Ad Hoc Prevention of Double-Spending in Offline Payments

Amir M. Aghapour[1], Erfan Bahrami[1], and Morteza Amini[1]

[1] Sharif University of Technology

March 19, 2025

Abstract

Digital payments are vital for daily activities, and they need to be available even when communication with payment service providers is impossible. The major problem faced in offline payments is that the same money can be spent multiple times, known as double-spending. Most solutions address doubles-pending either by deterrence or depend on secure hardware. This paper introduces an offline payment method to prevent double-spending with resource-constrained mobile devices. The provided solution is based on an arbitrary digital payment system and enables users to prevent double-spending by sharing information over ad hoc contacts. To reduce the overhead of sharing this data, we recommend using Bloom filter to store the spent money identifiers and suggest an algorithm to control the false positive rate. To further reduce the likelihood of doubles-pending, the payment protocol is designed to ensure that the funds used in each transaction are not under the control of the payer. We evaluated the proposed solution using several real-world mobility datasets by considering users' contacts as payment and information-sharing opportunities. The findings indicate that our proposed solution effectively prevents double spending, akin to ideal conditions where users have unlimited storage and bandwidth. The evaluation results show that with 240 bytes to store and share information about spent funds, the system achieves an accuracy of over 85% in detecting doubles-pending attempts.

Ad Hoc Prevention of Double-Spending in Offline Payments

Amir M. Aghapour, Erfan Bahrami, Morteza Amini

***Abstract*—Digital payments are vital for daily activities, and they need to be available even when communication with payment service providers is impossible. The major problem faced in offline payments is that the same money can be spent multiple times, known as double-spending. Most solutions address double-spending either by deterrence or depend on secure hardware. This paper introduces an offline payment method to prevent double-spending with resource-constrained mobile devices. The provided solution is based on an arbitrary digital payment system and enables users to prevent double-spending by sharing information over ad hoc contacts. To reduce the overhead of sharing this data, we recommend using Bloom filter to store the spent money identifiers and suggest an algorithm to control the false positive rate. To further reduce the likelihood of double-spending, the payment protocol is designed to ensure that the funds used in each transaction are not under the control of the payer. We evaluated the proposed solution using several real-world mobility datasets by considering users' contacts as payment and information-sharing opportunities. The findings indicate that our proposed solution effectively prevents double spending, akin to ideal conditions where users have unlimited storage and bandwidth. The evaluation results show that with 240 bytes to store and share information about spent funds, the system achieves an accuracy of over 85% in detecting double-spending attempts.**

***Index Terms*—Offline Payment, Double-Spending Prevention, Ad hoc Contact, Bloom filter**

I. Introduction

CASH has proven to be a suitable means of payment, whether considering financial inclusion or security aspects; however, an urge for digital replacements exists due to the physical limitations of paper money [1]. Digital payment systems have provided users with a fast and easy way to transact over the Internet, yet users need to have access to a remote ledger to be able to use their money. This means they cannot do so in offline environments where this access is impossible due to network or server failures. This limitation of commonly used digital payment systems has kept cash as a common and reliable, even if not always convenient, means of payment.

At the core of any payment system, there is a ledger that keeps track of money, and in the case of offline payments, this ledger can be viewed as a distributed database since it must store the account balances or token states of possibly disconnected users. In this case, the CAP theorem [2] can be applied to model offline payment systems where the ledger is considered *consistent* if the correct balance of an account or state of a token (being spent or not) is provided for all transactions. If the transaction is accepted without any guarantee about the correctness of the result, the ledger is considered *available*, and if users can make payments without access to the ledger, that is when a failure in the network or server occurs, the payment system is *partition tolerant*. According to the CAP theorem, an offline payment system can only satisfy two of the aforementioned three properties, among which partition tolerance is by definition necessary for offline payments. As a result, an offline payment system can either keep availability or consistency. Dealing with this trade-off is the main challenge in designing an offline payment solution.

Different approaches to enable offline payment can be identified by how they make the trade-off imposed by the CAP theorem. Solutions based on *E-cash* [3], [4] keep availability yet lose consistency as they allow double-spending to happen and only deter attackers by making invalid transactions detectable when online communication is possible. On the other hand, solutions based on *E-vouchers* [5] do not provide availability since users can only spend their money in predetermined places. When using secure hardware [6], we make certain assumptions about what an attacker can do. These assumptions effectively ignore the idea of partitioning. The reliance on secure hardware also carries risks; any failure in the trusted module may lead to widespread inconsistency. Finally, some solutions [7]–[10] use connections between users to prevent double-spending. An overview of this approach is demonstrated in Fig. 1. At first glance, it might seem that these solutions overlook partition tolerance, but if designed correctly, this approach does not contradict the offline definition. This is because users do not maintain the ledger and are not responsible for finalizing transactions; they only interact with each other to prevent receiving any invalid transactions.

In this work, we aim to maximize the consistency of the payment system in unexpected offline environments without relying on secure hardware or specific ledger design. We achieve this by designing a payment protocol that reduces the payer's control over the funds used in the payment. We then utilize the information flows between users to detect double-spent funds. One of the primary challenges of sharing information is the high volume of data that needs to be transferred and stored by users. We tackle this issue by using Bloom filter [11] while controlling its false positive rate. The provided service is best-effort, meaning that the solution would not conflict with the CAP theorem as it does not guarantee consistency but aims to minimize the probability of successful double-spending. Contributions of this work can

Amir M. Aghapour, Erfan Bahrami, and Morteza Amini are with the Department of Computer Engineering, Sharif University of Technology, Tehran 1458889694, Iran (e-mail: amir.aghapour@sharif.edu; erfan.bahrami98@sharif.edu; amini@sharif.edu).

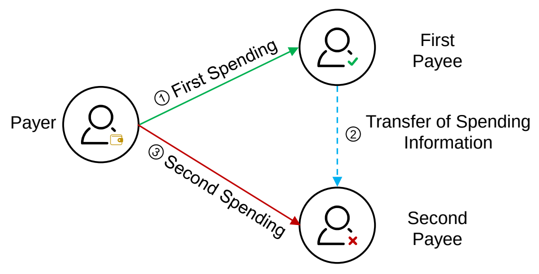

Fig. 1. Prevention of double-spending. Communication in step ② can be done directly or by intermediaries. The second spending in step ③ is denied because the second payee is informed about the first spending beforehand.

be summarized as follows:

- Propose a unified model of money to support different types of money representation.
- Design a double-spending prevention mechanism in payment protocol to minimize the risk of accepting invalid funds by merchants.
- Design an algorithm to share Bloom filters made out of spent money identifiers to prevent double-spending with a low false positive rate.

The rest of the paper is organized as follows: Section II covers related work. Following that, Section III describes the proposed solution by providing the details of the double-spending prevention algorithm. The performance analysis and overall evaluation of the proposed solution is presented in Section IV. Finally, the paper concludes with Section V.

II. Related Work

Enabling offline payment is a thoroughly researched topic, with various approaches that can be categorized by different criteria. One key aspect of the solution is the mode of payment, which can be classified as full, intermittent, or staged [12]. In fully offline payments, similar to cash, the same money can circulate indefinitely offline. In intermittent mode, money is submitted online at regular intervals. In staged mode, the online submission should happen after each payment. Our review takes into account all these forms of offline payment.

Another way to categorize offline payment solutions is the way they handle double-spending. The solutions can either deter double-spending or prevent it. In this section, we explore related works in these two categories in depth. TABLE I provides an overview of the surveyed works.

A. Double-Spending Detection

All forms of digital payment require transactions to be recorded on a ledger, as there is no alternative method to prevent double-spending [13]. The earliest solution for enabling offline payment by digital money, known as E-cash [14], uses a centralized ledger to detect double-spending. Significant advancements have been made in the development of E-cash, such as making it compact [15], transferable [16], and fair [17]. These days, there are production-ready solutions available based on E-cash [18]. The security of this type of offline payment relies on deterrence, where double-spending is only detected after it has occurred. After the detection, the double-spender is identified and punished if possible. The sole reliance on deterrence makes these schemes risky and impractical by themselves. In cases such as anonymous or pseudonymous payment solutions [19], where there is no assigned identity, deterrence is even less effective.

The method used to detect double-spending depends on the representation of money. The representation used by E-cash schemes is known as *bill* [20]. This format depicts money as a fixed value with a variable owner. Transactions involving bills alter the owner; therefore, double-spending occurs when a bill has more than one owner at a point in time. While this representation shares similarities with physical cash and offers advantages such as parallel processing and traceability, it is unsuitable for offline transactions due to its fixed value, which increases storage requirements and necessitates special handling of change money [21]. More recent works on offline payments use two alternative representations of money: *account* and *UTxO* (Unspent Transaction Output). Account representations link a balance to a fixed owner identity, causing each transaction to modify the balances of the sender and receiver. Both bills and accounts possess unique identifying numbers that remain unchanged throughout a payment; therefore, double-spending in account representation can similarly happen when an account has different balances at a point in time. In contrast, UTxO defines money as a token with an arbitrary value and owner while having a unique identity. In a UTxO system, transactions destroy tokens and create new ones with different identifiers. Each token can only be spent once, making it a suitable representation for offline payments [13], as the sole requirement is to verify whether a specific token is unspent. In other words, in a UTxO system, double-spending can only occur if an already spent token is used in another transaction later.

B. Double-Spending Prevention

In addressing the issue of double-spending, another method involves prevention. This approach is practical as it mitigates the risk of offline payment and complements the detection approach. Three subcategories can be identified within the prevention approach as follows.

1) Secure Hardware: Secure hardware is commonly used to prevent undesired user actions. The proposed secure element could be tamper-resistant hardware dedicated to payment, such as smart cards, or a trusted execution environment with additional functionalities beyond payment [23], [27]. Other suggested solutions include specialized hardware based on Physically Unclonable Functions [6], one-time readable memory [22], and even quantum computers [28] for offline payment. Although they support *fully offline* mode and are widely implemented, the universal applicability of secure hardware-based methods faces challenges, including the risk of hardware damage or loss, potential privacy violations, lack of graceful degradation [13], and the cost associated with providing hardware [29]. Therefore, we argue that while using secure hardware undoubtedly enhances payment security, it should be considered for hardening rather than the cornerstone of the system.

TABLE I
COMPARISON WITH OTHER OFFLINE PAYMENT SOLUTIONS.

	Ledger			User Requirements		Payment Properties	
Research	Organization	Money Representation	Mode of Payment	Hardware-Independent	Network Failure Resilient	Transferable	Double-Spending Prevention Method
OPERA [22]	Centralized	Bill	Full	✗	✓	✓	Secure Hardware
FRoDO [6]	Both	Bill	Staged	✗	✓	✗	Secure Hardware
Jie *et al.* [23]	Decentralized	UTxO	Intermittent	✗	✓	✓	Secure Hardware
Blaze *et al.* [24]	Centralized	Bill	Staged	✓	✓	✗	None
Bauer *et al.* [16]	Centralized	Bill	Staged	✓	✓	✓	None
EuroToken [25]	Decentralized	Account	Intermittent	✓	✗	✓	Fixed Network
Hoepman *et al.* [8]	Both	Bill	Intermittent	✓	✗	✓	Fixed Network
Nirvana [26]	Decentralized	Account	Staged	✓	✗	✓	Fixed Network
Our Solution	Both	All	Intermittent	✓	✓	✓	Ad hoc Network

2) Non-Fungible Money: Non-fungible money refers to individual units that are not interchangeable. In the context of offline payment, it refers to specific money issued for use with a particular merchant. Examples of such money include electronic vouchers [5], [30] and checks [24]. Non-fungible money enables merchants to prevent double-spending by only accepting funds exclusive to them. Another characteristic of non-fungible money is its *staged* nature, which requires the recipient to deposit and convert it online at a bank to unlock its value for future transactions. Although this approach is easy to implement and does not require special hardware, it is only practical when the payer is aware of purchases and offline situations in advance, such as during an airplane trip. This approach does not work in unexpected offline situations like during power outages from natural disasters or when the merchant is arbitrary.

3) Communication: Another approach for enabling offline payment is by introducing connectivity among users. It's important to distinguish this method from approaches involving a partitioned ledger [25], satellite network [31], or delay-tolerant networking [32]. These solutions do not qualify as offline payment, as they change the type of the ledger or the way it is accessed while keeping the necessity to submit transactions to the ledger. What we mean here is preventing double-spending through user communication in an offline environment where submitting transactions to the ledger is not feasible. Existing solutions use a fixed [7], [9], [33] or dynamic [8], [10], [34] set of nodes in a peer-to-peer network as witnesses to transactions. Another method involves random polling [35], which can also take the form of a lottery [36]. In this approach, transactions are audited randomly, sending only a fraction of them to the ledger in real-time while the rest is sent to the ledger later. Another recently suggested solution involves using a loan network [37] to minimize the impact of double-spending by distributing responsibility.

Our proposed solution in this work goes into this category, but we do not rely on the accessibility of specific nodes in each transaction. We use customers shared by merchants, local area networks and foot traffic as a means to share information among users to prevent double-spending. While it is suggested to use local network for offline payments [38], doing so in an ad hoc fashion with countermeasures against double-spending is the main contribution of the proposed approach in this paper.

III. PROPOSED SOLUTION FOR DOUBLE-SPENDING PREVENTION

We aim to propose a method to prevent double-spending in unexpected offline environments using ad hoc communications. First, in subsection III-A, an overview of the proposed solution and its scope is presented. After that, details about money representation (subsection III-B), payment protocol (subsection III-C), and information sharing (subsection III-D) are given. Finally, in subsection III-E, we suggest incentives to lower the risk of accepting offline payments that can be used in conjunction with our solution.

A. Overview

The proposed payment system consists of three operations: withdrawal, payment, and redemption. Users are either merchants or customers, although nothing prevents merchants from acting as customers or vice versa. Initially, in an online environment, customers must withdraw money from the ledger. This step is essential to ensure that the funds are usable offline, as there needs to be a reliable anchor for trust. Additionally, an expiration time should be specified to limit the duration for which the withdrawn funds remain valid. The ledger, whether centralized or decentralized, serves as the trusted entity for verifying these transactions. To standardize the representation of money across various types of ledgers and to delineate the management of offline operations from online transaction processing, an independent intermediary or features built into the ledger—such as a smart contract or script—may be utilized.

In order to make a purchase, the customer usually creates and authorizes the transaction and then sends it to the merchant to validate. But in our designed payment protocol, the customer cannot use arbitrary money for its transaction. Instead, the transaction is created using deterministic pseudo-random permutation of the customer's assets. By doing so, the probability of a successful double-spending attempt is lower.

To prevent double-spending, merchants can interact with each other. This can be done in two ways:

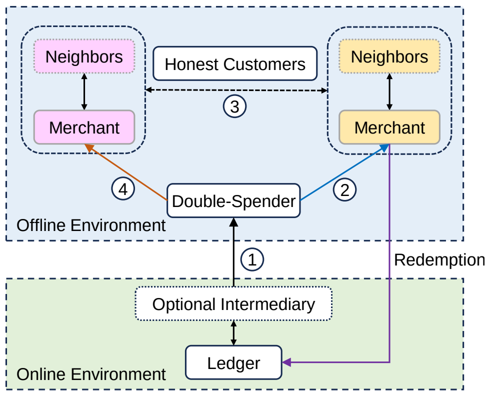

Fig. 2. Overview of the proposed solution. ①: Withdrawal of money from the ledger directly or through an intermediary. ②: Valid payment. ③: Sharing of payment information through neighbors and customers. ④: Prevention of double-spending.

- **Querying**: Merchants can query their neighbors, those reachable directly by the local network, to check whether the received money has been spent before. This approach requires a two-way connection and routing of messages, which are not always feasible or reliable. This approach is not the focus of our work.
- **Sharing information**: Merchants can share information about spent funds through broadcasting on a local network or with the help of customers and passersby. We use Bloom Filter to reduce the bandwidth and storage needed to keep and share this information. This data structure significantly reduces the size of the set of spent funds, but it also introduces false positives. We propose an algorithm that minimizes both false positives and false negatives for funds spent closer to a merchant. This design is based on the understanding that attackers operating in an offline environment must physically move, which creates a spatial connection between multiple attempts at double-spending. Even if attackers attempt to travel long distances, the routes they take are likely to be shared with other users, making it easier to detect any instances of double-spent money.

The redemption of received funds in offline transactions is done by submitting them to the ledger. This is only possible when the user gets online before the expiration of funds. In order to encourage the completion of this settlement and reduce the risk of accepting offline payments, a reward could be given to anyone who submits offline transactions to the ledger. This reward can be part of the customer's fee, or it can be covered by the merchant. These three steps in the proposed offline payment system are shown in Fig. 2 and detailed in the following subsections.

B. Money Representation

We keep the underlying money representation used by the ledger intact and only introduce auxiliary information needed to prevent double-spending in an offline environment. This information can be kept out of the ledger or integrated into it by using programmability features. As a result, information (such as value and current owner of money) or format of transaction is not the concern of our solution and only additional attributes are described in this subsection. There are three components in the proposed representation:

1) **Spending Identifier**: This identifier (demonstrated with $\mathrm{ID}_{\mathrm{spend}}$) changes after each transaction such that using it twice for transacting equals double-spending. We demonstrate that it can be derived from other representations in common payment processing systems, specifically UTxO, as well as bill and account.
   Creating the identifier $\mathrm{ID}_{\mathrm{spend}}$ for UTxO representation is straightforward. The identifier used for a UTxO token can be directly utilized as $\mathrm{ID}_{\mathrm{spend}}$ since using the same UTxO for multiple transactions essentially results in double-spending.
   The account representation differs from the UTxO representation because the account identifier ($\mathrm{ID}_{\mathrm{account}}$) cannot be used directly to detect double-spending. This is due to the fact that the $\mathrm{ID}_{\mathrm{account}}$ remains constant during transactions. To derive $\mathrm{ID}_{\mathrm{spend}}$ for account representation, we suggest creating a digest from the concatenation of the account identifier and its nonce (used to prevent replay of the last transaction) with a hash function (h) as shown in equation 1. This way, after each spending from an account, $\mathrm{ID}_{\mathrm{spend}}$ changes.

$$\mathrm{ID}_{\mathrm{spend}} = h(\mathrm{ID}_{\mathrm{account}} \| \mathrm{nonce}) \tag{1}$$

   In bill representation of money, each bill has a unique identifier ($\mathrm{ID}_{\mathrm{bill}}$) that cannot be used to detect double-spending because it stays unchanged while the bill is circulating. We therefore need to incorporate the history of owners for a bill in its spending identity creation. To do so, we denote the n-th owner of the bill with o_n and define the $\mathrm{ID}_{\mathrm{spend}}$ for a bill recursively.
   - For the first owner of the bill, we define the spending identifier according to equation 2.

$$\mathrm{ID}_{\mathrm{spend}} = h(\mathrm{ID}_{\mathrm{bill}} \| o_1) \tag{2}$$

   - If $\mathrm{ID}^{\mathrm{n}}_{\mathrm{spend}}$ denotes the spending identifier of the bill when it is owned by o_n, after it is transferred to o_{n+1} the new identifier is calculated according to equation 3.

$$\mathrm{ID}_{\mathrm{spend}} = h(\mathrm{ID}^{\mathrm{n}}_{\mathrm{spend}} \| o_{n+1}) \tag{3}$$

   By using this definition, it is not necessary to keep the complete history of owners for computing the new $\mathrm{ID}_{\mathrm{spend}}$ after every spending. Additionally, the same bill cannot be used more than once without reusing the same identity.
   We have effectively transformed all forms of money into uniquely identifiable tokens. The detection of double-spending in this context is straightforward: duplicate usage of $\mathrm{ID}_{\mathrm{spend}}$ indicates double-spending.
2) **Expiration Date**: We use a timestamp to denote the expiration date of each money. Expired money is considered lost and not accepted in any transaction. This limits the time during which the money can stay offline

and also allows recovery of lost money as it can be withdrawn again from the ledger after it has expired.

3) **Authenticity Proof**: In an offline environment, it is not possible to query the authenticity of money from the ledger, hence forgery is possible. To provide a way for merchants to check the authenticity of received money, we require a proof to be provided for each money. In an online environment, this proof is not needed as the existence of funds in the ledger shows the authenticity; whereas in an offline environment, this existence cannot be checked. This proof should cover all attributes of money and be verifiable locally. It can be in different forms such as a digital signature by a trusted party, offline-traversible blockchain [39], or membership proof in an accumulator [40]. This proof assures the offline merchant that the received money has existed in the ledger at some point, but it gives no guarantee regarding the current validity of the money.

If these attributes can be incorporated into the ledger, no intermediary for withdrawal is needed; otherwise, a trusted intermediary is required to include these attributes in the withdrawn money to be used in an offline environment.

C. Payment Protocol

As described before, double-spending occurs when a user manages to use a money in more than one transaction. To commit double-spending in a physical payment scenario, the payer needs to make the merchant accept forged cash; but what if the merchant, instead of the payer, chooses the cash from the payer's wallet? In this way, the attacker cannot always use the same money and as a consequence, the double-spending success rate is reduced. The proposed solution applies the same idea in the digital scenario.

To enable the merchant to select inputs for a transaction, it must be aware of the payer's assets. The naive approach is to have the payer provide a list of assets for selection. This approach has many problems, such as communication overhead and violation of the payer's privacy, but the biggest issue is that the payer can hide part of its assets to increase the probability of double-spending success. To avoid this problem, we suggest the use of Merkle-Patricia trie (MPT) [41] to keep the assets of the user. The choice of this data structure is primarily due to its adoption in popular blockchains and its capability to store and retrieve key-value pairs efficiently. It can also be used to prove the inclusion or exclusion of specific key-value pairs. An example of Merkle Patricia Trie (MPT) is illustrated in Fig. 3. The keys are represented as paths starting from the root of the trie. The smallest unit of a path is referred to as a *nibble*. There are three types of nodes in MPT:

- **Leaf node**: They contain the last nibble of each key and its corresponding value.
- **Branch node**: They keep a reference to the next node for all possible nibbles. It is also possible for a branch node to act as a leaf node and keep a value.
- **Expansion node**: They are used for specifying multiple nibbles in a single node to shorten the height of the trie.

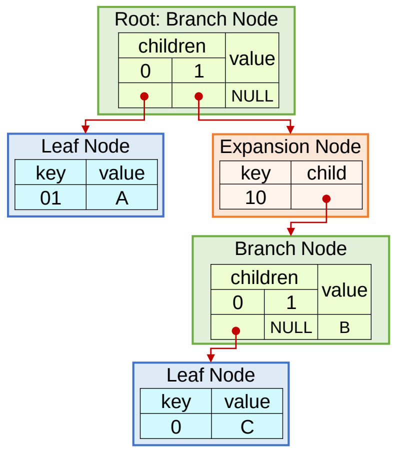

Fig. 3. An example Merkle-Patricia trie with binary nibbles. The stored key-value pairs are: $(001, A) - (110, B) - (11000, C)$.

All the withdrawn monies by each user are kept in an MPT. The $\mathrm{ID}_{\mathrm{spend}}$ is used as the key and other attributes of money are stored as value. The root of the MPT should be authenticated, by using the signature of a trusted entity or proof of inclusion in the ledger. Customers choose monies to include in transactions from this MPT. The algorithm used to choose the inputs of a transaction should satisfy two properties:

1) The algorithm should be deterministic because if the choice is made randomly for each request of payment, the attacker could keep trying until its desired money is chosen.
2) The choice of money should not solely depend on values provided by the payer as it would lead to the same order of money used by different merchants and actually help double-spending!

To satisfy these two requirements, it is proposed in Algorithm 1 to choose inputs of transaction using a pseudo-random permutation (PRP) seeded with a deterministic value ($\mathcal{S}$) chosen by the merchant. The range of the PRP should be the domain used for nibbles. To simplify the algorithm, we do not consider the count of nodes in each sub-trie which is required to sample uniformly. This simplification can be justified by the fact that, unlike a general MPT, the inserted keys in our case are already uniform which leads to balanced tries.

The proposed algorithm is designed recursively to find the list of $\mathrm{ID}_{\mathrm{spend}}$ in the order specified by the seed $\mathcal{S}$ with the prefix $\mathcal{P}$ for all keys in the trie rooted in $\mathcal{T}$. Initially, the algorithm is called with the authenticated root of MPT and an empty prefix to get the order of spending for all the client's assets. For the sake of brevity, we did not include the value stored in MPT or proof of inclusion for each $\mathrm{ID}_{\mathrm{spend}}$ in the output of the algorithm.

Now we describe the proposed payment protocol executed between a merchant and customer. We assume that an authenticated and secure channel is used, so attacks related to the communication such as replay and man-in-the-middle are not considered. This type of channel between a merchant and a customer is relatively easy to create in an offline environment

Algorithm 1 Choosing Money from Merkle-Patricia trie

```
Input: Seed of PRP ($\mathcal{S}$), MPT Root ($\mathcal{T}$), Key Prefix ($\mathcal{P}$).
Output: List of chosen $\text{ID}_{\text{spend}}$.
 1: function CHOOSE($\mathcal{T}, \mathcal{P}$)
 2:     if $\mathcal{T}$ is Leaf Node then
 3:         return $\mathcal{P} \| \mathcal{T}$.key
 4:     else if $\mathcal{T}$ is Extension Node then
 5:         return CHOOSE($\mathcal{T}$.child, $\mathcal{P} \| \mathcal{T}$.key)
 6:     else                                        ▷ $\mathcal{T}$ is Branch Node
 7:         if $\mathcal{T}$.value $\neq$ NULL then
 8:             return $\mathcal{P}$                  ▷ $\mathcal{T}$ is Terminal Branch Node
 9:         else
10:             idList $\leftarrow$ Empty List
11:             for all nibble $\in$ Nibbles do
12:                 $k \leftarrow$ PRP($\mathcal{S}$, nibble)
13:                 $c \leftarrow \mathcal{T}$.children[$k$]
14:                 if $c \neq$ NULL then
15:                     idNext $\leftarrow$ CHOOSE($c, \mathcal{P} \| k$)
16:                     idList $\leftarrow$ append(idList, idNext)
17:                 end if
18:             end for
19:             return idList
20:         end if
21:     end if
22: end function
```

where two parties can have a direct physical connection. One of the most commonly used protocols that can be used for this purpose is Near Field Communication (NFC) which enables short-range communications.

For each payment, the merchant first sends the request for payment to the customer. This request includes the seed used in Algorithm 1 along with the requested value. The payer executes the algorithm with the given seed to get an order of monies to include in the transaction. For each money, if it was unspent, the client includes it in the transaction. Otherwise, if the selected money is already spent, the transaction that has spent that money, along with resulted monies, is provided to the merchant. An optimization is possible to lower the count of transaction data that the customer needs to send to the merchant. One way to do so is to ask the merchant to remember the last-used ID_{spend} for each customer. We do not consider this optimization though because this overhead is negligible for the direct and local links where this protocol is executed.

To support transferability, the client takes the same procedure as before with the difference that the choice of monies is done not only from its own MPT but also from the MPT of previously received monies. This can be done since the membership proof of monies and authenticity proof of MPT root are transferable.

The customer keeps choosing money for a transaction until the total value of the chosen monies is more than or equal to the requested value by the merchant. After creating the transaction, the customer sends it to the merchant, along with the membership proof of the used monies in MPT, without signing the transaction. This is done to prevent the merchant from repudiating the payment request and also to verify that the monies used in the transaction are indeed unspent. The merchant has the chance to consider the risk of accepting the transaction according to different factors like expiration dates of the used monies, history of interaction with the user, and the output of the double-spending prevention method discussed in Section III-D. If the merchant accepts the transaction, it signs it and sends it back to the customer. The customer in turn authorizes the transaction and shares it with the merchant, finalizing the offline transaction.

It is worth noting that the proposed solution does not defend against an attacker who can spend all of its money at once by colluding with a merchant. In this way, no matter the choice for creating a transaction, double-spending occurs. We argue that even in this case, the attacker is not guaranteed to succeed since the offline environment is assumed unexpected, so the legit merchant has the chance to submit its received money sooner than double-spender by itself or with the help of customers, effectively neutralizing the attack.

D. Sharing Spent Monies

Our approach to preventing double-spending in an offline environment is to make users keep and share the identities of spent monies with each other. In this way, users try their best to detect malicious transactions before accepting them. Normally this task is done by the ledger of the system, but in an offline environment where the ledger is inaccessible, merchants have to keep this information locally and synchronized among themselves. Two problems arise while doing so:

- **Lack of connectivity**: The lack of connectivity among users causes inconsistencies in their views of the spent monies. This problem is an inherent property of offline environments. It can be remedied by using networks with longer ranges or increasing the mobility of users.
- **Large data**: With limited storage and network bandwidth, the size of the shared data, which includes identifier of spent monies, poses a challenge. We suggest using Bloom Filter (BF) for keeping and sharing the set of spent monies. The use of BF also alleviates the first problem as it lowers the requirement for the used network, allowing more devices to take part in the sharing of information.

We now discuss the details of the proposed method to store and share spent money identifiers. Ideally, the users would be able to keep and share all the spent money identifiers, but to reduce the resource requirements, we assume that each user can only keep the ID_{spend} of the monies received as a payee and also a single BF with fixed capacity. This assumption can be justified by the fact that a merchant's resources are usually proportional to the number of its customers; therefore, it can keep its received monies. On the other hand, since the BF is shared among all the merchants, keeping and sharing it should be possible for everyone.

Users can receive BFs through broadcast messages, without requiring a payment to be performed. In order to lower the requirement for the network's capabilities, we assume that each broadcast message can only contain a single BF; therefore, solutions that require multiple exchanges of information, such as set reconciliation [42] or those that make use of dynamic filters [43], are not applicable.

For each payment, the payee checks for double-spending against its own set of previously received monies and the BF it has. Each user can opportunistically broadcast a BF to other users nearby. The remaining question is: what should a user send to others? A user can send a BF composed solely of its own received monies, but it limits the ability to detect double-spending. If the user combines all the filters it receives with its own funds, this results in an excessive false positive rate (FPR) in the resulting Bloom filter, as it eventually exceeds the designated capacity.

It is tempting to merge the BFs in such a way that the resulting filter contains a random sample of the union. However, our evaluation of previously suggested solution using this approach in Section IV-A, shows that this approach leads to unacceptable performance. The poor performance of this approach can be attributed to the randomness of the stored data in the BF and the fact that the receiving user has no prior information about them.

An overview of the usage of the suggested algorithm to create the shared BF is shown in Fig. 4 and details of the method is provided in Algorithm 2. There are two desired properties for the shared BF:

1) It should have an acceptable FPR with constant size. To support this requirement, we keep the approximate number of items in the BF lower than its capacity and consequently lower the FPR at the inevitable cost of introducing false negatives.
2) It should contain all the received monies in payments by the sharing user. This necessity is because double-spending them is a direct loss to the user.

To achieve the first goal, the algorithm ensures that the approximated size of the shared BF is less than its capacity. For the second goal, the user should always include its own monies in the shared BF, which is why the algorithm first checks if this is possible by comparing the size of the user's monies and the capacity of the BF. If it is possible, a BF is created from the user's monies and unified with the previously shared BF. The union is calculated using logical OR operator between the BFs. On the other hand, if the size of the user's monies exceed the BF's capacity, the second goal cannot be achieved. In this case, the user can consider a sample of its monies for the input of the algorithm. It should be noted that

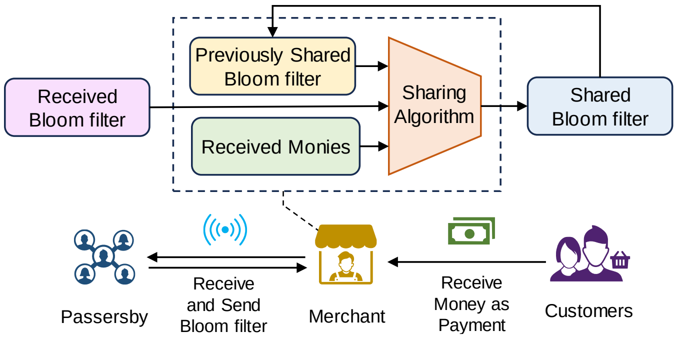

Fig. 4. Sharing Bloom filter of spent monies.

the parameters of the system should be chosen such that this case rarely happens, because as our results in Section IV-B show, the performance of the system is degraded if lots of users experience this situation.

The main idea behind the suggested algorithm is to include as much BFs as possible in the shared BF without violating its capacity. Doing so becomes impossible as more BFs are received, at which point we suggest refreshing the shared BF and only keep the most recently received one. To implement this idea, the algorithm instructs the user to compute the union between the previously shared BF and the newly received one. The user then approximates the number of items in the resulting BF with equation 4 [44]. In this equation, $\ln(.)$ denotes the natural logarithm and k denotes the number of hash functions used to create the BF with m bits that s of which are set.

$$|\mathcal{F}| \approx \frac{\ln\left(1-\frac{s}{m}\right)}{k\ln\left(1-\frac{1}{m}\right)} \tag{4}$$

If the approximated size of the resulting BF from this union is not more than the BF's capacity, it is shared. Otherwise, the same procedure is carried out for the BF made out of user's monies and the newly received BF, effectively refreshing the shared BF. Finally, if neither of the previously calculated BFs can be shared, the user either does not make any change to the previously shared BF or only share its own received monies.

Algorithm 2 Sharing Spent Monies

Input: Previously Shared BF ($\mathcal{F}_S$), Received BF ($\mathcal{F}_R$), Set of All Received Monies (M), Capacity of BFs (c).
Output: Newly shared Bloom filter or $\perp$.

```
1:  if $|M| \leq c$ then                 ▷ Exact size of the set is used
2:      $\mathcal{F}_M \leftarrow$ Created BF from $M$
3:  else
4:      return $\perp$                   ▷ Cannot include the received monies
5:  end if
6:  $\mathcal{F}_S \leftarrow \mathcal{F}_S \cup \mathcal{F}_M$
7:  if $|\mathcal{F}_S \cup \mathcal{F}_R| \leq c$ then
8:      return $\mathcal{F}_S \cup \mathcal{F}_R$      ▷ Include the received BF
9:  else if $|\mathcal{F}_M \cup \mathcal{F}_R| \leq c$ then
10:     return $\mathcal{F}_M \cup \mathcal{F}_R$      ▷ Reset the shared BF
11: else if $|\mathcal{F}_S| \leq c$ then
12:     return $\mathcal{F}_S$                          ▷ Make no change
13: else
14:     return $\mathcal{F}_M$           ▷ Share only the received monies
15: end if
```

Although the probabilistic nature of the prevention method creates an opportunity for an attacker to evade detection by broadcasting made-up BF, the probability of false detection would stay within desired range and successful double-spending can still be prevented by honest customers.

E. Incentivize

Since the proposed way to prevent double-spending requires the participation of users in sharing their set of received monies, there needs to be an incentive for them to do so. This incentive acts as a complement to the deterrence made by persecution of detected double-spenders. Another advantage of encouraging users to participate in offline operations even when they have access to online services is to lower the costs of banks processing the transactions, especially those with lower amounts. We propose two incentives in this work:

- **Mutual security**: Users secure the system by helping each other detect double-spenders. Detection of double-spenders also lowers the risk of accepting payments for merchants, leading to an increase in its adaption that ultimately benefits both customers and merchants. With this reasoning, we can assume that most users will always help prevent double-spending. These users share the environment with double-spenders and can visit the same merchants that the attackers can, which leads to effective prevention of double-spending.
- **Reward**: A reward can be paid by merchants to whoever submits their offline transactions to the ledger. This reward can be claimed by anyone with offline transaction data in an online environment. Merchants can decide the amount of reward and whom to share the data with. To lower the privacy concerns of sharing transaction information with arbitrary users, the privacy-enhancing technologies used in the underlying ledger or simply encryption of data with the key of the entity responsible for managing the ledger can be used.

The incentives can encourage people to improve their connectivity by different means such as using network devices and increasing their mobility. Depending on the cost for the participating for the users, there might even be an opportunity for them to make profit.

IV. Evaluation

To evaluate the performance of the proposed solution, we first focus on the algorithm used to share spent monies among users, as it plays a crucial role in the successful detection of double-spending attempts. After that, we use real-world datasets to measure the effectiveness of the solution in preventing double-spending.

A. Bloom Filter Sharing

As described in Section III-D, each user shares a BF made from their own received monies and mixes it according to Algorithm 2 with each received BF. The mixing approach is greedy, meaning users add filters until the empty space runs out, at which point the BF resets. To show the result of this process, Fig. 5 shows different approaches to mixing BFs in an illustrative scenario depicted as a directed graph. In this graph, each node represents a user with 225 received monies, and each edge represents a transfer of a BF. We use a random topological sort of graph to determine the transfer order. To mitigate the effects of randomness, the results are averaged over 100 executions. The color of each node indicates the measured False Negative Rate (FNR) of detecting the monies of that node by the central node. The FPR is estimated using 2000 random samples. We compare our proposed solution with three alternatives:

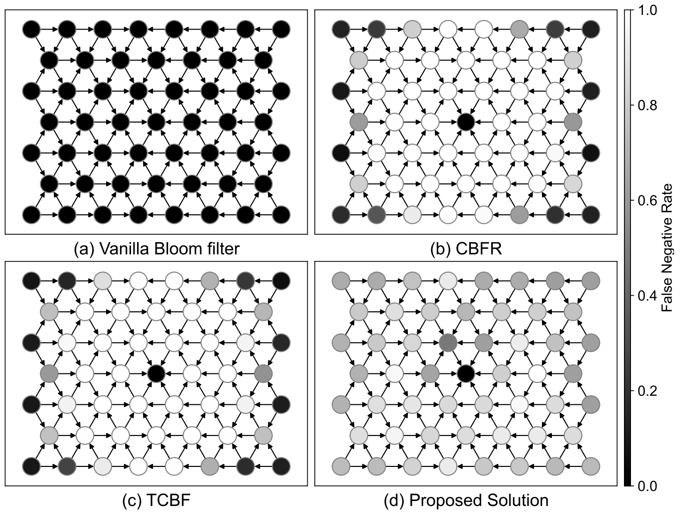

Fig. 5. Distribution of FNR of Bloom filters for sharing spent monies.

1) **Vanilla BF**: By utilizing BFs in their original form, the received BFs are combined using the logical OR operator and no measures are taken to control the FPR.
2) **CBFR** [45]: This data structure is a variant of counting Bloom filter [46] where each counter is modeled as a leaky bucket. The counters periodically decrease after a fixed timeout to control the FPR. Merging is done by incrementing each common element by one.
3) **TCBF** [47]: This data structure is also based on counting Bloom filter with decaying values. For merging, element-wise summation of two filters is calculated. To add an element, the counters associated with it are set to a fixed value.

All these three methods can be used as a drop-in for our proposed algorithm of sharing BFs. Through this comparison, we mainly intend to show the difference in the distribution of false negatives. To ensure a fair comparison, we carefully select the parameters of each filter through trial and error, aiming for a similar average false negative rate with equal capacity. For equal capacity, the storage size of CBFR and TCBF is four times larger than that of a standard Bloom filter because they utilize a 4-bit counter for each filter element instead of a single bit.

As is apparent in Fig. 5, using vanilla BF results in the central node having zero FNR for detecting almost all the spent money; however, this is because it has an FPR of over 57 percent! This high FPR means that the result of a membership check is unreliable, and the prevention of double spending leads to the rejection of legitimate payments. While CBFR and TCBF reduce the FPR, the FNR remains high for all nodes near the central node. Our proposed solution achieves close average values of FPR and FNR to CBFR and TCBF. The primary benefit of our proposed solution is the distribution of FNR among users. The proposed method keeps the FPR low by sacrificing the FNR of all nodes, while the other two methods are biased toward keeping the FNR lower for farther nodes. We argue that our choice is more effective for detecting double spending in offline environments because attackers need to move physically between merchants, which makes attempting double spending between distant users more improbable. We examine the correctness of this argument in Section IV-B by comparing the results from using different BFs under the same circumstances.

B. Double-Spending Prevention

To avoid double-spending, the payee must verify if the received funds have been used elsewhere. This information is normally available through the ledger but in an offline environment, it can only be obtained by communicating with other users; therefore, the success in prevention depends on the connectivity between the users.

We could not find a dataset that can be directly used to evaluate our work. This was expected as we would need the payment behavior along with the mobility of the users. This information is highly sensitive and hard to capture. We repurposed existing datasets of user mobility, originally intended for evaluating routing algorithms. All the chosen datasets contain one-way asynchronous contacts of mobile users and stationary devices with different granularity. The granularity of a trace refers to the minimum time between consecutive contacts of the same nodes in it. We consider five such datasets that can be used in our work:

1) **BLEBeacon** [48]: This dataset contains Bluetooth Low Energy (BLE) traces generated by beacons carried by individuals as they followed their daily routines inside a university building for over a month.
2) **Haggle** [50]: This dataset contains the results of 6 experiments. In each experiment, participants were given Intel® Motes, a sensor node platform equipped with Bluetooth capability, and instructed to roam freely. Afterward, the traces stored in the devices were collected.
3) **HumaNet** [51]: This dataset contains one day of Bluetooth mobility data. The proximity traces were collected using a customized Bluetooth system with optimized hardware to enhance granularity and reliability. Since the stationary nodes in this dataset did not send broadcast messages, we assumed a symmetric connection for each contact to allow merchants to share information.
4) **ShoppingMall** [52]: This dataset comprises real-world Bluetooth contact information gathered from shop employees at a shopping mall over a duration of six days. The data was recorded during working hours through software installed on the participants' smartphones. This dataset aligns well with the intended use case for the proposed solution.

The properties of the aforementioned datasets are reported in TABLE II. Unfortunately, we could not find the data from the fifth experiment of the Haggle dataset. We had to discard the first three Haggle experiments too as they have less than two stationary nodes.

To use these datasets for our offline payment scenario, we make the following assumptions:

- All the stationary nodes represent merchants, while the mobile devices serve as customers. As a consequence of this assumption, mobile devices that were not part of the experiment of generating the datasets are considered customers.
- When a customer interacts with a merchant, these interactions are considered payments and involve two-way communication. If there are several interactions between the same customer and merchant in a row, we treat only the first one as a payment. The following interactions are seen only as sharing information. This rule prevents recording too many payments when two parties stay close to each other for a long time.
- We divide each dataset into 12-hour epochs because the duration of datasets is longer than the expected time for an offline situation. After each epoch, we reinitialize all variables, as if all spent monies are either redeemed or expired.
- A certain fraction of customers are assumed to be attackers who try to double-spend at every chance they get and do not participate in sharing spent monies. We also assume that an attacker may attempt to retry a payment with different funds if the victim merchant identifies the payment as a double-spending attempt, similar to situations where a merchant permits the customer to retry the payment.
- Each payment, whether legit or double-spending attempt, is done using a single money, and each customer, whether attacker or legit, has a certain number of monies to spend. This assumption can be enforced in practice by enforcing values for monies and asset count (number of owned tokens) of users in the withdrawal phase. Another reason for this assumption is that offline payments are typically retail transactions with low values. Therefore, if an attacker attempts to use an excessive amount of money, the merchant can reject the payment.
- We consider two types of attackers: stateless and stateful. Stateless attackers do not keep track of their double-spending attempts and may try to pay the same merchant with the same money twice. In contrast, stateful attackers do not repeat their attacks. There may be more sophisticated attackers that we do not address in our evaluation. It should also be noted that merchants are always stateful.

We have implemented our proposed solution according to the assumptions described above. In order to evaluate different aspects of the solution, we consider three scenarios:

- **Ideal-sharing**: All the merchants and customers have enough storage and bandwidth to share spent monies they

TABLE II
PROPERTIES OF DATASETS.

Dataset	Duration	Mobile Devices	Stationary Devices	Contacts	Granularity
BLEBeacon	41 days	63	37	834,861	30 s
Haggle/1	3 days	127	1	24,088	120 s
Haggle/2	5 days	223	0	40,740	120 s
Haggle/3	3 days	274	0	89,768	120 s
Haggle/4	24 days	11,403	18	107,580	120-600 s
Haggle/6	4 days	4,499	20	620,912	120 s
HumaNet	1 day	55	30	933,300	5 s
ShoppingMall	6 days	759	25	300,623	120 s

know about in every contact. This case gives an upper bound on how well double-spending can be prevented in each experiment.

- **No-sharing**: No spent money is shared among merchants; therefore, the only mechanism that prevents double-spending is the random choice of money according to Algorithm 1. This case gives a lower bound on how well any mechanism for sharing spent information among users can perform.
- **Bloom filter-sharing**: Merchants and Non-attacker customers share the set of spent monies using Bloom filter according to Algorithm 2. This case shows how well our solution performs.

For the evaluation metrics, we consider the double-spending attack success rate (FNR), the failure rate for valid transactions (FPR), and the correct decision rate (Accuracy). The formula for calculating each of these metrics is given in equation 5, 6, and 7. We argue that maintaining a low FPR is more crucial than minimizing the FNR because the harm caused by a false negative can be mitigated by managing the risk of accepting payments through methods such as assigning credit scores to users. On the other hand, a false positive means a loss of sale opportunity for the merchant that cannot be recovered. The accuracy gives an overall perspective of how well the system generally works and can be used to compare the proposed solution to traditional payment methods.

$$\mathrm{FNR} = \frac{\text{Accepted Invalid Payments}}{\text{Invalid Payments}} \tag{5}$$

$$\mathrm{FPR} = \frac{\text{Denied Valid Payments}}{\text{Valid Payments}} \tag{6}$$

$$\mathrm{Accuracy} = \frac{\text{Accepted Valid} + \text{Denied Invalid}}{\text{Total Payments}} \tag{7}$$

We first compare the FNR of *ideal* and *no-sharing* scenarios for each dataset to illustrate the effect of user's behavior on preventing double-spending. In this experiment, the attackers are stateless because the FNR in the no-sharing scenario would be 1 for stateful attackers, rendering the results unhelpful for our comparison. The measurements and the average value across all datasets are presented in Fig. 6. The results show that the FNR is different among datasets because the behavior of users and placement of stationary nodes varies among them. It is observed that sharing information consistently reduces the FNR, though the extent of reduction varies among datasets. Notably, the FNR for the Mall dataset is reduced by over 350% compared to the ideal scenario that shows strong connectivity among merchants of this dataset. Another interesting point is that although the FNR of BLE and Haggle/6 are close to each other in no-sharing scenario and the range of devices used in BLE dataset is larger, but in ideal scenario the FNR of Haggle/6 is much less than the other. This difference shows that mobility and placement of merchants plays more crucial role in preventing double-spending.

We conduct another experiment to study the effect of asset count and capacity of shared Bloom filters on the correct operation of the system. In order to do so, we measure the average accuracy over all datasets and compare them using the

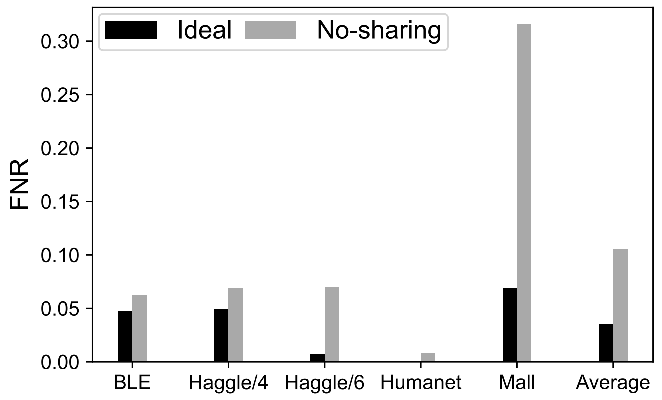

Fig. 6. FNR of double-spending detection in datasets for ideal and no-sharing scenarios.

heat map presented in Fig. 7. The results show that increased asset count improves accuracy if the capacity of the Bloom filter is high enough. This is the effect of using Algorithm 1 for payment that prevents attackers from choosing the assets used for payment freely. The increase in accuracy can be explained by the fact that with more assets, the probability of selecting the same money by two merchants decreases. Regarding the capacity of the Bloom filter, the increase in capacity results in an increase in accuracy. A notable point is that the accuracy improves more for lower asset counts. The reason for this is that when users have fewer assets to spend, the filters are less likely to reach their capacity and can therefore propagate over longer distances. In practice, users typically have limited assets in an offline environment, allowing for the selection of an appropriate capacity for the Bloom filter.

Finlay, we study the effectiveness of our proposed algorithm for sharing Bloom filters by comparing it with other methods previously detailed in Section IV-A. We measure the accuracy and FPR for each method with different attacker ratios, which is the fraction of users that act as attackers and try to double-spend. The parameters for the Bloom filter are selected to ensure its size is 240 bytes, which is well within the maximum message size limits of Bluetooth (512 bytes)

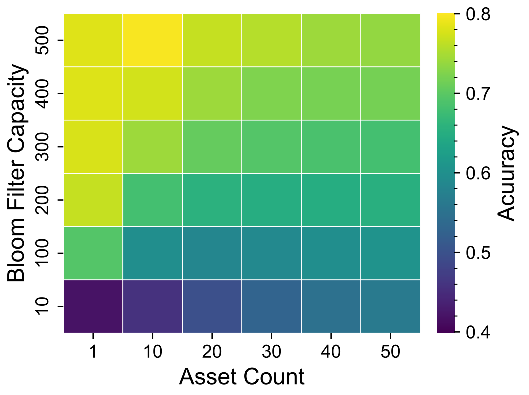

Fig. 7. Accuracy of the proposed method for different Bloom filter capacities and asset counts.

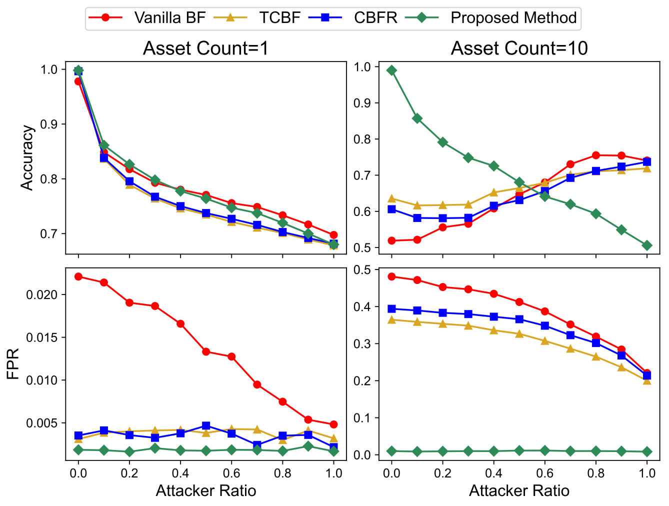

Fig. 8. Comparison of accuracy and FPR of the proposed method and other Bloom filter variants.

and WiFi Aware (256 bytes) [53]. Both of these networks are suitable for our solution. We consider two values for users' asset count separately to show how the methods under study perform under different circumstances. The results of the measurements are shown in Fig. 8. Each of the four plots represents measurements of our proposed solution along with three other methods.

The results in Fig. 8 show that when users have a single money to spend (left column), all methods behave the same. In this scenario, the accuracy of all methods decreases as the attacker ratio increases. Things become more interesting when each user has 10 monies (right column). Our proposed solution outperforms other methods when fewer than half of the users are malicious, which is typically expected in practice. On the other hand, when the attacker ratio increases above 0.5, the accuracy of our solution falls below other methods while the accuracy of other methods keeps improving! This observation can be attributed to the high FPR of other methods. In cases where most payments involve double spending, what seems like a false detection often turns out to be a true detection, resulting in increased accuracy. Additionally, it is worth noting that our proposed method demonstrates flexibility by consistently maintaining low FPR values across various conditions.

C. Discussion

The results of our experiments show that the success of preventing double-spending by sharing information among users depends on the mobility behavior of users and the amount of money circulating in the offline environment. Therefore, this approach should be complemented by measures that enhance connectivity among users and regulate the amount of spendable money in the offline environment. Even if these measures are not taken, since we consider low-value retail payments and the proposed method keeps the FPR low, the failure results in double-spending that can be recovered in the redemption phase by persecuting the double-spender.

There can be an attacker that broadcasts arbitrary Bloom filters to lower the effectiveness of the prevention method by polluting the Bloom filters of neighbor users. The risks associated with this kind of attacker are limited due to the refreshing property of the proposed sharing algorithm. Even though, separate work can be done to further evaluate the effect of this act on the effectiveness of the solution. An approach to deal with this denial-of-service type of attack is computing trust levels for users according to their activity and incorporating it into the payment protocol.

V. Conclusion

In this work, by using a random selection of money for each payment, setting the expiration date for monies, and sharing the identifiers of spent monies among users in an offline environment, we managed to prevent double-spending with an acceptable probability. Due to the usage of Bloom filters, the proposed solution has low overhead and can be used with resource-constrained devices. We evaluated the effectiveness of our proposed solution using real-world datasets. The findings indicate that the proposed method is suitable for offline retail payments under practical assumptions.

References

[1] J. Meyer and F. Teppa, "Consumers' payment preferences and banking digitalisation in the euro area," *SSRN Electronic Journal*, 2024, doi: 10.2139/ssrn.4757455.
[2] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.
[3] I. Simplot-Ryl, I. Traoré, and P. Everaere, "Distributed architectures for electronic cash schemes: a survey," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 24, no. 3, pp. 243–271, 2009.
[4] J. Enarsson and J. Holgersson, "On the viability of digital cash in offline payments," Ph.D. dissertation, Blekinge Institute of Technology, 71 79 Karlskrona, Sweden, 2022.
[5] V. D. Gauthier, K. Wouters, H. Karahan, and B. Preneel, "Offline NFC payments with electronic vouchers." ACM Press, 2009, pp. 25–30.
[6] V. Daza, R. Di Pietro, F. Lombardi, and M. Signorini, "FRoDO: Fraud resilient device for off-line micro-payments," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 296–311, 2015.
[7] I. Osipkov and E. Y. Vasserman, "Combating double-spending using cooperative P2P systems," *27th International Conference on Distributed Computing Systems (ICDCS '07)*, pp. 41–41, 2007.
[8] J.-H. Hoepman, "Distributed double spending prevention," in *Security Protocols: 15th International Workshop, Brno, Czech Republic, April 18-20, 2007. Revised Selected Papers 15*. Springer, 2010, pp. 152–165.
[9] K. Wei, A. J. Smith, Y.-F. Chen, and B. Vo, "Whopay: A scalable and anonymous payment system for peer-to-peer environments," in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*. IEEE, 2006, pp. 13–13.
[10] Z. Jia, S. Tiange, H. Liansheng, and D. Yiqi, "A new micro-payment protocol based on p2p networks," in *IEEE International Conference on e-Business Engineering (ICEBE'05)*. IEEE, 2005, pp. 449–455.
[11] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[12] B. I. Hub, *A handbook for offline payments with CBDC*. Bank for International Settlements, May 2023.
[13] H. Armelius, C. A. Claussen, and I. Hull, "On the possibility of a cash-like cbdc," Stockholm, Sveriges Riksbank Staff memo, 2021. [Online]. Available: https://hdl.handle.net/10419/231485
[14] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Annual International Cryptology Conference*, 1990, pp. 319–327.
[15] A. Rial and A. M. Piotrowska, "Compact and divisible e-cash with threshold issuance," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 381–415, 2023.

[16] B. Bauer, G. Fuchsbauer, and C. Qian, "Transferable e-cash: A cleaner model and the first practical instantiation," in *IACR International Conference on Public-Key Cryptography*. Springer, 2021, pp. 559–590.

[17] X. Zhou, "Threshold cryptosystem based fair off-line e-cash," in *2008 Second International Symposium on Intelligent Information Technology Application*. IEEE, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1109/iita.2008.87

[18] J. Burdges, F. Dold, C. Grothoff, and M. Stanisci, "Enabling secure web payments with GNU Taler," in *Security, Privacy, and Applied Cryptography Engineering*. Springer International Publishing, 2016, pp. 251–270.

[19] N. Andola, Raghav, V. K. Yadav, S. Venkatesan, and S. Verma, "Anonymity on blockchain based e-cash protocols–a survey," *Computer Science Review*, vol. 40, p. 100394, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013721000344

[20] A. Buldas, M. Saarepera, J. Steiner, and D. Draheim, "A unifying theory of electronic money and payment systems," May 2022. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.14994558

[21] S. Canard, D. Pointcheval, O. Sanders, and J. Traoré, "Divisible e-cash made practical," *IET Information Security*, vol. 10, no. 6, pp. 332–347, 2016.

[22] K.-W. Park and S. H. Baek, "Opera: A complete offline and anonymous digital cash transaction system with a one-time readable memory," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 10, pp. 2348–2356, 2017.

[23] W. Jie, W. Qiu, A. S. V. Koe, J. Li, Y. Wang, Y. Wu, J. Li, and Z. Zheng, "A secure and flexible blockchain-based offline payment protocol," *IEEE Transactions on Computers*, vol. 73, no. 2, pp. 408–421, 2023.

[24] M. Blaze, J. Ioannidis, and A. D. Keromytis, "Offline micropayments without trusted hardware," in *Financial Cryptography*. Springer Berlin Heidelberg, 2002, pp. 21–40.

[25] R. W. Blokzijl, "EuroToken: An offline capable central bank digital currency," Master's thesis, Delft University of Technology, Jul. 2021. [Online]. Available: http://repository.tudelft.nl/

[26] A. Madhusudan, M. Sedaghat, P. Jovanovic, and B. Preneel, "Nirvana: Instant and anonymous payment-guarantees," *IACR Cryptol. ePrint Arch.*, p. 872, 2022.

[27] Y. Chu, J. Lee, and S. Kim, "Review of offline payment function of CBDC considering security requirements," *Applied Sciences*, vol. 12, p. 4488, Apr. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/9/4488

[28] S. Wiesner, "Conjugate coding," *ACM Sigact News*, vol. 15, no. 1, pp. 78–88, 1983.

[29] Z.-Y. Lee, H.-C. Yu, and P.-J. Ku, "An analysis and comparison of different types of electronic payment systems," in *PICMET'01. Portland International Conference on Management of Engineering and Technology. Proceedings Vol. 1: Book of Summaries (IEEE Cat. No. 01CH37199)*. IEEE, 2001, pp. 38–45.

[30] E. Foo and C. Boyd, "A payment scheme using vouchers," in *International Conference on Financial Cryptography*. Springer, 1998, pp. 103–121.

[31] S. Sravan, S. Mandal, and P. Alphonse, "LIO-Pay: Sustainable low-cost offline payment solution," *Electronic Commerce Research and Applications*, vol. 67, p. 101440, 2024.

[32] C. Chakrabarti, "iCredit: A credit based incentive scheme to combat double spending in post-disaster peer-to-peer opportunistic communication over delay tolerant network," *Wireless Personal Communications*, vol. 121, no. 3, pp. 2407–2440, 2021.

[33] A. Madhusudan, M. Sedaghat, S. Tiwari, K. Cong, and B. Preneel, "Reusable, instant and private payment guarantees for cryptocurrencies," in *Australasian Conference on Information Security and Privacy*. Springer, 2023, pp. 580–605.

[34] P. Everaere, I. Simplot-Ryl, and I. Traoré, "Double spending protection for e-cash based on risk management," in *International Conference on Information Security*. Springer, 2010, pp. 394–408.

[35] Y. Yacobi, "Risk management for e-cash systems with partial real-time audit," *Netnomics*, vol. 3, pp. 119–127, 2001.

[36] R. L. Rivest, "Electronic lottery tickets as micropayments," in *International Conference on Financial Cryptography*. Springer, 1997, pp. 307–314.

[37] C. Beer, S. Zingg, K. Kostiainen, K. Wüst, V. Capkun, and S. Capkun, "Payoff: A regulated central bank digital currency with private offline payments," *arXiv preprint arXiv:2408.06956*, 2024.

[38] A. Kurt, A. Sahin, R. Harrilal-Parchment, and K. Akkaya, "Lnmesh: Who said you need internet to send bitcoin? offline lightning network payments using community wireless mesh networks," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2023, pp. 261–270.

[39] K. Nikitin, E. Kokoris-Kogias, P. Jovanovic, N. Gailly, L. Gasser, I. Khoffi, J. Cappos, and B. Ford, "CHAINIAC: Proactive software-update transparency via collectively signed skipchains and verified builds," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1271–1287.

[40] I. Ozcelik, S. Medury, J. Broaddus, and A. Skjellum, "An overview of cryptographic accumulators," *arXiv preprint arXiv:2103.04330*, 2021.

[41] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[42] S. Lee, H. Byun, and H. Lim, "Set reconciliation using ternary and invertible bloom filters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11 885–11 898, 2023.

[43] L. Luo, D. Guo, R. T. Ma, O. Rottenstreich, and X. Luo, "Optimizing bloom filter: Challenges, solutions, and comparisons," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1912–1949, 2018.

[44] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality estimation and dynamic length adaptation for bloom filters," *Distributed and Parallel Databases*, vol. 28, pp. 119–156, 2010.

[45] A. Reinhardt, O. Morar, S. Santini, S. Zöller, and R. Steinmetz, "CBFR: Bloom filter routing with gradual forgetting for tree-structured wireless sensor networks with mobile nodes," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2012, pp. 1–9.

[46] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM transactions on networking*, vol. 8, no. 3, pp. 281–293, 2000.

[47] Y. Zhao and J. Wu, "The design and evaluation of an information sharing system for human networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 796–805, 2013.

[48] D. Sikeridis, I. Papapanagiotou, and M. Devetsikiotis, "BLEBeacon: A real-subject trial dataset from mobile bluetooth low energy beacons," *arXiv preprint arXiv:1802.08782*, 2018.

[49] A. Natarajan, M. Motani, and V. Srinivasan, December 11, 2022, "Crawdad nus/bluetooth," IEEE Dataport, doi: https://dx.doi.org/10.15783/C74K5N.

[50] A. Chaintreau, P. Hui, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Impact of human mobility on opportunistic forwarding algorithms," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 606–620, June 2007, (previously published in the Proceedings of IEEE INFOCOM 2006). [Online]. Available: http://www.thomson.net/~chaintre/pub/chaintreau07impact.pdf

[51] J. M. Cabero, V. Molina, I. Urteaga, F. Liberal, and J. L. Martin, November 29, 2022, "Crawdad tecnalia/humanet," IEEE Dataport, doi: https://dx.doi.org/10.15783/C74G60.

[52] A. Galati, K. Djemame, and C. Greenhalgh, "A mobility model for shopping mall environments founded on real traces," *Networking Science*, vol. 2, pp. 1–11, 2013.

[53] D. Camps-Mur, E. Garcia-Villegas, E. Lopez-Aguilera, P. Loureiro, P. Lambert, and A. Raissinia, "Enabling always on service discovery: Wifi neighbor awareness networking," *IEEE Wireless Communications*, vol. 22, no. 2, pp. 118–125, 2015.