Secure Wallet-Assisted Offline Bitcoin Payments with Double-Spender Revocation

Alexandra Dmitrienko
Institute of Information Security, ETH Zurich
alexandra.dmitrienko@inf.ethz.ch

David Noack
Technische Universität Darmstadt
david.noack@cased.de

Moti Yung
Snapchat, Inc. and Columbia University
moti@cs.columbia.edu

ABSTRACT

Bitcoin seems to be the most successful cryptocurrency so far given the growing real life deployment and popularity. While Bitcoin requires clients to be *online* to perform transactions and a *certain amount of time* to verify them, there are many real life scenarios that demand for *offline* and *immediate* payments (e.g., mobile ticketing, vending machines, etc). However, offline payments in Bitcoin raise non-trivial security challenges, as the payee has *no* means to verify the received coins without having access to the Bitcoin network. Moreover, even online immediate payments are shown to be vulnerable to double-spending attacks.

In this paper, we propose the first solution for Bitcoin payments, which enables secure payments with Bitcoin in offline settings and in scenarios where payments need to be immediately accepted. Our approach relies on an offline wallet and deploys several novel security mechanisms to prevent double-spending and to verify the coin validity in offline setting. These mechanisms achieve probabilistic security to guarantee that the attack probability is lower than the desired threshold. We provide a security and risk analysis as well as model security parameters for various adversaries. We further eliminate remaining risks by detection of misbehaving wallets and their revocation.

We implemented our solution for mobile Android clients and instantiated an offline wallet using a microSD security card. Our implementation demonstrates that smooth integration over a very prevalent platform (Android) is possible, and that offline and online payments can practically co-exist. We also discuss alternative deployment approach for the offline wallet which does not leverage secure hardware, but instead relies on a deposit system managed by the Bitcoin network.

Keywords

Bitcoin; Offline payments; Zero confirmation transactions; Double-Spender revocation; Secure hardware

AsiaCCS '17, April 4–6, 2017, Abu Dhabi, United Arab Emirates

ISBN 978-1-4503-4944-4/17/04…$15.00.

DOI: http://dx.doi.org/10.1145/3052973.3052980

1 Introduction

Electronic payments and mobile banking have experienced rapid development and deployment with hundred millions of customers worldwide [12]. Mobile banking services have particularly been successful in developing countries and enable financial services to the large part of the population that is unbanked or cannot use solutions offered by banks [19].

Electronic cash has been an attractive subject of research and investigation since the late 1980s, starting with the seminal paper by David Chaum [26] followed by a large body of subsequent works. Cryptographic techniques were introduced to tackle security and privacy challenges such as protecting against forgery and double-spending and providing anonymity. However, almost all the proposed solutions so far involve trusted third parties such as banks to generate and validate digital cash. Moreover, none of these cryptocurrencies have ever made it to be used for real-world payments.

In contrast, Bitcoin [49] relies neither on banks nor on any other central authority for coin issuing or verifying transactions and seems to be the most successful cryptocurrency that is used for real world payments. There were at least over 100,000 merchants worldwide that accepted bitcoins in 2015 [29], and large support by merchant tools and payment processors [13, 23] is likely to increase this number to millions in 2016.

Indeed Bitcoin has initially celebrated enormous success driven by the large interest from users as well as academia and markets. Within a short time Bitcoin has been thoroughly analyzed with regards to security [42, 20, 35], privacy issues [15, 51, 46] as well as economic aspects [16, 44]. Moreover, a number of alternative cryptocurrencies (*altcoins*) were proposed that have made substantial changes to initial Bitcoin design with different goals. For instance, Zerocoin [47], Zerocash [21], CryptoNote [55] and PinnochioCoin[30] aim at providing advanced anonymity, Litecoin [9] and Dogecoin [6] use "memory-hard" puzzles which can be effectively solved using commodity hardware, Primecoin [43] introduced the useful puzzle which requires finding sequences of large prime numbers, while Ethereum [24] extends Bitcoin's transaction semantics to enable advanced transaction types. However, Bitcoin seems to still be the most popular and promising cryptocurrency for deployment in practice and believed to continue to co-exist with traditional financial systems.

The most challenging security requirements of digital currency, unforgeability and double-spending prevention, are addressed in Bitcoin by means of asymmetric cryptography

and a distributed time-stamping service based on proof-of-work (PoW). As a consequence, transactions cannot be considered definite as soon as they are received, because it takes some time for the Bitcoin network to verify and integrate them in a state that is very hard to change. Hence, a receiver of a Bitcoin transaction requires an *online connection* to the Bitcoin network in order to determine validity of the transaction, as well as a *certain amount of time* (10 min. in average). This makes, however, offline payment with bitcoins extremely challenging, although in many real world scenarios (e.g., taxis, mobile ticketing, vending machines, etc.) offline payments are highly desirable. Moreover, Bitcoin payments are increasingly used at POS terminals for immediate payments, where purchased items are released within a few seconds after the payment and before the transaction confirmation have been generated by the network, although it was already shown that such deployments are vulnerable to double-spending attacks [42].

Goals and Contributions

We present an extension to Bitcoin protocol which enables secure payments with Bitcoin in offline settings and in scenarios where payments need to be immediately accepted. In particular, our contributions are as follows:

- *Solution for immediate and offline payments.* We analyze non-trivial challenges of offline and immediate payments with Bitcoin (Section 3) and, for the first time, propose a solution which requires neither payer nor payee to be online during the payment and in which payments can be instantly accepted (Section 4). The solution relies on an offline wallet of a payee and incorporates probabilistic security mechanisms, which provide guarantees that the attack probability is lower than the desired threshold. Moreover, we eliminate remaining risks by introducing safe yet usable transaction limits and revocation mechanisms for double spending wallets in order to make attacks unprofitable for an adversary. We provide a rigorous risk analysis of our solution and suggest exemplary values for the security parameters (Section 5).

- *Implementation and Evaluation.* We provide a proof-of-concept implementation for mobile Android platforms and instantiate an offline wallet using a microSD security card (Section 6). Minor changes to Bitcion miners for revocation support are integrated into `bitcoind` client. We report performance measurements of operations performed on resource constrained microSD card, which demonstrate the feasibility of our approach for deployment in practice. We further discuss an alternative deployment option for the offline wallet, which relies on Bitcoin-based deposit system and does not require any secure hardware (Section 7).

Our scheme provides a valuable solution to a challenging problem and enables the promising use of Bitcoin in rising developing markets with the need for innovative and independent payment solutions.

2 Bitcoin Basics

We begin with a brief description of Bitcoin ecosystem. For a more detailed description we refer the reader to [36].

Involved Parties Generally, Bitcoin system assumes two types of users: Regular users $\mathcal{U}$ and miners $\mathcal{M}$. A regular user $X \in \mathcal{U}$ can utilize the Bitcoin network for exchanging bitcoins with another user $Y \in \mathcal{U}$ by means of transactions, either spending or receiving them. Regular users own (pseudonymous) accounts identified by addresses and associated asymmetric key pairs. Generally, a single transaction can transfer funds between several accounts at once, i.e., it can involve several sender and destination accounts, however for simplicity and without loss of generality we will assume throughout the paper that transactions involve one sender and one destination account.

Miners $\mathcal{M}$ are the actual operators of Bitcoin network. Each miner $M \in \mathcal{M}$ works on verifying transactions and includes them into the public history of all transactions, called the blockchain $\mathcal{B}$. There are no special miner accounts in the network, however miners typically own accounts of regular users in order to receive rewards for transaction processing.

Blockchain operation The blockchain is simply a chain (or sequence) of blocks, which can be extended by appending a new valid block. Each block in the chain references the previous block, which defines the unique order of blocks in the sequence. Creation of a new valid block requires miners to solve a cryptographic puzzle, which requires significant computational effort. The puzzle used by Bitcoin network requires to find an input to a hash function (randomized by a nonce) which results in a hash value less than a specific target value. Hence, miners usually have to compute a large number of hashes until they solve the puzzle. The target value is a security parameter which regulates difficulty of the puzzle, which is adjusted to the computational power available in the network.

As soon as the new block is created, all the transactions included into the block are considered as being confirmed by the network. The more subsequent blocks have been appended to the current block, the harder it gets to tamper with included transactions, as this would require to recompute all the subsequent blocks. Hence, one could say that the transaction τ is confirmed by n-transaction confirmation $n\text{-T} = \{B_i, ..., B_{i+n}\}$, if it is included into the block B_i and there are n subsequent blocks appended to B_i. The larger n is, the higher is the confidence in the validity of τ. For simplicity, we will refer to blocks from n-transaction confirmation as *confirmation blocks* throughout the paper.

Generally, transactions can be verified on different levels. While miners and powerful nodes perform a full verification, many (especially mobile) clients perform a more lightweight processing, called simple payment verification (SPV). The full verification includes checks of correctness of the transaction syntax, verification of all preceding transactions, which increased balance of the current account, as well as ensures that the account balance has not been already spent (by checking the entire blockchain). In contrast, SPV clients verify only transaction syntax and rely on n-transaction confirmation issued by the network, which implies that all the other verifications were successfully performed by full nodes. Confirmation blocks are verified by SPV clients at the level of *blockheaders* (i.e., without verifying transactions included into blocks), which is sufficient to ensure that confirmation blocks are the part of the entire blockchain and they were generated using the appropriate difficulty.

3 Threat Analysis and Challenges

In this section we analyze threats of offline Bitcoin payments and discuss associated challenges.

3.1 Coin forgery attacks

Coin forgery attacks do not impose threat to immediate payments for full clients, as they have a local copy of the blockchain and can verify validity of all preceding transactions on their own, without relying on transaction confirmations issued by the network. However, in offline scenario detection of forged coins is challenging even for full nodes, as they may not have a part of the blockchain which is necessary for verification of (one of) the preceding transaction(s). This may happen if the node got offline before the preceding transaction have been integrated into the blockchain.

To address this threat, we propose a new *time-based transaction confirmation verification* mechanism (cf. Section 4.2). Additionally to regular checks performed by the standard transaction confirmation verification of Bitcoin, it considers a time window within which the transaction confirmation was generated. As a result, it can provide a high confidence even in offline settings that the transaction confirmation was produced by the Bitcoin network rather than by an adversary.

3.2 Double-Spending Attacks

The risk of double-spending arises from the fact that a payer has always access to private keys of bitcoins he owns, and, hence, can always reuse them.

While online Bitcoin clients can protect themselves from double-spending attacks by observing the Bitcoin network and verifying a transaction confirmation issued by the network, offline clients do not have this option. Further, in immediate payments, even connected clients are vulnerable to double-spending, as they consider transactions being valid as soon as they appear in the Bitcoin network, but before their confirmations are generated.

Our approach to deal with double-spending attacks in offline scenarios is to rely on an offline wallet, which either behaves correctly, or its misbehavior is penalized. One way to achieve this in practice is to realize such a wallet on top of tamper-proof secure hardware, which can be compromised only at significant costs [53]. Furthermore, even if compromised, one can detect and revoke it, so that it can only misbehave for a limited period of time.

While the idea of using secure hardware may seem controversial in context of Bitcoin payments, as it would normally imply some trust assumptions on origin of this hardware, we argue that (online) Bitcoin users already trust in similar way to manufacturers of their computing platforms. There are also other implicitly trusted parties in Bitcoin ecosystem, such as developers of bitcoin wallets or large mining pools [38]. Hence, we believe that it is reasonable to trust manufacturers of secure hardware, such as ARM TrustZone [14] or Intel SGX [45], as long as users already use ARM and Intel processors to run their online bitcoin wallets. Nevertheless, in Section 7 we discuss an alternative deployment approach for offline wallets, which does not rely on secure hardware.

State-of-the-art payment solutions, such as GoogleWallet [7] and ApplePay [2], already rely on tamper-proof wallet environments. However, they are inapplicable in case of Bitcoin and cannot be used to achieve offline Bitcoin payments, as they commonly rely on a trusted authority, such as a bank, to pre-load coins into the wallet – an entity which does not exist in Bitcoin ecosystem. Moreover, as we elaborate in the following, limitations and constrains of commodity secure wallet environments make it challenging to verify validity of bitcoins from within the wallet even for online users.

3.3 Challenges

(1) The first challenge is related to resource constrains of commodity wallet environments which are likely to render full validation of the blockchain (and, hence, transaction validation) within the wallet environment infeasible, and even while being online. In fact, it takes days to download and verify the whole blockhain even on resource-reach platforms such as PCs [5], while wallet environments are typically much more constrained. Especially mobile wallets may not be able to perform even lightweight SPV verification which is carried out at the level of blockheaders (cf. Section 2 for more details), as they may not have enough resources to store sufficient number of blockheaders. This makes it challenging to ensure within the wallet environment that the transaction confirmation blocks are linked to the blockchain. We approach this problem in the same way as the problem of coin forgery attacks against offline clients (cf. Section 3.1), and apply *time-based transaction confirmation verification* to verify (online) transactions from within wallet environments.

(2) The second challenge is imposed by the fact that typical wallet environments do not feature direct access to networking interfaces, but their network access is rather mediated by hosting platforms to which they are connected to. This is specifically a problem if a host platform is controlled by an adversary, who can then mediate network access of the wallet and manipulate its system view in the similar manner as performed during Eclipse attacks [41]. For instance, the wallet can be tricked to accept a fake transaction as valid by providing an adversarial blockchain, generated with low difficulty, in which this fake transaction is referenced. To address this challenge, we propose to use *delayed parameters verification*, which postpones the verification of critical parameters to a later offline payment stage and involves the payee to verify them.

(3) The third challenge is related to the estimation of time window within which the transaction confirmation was produced by the Bitcoin network, which is a security-sensitive verification parameter of our *time-based transaction confirmation verification*. This requires either a local time source (e.g., a timer), or access to external time provider. However, a secure timer is not a standard feature of commodity wallet environments. Further, although correct time can be obtained from network-based time services and even in authenticated manner [10], it is preferable to avoid relying on third party services. To overcome this challenge, we propose a method to reliably estimate an upper bound for the time window and compensate over-approximation by inferring *limits on transaction amounts*, which should render respective attacks unprofitable for an adversary.

(4) The fourth challenge arises from a threat of the offline wallet being compromised. And, although it is known to be difficult to compromise commodity secure wallet environments in practice [53], advanced adversaries might invest

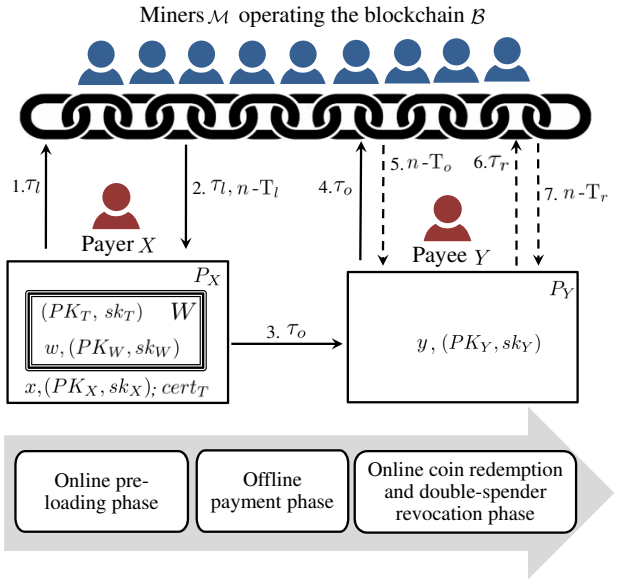


Figure 1: System model for offline Bitcoin payments

significant amount of time and resources into an attack if in return they can double-spend without any limit. Hence, inspired by previous works on digital cash [27], we propose to detect misbehaving wallets and enforce their revocation and present a new *distributed wallet revocation* scheme which does not rely on any external third party, but utilizes Bitcoin network to distribute revocation information.

4 Secure Offline Bitcoin Payments

In this section we present details of our solution. We begin by specifying our system and adversary model, then introduce new security mechanisms and, finally, present protocols which illustrate integration of new building blocks into the payment scheme.

4.1 System Model

Our system model is depicted in Figure 1. It includes the Bitcoin infrastructure consisting of the blockchain $\mathcal{B}$ and the miners $\mathcal{M}$. Further, it includes two regular users $X, Y \in \mathcal{U}$, where X is a payer who sends an offline transaction τ_o to the payee Y. Both users have computing platforms P_X and P_Y, respectively. Each platform P_X (resp. P_Y) executes the Bitcoin client software, which manages respective user accounts x, y and corresponding key pairs (PK_X, sk_X) and (PK_Y, sk_Y). Additionally, the platform P_X has an offline wallet W which manages its own account w and corresponding key pair (PK_W, sk_W). Furthermore, the wallet W has a certified key pair (PK_T, sk_T)[1], and its certificate $cert_T$ is known to P_X.

Because both, the payer X and the payee Y, have no online connection during the payment, the offline transaction τ_o is sent via local interfaces (e.g., Near Field Communication (NFC) or Bluetooth Low Energy (LTE)), in contrast to regular (online) Bitcoin transactions which are transferred via the Bitcoin peer-to-peer network. Nevertheless, both parties occasionally go online, for instance, X to receive preceding transactions to their account, while Y to redeem the received offline transaction after the payment[2].

Payment scenario Our solution consists of 3 phases (cf. Figure 1): (i) online coin preloading, (ii) offline payment, and (iii) online coin redemption and double-spender revocation. In the first phase, the payer X generates pre-loading transaction τ_l (step 1) that transfers some bitcoins from her standard Bitcoin account x to the offline wallet's account w, so that the balance of w becomes positive. This is done by means of standard online Bitcoin transaction, for which the network generates n-transaction confirmation n-T$_l$. In the second phase the payer X requests W to generate an offline transaction τ_o with the desired amount destined to the account y (step 3). In the third phase, the payee Y redeems the bitcoins he received offline by broadcasting τ_o into the Bitcoin network (step 4) and optionally obtaining network confirmation n-T$_o$ (step 5). The network confirmation will only be issued, if the network has not detected a double-spending attack against τ_o. Otherwise, the payee Y will trigger an optional double-spender revocation procedure, which includes sending a double-spender revocation transaction τ_r (step 6) to the Bitcoin network and obtaining corresponding confirmation n-T$_r$ (step 7). We indicate optional and conditional steps in Figure 1 using dashed arrows.

Adversary model and assumptions Our adversary model is similar to that of Bitcoin, where an adversary $\mathcal{A}$ is a malicious user $X \in \mathcal{U}$, and, optionally, $X \in \mathcal{M}$. In particular, X aims to get financial benefit by paying with invalid bitcoins such as forged or double-spent transactions and/or get them included into the blockchain. The most fundamental assumption is that even if $X \in \mathcal{M}$, she does not control more than 50% of the computational power available to the Bitcoin network. Moreover, it is assumed that the adversary $\mathcal{A}$ has control over her own platform P_Y, but she cannot compromise platforms of other users, i.e., she has no control over the platform of the payee P_Y[3].

We also assume that the malicious payer X can compromise her offline wallet W, however, the attack is associated with significant costs. The latter assumption is beyond standard adversary model of Bitcoin, and, as we discussed already in Section 3.2, it can be fulfilled by utilizing tamper-resistant wallet environments, which are widely available on both, mobile and PC platforms (e.g., Intel SGX [45], ARM TrustZone [14] and ASSD cards [39], to name some), and already commonly used by payment solutions (e.g., GoogleWallet [7] and ApplePay [2]).

In Section 7 we discuss an alternative deployment option of the offline wallet which neither relies on secure hardware nor on their manufacturers for certification.

4.2 Security Mechanisms

In the following we present our new security mechanisms which are motivated by threats and challenges discussed in Section 3.

[1]Such keys are typically available within hardware-based secure environments and certified by secure hardware manufacturers.

[2]Note that even offline devices like vending machines may have temporary connectivity (e.g., provided via hotspots of personnel supplying items).

[3]Otherwise the adversary could trivially trick the payee Y to accept any invalid translation by displaying the fake notification that the transaction is valid.

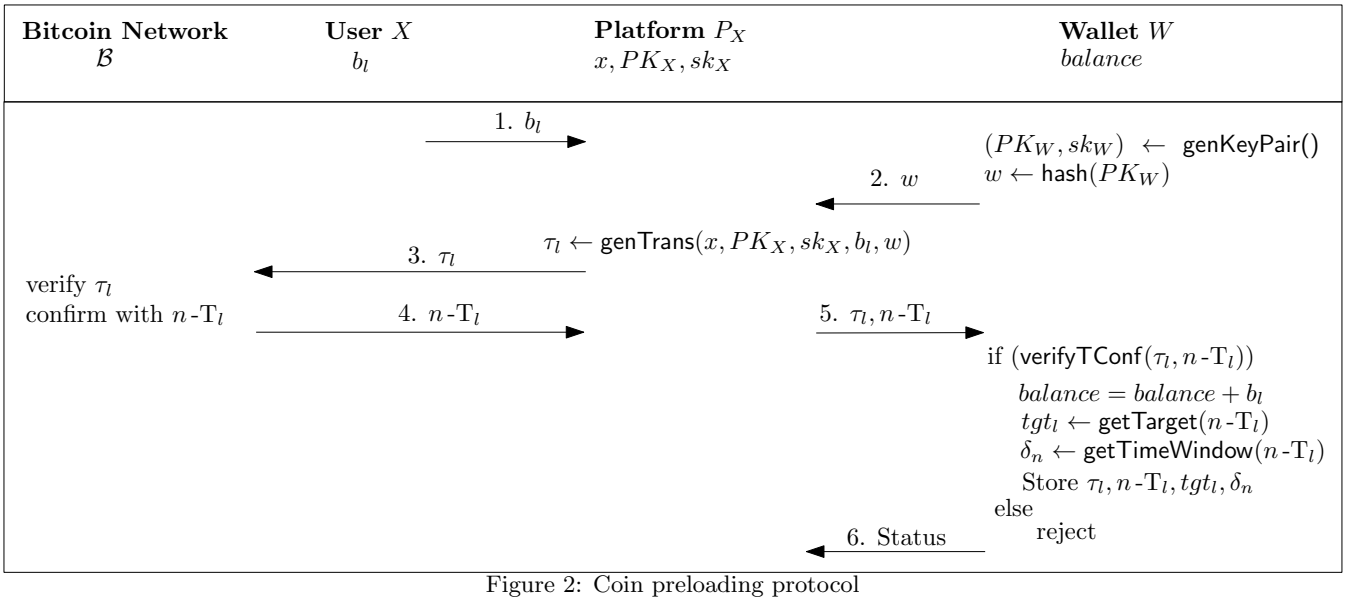


Figure 2: Coin preloading protocol

Time-based Transaction Confirmation Verification This mechanism enables us to get a high confidence in validity of transaction confirmation even without the ability to check that the transaction confirmation is linked to the entire blockciain. In a nutshell, it is an enhancement of a standard transaction confirmation of Bitcoin, which requires Bitcoin network to produce n-transaction confirmation n-T $= \{B_i, ..., B_{i+n}\}$ with the specified difficulty. Beyond that, our mechanism additionally requires n-transaction confirmation to satisfy time constraints $\delta_n := t_i - t_{i+n} \leq n \cdot \delta$, where t_i and t_{i+n} are time stamps extracted from blocks B_i and B_{i+n}, respectively, and δ is a security parameter.

We analyze this mechanism and potential attacks as well as provide estimations for reasonable values for δ and n in Section 5.1.

Delayed Parameters Verification To deal with the challenge of untrusted and potentially manipulated inputs within the wallet environment, we propose to split transaction confirmation verification process into two phases. In the first phase the wallet W validates the pre-loading transaction τ_l without getting assured that validation parameters (obtained from untrusted input sources) are correct, but storing them for future use. In the second phase, which takes place during an offline payment, these parameters are compared against reference values provided by P_Y, which are trustworthy, as there is no incentive for P_Y to manipulate the values (otherwise it would risk to accept an invalid transaction).

The values which can be verified using this method are the security parameter δ, the number of confirmation blocks n, and the target value tgt. However, obtaining the reference values for t_i and t_{i+n} is more challenging, as there might be no n-T$_l$ available on P_Y (e.g., if the payee Y got offline before τ_l have been confirmed by the network). Furthermore, the time stamps extracted from n-transaction confirmation received by W during pre-loading phase are not trustworthy, as they could be forged by an adversary controlling less than 50% of the network power. In particular, an adversary $\mathcal{A}$ could search for confirmation blocks over longer time, while manipulating time stamps in such a way that they look like the confirmation was generated faster. Even more crucial, once generated, such a spoofed transaction confirmation could be re-used to cheat multiple wallets.

As a countermeasure, we propose to estimate bounds for $\delta_n := t_i - t_{i+n}$ by (i) creating a fresh bitcoin address for every pre-loading transaction, which effectively bounds t_i[4] and also prevents an adversary from re-using the spoofed confirmation with several wallets. Further, we suggest to (ii) use time of offline payment t_p as an upper bound of t_{i+n}.

Transaction Limits Depending on time passed in between the transaction τ_l was loaded and until τ_o is spent, verification of a condition $\delta_n := t_p - t_i \leq n \cdot \delta$ may fail for reasonable δ values even for legitimate confirmations, which may impact usability.

To mitigate this side effect of our over-approximation, we suggest that it is safe for Y to relax δ parameter if transaction amounts for offline payments are limited in such a way that transaction confirmation spoofing attacks become unprofitable. In Section 5.2 we provide cost analysis for transaction confirmation spoofing attacks and show that they are sufficiently high to allow for safe yet usable transaction limits.

Distributed Wallet Revocation Bitcoin transactions can carry out some limited amount of data (up to 80 bytes per record). Inclusion of data renders the actual amount transferred by such a transaction unspendable, but the transaction gets still integrated into the blockchain. We leverage this feature in order to instantiate Bitcoin-based distributed revocation manager. In particular, we represent revocation requests in the form of a special transaction, which is sent by a cheated payee Y to the special revocation address which is publicly known and for which there is no corresponding signing key. The revocation transaction contains the public key of the accused wallet in the data field, and, once it is integrated into the blockchain, it serves as an entry in the

[4]This is because neither the network, nor an adversary could produce correct n-T$_l$ before such an address was generated.

revocation list and can be downloaded by Bitcoin clients. Revocation updates can be received by all the clients by monitoring all the transactions sent to the revocation address. Offline clients receive these updates whenever they go occasionally online, e.g., to redeem the offline transaction after the payment[5].

4.3 Protocol Design

In the following we present protocol design for each phase of our solution: (i) coin-preloading, (ii) offline transaction, and (iii) coin redemption and double-spending wallet revocation.

Notations In this section we use the following notations and conventions. We denote an algorithm as $out \leftarrow \mathsf{A}(in)$, where A is the name of the algorithm, in is the list of input parameters and out is the list of output values, potentially a boolean value. With $\sigma \leftarrow \mathsf{sign}(sk; m)$ we denote a signature on message m under signing key sk which can be verified by $\{true, false\} \leftarrow \mathsf{verify}(PK; m, \sigma)$. When checking for the result of a function that outputs boolean values, we simply write $\mathsf{A}(in)$ instead of $\mathsf{A}(in) \stackrel{?}{=} true$ for brevity.

4.3.1 Coin Preloading

In the coin-preloading protocol, shown in Fig. 2, the payer X first indicates the amount of bitcoins b_l she would like to preload into her wallet W (step 1). Next, P_X requests a new account w from the wallet (step 2), then creates the pre-loading transaction τ_l transferring b_l bitcoins from her x account to w and commits it to the network (step 3). As soon as τ_l is verified by the Bitcoin network and confirmation n-T$_l$ is issued (step 4), X provides τ_l and n-T$_l$ to the wallet W (step 5), which in turn runs an algorithm $\mathsf{verifyTConf}$ with the parameters τ_l and n-T$_l$ to perform time-based transaction confirmation verification (cf. Section 4.2). If successful, W increases its *balance* by b_l, extracts difficulty tgt from confirmation blocks and estimates δ_n using time stamps t_i and t_{i+n} (extracted from n-T$_l$). It then stores values τ_l, n-T$_l$, tgt, δ_n for future use. When done, it replies to P_X with status (step 6), which notifies the user whether the transaction was accepted by the wallet.

4.3.2 Secure Offline Transaction

The secure offline transaction protocol is shown in Fig. 3. It is initiated by Y, who indicates to his platform P_Y the demanded amount b_o (step 1). Next, P_Y sends the public key PK_Y to the platform P_X which immediately forwards it to W[6] (step 2). P_X replies to P_Y with $cert_T$, the certificate issued to the wallet environment by its manufacturer (step 3). P_Y validates $cert_T$ and, if correct, runs Diffie-Hellman key exchange protocol with W to establish a session key K (step 4), which is then used to protect all the subsequent messages[7]. The next message from P_Y to W transfers b_o and payee's reference values of verification parameters tgt_Y, n_Y and δ (step 5). Meanwhile, W calculates Bitcoin address y of Y by hashing PK_Y. As soon as W receives b_o (along with the security parameters), it displays to the payer X the transaction destination y and amount to pay b_o (step 6). If acknowledged (step 7), W ensures that it has sufficient funds, i.e., $b_o \leq balance$, and verifies if τ_l was confirmed by a sufficient number of blocks by checking if n equals or greater than n_Y. It also compares the target values tgt and tgt_Y to ensure that n-T$_l$ was calculated with a satisfying difficulty. Finally it validates if the security condition $\delta_n \leq n_Y \cdot \delta$ holds. If all checks pass, it generates a transaction τ_o, which transfers b_o amount of bitcoins from wallet's address w to payee's address y. The transaction is signed with the wallet's Bitcoin key sk_W. Further, W generates a proof that this transaction was created within the secure wallet environment by signing τ_o with its certified key sk_T.

The resulting τ_o is sent to P_Y along with the signature $proof$ and the time stamp t_i (extracted from the block B_i of n-T$_l$) (step 8). Upon receive, the transaction is verified to match the address y, to include amount b_o and to be syntactically correct (e.g., correctly signed). Furthermore, $proof$ is verified to be a valid signature made over the transaction τ_o. Additional check is performed to verify if $t_p - t_i$ (where t_p is a current time) equals or greater than $n_Y \cdot \delta$, and, if failed, an additional condition is triggered to ensure that the transaction amount b_o does not exceed a safe limit b_{max}.

If successful, P_Y stores τ_o, $cert_T$ and $proof$ for future use. Finally, it replies to W with the status (step 9), which, in turn, decreases *balance* by b_o and, in case there are no more funds in the account w left, deletes its key sk_W.

4.3.3 Coin Redemption and Wallet Revocation

The coin-redemption and wallet revocation protocol is depicted in Figure 4. It involves a payee Y and the Bitcoin network as communicating parties and relies neither on the payer X nor on his or her offline wallet W. The protocol begins by a payee Y, who redeems the coins received earlier from X by broadcasting τ_o to the Bitcoin network (step 1). Next, the Bitcoin network either confirms τ_o by sending transaction confirmation n-T$_o$ (step 2), or, if it detects double-spending transaction, no transaction confirmation is issued[8]. In the latter case the payee Y initiates revocation by creating a revocation transaction τ_r, which is sent from y signed with sk_Y. It is destined to the pre-defined revocation address and includes a hash of the public key PK_T of the compromised offline wallet to be revoked. The revocation transaction is then committed to the Bitcion network along with τ_o, $cert_T$ and $proof$ (step 3). In turn, the Bitcoin network verifies the proof that the double-spending transaction was indeed signed by PK_T (extracted from $cert_T$), and if correct, the revocation request is accepted and the revocation transaction τ_r is integrated into the blockchain.

5 Analysis of Security Parameters

In this section we elaborate on how the security parameters n and δ should be determined. We recall that they are used by our *time-based transaction confirmation verification* mechanism (cf. Section 4.2) to decide whether an n-transaction confirmation comes from the Bitcoin network

[5]Even disconnected platforms like vending machines can be provided regular updates through mobile hotspots brought, e.g., by a personnel supplying snacks.

[6]Any communication with W is mediated by P_X, which we do not show in the figure for brevity.

[7]Note that messages between the offline wallet W and the payee P_Y (including the offline transaction) must be exchanged via a secure channel to prevent their malicious manipulation. This is different for online Bitcoin payments where transactions go through the decentralized Bitcoin peer-to-peer network unencrypted, as the trust is put on the validity of the blockchain.

[8]We rely on standard mechanisms of Bitcoin network for detection of double-spending transactions.

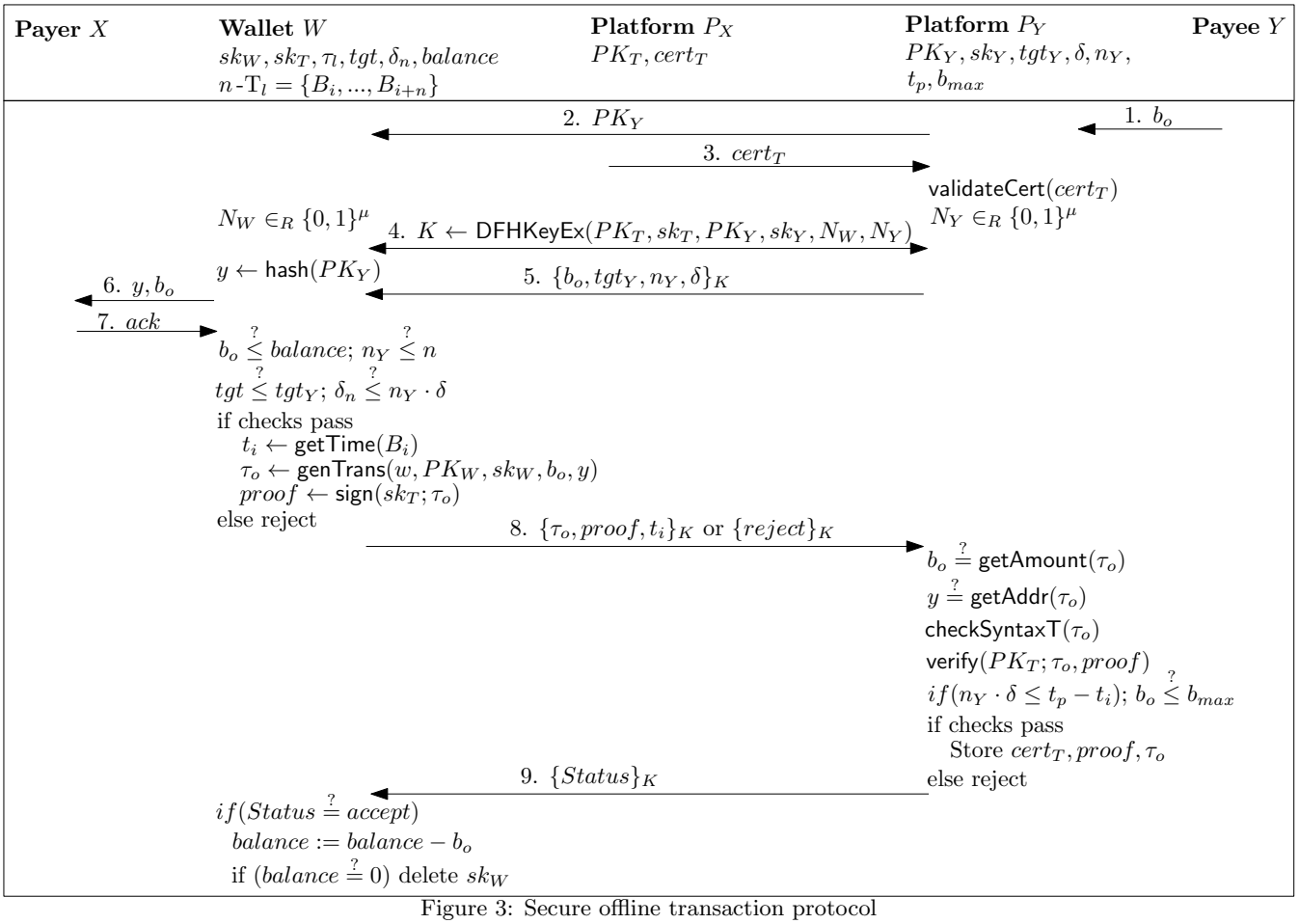


Figure 3: Secure offline transaction protocol

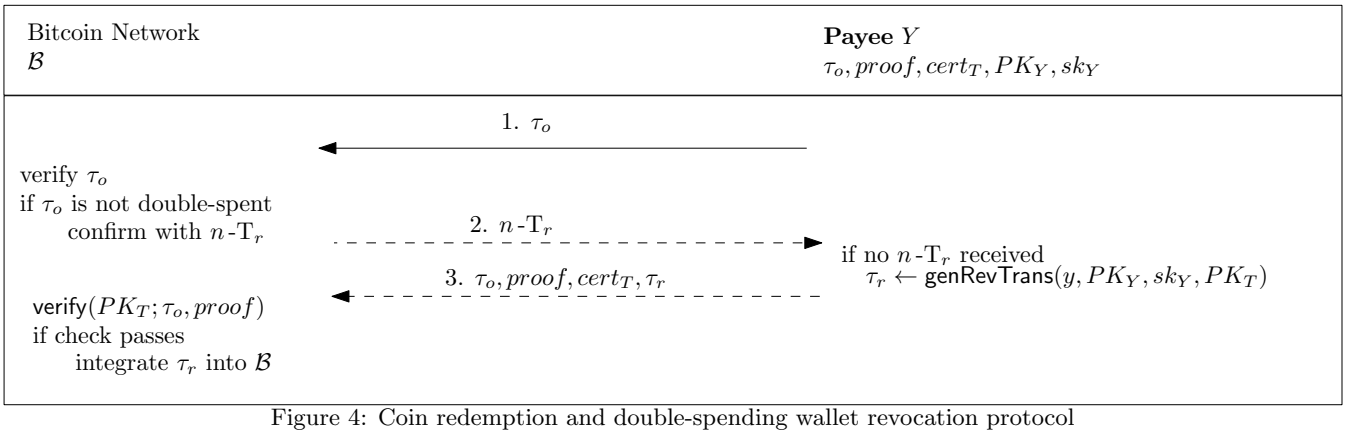


Figure 4: Coin redemption and double-spending wallet revocation protocol

or from an adversary. Therefore, we analyze the probability for an adversary to successfully spoof a transaction confirmation and determine n and δ for bounding this probability. We then analyze costs associated with transaction confirmation spoofing attacks in order to identify transaction limits which would render these attacks unprofitable.

Preliminaries Let parameters r_{net} and $r_{\mathcal{A}}$ be the total hashrate of the network and the hashrate of the adversary $\mathcal{A}$ in hashes per second (H/s), respectively. We assume that $r_{\mathcal{A}} < r_{net}$. The target value for block generation is denoted by tgt, the number of blocks required to confirm a transaction by n and the upper bound for generation by δ in seconds. For the following considerations we suppose tgt is the correct current target value, which can be derived from the blockchain. Consequently, since tgt is chosen such that block generation takes 10 minutes (or 600 seconds) in average, it follows that

$$r_{net} = \frac{2^{256}}{tgt} \cdot \frac{1}{600}. \tag{5.1}$$

Further, n-T $= \{B_i, ..., B_{i+n}\}$ is a n-transaction confirma-

tion, δ_n is the time it took to generate n-T and let $\alpha \in [0,1]$ be an arbitrary, but fixed parameter, which we call security level and which is denoting the upper bound for the attack success probability of $\mathcal{A}$.

5.1 Analysis of Attack Probability for Transaction Confirmation Spoofing

To produce a valid transaction confirmation within defined time constraints, the adversary $\mathcal{A}$ needs to compute the required n confirmation blocks for this transaction. Without tampering with any of the relevant inputs to the block generation procedure she is expected to take much longer than the (honest) Bitcoin network and therefore $n \cdot \delta$ can be used as an upper bound to decide whether a sequence of n blocks comes from the network or from $\mathcal{A}$.

Eventually, we want to control the probability that an adversary can find n confirmation blocks in this time. Therefore, we will first analyze the probability of finding one block, then of finding n blocks and finally show that these probabilities can differ significantly for the network and the adversary for different values of δ. From this we derive how to determine δ and n such that $\mathcal{A}$'s probability to find n-T within $\delta_n \leq n \cdot \delta$ is less than a required security level α, which is used to bound this probability.

Let X_i be a geometrical distributed random variable with parameter p, denoting the number of hashes that have to be computed until a valid hash for block B_i is found, and p_{suc} being the success probability of finding a valid hash. Recall that in order to find a valid block, a nonce has to be discovered such that the hash of the block header is less or equals the 256 bit long target value tgt. Consequently, since Bitcoin uses SHA-256 there are exactly tgt out of 2^{256} possible valid hashes. Hence, the probability that any computed hash is valid is

$$p_{suc} = \frac{tgt}{2^{256}}. \tag{5.2}$$

The relationship between the number of hashes needed for one block h_1 and the time δ in which hashes are computed at rate r is as follows:

$$h_1 = r \cdot \delta. \tag{5.3}$$

Hence, the probability of finding a valid hash in less then h_1 attempts (i.e., within δ) is given by the cumulative distribution function (c.d.f.) of the geometrical distributed X_i:

$$\Pr(X_i \leq h_1) = 1 - (1 - p_{suc})^{h_1} \tag{5.4}$$

Next, we extend it to the case of finding several confirmations in time as needed for n-confirmation transactions. Let Y_i be the random variable denoting the total number of hashes that have to be computed until n valid confirmations have been found. It follows that $Y_i = \sum_{j=i}^{i+n} X_j$ is the sum of n i.i.d. variables following the geometrical distribution with parameter $p = p_{suc}$. Hence Y_i follows the negative binomial distribution $NB(r,p)$ with parameter $r = n$ being the number of successes until the experiment is stopped and $p = p_{suc}$ being the single success probability of one trial.

The number of hashes that can now be computed within $n \cdot \delta$ and with hashrate r is

$$h_n = r \cdot n \cdot \delta. \tag{5.5}$$

Therefore, the probability of finding n valid hashes within these h_n attempts (i.e. the probability that it took less time than $n \cdot \delta$ to find them) is given by the c.d.f. of the negative binomial distribution which can be calculated in the following ways:

$$\begin{aligned} \Pr(Y_i \leq h_n) &= 1 - I_{1-p}(h_n + 1, n) && (5.6)\\ &= I_p(n, h_n + 1) && (5.7)\\ &= \frac{\int_0^p t^{n-1}(1-t)^{h_n}\mathrm{d}t}{\int_0^1 t^{n-1}(1-t)^{h_n}\mathrm{d}t} && (5.8) \end{aligned}$$

where $I_x(a,b)$ is the regularized incomplete beta function.

When substituting h_n according to equation (5.5), the above probability depends on the hashrate r, the number of required confirmations n and the allowed time δ for finding these confirmations. Since the hashrate is a fixed parameter either given by the current target value tgt for r_{net} or by assumption for $r_{\mathcal{A}}$, the only remaining influential parameters are δ and n. For better readability we define

$$\mathrm{P}(r; n, \delta) := \Pr(Y_i \leq r \cdot n \cdot \delta)$$

Hence, the following equation should be satisfied:

$$\mathrm{P}(r_{\mathcal{A}}; n, \delta) < \alpha \tag{5.9}$$

Note that for a given pair of (n, δ) that satisfies Eq. 5.9 there is always a probability $\beta = \mathrm{P}(r_{net}; n, \delta)$ that the Bitcoin network will not be able to produce the correct transaction confirmation in the given time limits. Hence, it is also important to choose δ in such a way that for increasing n the probability $\mathrm{P}(r_{net}; n, \delta)$ increases but $\mathrm{P}(r_{\mathcal{A}}; n, \delta)$ decreases.

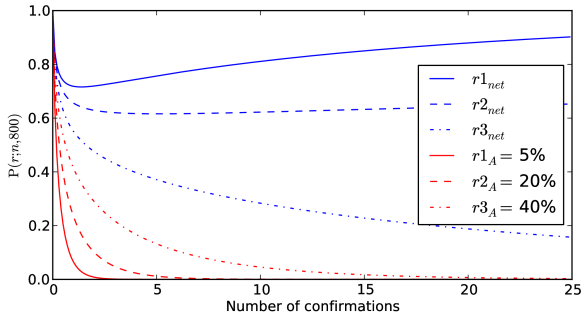
Furthermore, for best usability n should be as small as possible in order to reduce overhead for the wallet, while δ should be as large as possible to reduce the probability that $\delta_n < n \cdot \delta$. We model this as follows:

$$\begin{aligned} &\mathsf{max}\ \frac{1}{n} \cdot \mathrm{P}(r_{net}; n, \delta) - \mathrm{P}(r_{\mathcal{A}}; n, \delta) && (5.10)\\ &\mathsf{s.t.}\\ &\lim_{n\to\infty} \frac{\int_0^p t^{n-1}(1-t)^{r_{net}\cdot n\cdot\delta}\mathrm{d}t}{\int_0^1 t^{n-1}(1-t)^{r_{net}\cdot n\cdot\delta}\mathrm{d}t} = 1\\ &\lim_{n\to\infty} \frac{\int_0^p t^{n-1}(1-t)^{r_{\mathcal{A}}\cdot n\cdot\delta}\mathrm{d}t}{\int_0^1 t^{n-1}(1-t)^{r_{\mathcal{A}}\cdot n\cdot\delta}\mathrm{d}t} = 0\\ &\mathrm{P}(r_{\mathcal{A}}; n, \delta) \leq \alpha \end{aligned}$$

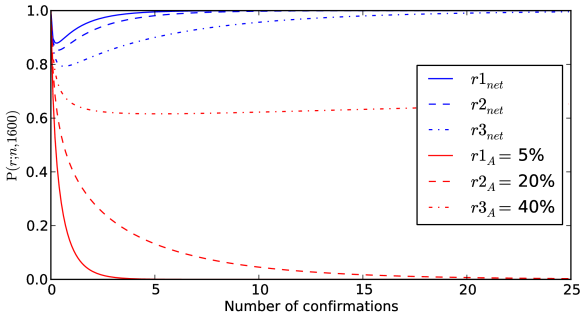
The resulting pair (n, δ) guarantees that any valid n-T will be accepted and $\mathcal{A}$'s probability of success is bound by α.

Parameter examples We calculated several exemplary values for security parameters for the attack probability $\alpha = 0.1\%$. Particularly, Table 1 presents examples for parameters n and δ for different attackers' hashrates. The solutions have been numerically approximated according to Eq. 5.10. For the network values we refer to block 287500 of the blockchain. Here the target value is $1.5E+58$, which means a hashrate r_{net} of approximately 22 petahashes per second (cf. Eq. (5.1)).

Fig. 5 shows how $\mathcal{A}$'s probability of success differs from the networks' probability of success and how it diminishes as more confirmations are required. Note especially that a



(a) Probability distributions for δ =800



(b) Probability distribtutions for δ =1600

Figure 5: Comparison of $P(r_{net}; n, \delta)$ and $P(r_{\mathcal{A}}; n, \delta)$ pairs for different adversary models ($\mathcal{A}$'s hashrate is given as percentage of the network hashrate). The *red/lighter* (lower) graphs denote $\mathcal{A}$'s probability and the *blue/darker* ones the corresponding probability of the network

$r_{\mathcal{A}}$	5%	10%	20%	30%	45%
δ	1400s	1100s	1000s	1000s	1200s
n	5	6	12	16	> 600

Table 1: Solutions for the confirmation generation time limit (δ) for different assumptions on the hashrate of the adversary ($r_{\mathcal{A}}$) and the number of confirmations (n) required in each case in order to satisfy a security level α of 0.1%.

very strong attacker (rA = 40%)[9] will still be capable to succeed for δ = 1600s, but not any longer for δ = 800s.

5.2 Cost Estimations of Transaction Confirmation Spoofing

In the following, we calculate the costs for $\mathcal{A}$ to compute n confirmations in order to determine the upper limit for the amount which can be safely accepted by payees to render timing attacks unprofitable. Our analysis considers costs incurred by electricity consumed during required computations and excludes costs for hardware.

Let $r_{\mathcal{A}}$ be $\mathcal{A}$'s hashrate in H/s, c her power costs in ct/kWh and w her power consumption in kWh. Then computing the expected number of hashes per block ($h_E = E(X_i)$) will take her t_E seconds in average and result in c_E costs for electricity for one block in average.

$$h_E = E(X_i) = \frac{2^{256}}{tgt} \tag{5.11}$$

$$t_E = \frac{h_E}{r_{\mathcal{A}}} \tag{5.12}$$

$$c_E = w \cdot c \cdot \frac{t_E}{3600} \tag{5.13}$$

Hence, $\mathcal{A}$ would need to load bitcoins at least worth c_E into her W. Note that $\mathcal{A}$'s costs do not depend on her fraction of the network hashrate, but only on the total network hashrate. Increasing $r_{\mathcal{A}}$ decreases the time in which $\mathcal{A}$ can fake blocks, but at the same time increases her power consumption to the same degree.

Note that this restriction can not be applied immediately, since the possible profit is in BTC and the costs are in common fiat currency such as euros or dollars. Therefore, the actual limit that needs to be applied depends on the exchange rate between BTC and the reference currency, e.g., USD.

$$c_{max} = c_E \cdot \frac{BTC}{USD} \tag{5.14}$$

Restricting the amount that a trusted wallet can be charged with one transaction to c_{max} renders timing attacks unprofitable, given $\mathcal{A}$'s $r_{\mathcal{A}}$, c and w.

Exemplary values For exemplary purposes, consider a single adversary $\mathcal{A}$ that acquired a small number of recent mining hardware worth of $60,000. Suppose $\mathcal{A}$ controls 10 TerraMiner IV, which is one of the most efficient hardware as of today. With 2000 GH/s per rig $\mathcal{A}$'s hashrate $r_{\mathcal{A}}$ is 20000 GH/s, constituting about 0.09% of the total network hashrate. With this hashrate she needs t_E = 187 hours in average for one block. Assuming c = 8 ct/kWh, an estimate of electricity cost in China [3] (where large Bitcoin miners are active), then for w = 1.2 kWh per rig and an exchange rate of 600 USD/BTC this costs her c_E = $300 for one block. Suppose the offline wallet requires at least 6 confirmations for a pre-loading transaction, then $\mathcal{A}$ is expected to take over 46 days for computation and faces power costs of $1800.

6 Implementation

Our implementation consists of three components: (i) Bitcoin miner, (ii) Bitcoin client of the payee and (iii) offline wallet of the payer.

Miners Changes required by miners were integrated into `bitcoind` Bitcoin client. Modifications to the basic Bitcoin protocol are minimal – we added functionality for miners to verify the proof that the accused wallet indeed signed the double-spending transaction before the revocation transaction is further processed.

Payee's client To implement functionality of a payee, we extended the Android Bitcoin wallet [1]. Changes concern offline payment and revocation phases, while redemption of bitcoins is performed in the same way as in the standard Bitcoin scheme. Further, we enabled a client to listen to transactions sent to the revocation address.

Payer's Offline Wallet The offline wallet was prototyped using a cgCard [25], which runs JCOP 2.4.1 R3, supports

[9]Remarkably, it was shown [52] that such a strong adversary has a non-negligible probability to succeed in double spending even in online Bitcoin payments.

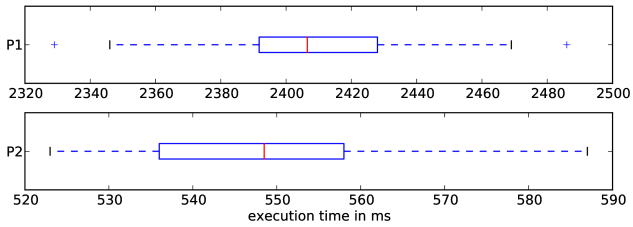


Figure 6: Timing results of 50 independent runs of the coin preloading protocol (P1) and the offline transaction protocol (P2). Confirmation size was $n = 7$.

JavaCard API of version 2.2.2 and the global platform standard in version 2.2.1. It has an NXP P5CD081 Chip based on 8-bit CPU and about 81 kbyte of EEPROM. The Java Card technology is an extremely stripped down Java derivate designed to run on small memory footprint devices, such as smart cards. While Java Card is a precise subset of Java, it offers none of the common conveniences and approaches of conventional Java programming. Yet we were able to realize full support of our protocol. In particular we are able to perform parsing and validation of transactions, blockheaders and Merkle trees as well as key and transaction generation.

The memory footprint is about 2 kB for transient memory (RAM) and 3 kB for persistent data excluding transactions, keys and the applet bytecode itself.

Fig. 6 shows the performance of the card for coin preloading and offline payment protocols. Measurements include computations on the card and communication with the host. The longest time ($\approx$ 1100 ms in average) of the coin preloading protocol was required for transaction confirmation verification, while verification of a single blockheader took about 160 ms in average, which we consider as a good result[10]. With a median of 548 ms the generation of offline transactions should not notably interrupt any payment process.

7 Discussion

So far we described in details how to instantiate an offline wallet by leveraging secure hardware. However, as we already mentioned in Section 4.1, our solution can be instantiated using an alternative approach which neither relies on secure hardware nor on their manufacturers for certification.

Our primary motivation for leveraging secure hardware was to impose financial loss to misbehaving users, so that an adversary would need to invest significant resources in order to compromise the wallet environment, but could misuse it only for a limited period of time, until the wallet is detected and revoked. This is likely to make wallet compromise attacks unprofitable, especially given significant costs of attacks against secure hardware and limited transaction amounts.

An alternative way to achieve similar objective is to utilize a deposit system. Such a deposit system can be managed by the Bitcoin network itself and without relying on any external parties. In particular, it could be instantiated using *decentralized anonymous credentials* system of Garman et al. [37] which enables a Bitcoin-based distributed certification authority which can issue certificates to our offline wallets. Such a certification authority can be extended to issue certificates only to wallets that can prove they have sent certain amount of cryptocurrency to a special deposit account.

Note that the naïve approach to return deposits to well-behaving wallets is to establish a trusted party which would have control over the signing key of the deposit account. In a more elegant solution, however, one could rely on a deposit account of special type for which there is no signing key exists. When currency is transferred to such an account, it gets permanently destroyed. To enable withdrawal of deposits, we suggest to introduce a transaction of special type, which has no source address, but only destination. Similar transactions already exist in Bitcoin network – *coinbase* transactions which are used to reward miners for mining blocks. In this way, the destroyed currency can be returned to the system and be transferred back to users of well-behaving wallets[11].

Remarkably, solution which we described in details in this paper relies on wallet revocation to merely punish double-spenders, while coin forgery attacks are prevented by other means (cf. Section 3.1 for details). This is due to the fact that coin forgery attacks could be launched against non-compromised wallets. Indeed, an attacker controlling user's platform, but not the offline wallet, could spoof transaction confirmation of the pre-loading transaction (cf. Section 4.2). Hence, the successful attacker would not bear the costs of wallet compromise, but only costs of transaction confirmation spoofing (cf. Section 5.2). In case of deposit-based wallet deployment, however, it might be reasonable to use wallet revocation to defeat both attack classes, which would simplify transaction verification in offline phase.

We would like to investigate deposit-based approach in our future work, and in particular to model important system parameters, such as size of the deposit, transaction limits, and the amount of time the payee stays offline.

Bitcoin limitations Recent research [28] has shown that Bitcoin faces scalability problems, which limit transaction throughput by a few transactions per second, hence, in a long perspective Bitcoin might not be the best choice for low amount payments. This, however, does not undermine our results: While we have chosen Bitcoin due to its widespread adoption, our solution can also be applied to other blockchain-based cryptocurrency systems, including systems with larger transaction throughput (e.g., Ethereum [24] which has block generation time of 10 seconds).

8 Related Work

Bitcoin was originally presented in [49] and since then was thoroughly analyzed with regards to security, privacy and beyond. A technical survey on decentralized digital currencies, including Bitcoin, is available in [54], and systematization if knowledge in this domain is provided by [22].

Hashrate-based double-spending attacks initially discussed in [49] were analyzed in more detail in [52]. The first comprehensive study on Bitcoin was presented in [20], which considered such aspects as deflation due to capped amount of bitcoins, forking the chain, malware attacks and scalability. Further, the Bitcoin protocol was modeled in [48], and it was shown that it reaches a Byzantine consensus. The core

[10]For comparison, Gura et al. [40] reported 0.81 sec for a single 160-bit ECC multiplication on 8-bit CPU.

[11]Withdrawal transactions should reference corresponding deposit transactions to prevent illegitimate withdrawals.

of the Bitcoin protocol was analyzed in [36] and was proven secure if the network is tightly synchronized and the adversary's hashing power is strictly less than 50%, but shown that the adversarial bound decreases as the synchronization gets looser.

Essential aspects of network synchronization and information propagation were studied in [31, 33]. Eclipse attacks on Bitcoin network were shown [41], which allow attackers to perform n-confirmation double spending attacks, selfish mining, and facilitate adversarial forks in the blockchain [41].

Closest to our interest are works [42, 17] which analyzed double-spending attacks in the context of fast payments and suggested countermeasures. However, they rely on online detection of double-spending transactions appearing in the Bitcoin network, and, hence, not applicable in offline payment scenarios. Further, OtherCoin [34], TREZOR [50] and BlueWallet [18] similar to us use secure hardware to protect Bitcion signing keys. However, their main purpose is to protect private keys from malware or other attempts to extract them, while we aim to limit access of bitcoin owners to corresponding signing keys to prevent double-spending.

CoinBlesk [11] is a mobile Bitcoin payment solution suitable for fast payments. Similarly to our solution, transactions in CoinBlesk are performed directly from the payer to the payee (over NFC). However, differently from us, at least one party requires online connection during transaction, and, hence, the solution does not solve the problem which we aim to address. Moreover, an idea of duplex micropayment channels initially introduced in the context of Bitcoin contracts [4] was adapted in [32] for off-blockchain payments with a goal to minimize amount of transactions to be confirmed by the network. The resulting solution can be used for offline payments, however, only for cases when the payer and payee know each other a priory the payment. Finally, Green Addresses solution [8] solves "confirmation delay" problem by introducing a trusted third party, which facilitates assured, zero-confirmation transactions. Generally, Green Addresses provide an evidence that trusted third parties can be accepted in Bitcoin ecosystem, but does not solve a problem of offline payments.

9 Conclusion

In this work we aimed to tackle a problem of secure payments with Bitcoins in scenarios where parties have no online connection during the payment, or the connection is available, but purchased items are released immediately after the payment, and before the transaction has been confirmed by the Bitcoin network. Our solution relies on an offline wallet residing on the platform of the payer. Such a wallet can be instantiated differently, e.g., using secure hardware or by means of utilizing deposit system. In this paper, we investigated in details the first approach. In particular, we proposed new security mechanisms as building blocks, provided their rigorous analysis and showed how they can be integrated into payment processes. As a proof of concept, we prototyped our solution for Android clients and used a JavaCard to host an offline wallet. Our implementation shows that smooth integration over a very prevalent platform (Android) is possible, and that offline and online payments can practically co-exist. We further discussed second deployment approach and outlined how the offline wallet could be instantiated using a wallet deposit system, which is managed by the Bitcoin network itself and does not rely on any external third parties.

10 References

[1] Android Bitcoin wallet. https://play.google.com/store/apps/details?id=de.schildbach.wallet.

[2] Apple Pay payment solution. http://www.apple.com/apple-pay/.

[3] Average electricity prices around the world: USD/kWh. https://www.ovoenergy.com/guides/energy-guides/average-electricity-prices-kwh.html.

[4] Bitcoin contracts. https://en.bitcoin.it/.

[5] Blockchain taking years to download? http://www.reddit.com/r/Bitcoin/comments/1pssvp/blockchain_taking_years_to_download/.

[6] DOGE. http://dogecoin.com.

[7] Google Wallet: Shop. Save. Pay. With your phone.

[8] Greenaddress. https://greenaddress.it/en/.

[9] LTC. http://litecoin.org.

[10] The NIST authenticated NTP service. http://www.nist.gov/pml/div688/grp40/auth-ntp.cfm.

[11] Coinblesk, a mobile bitcoin payment solution, 2014. https://github.com/coinblesk.

[12] Edgar Dunn & Company. Advanced payments report, 2014. http://www.paymentscardsandmobile.com/wp-content/uploads/2014/02/PCM_EDC_Advanced_Payments_Report_2014_MWC.pdf.

[13] 10 best payment processors for Bitcoin for merchants, 2016. https://toughnickel.com/personal-finance/Best-Payment-Processors-for-Bitcoin-Bitcoin-for-Merchants.

[14] T. Alves and D. Felton. TrustZone: Integrated hardware and software security. *Information Quaterly*, 3(4), 2004.

[15] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in Bitcoin. In *Financial Cryptography and Data Security*, 2013.

[16] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and red balloons. In *ACM Conference on Electronic Commerce*, 2012.

[17] T. Bamert, C.Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with Bitcoins. In *13-th IEEE International Conference on Peer-to-Peer Computing*, 2013.

[18] T. Bamert, C. Decker, R. Wattenhofer, and S. Welten. BlueWallet: the secure Bitcoin wallet. In *International Workshop on Security and Trust Management*, 2014.

[19] L. Bångens and B. Söderberg. Mobile banking – financial services for the unbanked? 2008. http://spidercenter.org/polopoly_fs/1.146036.1378747792!/menu/standard/file/Mobile%20banking%20-%20financial%20services%20for%20the%20unbanked.pdf.

[20] S. Barber, X. Boyen, E. Shi, and E. Uzun. Bitter to better – how to make Bitcoin a better currency. In *Financial Cryptography and Data Security*, 2012.

[21] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Practical decentralized anonymous e-cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, May 2014.

[22] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A.

Kroll, and E. W. Felten. Research perspectives and challenges for Bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.

[23] J.-P. Buntinx. 27 million more merchants can now accept Bitcoin, 2016. https://news.bitcoin.com/bitpay-enables-27-million-ingenico-retailers-accept-bitcoin/.

[24] V. Buterin. A next-generation smart contract and decentralized application platform, 2014. https://github.com/ethereum/wiki/wiki/White-Paper.

[25] Certgate. Certgate products. cgCard. Texas Instruments White Paper, 2012. http://www.certgate.com/wp-content/uploads/2012/09/20131113_cgCard_Datasheet_EN.pdf.

[26] D. Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, 1983.

[27] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology*, 1990.

[28] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer. On scaling decentralized blockchains (a position paper). In *Bitcoin Workshop*, 2016.

[29] A. Cuthbertson. Bitcoin now accepted by 100,000 merchants worldwide. http://www.ibtimes.co.uk/bitcoin-now-accepted-by-100000-merchants-worldwide-1486613.

[30] G. Danezis, C. Fournet, M. Kohlweiss, and B. Parno. PinocchioCoin: building Zerocoin from a succinct pairing-based proof system. In *PETShop*, 2013.

[31] C. Decker and R. Wattenhofer. Information propagation in the Bitcoin network. In *IEEE International Conference on Peer-to-Peer Computing*, 2013.

[32] C. Decker and R. Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, 2015.

[33] J. A. D. Donet, C. Perez-Sola, and J. Herrera-Joancomart. The Bitcoin P2P network. In *Financial Cryptography and Data Security*, 2014.

[34] R. Dragomirescu. OtherCoin. 2013. http://www.othercoin.com/OtherCoin.pdf.

[35] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, 2014.

[36] J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT*, 2015.

[37] C. Garman, M. Green, and I. Miers. Decentralized anonymous credentials. In *Network and Distributed System Security Symposium*, 2014.

[38] A. Gervais, G. Karame, S. Capkun, and V. Capkun. Is bitcoin a decentralized currency? In *IEEE Security and Privacy*, 2014.

[39] Giesecke & Devrient Secure Flash Solutions. The Mobile Security Card SE 1.0 offers increased security. http://www.gd-sfs.com/the-mobile-security-card/mobile-security-card-se-1-0/.

[40] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Workshop on Cryptographic Hardware and Embedded Systems (CHESS)*, 2004.

[41] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg. Eclipse attacks on Bitcoin's peer-to-peer network. In *USENIX Security Symposium*, 2015.

[42] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending attacks on fast payments in Bitcoin. *ACM Conference on Computer and Communications Security*, 2012.

[43] S. King. Primecoin: Cryptocurrency with prime number proof-of-work, 2013. http://academictorrents.com/details/d0f9accaec8ac9d538fdf9d675105ae1392ea32b.

[44] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of Bitcoin mining or, Bitcoin in the presence of adversaries. *Workshop on the Economics of Information Security*, 2013.

[45] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In *International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

[46] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of Bitcoins: characterizing payments among men with no names. In *Conference on Internet Measurement Conference*, 2013.

[47] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, 2013.

[48] A. Miller and J. LaViola. Anonymous Byzantine consensus from moderately-hard puzzles: A model for Bitcoin. Technical Report CS-TR-14-01, University of Central Florida, April 2014.

[49] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical Report, 2008. http://www.vsewiki.cz/images/archive/8/89/20110124151146!Bitcoin.pdf.

[50] M. Palatinus and P. Rusnak. Trezor, 2013. www.bitcointrezor.com.

[51] D. Ron and A. Shamir. Quantitative analysis of the full Bitcoin transaction graph. *Financial Cryptography and Data Security*, 2012.

[52] M. Rosenfel. Analysis of hashrate-based double-spending. In *ArXiv Preprint: 1402.2009v1*, 2012. http://arxiv.org/abs/1402.2009.

[53] S. Skorobogatov. Chapter 7: Physical attacks and tamper resistance. In *Introduction to Hardware Security and Trust*. Springer New York, 2012.

[54] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. In *Cryptology ePrintArchive,Report2015/464*, 2015.

[55] N. van Saberhagen. Cryptonote v 2.0, 2013. https://cryptonote.org/whitepaper.pdf.