Short Paper: Secure Offline Payments in Bitcoin

Taisei Takahashi and Akira Otsuka

Institute of Information Security
Yokohama, Japan
{mgs174503,otsuka}@iisec.ac.jp

Abstract. Double-spending attacks on fast payments are one of the fatal architectural problems in Cryptocurrencies. Dmitrienko et al. proposed an offline fast payment scheme that relies on tamper-proof wallets produced by trustworthy manufacturers. With the wallets, the payee can immediately trust the transactions generated by the wallets without waiting for their registration to the blockchain. Secure coin-preloading to the wallet is important, while illegal coin-preloading can cause over/double-spending by the trusted wallets. For this, they proposed an interesting protocol that makes use of a fragment of the main blockchain to prove to the wallets the legitimacy of preloaded coins. One drawback is that, in proving that the fragment are from honest miners, their protocol requires a trusted online time-stamp server so that the wallets can verify the timestamps to see if the blocks in the fragment is mined with sufficiently large amount of computing resources. Otherwise, it sacrifices usability. In order to eliminate such an online trustee, in this paper we took the opposite approach that the payee (not the wallets) verifies the legitimacy of preloaded coins at the time of offline payment. As a consequence, our result shows that, with light-weight tamper-proof wallets, completely decentralized offline payment is possible without any modification to the existing Bitcoin network.

Keywords: blockchain · offline payment · tamper-proof wallet.

1 Introduction

Double-spending attacks on fast payments [4] are the one of the fatal architectural problems in Cryptocurrencies. Double spending refers to the payment where the same coin is spent twice in a way that the receiving party cannot notice the invalidity of the payment. We study an offline immediate payment scheme based on blockchain secure against the double-spending attacks on fast payment assuming the security of tamper-proof wallet. Dmitrienko et al. [1] pointed out that Bitcoin requires clients to be online to perform transactions and a certain amount of time to verify them, and also offline payments raise non-trivial challenges, as the payee has no means to verify transactions. Even online, fast payments are shown to be vulnerable to double-spending attacks [4]. Offline immediate payments is long demanding in cryptocurrencies. In practice, without changing the ongoing systems, "fast payment" is widely used such that the payee could accept the transaction immediately by checking the signature, and the payer's balance for confirming that the payer has enough money to spend. Karame et al. [4] pointed out that, for such a fast payment scheme, double-spending attack is possible if a malicious payer makes use of the race condition of the double-spending transactions that reach the payees with different timing in the peer-to-peer network. Dmitrienko proposed a solution for immediate payment and offline payments that relies on tamper-proof wallets manufactured by trustworthy manufacturer and deploys time-based transaction confirmation verification mechanism. They make use of a fragment of the main blockchain to prove the legitimacy of preloaded coins to the wallets. Their scheme solved the double-spending problem as Karame et al. suggested. The wallet loses their credibility if a malicious user succeeds to pre-load illegal coins by intention. Thus, it is enormously important to establish secure coin preloading to the wallet. In order to achieve this, Dmitrienko propose an interesting protocol that proves the fact that a pre-loading payment to the wallet existed using a subchain, a fragment of the main blockchain. They considered that a subchain is a part of the mainchain if and only if the total PoW to mine the blocks in the subchain is greater than some predetermined lower-bound and the average time to produce the subchain is less than some constant time period. Thus, to verify the proof inside the tamper-proof wallet, it is necessary to measure the total computation time consumed to generate the subchain objectively. (A) A trivial solution to this is to assume a trusted time-stamp server, which supplies a time-stamp to a block everytime the corresponding block is created. (B) Another solution proposed in [1] is to set the expiration time of the pre-loading coins. This also convinces the wallet that the subchain is produced within some bounded time period. Apparently, the construction (A) with the trusted time-stamp server requires the additional trusted third party. The construction (B) is decentralized, but it sacrifices the usability to the great extent.

Other studies of interest include Teechan, an offline payment channel, proposed by Lind et al. [5]. It assumes tamper-proof wallets in both side and achieves offline payments. However, it has to deposit some fund, which corresponds to our preloading, into the 2-of-2 multisig address between the payee and the payer sufficiently before the offline payment occurs. Thus, the setting differs from ours. In the theoretical aspect, Juan Garay et al. [3] formulated the basic properties of the blockchain in Bitcoin [6] such as "common prefix" and "chain quality," assuming that the hashing power of an adversary controlling a fraction of the parties is strictly less than 1/2. Further extensions are made for variable difficulty [2], a security analysis in the "semi-synchronous" network model [7].

We propose a novel offline payment scheme alternative to the Dmitrienko's protocol. The advantage of our scheme is its fully decentralization, that is, it does not assume any external trusted time-stamp server. Further, we do not need to set any expiration time to the pre-loaded coins.

2 Preliminaries

- $\mathbb{C}$: a blockchain.

- B_i: i-th block in $\mathbb{C}$ is a triple $\langle \mathsf{hash}(B_{i-1}), \boldsymbol{x}_i, nonce\rangle$.
- τ: a transaction of a form $\mathsf{Sign}(sk_A; A \to B, value)$.
- $\boldsymbol{x}_i$: a root of Merkle Tree for a set of transactions $\{\tau_1, \tau_2, \ldots\}$ in B_i.
- $\mathcal{U}$: a set of users.
- $\mathcal{H}$: a set of honest users, $\mathcal{H} \subseteq \mathcal{U}$.
- $X, Y \in \mathcal{U}$: (typically, X as a payer, Y as a payee).

To analyze the security of our scheme, we follow the notions introduced by Garay et al. [3]. All players are bounded interactive turing machines and are all synchronized and message are exchanged in a discrete time frame called round. $\{EXEC^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)\}_{z\in\{0,1\}^*}$ denotes the random variable ensemble that determines the output of the environment $\mathcal{Z}$ on input z for a protocol Π with adversary $\mathcal{A}$. $VIEW^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)$ denotes the concatenated view of all parties after the completion of an execution $EXEC^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)$. A "flat" model is assumed where all parties executes exactly q mining trials (hash queries) per round. Then, each mining trials by various players are modeled as a Bernoulli distribution with different parameters and the deviation from the expected probability is estimated. We denoted by κ the length of hash function output, by η parameter determining block to round translation and by f probability at least one honest party succeeds in finding a POW in a round. Garay et al. [3] showed that any consecutive rounds S of length $|S| > \eta\kappa$ in the blockchain protocol is a "typical execution" with probability 1− $e^{-\Omega(\kappa)}$ where honest and adversarial mining trials succeed as expected within a bounded probability fluctuation of at most ϵ. Then, we have the followings.

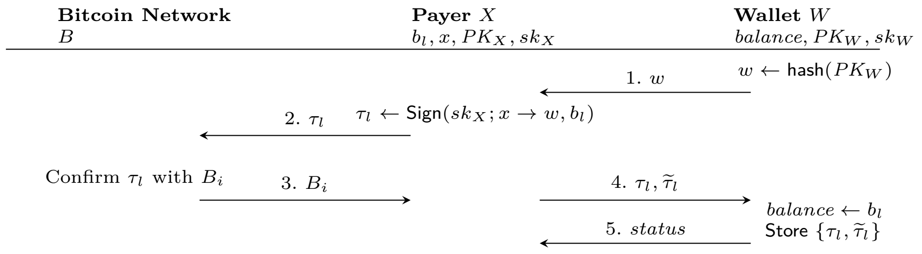
Lemma 1. *(Common-Prefix Lemma [3]) Assume a typical execution and consider two chains* $\mathbb{C}_1$ *and* $\mathbb{C}_2$ *such that* $\mathrm{len}(\mathbb{C}_2) \geq \mathrm{len}(\mathbb{C}_1)$. *If* $\mathbb{C}_1$ *is adopted by an honest party at round* r, *and* $\mathbb{C}_2$ *is either adopted by an honest party or diffused at round* r, *then* $\mathbb{C}_1^{\lceil k} \leq \mathbb{C}_2$ *and* $\mathbb{C}_2^{\lceil k} \preceq \mathbb{C}_1$, *for* $k \geq 2\eta\kappa f$.

Theorem 1. *(Chain Growth Theorem [3]) Assume a typical execution, for any honest party* P *with chain* $\mathbb{C}$ *in* $VIEW^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}$, *it holds that for any* $s \geq \eta\kappa$ *rounds there are at least* $(1-\epsilon)f \cdot s$ *blocks added to the chain of* P.

3 Secure Offline Bitcoin Payments

Our immediate payment protocol construction basically follows the Dmitrienko's protocol [1] but without the existence of the time-stamp server. The main difference of the protocol is that the correctness of the coin-preloading to the tamper-proof wallet is verified by the payee in our construction, not the payer as in the Dmitrienko's protocol. Similarly to their scheme [1], our construction also assumes the tamper-resistant wallet to incorporate overspending prevention. Tamper-resistant wallet has a secret key sk_T which was created by the wallet manufacturer T. The wallet also has a variable *balance* (≥ 0). It increases in coin preloading phase and decreases in offline payment phase. Our construction has 3 phases: (i) online coin preloading, (ii) offline payment, and (iii) coin redemption and wallet revocation. The details are as follows.

Fig. 1. Coin Pre-loading protocol


Bitcoin Network
B
Payer X
b_l, x, PK_X, sk_X
Wallet W
$balance, PK_W, sk_W$
$w \leftarrow \mathsf{hash}(PK_W)$
1. w
2. τ_l
$\tau_l \leftarrow \mathsf{Sign}(sk_X; x \to w, b_l)$
Confirm τ_l with B_i
3. B_i
4. $\tau_l, \widetilde{\tau_l}$
$balance \leftarrow b_l$
5. $status$
Store $\{\tau_l, \widetilde{\tau_l}\}$

Coin Preloading In the coin-preloading phase, the payer X requests a new account w from the wallet (Step 1), then create the preloading transaction τ_l transferring b_l bitcoins from her account x to w and commit it to the networks (Step 2). As soon as τ_l is verified and integrated into the Bitcoin network in a block, say B_i, X takes B_i (Step 3), and provides τ_l and the witness of the membership proof $\widetilde{\tau_l}$[1] to W (Step 4). W sets its *balance* to b_l and stores τ_l, $\widetilde{\tau_l}$, and replies status (Step 5)[2]. For simplicity, we assume one-time coin preloading for every account w such that once an amount b_l is preloaded to w, the wallet W never accepts preloading transaction to w any more and only makes payments while $balance \geq 0$. It is not hard to extend it to multiple coin preloading.

Secure Offline Transaction In the offline transaction phase, the payee Y sends the public key PK_Y and the requested amount b_o to the payer X which are immediately forwarded to W (Step 1). W checks the balance and if $balance \geq b_o$, it decreases *balance* by b_o and generates a transaction $\tau_o = \mathsf{Sign}(sk_w; w \to y, b_o)$. Further, W generates a $proof = \mathsf{Sign}(sk_T; \tau_o, \tau_l, \widetilde{\tau_l})$ that shows τ_o was created within the tamper-proof wallet by signing with sk_T. The resulting τ_o, *proof* and $cert_T$, a trustworthy vendor certificate, are sent to Y (Step 3). Y accepts the transaction if τ_l is $\mathsf{confirmed}$ and τ_o is valid and issued by a tamper-proof wallet. More formally, Y accepts τ_o and $proof$ if and only if

$$\begin{cases} cert_T \text{ is trustworthy} \\ \mathsf{Verify}(PK_T; proof) = 1 \\ \tau_o \in (\mathcal{V}_{\mathbb{C}_Y} \cap \mathsf{Sign}(sk_W; w \to \cdot, b_o)) \\ \tau_l \in \left(\mathbb{C}_Y^{\lceil k} \cap \mathsf{Sign}(\cdot; \cdot \to w, b_l)\right) \end{cases}$$

If all checks succeed, Y stores τ_o, $proof$ and replies to W with status (Step4).

Coin Redemption and Wallet Revocation When Y gets online, Y proceeds to the coin redemption and wallet revocation phase. Y broadcasts τ_o to the Bit-

[1] $\widetilde{\tau_l}$ is a set of hash values such that $\mathsf{member}(B, \tau, \widetilde{\tau}) = 1$.

[2] The wallet does not check the validity of coin preloading transaction τ_l. Payments made from $\mathsf{unconfirmed}$ τ_l will be rejected by payees.

Fig. 2. Offline transaction protocol

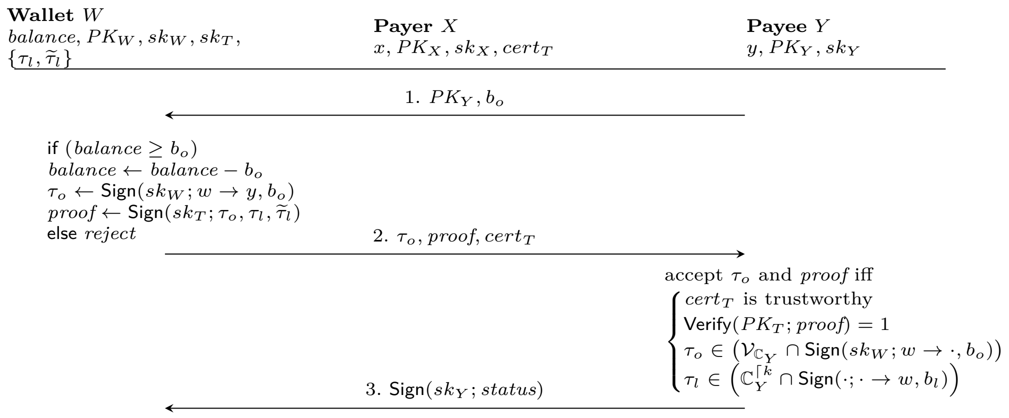
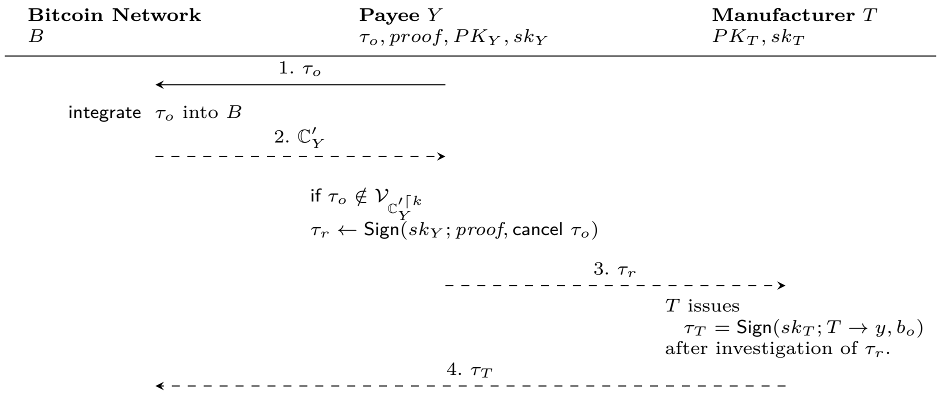

Wallet W
$balance, PK_W, sk_W, sk_T, \{\tau_l, \widetilde{\tau}_l\}$
Payer X
$x, PK_X, sk_X, cert_T$
Payee Y
y, PK_Y, sk_Y
1. PK_Y, b_o
if $(balance \geq b_o)$
$balance \leftarrow balance - b_o$
$\tau_o \leftarrow \mathsf{Sign}(sk_W; w \rightarrow y, b_o)$
$proof \leftarrow \mathsf{Sign}(sk_T; \tau_o, \tau_l, \widetilde{\tau}_l)$
else $reject$
2. $\tau_o, proof, cert_T$
accept τ_o and $proof$ iff
$cert_T$ is trustworthy
$\mathsf{Verify}(PK_T; proof) = 1$
$\tau_o \in \left(\mathcal{V}_{\mathbb{C}_Y} \cap \mathsf{Sign}(sk_W; w \rightarrow \cdot, b_o)\right)$
$\tau_l \in \left(\mathbb{C}_Y^{\lceil k} \cap \mathsf{Sign}(\cdot; \cdot \rightarrow w, b_l)\right)$
3. $\mathsf{Sign}(sk_Y; status)$

Fig. 3. Coin redemption and double-spending wallet revocation protocol


Bitcoin Network
B
Payee Y
$\tau_o, proof, PK_Y, sk_Y$
Manufacturer T
PK_T, sk_T
1. τ_o
integrate τ_o into B
2. $\mathbb{C}'_Y$
if $\tau_o \notin \mathcal{V}_{\mathbb{C}'^{\lceil k}_Y}$
$\tau_r \leftarrow \mathsf{Sign}(sk_Y; proof, \mathsf{cancel}\ \tau_o)$
3. τ_r
T issues
$\tau_T = \mathsf{Sign}(sk_T; T \rightarrow y, b_o)$
after investigation of τ_r.
4. τ_T

coin network in order to redeem the coins received from W (Step 1). Next, the Bitcoin network verifies τ_o and integrates it to the blockchain. The payee Y observes the Bitcoin network and periodically updates its local chain $\mathbb{C}'_Y$ reflecting the newly mined blocks (Step 2). Y waits until τ_o is confirmed or τ_o becomes invalid, $\tau_o \notin \mathcal{V}_{\mathbb{C}'_Y}$. If τ_o is invalid, Y initiates revocation by creating a revocation transaction $\tau_r = \mathsf{Sign}(sk_Y; proof, \mathsf{cancel}\ \tau_o)$ and send it to the Manufacturer T (Step 3). T investigate τ_r and in order to compensate Y for the damage of b_o, issues τ_T then committed to the Bitcoin network (Step 4).

4 Security Model

Definition 1. *τ is said to be* valid *with respect to a blockchain $\mathbb{C}$ if and only if τ satisfies*[3] *all the pre-agreed requirements*[4] *with respect to the blockchain $\mathbb{C}$. Further, we denote by $\mathcal{V}_{\mathbb{C}}$ a set of all possible* valid *transactions with respect to $\mathbb{C}$ such that*

$$\mathcal{V}_{\mathbb{C}} = \{\tau \mid \tau \textit{ is } \mathsf{valid} \textit{ with respect to } \mathbb{C}\}.$$

valid transactions are correctly formed and not over-spent more than the balance of the payer's account as far as regarding the given blockchain $\mathbb{C}$ as the mainchain. Once valid transactions are broadcasted to Bitcoin network by peers, they may be integrated into the mainchain if the given chain $\mathbb{C}$ is shared by honest miners, and thus the transaction is valid for those honest miners. Note that this is not guaranteed. Because the given chain $\mathbb{C}$ may contradicts with the localchains of other honest miners. Furthermore, the set of valid transactions will change as chains evolve.

Definition 2. *A transaction τ is said to be* confirmed[5] *if and only if for every honest player $X \in \mathcal{H}$ the transaction τ can be found on his localchain $\mathbb{C}_X$, that is, $\tau \in \mathbb{C}_X$ for all $X \in \mathcal{H}$.*

Once a transaction τ is confirmed, the transaction must be in the mainchain $\mathbb{C}$ where $\mathbb{C}$ is the longest prefix of all localchains held by honest players such that $\mathbb{C} \preceq \mathbb{C}_X$ for all $X \in \mathcal{H}$. where $\preceq$ is a prefix relation [3]. We consider confirmed transactions are valid. That is, τ is $\mathsf{confirmed} \implies \tau \in \mathcal{V}_{\mathbb{C}_X}$ for all $X \in \mathcal{H}$.

Definition 3. *Given a security parameter k, τ_o is said to be* verified *by a proof of a form $\mathsf{Sign}(sk_T; \tau_o, \tau_l)$ with respect to a localchain $\mathbb{C}$ if and only if*

$$\begin{cases} cert_T \textit{ is issued by a trustworthy provider} \\ \mathsf{Verify}(PK_T; proof) = 1 \\ \tau_o \textit{ is a transaction of a form } \mathsf{Sign}(sk_W; w \to \cdot, b_o) \\ \tau_l \textit{ is a transaction of a form } \mathsf{Sign}(\cdot; \cdot \to w, b_l) \\ \tau_o \in \mathcal{V}_{\mathbb{C}} \\ \tau_l \in \mathbb{C}^{\lceil k} \end{cases} \tag{1}$$

[3] Dmitrienko [1] have introduced a slightly different term CheckSyntaxT for transaction validation. CheckSyntaxT refers to the syntactical conformance of transactions to the requirements in the blockchain. On the other hand, valid refers to all the requirements for integrating into the blockchain. Those are the syntactical conformance of transactions, correctness of payer's signature and further requirements such as "the payer's account must exist", "balance after the transaction must not be negative."

[4] In Bitcoin, the requirements (e.g., Bitcoin Improvement Proposals) are subject to change. New proposals will be incorporated in the requirements after agreed among majority of miners through voting process in the blockchain.

[5] In practice, a transaction is *confirmed* after the block that contains the transaction has at least 6 blocks built on top. This is the situation where our notion of confirmed is satisfied with high probability.

where $cert_T$ is a certificate issued to a public key PK_T which is generated within a tamper-proof wallet W at the production time. (PK_W, sk_W) is a key pair generated by W, and w is an account related to PK_W.

In the offline payment, a payee Y verifies the received transaction τ_o by a proof $\mathsf{Sign}(sk_T; \tau_o, \tau_l)$ with his localchain $\mathbb{C}_Y$. If all of the above conditions are satisfied, Y is convinced that the coin-preloading transaction to the payer's wallet τ_l is confirmed by all honest players with overwhelming probability in k. As far as the tamper-proof wallet honestly produces the payment transaction τ_o, Y can believe that τ_o is not an overspending transaction and will be confirmed later on. To see whether τ_l satisfies the last condition $\tau_l \in \mathbb{C}^{\lceil k}$, in a naive construction, the payee must keep the whole set of transactions previously registered in the blockchain $\mathbb{C}$. For efficiency purposes, we assume that all transactions in a block B are kept in a form of a Merkle Tree and B only keeps the root hash value of the tree. Let $\widetilde{\tau_l}$ be a witness of the membership proof, or a set of all sibling hash values in every branch in the path from the root to the leaf τ_l. Using the witness $\widetilde{\tau_l}$, the membership proof can be efficiently proved since there exists a function $\mathsf{member}(\cdot)$ such that

$$\mathsf{member}(B, \tau, \widetilde{\tau}) = \begin{cases} 1 & \text{if } \tau \in B \\ 0 & \text{otherwise} \end{cases}$$

We replace the *proof* with $\mathsf{Sign}(sk_T; \tau_o, \tau_l, \widetilde{\tau_l})$ where offline payee needs efficiency.

Theorem 2. *We assume there exists a tamper-proof wallet W. Given a security parameter k and an offline transaction τ_o accepted by a payee Y such that τ_o is* verified *by a proof $= \mathsf{Sign}(sk_w; \tau_o, \tau_l)$ with respect to the payee's localchain $\mathbb{C}_Y$, the probability that the transaction τ_o, later on, changes its state to* invalid *is negligible in k. That is, with $\mathbb{C}_Y^{\lceil k} \preceq \mathbb{C}'_Y$, we have*

$$\Pr[\tau_o \notin \mathcal{V}_{\mathbb{C}'_Y}^{\lceil k} \mid \tau_o \textit{ is } \mathsf{verified} \textit{ by proof with respect to } \mathbb{C}_Y] < \mathsf{negl}(k).$$

Proof. Given τ_o is verified by *proof* with respect to $\mathbb{C}_Y$, equation (1) holds. Under the tamper-proof assumption, the tamper-proof wallet W never overspends exceeding the preloaded balance b_l. Therefore, if the preloading transaction τ_l is confirmed, that is, $\tau_l \in \mathbb{C}_X$ for all $X \in \mathcal{H}$, then the offline transaction τ_o cannot become invalid with respect to $\mathbb{C}'_Y$.

Since τ_l is already in a localchain of Y, that is, $\tau_l \in \mathbb{C}_Y^{\lceil k}$, for τ_l not to be confirmed, there must exist an honest player Z with a localchain $\mathbb{C}_Z$ such that $\tau_l \notin \mathbb{C}_Z$. The common-prefix property states that all localchain $\mathbb{C}$ held by honest players must satisfy $\mathbb{C}^{\lceil k} \preceq \mathbb{C}_Z$ with negligible error probability in k. Substituting $\mathbb{C} \rightarrow \mathbb{C}_Y$, $\tau_l \in \mathbb{C}_Y^{\lceil k}$ contradicts with $\tau_l \notin \mathbb{C}_Z$. Thus, τ_l must be confirmed with arbitrarily high probability $1 - \mathsf{negl}(k)$ with the security parameter k. Hence the theorem. $\square$

In the case $\tau_o \notin \mathcal{V}_{\mathbb{C}'^{\lceil k}_Y}$ where the offline transaction τ_o is turned out to be invalid later on with respect to the payee's evolved localchain $\mathbb{C}'_Y(\succeq \mathbb{C}_Y^{\lceil k})$, this must be the case where the tamper-proof wallet assumption is broken and τ_o is found to be an overspent transaction. Even in this case, the preloading transaction τ_l must still be $\mathsf{confirmed}$ with $1 - \mathsf{negl}(k)$. Therefore, the $\mathit{proof} = \mathsf{Sign}(sk_w; \tau_o, \tau_l)$ still satisfies the following 5 of all 6 conditions in (1) for all honest player $X \in \mathcal{H}$ with $1 - \mathsf{negl}(k)$.

$$\begin{cases} \mathit{cert}_T \text{ is issued by a trustworthy provider} \\ \mathsf{Verify}(PK_T; \mathit{proof}) = 1 \\ \tau_o \text{ is a transaction of a form } \mathsf{Sign}(sk_W; w \to \cdot, b_o) \\ \tau_l \text{ is a transaction of a form } \mathsf{Sign}(\cdot; \cdot \to w, b_l) \\ \tau_l \in \mathbb{C}_X{}^{\lceil k} \end{cases}$$

This fact convinces all honest players. Given τ_o is $\mathsf{invalid}$ with respect to $\mathbb{C}'_Y$, that is, $\tau_o \notin \mathcal{V}_{\mathbb{C}'^{\lceil k}_Y}$, the redeeming transaction $\tau_r = \mathsf{Sign}(sk_Y; \mathit{proof}, \mathsf{cancel}\ \tau_o)$ becomes valid with respect to $\mathbb{C}_X \succeq \mathbb{C}'^{\lceil k}_Y$ for all $X \in \mathcal{H}$ with arbitrarily high probability $1 - \mathsf{negl}(k)$. The trustworthy provider of the tamper-proof wallet or an insurance company might compensate Y for the damage of b_o after τ_r is $\mathsf{confirmed}$.

5 Conclusion

In this paper, we have shown that, with light-weight tamper-proof wallets, completely decentralized offline payment is possible without any modification to the existing Bitcoin network. Our protocol requires the coin pre-loading transaction is confirmed and its block is delivered to every possible payee before the first offline payment is made. This should be the best possible for Bitcoin network.

Lastly, we thank anonymous reviewers for their valuable comments.

References

1. Dmitrienko, A., Noack, D., Yung, M.: Secure Wallet-Assisted Offline Bitcoin Payments with Double-Spender Revocation. AsiaCCS (2017)
2. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 10401, pp. 291–323. Springer (2017)
3. Garay, Juan A, Kiayias, Aggelos, Leonardos, Nikos: The Bitcoin Backbone Protocol - Analysis and Applications. EUROCRYPT **9057**(Chapter 10), 281–310 (2015)
4. Karame, G.O., Androulaki, E., Capkun, S.: Double-spending fast payments in bitcoin. In: CCS. pp. 906–917. ACM (2012)
5. Lind, J., Eyal, I., Pietzuch, P.R., Sirer, E.G.: Teechan: Payment channels using trusted execution environments. https://arxiv.org/abs/1612.07766 (2016), https://arxiv.org/pdf/1612.07766.pdf, accessed: 2017-03-09
6. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash systems (11 2008), https://bitcoin.org/bitcoin.pdf
7. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 10211, pp. 643–673 (2017)