Burning Zerocoins for Fun and for Profit

A Cryptographic Denial-of-Spending Attack on the Zerocoin Protocol

Tim Ruffing
Saarland University, Germany
tim.ruffing@mmci.uni-saarland.de

Sri Aravinda Thyagarajan
Viktoria Ronge
Dominique Schröder
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
name.separated.by.dots@fau.de

***Abstract*—Zerocoin (Miers et. al, IEEE S&P'13), designed as an extension to Bitcoin and similar cryptocurrencies, was the first anonymous cryptocurrency proposal which supports large anonymity sets. We identify a cryptographic *denial-of-spending* attack on the original Zerocoin protocol and a second Zerocoin protocol (Groth and Kohlweiss, EUROCRYPT'15), which enables a network attacker to destroy money of honest users. The attack leads to real-world vulnerabilities in multiple cryptocurrencies, which rely on implementations of the original Zerocoin protocol.[1]**

The existence of the attack does not contradict the formal security analyses of the two Zerocoin protocols but exposes the lack of an important missing property in the security model of Zerocoin. While the security definitions model that the attacker should not be able to create money out of thin air or steal money from honest users, it does not model that the attacker cannot destroy money of honest users. Fortunately, there are simple fixes for the security model and for both protocols.

I. Introduction

Traditional currency systems rely on a trusted central authority that keeps track of transactions and validates them in order to ensure that users do not spend more than what they can. With Bitcoin [21] we witnessed a new paradigm: a decentralized digital currency where transaction verification takes place in a distributed fashion on a public ledger, otherwise known as the blockchain. To enable public verifiability, sensitive information about the financial transactions are disclosed on the blockchain.

For example, each transaction in Bitcoin is authenticated with a digital signature issued by the sender. While signing each transaction is necessary to make sure that only the owner can spend the funds, it also puts the anonymity of users at risk. By simply looking at a transaction any observer can learn the involved parties' public keys along with the transactions. The observer can use transaction graph analysis techniques empowered with different heuristics to link transactions and ultimately break anonymity [19], [24], [11]. But anonymity is crucial for financial privacy of users individually and fungibility of a cryptocurrency globally.

The Zerocoin [20] protocol, proposed as an extension to Bitcoin, was among the first solutions to solve the linkability problem and provide anonymity in cryptocurrencies. This protocol as well as a second instantiation by Groth and Kohlweiss (EUROCRYPT'15) [14] use zero-knowledge proofs to achieve anonymity by hiding which coin in the blockchain is spent by a transaction.

At first glance, anonymity seems to conflict with the goal of preventing double-spending: If an observer cannot tell which zerocoin is spent, how can he tell whether two transactions spend the same zerocoin or not? The Zerocoin protocol solves this problem by essentially letting each zerocoin be a commitment to a so called *serial number* S. When a zerocoin is spent, the spender reveals the serial number S and proves in zero-knowledge that she knows a zerocoin with serial number S, without revealing which zerocoin. Verifiers of transactions can now prevent double-spending by simply rejecting transactions with serial numbers that have been used already. (And since the commitment scheme is binding, no malicious spender can spend the same zerocoin with two different serial numbers.)

The original Zerocoin protocol is used by multiple cryptocurrencies (Zcoin, PIXV, SmartCash, Zoin, and HexxCoin) with a combined market capitalization of about $66\,000$ BTC $\approx 660\,000\,000$ USD at the time of writing [2].

A. Contribution

Our contribution is a cryptographic *denial-of-spending attack* on both Zerocoin protocols. The attack enables a network attacker to burn a zerocoin of an honest user by spending a maliciously generated zerocoin with the *same (but supposedly unique)* serial number. Since any attempt by the honest user to spend her zerocoin will reveal the same, already used, serial number. Any such attempt will be falsely rejected as double-spend and the zerocoin of the honest user is unspendable.

We believe that attacks of this kind could potentially have large-scale consequences beyond burning money: A greedy attacker may not be interested in harming individuals but in creating suspicion and subsequent devaluation of the currency. The attacker can then profit by placing bets on a decline of the price of the currency.

The attack is interesting from a theoretical point of view, because the designs of both Zerocoin protocols are rigorous and follow the paradigm of provable security. While our attack does not contradict the formal security analysis of the two Zerocoin protocols, it exposes the lack of an important missing property in the security model of Zerocoin, reminding us that provable security is only as good as the security definitions.

[1] We stress that the findings in this paper apply to the Zerocoin protocol and *not* to the different Zerocash protocol as used in the cryptocurrency Zcash.

As countermeasures to the attack, we provide a definition of the missing security property *serial number unforgeability* and present a fix that applies to both Zerocoin protocols.

II. Overview on Zerocoin

Before we are ready to explain the attack, we review the basics of the Zerocoin protocol. We restrict our attention to what is necessary to understand the attack and refer the reader to Miers et al. [20] for details.

The Zerocoin extension acts as "cryptographic mixer" for an underlying base currency. The basic idea is to exchange one unit of the base currency for the ability to *mint* a zerocoin, which can later be *spent* anonymously, i.e., without revealing which minted zerocoin is spent.

There are four algorithms to make the scheme work: The trusted Setup algorithm takes a security parameter as input and outputs public parameters given as input to all other algorithms; this is necessary to model a trusted setup.

The Mint algorithm generates a *zerocoin* c, a *serial number* S and a *trapdoor* skc.[2] By putting the outputs of Mint in a *mint transaction* on the blockchain, a user can mint a zerocoin according to the consensus rules, which require the user to forgo a fixed denomination of the base currency in exchange for the zerocoin. (The user must sign the mint transaction under the public key associated with the forgone funds in the base currency to prove ownership of these funds.)

The Spend algorithm takes as input an (already minted) zerocoin c, a serial number S, a trapdoor skc for c, an anonymity set $\mathbf{C}$ of zerocoins, and an arbitrary transaction string R. It outputs a *spend proof* π, which authorizes the spending of zerocoin c with the effects specified in the transaction string R (according to the consensus rules of the currency) without revealing which zerocoin in the set $\mathbf{C}$ is actually spent.

The tuple $(\pi, S, R, \mathbf{C})$ is recorded in a *spend transaction* in the blockchain.[3] It can be verified using Verify, which outputs true iff π is a valid spend proof for S, R and $\mathbf{C}$.

No attacker can double-spend the same zerocoin, because he will be forced to reveal the same serial number S twice, and the second attempt can easily be detected as a double-spend.

Both known Zerocoin schemes [20], [14] realize the public part c of a zerocoin as a commitment $c = \mathsf{Com}(S, r)$ to a randomly chosen serial number S with randomness $r = skc$, which is the trapdoor necessary to spend c. Further details of the constructions are not relevant for our discussion.

III. The Attack

Since honestly generated serial numbers are drawn randomly from a sufficiently large space, the probability of having two honest zerocoins with the same serial numbers is negligible. However, the constructions do not prevent a maliciously generated zerocoin from having the same serial number as a honestly generated zerocoin! Observe that an attacker can choose the serial number freely when creating the commitment. This yields the following *denial-of-spending* attack:

1) An honest user $\mathcal{U}$ creates a spend transaction for her zerocoin c with serial number S, thereby revealing S in the transaction.
2) The attacker $\mathcal{A}$, listening on the honest user's network, obtains the transaction and blocks the propagation of the transaction into the network. The attacker $\mathcal{A}$ also blocks any further communication for $\mathcal{U}$.
3) The attacker $\mathcal{A}$ then mints a new maliciously generated zerocoin with commitment $c' = \mathsf{Com}(S, r')$, which is a commitment to the same serial number S.
4) The attacker $\mathcal{A}$ creates a spend transaction that spends c' with serial number S (to an address controlled by $\mathcal{A}$, so $\mathcal{A}$ gets his funds in the base currency back).

Now, transactions by $\mathcal{U}$ spending c will be rejected as double-spends by the cryptocurrency nodes, because they have already witnessed S in a previous spend transaction. Therefore, $\mathcal{A}$ has made $\mathcal{U}$'s zerocoin unspendable and effectively burned it.

We stress that an attacker does not need to have full control over the network to perform this attack. It suffices to control all outgoing connections of a single user, which is a concern in cryptocurrencies in general [15], [13] and in particular for thin clients relying on a semi-trusted server and clients connecting via relays, e.g., through the Tor network [13].

IV. Consequences and Discussion

The attack applies to the original Zerocoin protocol [20], and thus leads to real-world security vulnerabilities in all cryptocurrencies relying on this protocol, namely Zcoin [8], PIXV [6], SmartCash [7], Zoin [9], and Hexxcoin [3].

Furthermore, the attack applies to a second (unimplemented) Zerocoin protocol proposed by Groth and Kohlweiss [14].

a) Experimental Verification: To verify that the aforementioned cryptocurrencies are indeed vulnerable, we tested the attack on the library `libzerocoin` [4], written as a prototype implementation by the initial Zerocoin authors. It is used by all of the aforementioned projects. For testing, we used the version of the library that was deployed in Zcoin, when we discovered the attack.

We modified example code shipped with the library to confirm the attack. The modified version observes a spend transaction, creates a malicious mint transaction with the serial number taken from the observed spend transaction, and creates a malicious spend transaction that spends the malicious zerocoin. Further, it verifies the validity of the two malicious transactions, and that the serial numbers indeed collide. This confirms the attack.

b) Cost of the Attack: When carried out successfully, the attacker only incurs the transaction fee of his mint transaction and his spend transaction. He can get his base currency back by spending to one of his own addresses and therefore regains his investment fully except for the transaction fee.

[2] In the other formalizations [20], [14], the serial number is only later output by the Spend algorithm. This is only a syntactic difference. Our formulation simplifies our security definition.

[3] The anonymity set can have a short representation, e.g., a cryptographic accumulator [20], or even be implicit from the blockchain.

It is however important to note that the attacker risks his zerocoin: if the honest spend transaction wins, then the attack occurs effectively with reversed roles, i.e., the honest spend transaction burns the zerocoin of the attacker. If that is an concern for the attacker, he needs to make sure that either the user will not broadcast the honest transaction, or that the malicious transaction wins with very high probability. The attacker can achieve the latter by mining the transaction on its own such that the attack is effectively carried out only if it will succeed. In that case it is necessary to include both the malicious mint transaction and the malicious spend transaction in a single block.

c) Profiting from the Attack: The immediate impact of the attack is digital vandalism: an attacker can destroy funds of a honest user. This itself is a serious threat but the attacker is poised to profit from the attack as follows. The attacker short sells the currency, i.e., he bets on a falling price by borrowing and selling currency units, which must be sold back later. He then demonstrates publicly that he can destroy funds of the currency in the hope of creating panic among users, who will sell their currency. The attacker makes a sizable profit if the price drops as expected. This is infamously referred to as *insider trading* and is a very real threat. Similar attacks happened on other markets, where criminals tried to use their insider knowledge about their own forthcoming crimes to profit [12].

We are not aware of anyone who lends the affected cryptocurrencies, but we believe that there has just been not enough demand for borrowing due to the relatively low popularity of the affected currencies. In contrast, it is very easy to borrow popular cryptocurrencies such as Bitcoin and Ethereum. Moreover, nothing stops an attacker from borrowing units of the cryptocurrencies over the counter.

V. Fixing The Security Model

Despite the presence of this attack, both Zerocoin schemes [20], [14] are proven to be secure in the proposed security models. On the formal side, the problem is an arguably too weak security definition. Miers et al. [20] as well as Groth and Kohlweiss [14] define correctness notions, which capture that honest users are able to create spend transaction for their own zerocoins such that these spend proofs pass verification by the verification algorithm.[4]

However, the respective definitions fail to take the effect of double-spending prevention into account. The subtlety is that it is not the task of the verification algorithm to check for double-spending. Instead, the verification algorithm is just required to check whether a spend transaction is valid for a given serial number S, and checking whether S has been used already is outside the formal security model. (There is a good reason to use this approach: the verification algorithm can remain stateless, which simplifies the model.) As a result, the correctness property fails to capture this attack.

[4]Despite its name, Groth and Kohlweiss [14] define correctness as a security property in the sense that it is supposed to hold in presence of an attacker, who is allowed to select the anonymity set $\mathbf{C}$ of zerocoins adaptively. This is necessary to prevent other potential denial-of-spending attacks.

In the model by Miers et. al. [20], the language is ambiguous: Correctness, which is explicitly not listed as a security property, is supposed to hold for spends with anonymity sets $\mathbf{C}$ which can be "any valid set of coins" but it is not clear what "valid" refers to in this context.

$\mathsf{SNUnf}_{\Pi}^{\mathcal{A},p}(\lambda)$

$params \leftarrow \mathsf{Setup}(\lambda);\ \mathcal{T} \leftarrow \emptyset$
for $i \leftarrow 1, \ldots, p(\lambda)$
  $(c_i, S_i, skc_i) \leftarrow \mathsf{Mint}(params)$
$(\pi^*, S^*, R^*, \mathbf{C}^*) \leftarrow \mathcal{A}^{\mathsf{Spend}'}(params, (c_1, \ldots, c_{p(\lambda)}))$
return $\mathsf{Verify}(params, \pi^*, S^*, R^*, \mathbf{C}^*) = \mathsf{true}$
  $\wedge\ \exists i.\, S = S_i\ \wedge\ (\pi^*, S^*) \notin \mathcal{T}$

Oracle $\mathsf{Spend}'(i, R, \mathbf{C})$

if $c_i \notin \mathbf{C}$ **then**
  return $\perp$
$\pi \leftarrow \mathsf{Spend}(params, c_i, S_i, skc_i, R, \mathbf{C})$
$\mathcal{T} \leftarrow \mathcal{T} \cup \{(\pi, S_i)\}$
return (π, S_i)

Figure 1. Game for serial number unforgeability

To exclude the attack, we need an additional security property which captures that an attacker cannot forge a spend transaction *under a honestly generated serial number*, even if he is allowed to obtain other honest spend proofs (including serial numbers) of his choice. We denote this property, which is similar to *non-slanderability* in the context of ring and group signatures [18], [25], *serial number unforgeability*.

We note that in our model the attacker is allowed to query to oracle multiple times for the same zerocoin, i.e., the attacker is allowed to obtain double-spends from honest users. We have chosen this strong model, because honest users may double-spend the same zerocoin accidentally, e.g., when they keep the same zerocoins (or secret keys) in different wallets (desktop and mobile) and the wallets are not fully synchronized.

Definition 1 (Serial number unforgeability)**.** *A Zerocoin scheme* $\Pi = (\mathsf{Setup}, \mathsf{Mint}, \mathsf{Spend}, \mathsf{Verify})$ *is* serial number unforgeable *if for all* PPT *adversaries* $\mathcal{A}$ *and all polynomials* p *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for the game* $\mathsf{SNUnf}_{\Pi}^{\mathcal{A},p}$ *defined in Figure 1, we have*

$$\Pr\left[\mathsf{SNUnf}_{\Pi}^{\mathcal{A},p}(\lambda) = \mathsf{true}\right] \leq \mathsf{negl}(\lambda)\,.$$

VI. Fixing the Schemes

Fortunately, a simple and inexpensive fix is available to prevent the attack in each of the existing Zerocoin schemes.

Concretely, the modification (valid for each of the schemes) is as follows: Instead of using a fresh bitstring as serial number, we propose to use a fresh verification key of an ordinary signature scheme, which is strongly existentially unforgeable under chosen message attacks. The spender will additionally sign spend transactions under this verification key, and verifiers

will additionally verify these signatures using the verification key revealed as serial number.

Then the attacker will not be able to create a valid spend transaction under a honest same serial number, because this requires a signature under a honest verification key.

Theorem 1 (Unforgeability)**.** *The modified version of Miers et al.'s scheme and the modified version of Groth and Kohlweiss' scheme are serial number unforgeable.*

Proof sketch. In the modified schemes, the serial number is a verification key of a signature scheme, under which spend transactions need to be signed. Thus a forgery under a honest serial number implies a forgery under a honest verification key of the signature scheme. The theorem follows from the strong existential unforgeability of the signature scheme. □

Our modifications do not invalidate any other desired properties of the schemes: In particular, Miers et al.'s modified scheme fulfills *correctness*, *balance*, and *anonymity* in its respective security model [20], and Groth and Kohlweiss' modified scheme fulfills *perfect correctness*, *balance*, and *perfect anonymity* in its respective security model [14]. We do not present proofs, because the original proofs [20], [14] hold with trivial modifications. We note that our modification can be seen as a general transformation that turns any "secure" Zerocoin scheme without serial number unforgeability into a secure Zerocoin scheme with serial number unforgeability.

VII. Implementations and Responsible Disclosure

We contacted the authors of the original Zerocoin protocol, who confirmed the attack. Moreover, we contacted the Zcoin maintainers, who hired one of the authors of this paper to help them implement a fix. Unfortunately we noticed only later that the other mentioned cryptocurrencies also use the original Zerocoin protocol; we finally contacted the maintainers of the reference implementations of all of those cryptocurrencies.

In the case of Zcoin and Zoin, the respective maintainers have integrated the fix in the reference implementation of the wallets [16], [10], and zerocoins minted by the current version are not vulnerable to the denial-of-spending attack. However, old zerocoins (minted before with old versions) are still vulnerable at the time of writing, because there is nothing that prevents an attacker from minting new old-style zerocoins in order to attack honest spends of old zerocoins.

The Zcoin maintainers and the Zoin maintainers have indicated to us that an additional fix that prevents the attack mints will be released in the future. In the meantime, we feel that it is necessary to release our findings and inform users about the vulnerabilities. We currently recommend Zcoin users not to spend zerocoins minted with versions prior to v0.13.3 of the Zcoin wallet, and Zoin users not spend zerocoins minted with versions prior to v0.13.0 of the Zcoin wallet; in doubt, users should contact the maintainers of the their wallet software.

In the case of SmartCash, PIVX, and Hexxcoin, the maintainers have disabled the Zerocoin functionality: The SmartCash maintainers had already disabled Zerocoin [17] in response to other critical implementation vulnerabilities in libzerocoin.[5] Similarly a user from the Hexxcoin community went ahead and revived the dead project by releasing a client in which Zerocoin is disabled [1], and later, also the PIVX maintainers disabled Zerocoin.

Note that disabling Zerocoin is not a permanent solution: Beside the obvious drawback for privacy, it implies that minted but unspent zerocoins cannot be spent currently, effectively disabling these zerocoins temporarily. We believe that this interferes seriously with the savings of users, but nevertheless the respective communities seem to have accepted that Zerocoin is disabled.

VIII. Related Work

a) Zerocash: Zerocash [23], which can be considered a successor to Zerocoin, also makes use of serial numbers to prevent double-spending. In Zerocash however, a general *transaction non-malleability* property prevents the attack described above, because it includes serial number unforgeability as a special case. In the construction, the serial number is a output of a pseudorandom function (PRF). Since the PRF key is kept secret by honest users and required to authenticate spend transactions, the construction ensures that no attacker can create a spend transaction valid with a honest serial number.

b) Monero: In the anonymous cryptocurrency Monero [5], [22] the serial number[6] is constructed as $S = H(c)^{skc}$, where $c = g^{skc}$ is a public part of the coin, skc is the corresponding secret key and H is a hash function modeled as a random oracle. As in Zerocoin, the security model by Noether and Mackenzie [22] lacks the consideration that an attacker could produce a malicious spend proof valid under an honest serial number. However, the construction used in Monero ensures that an attacker, given an honest coin and its serial number, cannot find a coin with the same serial number. Consequently, Monero is not vulnerable to our denial-of-spending attack. Nevertheless, more work is required to determine the exact security properties of the construction used in Monero.

Sun et al. [25] propose a different construction and an improved security model for Monero that includes an explicit non-slanderability property that is similar to serial-number unforgeability as defined above.

References

[1] Forked Hexxcoin. https://github.com/hexxcointakeover/4.0.1.X.
[2] "CoinMarketCap," https://coinmarketcap.com/.
[3] "Hexxcoin," https://bitcointalk.org/index.php?topic=1171724.1180.
[4] "libzerocoin," https://github.com/Zerocoin/libzerocoin.
[5] "Monero," https://getmonero.org.
[6] "PIVX," https://pivx.org/.
[7] "SmartCash," https://smartcash.cc/.

[5]Those vulnerabilities have been found by the author implementing the fix for Zcoin. We note that `libzerocoin` was developed an academic prototype and includes very explicit warnings about security and non-readiness for deployment [4], but was nevertheless integrated effectively without changes in Zcoin, SmartCash, Zoin and Hexxcoin, which were all vulnerable to these implementation issues. Only the developers of PIVX audited the library before deployment and found those issues independently.

[6]In Monero, the term *key image* is used instead of serial number. For the sake of readability, we stick with the Zerocoin terminology.

[8] "Zcoin," https://zcoin.io/.
[9] "Zoin," https://zoinofficial.com/.
[10] "Zoin core release v.0.13.0.0," https://github.com/zoinofficial/zoin/releases/tag/v0.13.0.0.
[11] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in Bitcoin," in *FC'13*.
[12] BBC, "Borussia dortmund bombs: 'speculator' charged with bus attack," 2017, http://www.bbc.com/news/world-europe-39664212.
[13] A. Biryukov and I. Pustogarov, "Bitcoin over Tor isn't a good idea," in *S&P'15*.
[14] J. Groth and M. Kohlweiss, "One-out-of-many proofs: Or how to leak a secret and spend a coin," in *EUROCRYPT'15*.
[15] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on Bitcoin's peer-to-peer network." in *USENIX Security '15*.
[16] P. Insom, https://github.com/zcoinofficial/zcoin/commit/01e25084b88622a5d9dd1207a4cbebd78bed1dcb.
[17] JuicyG, "SmartCash Zerocoin related issues and the path forward," https://forum.smartcash.cc/t/smartcash-zerocoin-related-issues-and-the-path-forward/1359.
[18] J. K. Liu and D. S. Wong, "Linkable ring signatures: Security models and new schemes," in *ICCSA'05*.
[19] 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 *IMC'13*.
[20] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in *S&P'13*.
[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[22] S. Noether and A. Mackenzie, "Ring confidential transactions," *Ledger*, vol. 1, 2016, https://doi.org/10.5195/ledger.2016.34.
[23] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *S&P'15*.
[24] M. Spagnuolo, F. Maggi, and S. Zanero, "BitIodine: Extracting intelligence from the Bitcoin network," in *FC'14*.
[25] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero," in *ESORICS'17*.