecret* along with the purchaser's digital endorsement allows the merchant to confirm the legitimacy of the transaction.

As VendorScrip is non-reusable, the merchant can effectively detect any replay attacks. Furthermore, the application of encryption guarantees both confidentiality and data integrity throughout the process.

Upon receiving T_0, the merchant decrypts the designated portion of the message, verifies the digital signature, and validates the received VendorScrip. If the provided details are accurate, the merchant constructs message T_1, transmits it to the broker, and temporarily stores the VendorScrip.

Within T_1, the merchant's digital signature serves as authentication, proving to the broker that the transaction was authorized. Additionally, encryption ensures that the transmitted data maintains integrity and confidentiality. Once the broker receives this message, they process its components by decrypting the elements and verifying their respective signatures. The broker additionally authenticates the buyer's credentials. Upon successful validation, they check whether X_1 corresponds to X_2. If the values align, two distinct response signals are produced: one designated for the vendor (T_2) and another

for the purchaser (T_3). These are subsequently transmitted to their respective destinations, while a transaction entry is documented in a log file.

In T_3, the broker's signature serves as verification for the customer, confirming that the transaction was officially authorized. Encryption ensures that the data remains confidential and intact during transmission. The customer receives the message, decrypts its contents, and verifies the digital signature. Once verified, they remove the VendorScrip and the associated *CustomerSecret* from their local repository.

Similarly, T_2 maintains confidentiality and integrity through encryption. Additionally, the merchant gains definitive proof of authorization via the broker's digital signature. Upon receiving the message, the merchant verifies its authenticity and, if validated, retrieves and deletes the previously stored VendorScrip along with the associated *MasterScripSecret* and *MasterCustomerSecret*.

H. Expired Scrip

The scrip (BrokerScrip/VendorScrip) is associated with an expiration date, beyond which it becomes invalid. Upon expiration, a renewal process is triggered (Figure 14), where the expired scrip is sent back to its issuer for reauthorization. During this renewal, the *PrimaryScripKey* and *PrimaryUserKey*, which are linked to the scrip and stored within the issuer's lookup repository, are refreshed. Simultaneously, the respective *UserKey*, residing in the user's local storage, undergoes renewal.

In M_0, the digital signature, along with the corresponding *UserKey* of the scrip, serves as cryptographic proof that the user has authenticated the transaction. Additionally, encryption mechanisms safeguard the integrity and confidentiality of transmitted data. Moreover, the inherent non-reusable nature of the scrip guarantees that the transaction is not susceptible to replay attacks.

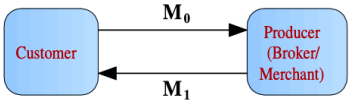
Upon receiving this request, the issuer decrypts the included elements and verifies the digital signature. Subsequently, the issuer validates the scrip and issues a replacement. The newly generated scrip contains:

- An updated expiration date,
- A fresh *ScripID* and *UserID*,
- A new certificate along with a newly assigned *UserKey*.

The only retained element between the old and the new scrip is their intrinsic value. The issuer then composes message M_1, transmits it to the user, and securely deletes the previous *PrimaryScripKey* and *PrimaryUserKey* associated with the expired scrip.

The digital signature appended to M_1 by the issuer serves as proof to the user that the transaction was properly authorized. Furthermore, encryption mechanisms ensure both the confidentiality and integrity of transmitted information.

Upon receiving M_1, the user decrypts its contents, verifies the authenticity of the signature, and confirms that the updated scrip holds the same value as the expired one. If all conditions are met, the user securely stores $PKEnc_{KU}(E_1)$ and $PKEnc_{KU}(USE_1)$ within their local repository.



C_0	*Expired scrip request*
X_0	E_0, CS_{E0}
M_0	**C_0, UID_C, $Enc_{K0}(Sign_{KC}(X_0))$, $PKEnc_{KP}(K_0)$**
C_1	*Expired scrip response*
X_1	E_1, CS_{E1}
M_1	**C_1, UID_P, $Enc_{K0}(Sign_{KP}(X_1))$, $PKEnc_{KC}(K_1)$**

Fig. 11. Expired Scrip

VII. Computational Overhead of the Broker

The proposed protocol is designed to minimize the broker's involvement, thereby reducing their operational and computational burden. As previously discussed, the broker represents financial institutions and plays a crucial role in facilitating transactions. Within the payment protocol, the broker functions as both the entity responsible for payment authorization and as a recorder of transaction details.

Considering the three fundamental transaction phases in the protocol—1. "Issuing electronic cash from the bank," 2. "Receiving electronic cash from the merchant," and 3. "Executing a purchase transaction" (with other steps being supplementary to these)—the broker has a dual role in the first two steps: verifying transactions and maintaining transaction records. This results in significant computational overhead due to the requirement for multiple cryptographic operations. However, in the third phase, their role is limited to passive observation and logging, requiring only two signature verifications and updates to the transaction ledger.

Optimizing the protocol involves ensuring that the first two transaction phases, which establish trust between peers, occur less frequently than the third phase. This design ensures that in a peer-to-peer transaction model, the broker's computational burden is effectively reduced.

VIII. Conclusion

This paper introduces a novel decentralized peer-to-peer (P2P) payment protocol that ensures secure, private, and anonymous transactions while addressing key limitations found in existing systems such as Venmo and CashApp. The protocol is particularly well-suited for decentralized and hybrid network architectures where third-party oversight is minimal or undesired.

Unlike centralized platforms like Venmo and CashApp that require full user identification and depend on central servers for identity verification and transaction processing, our system provides enhanced **anonymity** through the use of cryptographically secured tokens (BrokerScrip/VendorScrip) that do not directly reveal a user's real-world identity. Instead, **pseudonymous credentials** and digital signatures serve as proof of ownership and authorization, effectively separating user identity from transaction content.

The proposed system also guarantees **transaction integrity** through layered encryption and digital signatures applied to every message exchange between participants (customer, merchant, and broker). These measures protect the transaction from tampering and ensure that all actions are verifiable and non-repudiable.

To solve the problem of **key security**, the protocol adopts a token-based approach where each token (and its associated secret) is **non-reusable and time-bound**. In the event of key compromise, the damage is contained, and renewal mechanisms are available to refresh keys and associated credentials without affecting overall system integrity.

A critical innovation in the protocol lies in its **trust evaluation** mechanism. Trust is implicitly established in early transaction phases ("Issuing electronic cash from the bank" and "Receiving electronic cash from the merchant") through interaction with a trusted broker. However, once trust is established, the broker's role is minimized, and **direct P2P transactions** can proceed without continuous third-party intervention. This design reduces both the broker's computational burden and the customer's dependence on centralized trust anchors—something that platforms like CashApp and Venmo do not provide, as they fully mediate and authorize every transaction.

In summary, the key differentiators of our system include:

- **Decentralized identity management** via pseudonymous credentials.
- **No central authority required for every transaction**, promoting true P2P exchange.
- **Non-reusable and time-sensitive scrip** that mitigate key theft and replay attacks.
- **Trust bootstrapping via minimal broker involvement**, with full cryptographic logging and verifiability.
- **End-to-end encryption and digital signatures** for confidentiality, authenticity, and transaction integrity.
- **Built-in withdrawal and expiration mechanisms** for lifecycle control over digital cash.

The protocol therefore satisfies core security and privacy objectives—including confidentiality, authenticity, integrity, and anonymity—while also offering greater flexibility and resilience compared to traditional mobile payment systems. It represents a step forward in realizing a scalable, trust-minimized digital economy aligned with emerging demands for privacy-preserving financial technologies.

References

[1] E. Turban, D. King, J. K. Lee, and T.-P. Liang, *Electronic Commerce 2018: A Managerial and Social Networks Perspective*. Springer, 2018.

[2] F. Lachner and J. Trippas, "Digital trust in the age of e-commerce," *Journal of E-Commerce Studies*, vol. 12, no. 3, pp. 201–219, 2020.

[3] International Telecommunication Union, "Global connectivity report 2024." https://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx, 2024. Accessed April 5, 2025.

[4] Amazon.com, Inc, "Amazon." http://www.amazon.com/, 2025.

[5] eBay, Inc, "ebay." http://www.ebay.com/, 2025.

[6] Y. Benkler, *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, 2006.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet, a distributed anonymous information storage and retrieval system: Designing privacy enhancing technologies," in *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, vol. 2009 of *LNCS*, pp. 46–66, 2001.

[8] A. O. Freier, P. Kariton, and P. C. Kocher, "The ssl protocol: Version 3.0," 1996.

[9] C. Shirky, "What is p2p... and what isn't?." Online, November 2000. http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html.

[10] Venmo, "Venmo - mobile peer-to-peer payments." https://venmo.com, 2024. Accessed April 5, 2025.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, vol. 2008, pp. 1–9, 2008.

[12] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," in *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[13] C. Wade, "New models for providing payment services over the internet." Online. http://commerce.net.

[14] J. Swanson, "The platform economy and peer-to-peer commerce: How small businesses compete," *Business Horizons*, vol. 62, no. 4, pp. 429–440, 2019.

[15] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson, 2017.

[16] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[17] J. Garms and D. Somerfield, *Professional Java Security*. Wrox, 2001.

[18] Z. Y. Lee, H. C. Yu, and P. J. Kuo, "An analysis and comparison of different types of electronic payment systems," in *Management of Engineering and Technology, PICMET '01*, vol. 2, pp. 38–45, 2001.

[19] B. Yang and H. Garcia-Molina, "Ppay: Micropayments for peer-to-peer systems," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, pp. 300–310, October 2003.

[20] K. Wei, Y. F. R. Chen, A. J. Smith, and B. Vo, "Whopay: A scalable and anonymous payment system for peer-to-peer environments." Online. http://uther.dlib.vt.edu/.

[21] Y. Zhou and B. Zhang, "Secure mobile payment model based on trusted third party," *Journal of Computer and Communications*, vol. 1, no. 1, pp. 45–50, 2013.

[22] X. Hu and W. Liu, "Single point of failure in payment systems," in *Proceedings of the International Conference on Secure Systems*, pp. 234–239, 2005.

[23] M. Abadi and M. Burrows, "Principles of distributed trust management," in *Proceedings of the 12th International*

Conference on Secure Communication, pp. 321–332, 2005.

[24] A. Shamir, "Identity-based cryptosystems and signature schemes," *Advances in cryptology*, vol. 7, pp. 47–53, 1984.

[25] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[26] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Annual International Cryptology Conference*, pp. 1–15, 1996.

[27] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*, pp. 139–147, 1992.

[28] D. Chaum, "Blind signatures for untraceable payments," *Advances in cryptology*, vol. 82, pp. 199–203, 1983.

[29] T. Okamoto and K. Ohta, "An efficient divisible electronic cash scheme," in *Advances in Cryptology*, pp. 438–451, 1993.

[30] S. Goldwasser and S. Micali, "Probabilistic encryption," in *Proceedings of the ACM symposium on Theory of computing*, pp. 365–377, 1984.

[31] P. Wallish, "Cyber view: How to steal millions in champ change," *Science American*, pp. 32–33, August 1999.