Building a Private Currency Service Using Exonum

Darya Korepanova, Maria Nosyk, Alex Ostrovsky and Yury Yanovich

Abstract—Zero-knowledge proofs are an emerging cryptographic technology that have many potential applications for blockchains. Exonum is an extensible open-source framework for creating blockchain applications. In this article, we describe how zero-knowledge proofs, specifically bulletproofs, can be applied to build a privacy-focused service using Exonum. The token logic is implemented as a platform service and is a proof of concept.

I. INTRODUCTION

Zero-knowledge proof/argument (ZKP) [1], [2] is an emerging cryptographic technology that promises to bring us closer to the zenith of blockchain: providing data privacy for blockchain users without sacrificing auditability.

Potential applications for zero-knowledge proofs include, but are not limited to: inter-bank transfer systems [3], privacy-focused management of digital assets [4], know your customer [5], self-sovereign identity [6], voting [7].

Another application for zero-knowledge proofs is helping blockchains scale [8]. ZKPs allow for the "compressing" of computations for blockchain transactions without sacrificing security. In this article, we describe how zero-knowledge proofs (specifically, bulletproofs [9]) can be applied to build a privacy-focused service using Bitfury's Exonum platform (https://exonum.com/).

II. SITUATIONAL ANALYSIS

It is possible to achieve some level of data privacy in blockchain apps using the "walled garden" approach, where the data is hidden because access to it is restricted with the help of firewalls, role-based access control, moats and other perimeter security measures. Sensitive data in the blockchain may be encrypted (perhaps, with a public-key encryption scheme, with the relevant public keys managed by the same blockchain) and/or stored outside of the blockchain (in this case, the blockchain stores only hash fingerprints of the data). This approach is used in many permissioned distributed ledger frameworks [10], [11], [12], [13], [14].

However, the disadvantages of the "walled garden" approach are becoming increasingly apparent. Namely, the approach is antithetical to one of the main selling points of blockchain–auditability. If the data on the blockchain cannot be audited by consulting smart contract logic, the blockchain becomes a glorified linked timestamping service [15]. The fact that there is some data on the blockchain no longer means that this data is valid as per smart contract rules. The second major disadvantage to the walled garden approach is that it does not scale. R3's CTO Richard Brown, for example, aptly compared the privacy model of their solution to Slack channels—it is difficult to securely add or remove participants to/from the garden, even more so when there are no prior expectations as to the number and identities of these participants [16].

This is where zero-knowledge can be valuable. By design, zero-knowledge proofs and arguments convincingly prove a statement about private data without revealing anything about the data except the statement being proved. It is easy to make zero-knowledge proofs universally verifiable, without sacrificing any privacy! This feature is exactly what is needed to build a system that is both privacy-preserving and auditable at the same time.

III. OUR RESEARCH

To demonstrate the use of zero-knowledge proofs, we are going to build a cryptocurrency service with similar functionality to the tutorial services in the Exonum [17] documentation (https://exonum.com/doc/get-started/create-service/). The service makes it possible to register users and wallets (providing an initial token balance as a reward) and transfer tokens among registered parties. All transactions are authenticated with the help of a digital signature cryptosystem, Ed25519, which is built into Exonum services. We do not hide identities of transacting parties (i.e., their public keys), but we hide the number of tokens being transacted and the balance of each account in the system. We also discuss how we could improve the service to hide the transacting entities at the end of the article.

The service is fully open-sourced and can be accessed on Github https://github.com/exonum/private-currency.

IV. CRYPTOGRAPHY BACKGROUND

In order to understand how the service works, we first need to introduce ourselves to the core cryptographic primitive underpinning bulletproofs — a concept called Pedersen commitments [18]. A cryptographic commitment scheme is somewhat like a hash function: someone inputs secret data (the opening) and gets the output that is unrecognizably scrambled (the commitment). One can then reveal the opening to prove that the committed value corresponds to it.

The difference with hash functions is that besides being *binding* (no one can devise two different openings producing the same commitment), a commitment scheme is also expected to be hiding (it is impossible to reverse the scheme and produce an opening given a commitment). A hash function is *hiding* if its input is uniformly distributed about the entire input space, but this assumption most frequently does not hold for commitments (indeed, it must be possible to commit to a value from a very small set, such as Boolean). As such, the opening contains, besides the payload, the *blinding factor* which makes it (at least statistically) improbable to guess the payload given the commitment.

The Pedersen commitment scheme uses a prime-order group, in which the discrete logarithm problem (DLP) is believed to be hard [2], together with two generators, G and H. G and H must be chosen in such a way that the discrete log relation among them is unknown; in other words, no one knows k such that $H = kG$. The opening is a pair (x, r), where x is the committed value and r is the blinding factor; both are group scalars (essentially, integers with "overflow" akin to finite integer types used in most programming languages). The commitment is computed as $\mathrm{Comm}(x; r) = xG + rH$. It can be proven that if DLP in the group is hard, the Pedersen commitment is computationally binding and perfectly hiding.

The crucial property for Pedersen commitments is that they are additive: the sum (or difference) of two commitments is a commitment to the sum (or difference) of committed values. Indeed,

$$C_1 = x_1G + r_1H;\;\; C_2 = x_2G + r_2H \rightarrow$$
$$C_1 + C_2 = (x_1 + x_2)G + (r_1 + r_2)H =$$
$$= \mathrm{Comm}(x_1 + x_2; r_1 + r_2).$$

V. BUILDING A SERVICE

With our knowledge, we can securely hide account balances and transfer amounts with the help of Pedersen commitments. Using range proofs, we can prove/verify that a transfer is correct:

- The transferred amount is positive
- The sender has enough balance in his account.

For the first proof, we take the commitment to the transfer amount, C_a (it is directly present in the transfer transaction), and verify that the value committed in $C_a - \mathrm{Comm}(1; 0)$ lies in the range $[0, M)$. Indeed, this is equivalent to proving that C_a corresponds to a value in the range $[1, M]$. The sender can produce this proof, as he knows the transferred amount a.

For the second proof, we need to take the commitment for the sender's current balance, C_s, and verify that the value committed in $C_s - C_a$ lies in the range $[0, M)$. Again, the sender can produce this proof as he knows the opening to both C_s and C_a.

To apply the transfer to the blockchain state, we subtract amount commitment C_a from the sender's balance commitment (as we have verified, it cannot lead to a negative balance, or to the increase of the sender's balance), and then add C_a to the receiver's balance commitment.

VI. KEY DETAILS

It is important to note that there are a few conditions that can make the implemented service more complex.

The receiver of a transfer must find out the opening to C_a from somewhere; otherwise, he ceases to know the opening to his balance and can no longer do anything with his wallet. The opening is not present in the plaintext of the transfer transaction (which is the entire point). We could assume that the receiver reliably obtains the opening via an off-chain channel (for example, sent by the sender via Telegram), but that is not an illustrative scenario. So instead, we encrypt the opening using two-party public-key encryption based on Diffie-Hellman key exchange [19]. For the added benefit, Curve25519 keys [20] required for the box routine can be converted from Ed25519 keys, so we may continue to use a single keypair for each user instead of introducing separate encryption keys.

Once we introduce encryption, we can no longer apply the transfer atomically. Indeed, the sender can maliciously or unintentionally provide garbage instead of the opening encryption, and the blockchain logic won't be able to tell that this is the case. Thus, we request the receiver to explicitly accept the transfer via a separate transaction.

Before a transfer is accepted, it modifies the sender's balance commitment (otherwise, we would allow double-spending!), but not the receiver's one.

Once the acceptance transaction is confirmed by the blockchain network, the receiver's balance is updated, and the transfer is complete.

To prevent deadlocks, a transfer specifies a time-lock delay (in relative blockchain height, a la Bitcoin's CSV) for the receiver to signal acceptance. If the time lock is expired, the transfer is automatically refunded to the sender (Exonum allows this via `Service::beforeCommit()` hook).

Another issue is more intricate. In order to produce the proof of a sufficient balance, the sender needs to know his current balance, which may not always be the case. A stray acceptance transaction or refund may increase the sender's balance unwittingly to him/her; in this case, the transfer will fail verification, and the sender will be reasonably frustrated. To alleviate this problem, we allow transfers to reference what the sender thinks is his current wallet state (more precisely, the reference takes form of the number of events changing account balance — transfers and refunds). When checking the proof of a sufficient balance, we use the referenced state to obtain the sender's balance commitment. Additionally, we check that no outgoing transfers have occurred since the referenced state. If this is the case, we can be certain that if we subtract the transfer amount from the sender's current balance, we will end up with a non-negative value. Indeed, the events in the account history after the reference point (incoming transfers and refunds) can only increase the balance!

With reference points in place, the sender is still somewhat constrained; he must have no pending transfers when creating a new transfer. Still, this restriction is much less limiting than the requirement to know one's account state at the moment of the transfer; fundamentally, we make the sender dependent on what he did previously, but not on others' actions.

VII. IMPLEMENTATION

We use a bulletproofs library written in pure Rust (https://doc.dalek.rs/bulletproofs/), which has recently reached pre-release stage. Since the Exonum platform is written in Rust, it integrates with the library seamlessly. For added benefit (and unlike the version of bulletproofs described in the original whitepaper which is being developed in C and uses Bitcoin's secp256k1 elliptic curve), the library we use is based upon Curve25519, which is already used in Exonum as the main component of the Ed25519 [21] digital signature cryptosystem.

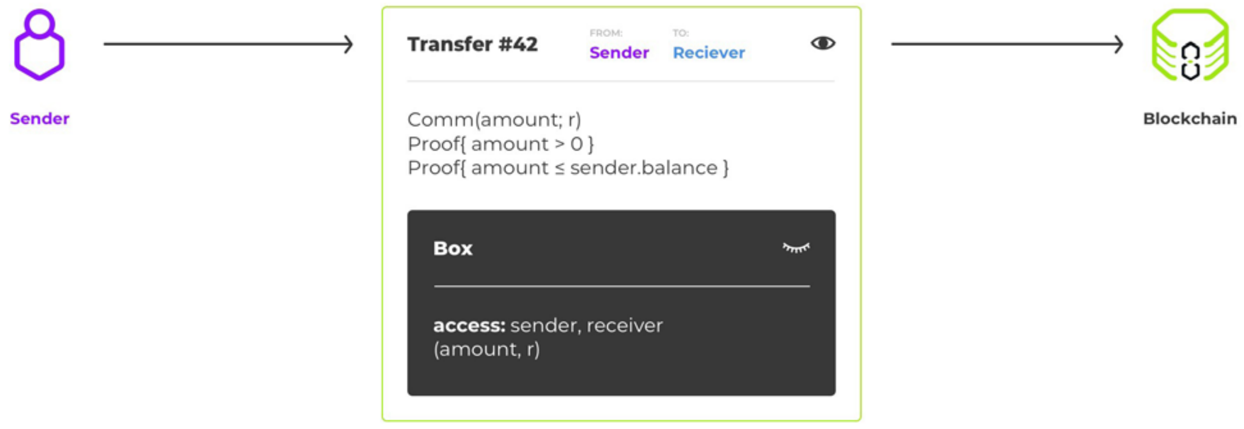
Implementing the service based on the above description is quite straightforward. The most difficult part was constructing Merkle proofs that authenticate information returned to the user so that he does not have to blindly trust the Exonum nodes he communicates with. Improving experience of service developers in this regard is one of the major goals of the Exonum 1.0 release.

VIII. NEXT STEPS

The service we have built does not hide the identities of the sender and the receiver of transfers, which is a major limitation for real-world applications. Fortunately, there are ways to solve this problem.

The generic technique used in zCash [22] is based on creating a Merkle tree of the system state. For example, zCash builds the note commitment tree, which is roughly equivalent to ever created transaction outputs in Bitcoin. Zero-knowledge proofs then encompass authentication paths (aka Merkle branches) in this tree, reveals something about an element of the tree without revealing which element is referred to. The downside of this approach is that cryptographic hash functions used to build Merkle trees are difficult to transfer into the zero-knowledge realm; the resulting proofs become computationally expensive — a single proof can take seconds or even minutes

Fig. 1. Record of transaction in blockchain



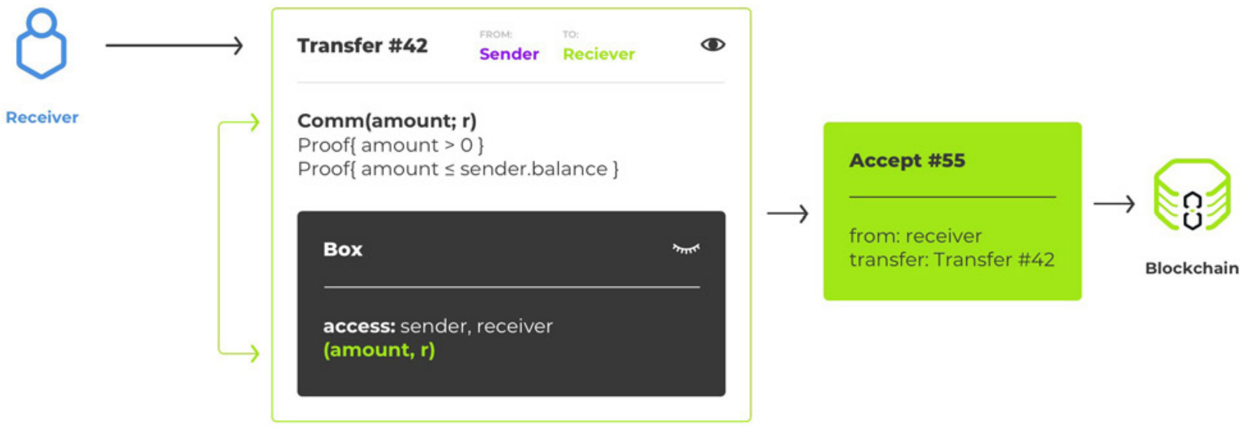


Fig. 2. Receiver verifies and accepts the transfer

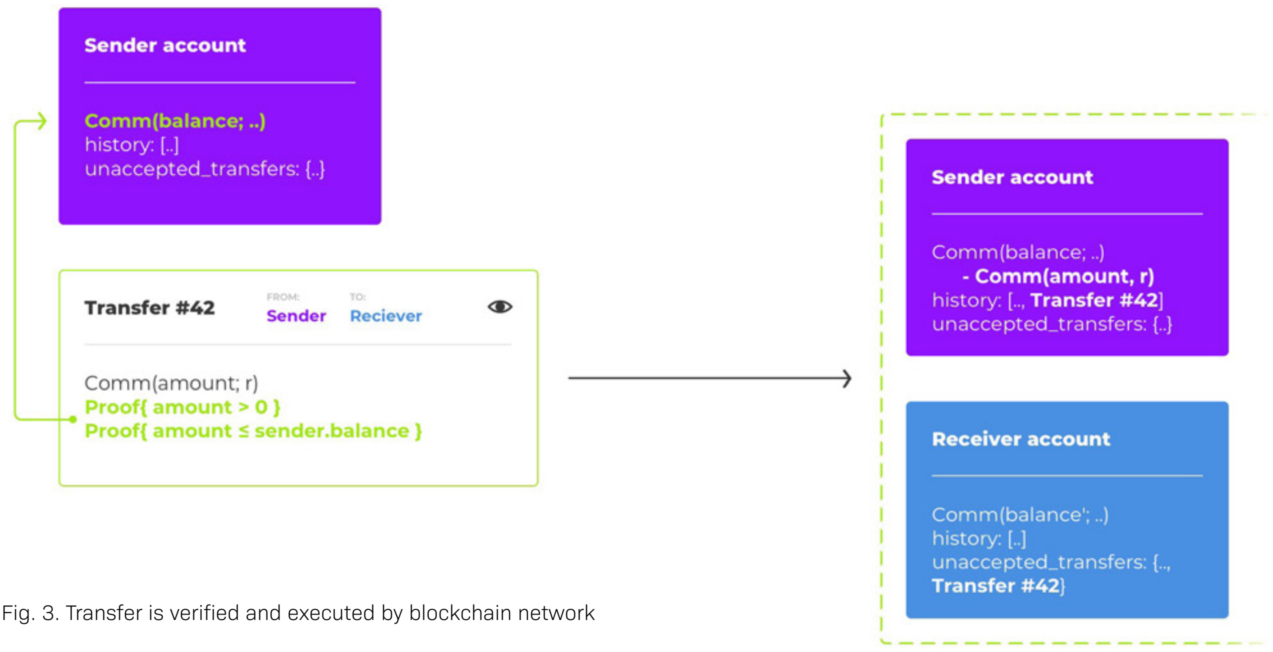


Fig. 3. Transfer is verified and executed by blockchain network

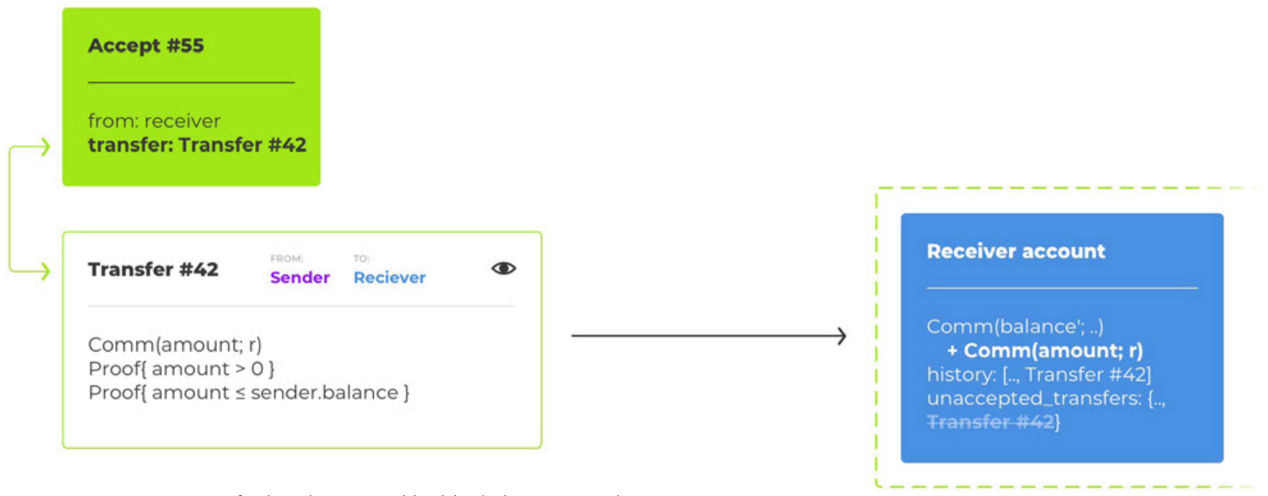


Fig. 4. Acceptance is verified and executed by blockchain network

to create. Searching for more "ZKP-friendly" cryptographic hash functions is an area of active research.

If we admit additional constraints, there may be an easier solution. For example, a recent paper by Narula et al. [3] describes a system with a limited, a priori known number of participants, which can transact among themselves without revealing participants or transferred amounts for any transaction.

On a more prosaic note, there are probably many technical improvements that the developed service can enjoy: more test coverage, separation of signing and encryption keys, benchmarking, etc. A major improvement to the service UX would be enabling deterministic ordering of transactions originating from the same user, which we plan to solve not long after releasing Exonum 1.0.

IX. CONCLUSION

We have described how to construct an account-based cryptotoken with strong privacy enabled by zero-knowledge proofs (specifically, bulletproofs). The token logic was implemented as an Exonum service. Although currently the service is just a proof of concept, it showcases how the Exonum platform can be used to build atop complex cryptographic primitives with very low overhead imposed by the execution environment.

References

[1] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", SIAM Journal on Computing, vol. 18, No. 1, pp. 186–208, 2 1989.

[2] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, ser. Discrete Mathematics and Its Applications. CRC Press, 10 1996.

[3] N. Narula, W. Vasquez, and M. Virza, "zkLedger: Privacy-Preserving Auditing for Distributed Ledgers", in NSDI'18. USENIX, 2018, pp. 65–80. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/narula.

[4] JPMorgan and Zcash, "ZSL Proof of Concept", 2018. [Online]. Available: https://github.com/jpmorganchase/quorum/wiki/ZSL.

[5] I. Allison, "ING Bank Launches Zero-Knowledge Tech for Blockchain Privacy", 2018. [Online]. Available: https://www.coindesk.com/ing-bank-launches-simplified-zero-knowledge-proofs-for-blockchain-privacy.

[6] K. Rannenberg, J. Camenisch, and A. Sabouri, Eds., Attribute-based Credentials for Trust. Cham: Springer International Publishing, 2015.

[7] A. Chekanov, T. Knowles, U. Sauer, M. Rexestrand, C. Calderon, A. Wyss, S. Lindsay, and G. Guerin, "General Meeting Proxy Voting On Distributed Ledger", Tech. Rep., 2017. [Online]. Available: https://www.six-group.com/swiss-sptc/dam/downloads/swiss-sptc/meeting-minutes/sptc-protokoll-40-csd-dlt-working-group.pdf.

[8] V. Buterin, "On-chain scaling to potentially ~500 tx/sec through mass tx validation", 2018. [Online]. Available: https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477.

[9] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More", in 2018 IEEE Symposium on Security and Privacy (SP), vol. 2018 May. IEEE, 5 2018, pp. 315–334.

[10] Bitfury Group and J. Garzik, "Public versus Private Blockchains. Part 1: Permissioned Blockchains", bitfury.com, pp. 1–23, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt1-1.pdf.

[11] C. Walsh, P. OReilly, R. Gleasure, J. Feller, S. Li and J. Cristoforo, "New kid on the block: a strategic archetypes approach to understanding the Blockchain", ICIS 2016 Proceedings, pp. 1–12, 12 2016. [Online]. Available: https://aisel.aisnet.org/icis2016/Crowdsourcing/Presentations/6.

[12] Bitfury Group, "On Blockchain Auditability," bitfury.com, pp. 1–40, 2016.

[13] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric", IBM Research, vol. July, 2016.

[14] E. Androulaki, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolič, A. Barger, S. W. Cocco, J. Yellick, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, and G. Laventman, "Hyperledger fabric", in Proceedings of the Thirteenth EuroSys Conference on EuroSys 2018. New York, New York, USA: ACM Press, 2018, pp. 1–15. [Online]. Available: https://arxiv.org/pdf/1801.10228.pdf, http://arxiv.org/abs/1801.10228, http://dl.acm.org/citation.cfm?doid=3190508.3190538.

[15] S. Haber and W. S. Stornetta, "How to Time-Stamp a Digital Document", in Advances in Cryptology-CRYPTO 90. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 437–455.

[16] R. G. Brown, "What Slack Can Teach Us About Privacy In Enterprise Blockchains", 2017. [Online]. Available: https://gendal.me/2017/07/20/what-slack-can-teach-us-about-privacy-in-enterprise-blockchains/.

[17] Y. Yanovich, I. Ivashchenko, A. Ostrovsky, A. Shevchenko and A. Sidorov, "Exonum: Byzantine fault tolerant protocol for blockchains", bitfury.com, pp. 1–36, 2018.

[18] T. P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing", in Advances in Cryptology CRYPTO 91. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 129–140.

[19] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, vol. 22, No. 6, pp. 644–654, 11 1976. [Online]. Available: http://ieeexplore.ieee.org/document/1055638/.

[20] D. J. Bernstein, "Curve25519: New Diffie-Hellman Speed Records", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Berlin, Heidelberg, 2006, vol. 3958 LNCS, pp. 207–228.

[21] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," Journal of Cryptographic Engineering, vol. 2, No. 2, pp. 77–89, 9 2012.

[22] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer and M. Virza, "Zerocash: Practical Decentralized Anonymous E-Cash from Bitcoin", in Proceedings of the 2014 IEEE Symposium on Security and Privacy. IEEE, 5 2014, pp. 459–474.