Unobservable Contracts from Zerocash and Trusted Execution Environments

Adrian Cinal[0009−0001−9789−0470]

NASK National Research Institute
adrian.cinal@nask.pl

Abstract. Privacy-oriented cryptocurrencies like Zerocash only support direct payments and not the execution of more complex contracts. Bitcoin and Ethereum, on the other hand, cannot guarantee privacy, and using them for contract execution leaves open questions about fungibility of the proceeds and requires contract designers to take frontrunning countermeasures. This work reconciles the two worlds and develops a practical framework for decentralized execution of complex contracts that (1) is undetectable to the network at large, (2) maintains anonymity of the potentially mutually distrustful counterparties, (3) guarantees fair termination, and (4) is immediately resistant to frontrunning and miner bribery attacks. This is achieved by leveraging the confidentiality and anonymity guarantees of Zerocash and the verifiability and flexibility of trusted execution environments.

Keywords: Fairness · Fungibility · Privacy coins · Trusted execution environments

1 Introduction

Whereas an extensive line of research [1,2,4,7,13,15,20,23] has been devoted to leveraging cryptocurrencies to enforce fairness in multiparty computation (MPC) by penalizing and financially compensating offenders and their victims, respectively, an overlooked aspect has always been the fungibility of the eventual proceeds from such "contracts." Bentov and Kumaresan in [7], when formalizing fair MPC, assume coins to be fungible and "perfectly indistinguishable from each other," which, say, bitcoins clearly are not [27,28]. In protocols studied therein, such distinguishable coins change hands in making and claiming deposits: "[t]he amount deposited equals the amount claimed" [7], but participants do not get their deposits back but instead those of their counterparties, and coins obtained from, e.g., gambling could be deemed less valuable.[1] Transactions used for implementing contracts are also non-standard or otherwise uncommon, interact

[1] While paying premium relative to the nominal value in order to account for future devaluations could address this, the feasibility of this approach is questionable, as quantifying the associated risk is a challenge [27,28], especially in prevailing regulatory landscape with new legislation, affecting the perceived value of coins, emerging yearly [30].

with smart contracts [4, 15, 17] or use features such as timelocks [1, 2, 7, 13, 23], making them clearly distinguishable from regular point-to-point payments. Furthermore, termination of (or any progress in) a decentralized contract is entirely at the discretion of miners who have the final say about when (or whether at all[2]) to include a transaction in the ledger. This raises concerns about frontrunning [13], censorship [33], and the threat of bribery, since miners see the details of a contract *before* it has been settled. Finally, it may be desirable for participants of a contract, say, a purchase of digital commodities, to keep the details of the purchase confidential [10] or even the very fact of it having occurred undetectable to the public. If the parties are distrustful of each other, they may also want to keep their identities secret from one another.

All the above concerns are addressed successfully in this work by leveraging *trusted execution environments* (TEEs) in a fully decentralized fashion and in conjunction with Zerocash [5]. This hybrid guarantees unobservability of a contract and makes proceeds from it indistinguishable from regular payments. With Zerocash only supporting point-to-point payments and no scripts, timelocks, multisignatures or other gadgets typically involved in decentralized contracts, the key technical challenge becomes formulating principles of contract design that have the TEEs emulate these missing pieces as well as heavyweight cryptography such as witness encryption and zero-knowledge proofs. While there is prior work relying on TEEs to enable new uses of cryptocurrency [3,6,13,16,17,22,24–26,34], privacy in the strongest sense considered in present paper has not, to the best of the author's knowledge, been studied before.

1.1 Preliminaries

This section introduces the two main primitives used in this work: Zerocash and trusted execution environments. Familiarity with elementary cryptocurrency concepts such as Proof-of-Work blockchains, mining, the UTXO model, or double-spending is assumed.

Zerocash. Zerocash is a privacy-oriented cryptocurrency protocol introduced by Ben-Sasson et al. in [5] and deployed as Zcash [18]. Unlike Bitcoin's, Zerocash transactions reveal neither the participants nor the transacted amounts in plaintext. Instead, the network of validators (miners) is convinced that a transaction is well-formed, i.e., spends valid and previously unspent transaction outputs and its amounts balance, through zero-knowledge proofs. Zerocash is based on a Proof-of-Work blockchain assumed to have standard security properties, namely, liveness, immutability, and consistency [13].[3]

[2] In Bitcoin, conservative miners have been observed to reject transactions with complex scripts [1, 23].

[3] For clarity of exposition, miner's fees and, in particular, their volatility are ignored, as they can be straightforwardly accounted for.

Trusted Execution Environments. Trusted execution environments (TEEs) are a promising alternative to heavyweight cryptography like zero-knowledge proofs and multiparty computation, enabling machine code to be executed with confidentiality and integrity guarantees even in the presence of a malicious operating system or hypervisor. As long as the physical CPU package is not tampered with, TEEs are protected from both software-based and physical attacks. TEEs typically support *remote attestation*, a procedure through which a remote relying party can be convinced that the correct program is executing on an authentic hardware platform in a correctly established TEE. While TEEs do guarantee data confidentiality as well as control flow and memory integrity,[4] any interaction with the outside world, in particular, network and disk access, must be intermediated by the untrusted host. This allows for malicious data to be injected into the TEE and messages to be reordered or dropped altogether. In particular, the TEEs cannot rely on their hosts' to give them accurate view of the blockchain, as it may be in their best interest to isolate the TEE, by itself having no meaningful sense of time, and schedule it off the CPU for sufficiently long to present it with a forged subchain.

1.2 Motivating Example

Suppose Alice the Auctioneer has a digital commodity x that she wishes to sell to the highest of three bidders: Betty, Bob, or Brian. Suppose each bidder specifies a (possibly different) predicate ϕ_i, for $i = 1, 2, 3$, that x must satisfy.[5] We want to achieve *fairness* with the following events happening atomically: Alice gets paid the highest bid by the auction's winner, the winner gets the commodity x, and losers get refunded any deposits they may have made. It should be impossible for one party to default on their obligations but nonetheless benefit from the contract or indeed for any party to unfairly cause losses for other parties. Suppose each party hosts a TEE that has attested in front of the others to the code it is executing, and that secure (i.e., encrypted and authenticated) channels have been established between Alice's TEE and the TEEs of the bidders. Alice can escrow x with the TEEs of the bidders, who in turn deposit their bids b_i in transactions t_i to the payment address A established by Alice's TEE. (Note that the bid transactions t_i need not overdeposit as in [13] in order to hide the actual bids, as the confidentiality guarantees of Zerocash ensure bids cannot be frontrun.) The transactions t_1, t_2, and t_3 can be given to Alice as, in Zerocash, they reveal no information about the bids or the bidders' identities. Alice can confirm they have been broadcast to the Zerocash network and appended to the blockchain, whereas her TEE can confirm they credit the bids (higher than some minimal amount) to the correct address A and clear the auction, i.e., select the highest bid $b = \max\{b_1, b_2, b_3\}$ (or, say, second highest) and issue a settlement transaction t that sends b coins to Alice and refunds the losers. It cannot be

[4] Side-channel and fault-attack vulnerabilities are outside the scope of this work.

[5] In this paper, ϕ_i can be practically arbitrarily complex, e.g., may be a neural network assessing the validity of its input according to some metric.

issued to Alice, however, as she would then learn the winning bid b ahead of time and could unfairly back out of the auction by refusing to broadcast t to the network. Instead, her TEE can send t to the TEEs of the bidders for them to broadcast it. t should be withheld from the winner, who would anyway not want to publish it, as they have likely overpaid,[6] but the losers can get it, since they want to get refunded. The moment one of them broadcasts t, and it gets accepted to the blockchain, Alice has been compensated, and the winner, say, Bob, can present t to his TEE to convince it that the contract has been settled, and that it should release x.[7] If the auction is aborted after Betty has already placed her bid, she can convince her TEE that the contract has timed out without settlement to get a refund transaction t'. By the time she manages to forge such a proof, however, her deposit will have been concurrently spent in t (and credited either to Alice or to Betty herself, depending on the auction's outcome) and the refund transaction t' will be worthless. This is elaborated in Section 2.

1.3 Contributions

The paper proposes, for the first time, combining Zerocash with trusted hardware for the purposes of contract execution and explores the advantages and pitfalls of such a hybrid. The contract engine proposed enables provably unobservable execution (the strongest privacy notion to date), where parties to a contract remain anonymous to each other, and the network at large cannot tell a contract was ever run, preserving the fungibility of the currency. No changes to the Zerocash protocol are required allowing the construction to be deployed today. In achieving this, a number of technical challenges are overcome posed by the minimal assumptions on the underlying blockchain including using Proof-of-Work puzzles as (probabilistic) proofs of contract timeout and ensuring settlement atomicity through transaction chaining. These are all of independent interest.

1.4 Related Work

Most relevant for this work is that of Das et al. [13], who propose FastKitten: a TEE-based contract execution framework where a single *operator* hosts a TEE and is held accountable by a timelocked *penalty transaction*. Present work can be viewed as extending FastKitten to the setting of privacy-oriented cryptocurrencies without even timelock capabilities. Castejon-Molina et al. in [10] construct contingent payments that are unlinkable by virtue of relying on coin mixers. Privacy of contracts settled on blockchains is considered in [4,9,10,19,22]. Full unobservability is outside the scope of all of them. Thyagarajan et al. in [32] stress the importance of preserving fungibility formalized as indistinguishability of contract execution from regular point-to-point payments. Wüst et al. in [35] are the first, to the best of the author's knowledge, to propose using TEEs in

[6] "The winner's curse."

[7] Since, in Zerocash, Bob will not know what t looks like ahead of time, he must make guesses submitting to his TEE a suffix of the blockchain that *could* contain it.

conjunction with Zerocash but do it only to offload blockchain monitoring and enable SPV clients. Finally, the synergy between TEEs and blockchains is also explored in a number of other works, including [6,11,12,14,16,17,20,25,26,34,35].

2 Unobservable Contracts

A *contract* is a protocol realizing a (randomized) functionality $\mathcal{F}$ that takes as inputs "commodities" $x_i \in \{0,1\}^*$ (of some economic value) and balances u_i of each *player* $i = 1, \ldots, N$ and outputs to each player $y_i \in \{0,1\}^*$ (which could be one of the input commodities x_j or a function thereof) and updates their balance to v_i. A contract is *fair* if it either terminates correctly or gets aborted with all honest parties remaining financially neutral (accounting for value of commodities) [13]. Say a contract is *undetectable* [29] if no efficient distinguisher, themselves not a party to the contract, can tell apart two ledgers: one on which the contract was executed and one on which it was not. A contract is *unobservable* [29] if it is undetectable, *and* furthermore each player i only learns negligible information about the ledger identities (addresses) of other players. The *ledger indistinguishability* property of Zerocash [5] guarantees that contracts built on top of it are *undetectable*. Anonymity will be achieved by virtue of intermediating all payments $v_i - u_i$ by TEEs hosted by the players themselves.[8] When coupled with network-layer obfuscation (e.g., onion routing) and proper handling of TEE remote attestation (see below), the above hybrid makes the parties to a contract anonymous to each other and their repeated interactions unlinkable. Without timelocks, however, the FastKitten construction of [13] no longer applies to Zerocash, leading to multiple players having to host TEEs. The TEEs can attest in front of all other players to the program they are executing, jointly establish a payment address A, and escrow both the inputs x_i (whose consistency they can verify) as well as *deposits* made to address A. In the end, one of the TEEs (cf. the "operator" in [13]) *clears* the contract and issues a *settlement transaction* t to parties that would benefit from it and are thus incentivized to broadcast it. (Observe how, in Zerocash, other players will not be able to identify t in the mempool; hence, to deny it, they would have to bribe the miners to stop mining altogether and account for their opportunity cost.) The core technical challenge is for the TEEs to be convinced that the deposit transactions presented to them have indeed been broadcast to the network and mined. This is resolved via incentive design and relying on counterparties to confirm each other's deposits. It is never in the interest of players to lie, namely, claim a deposit was made when in fact it was not, since this would invalidate the settlement transaction.[9] Other players in this case would eventually be able to be refunded by the TEEs they host. No external trusted or semi-trusted parties are assumed

[8] Note that, while payment channels do not satisfy the above definition of a contract, the principles developed in this paper can naturally be leveraged to bestow unobservability on the construction of [25] as well.

[9] It would reference as input a transaction that never made it to the blockchain.

such as reliable time servers [11] or arbiters for dispute resolution [15].[10] The remainder of this section elaborates on the technical challenges needed to be overcome when deploying unobservable contracts intermediated by TEEs.

Trusted Randomness and Secure Channels. With randomness not easy to come by in a TEE, this work has each player contribute a seed d_i (via a secure channel established in remote attestation), and the TEEs use a PRNG (all of them the same one) seeded with $g = \mathsf{Hash}(d_1, \ldots, d_N)$ as their source of randomness.[11] Using this PRNG, secure channels between the TEEs can be established as well as the deposit address A.

Anonymous Attestation. While, in prior work [3, 6, 11–14, 16, 17, 20, 24–26], the remote attestation process was taken for granted, it becomes of concern when striving for privacy guarantees as strong as in this paper. Indeed, remote attestation involves signing a hardware-generated report with a key that is derived from a hardware root of trust and certified by the CPU vendor, e.g., Intel. So as to prevent linking interactions with the same CPU (and hence, likely, the same counterparty), group signatures can be used for attestation as originally done by Intel [8],[12] or the CPU can be reprovisioned with a new attestation key for each contract.[13]

Transaction Atomicity and Chaining. In [13], the TEE operator is incentivized to settle through a separate penalty transaction, timelocked and distributed to all players as part of contract setup. In this work, the player clearing the contract must instead be incentivized by the settlement transaction itself. In many contracts of practical interest (e.g., auctions), a key enabling property is that there is at least one (often just one) player incentivized to settle. In that case, we will want that player to not only claim *their* funds but finalize the contract for *everyone*. For a small number of players N, one can bundle all the payouts in a single transaction, leveraging the fact that Zerocash transactions are not malleable [18] and signed as a whole with all their outputs fixed. (This settlement *atomicity* also disincentivizes players from lying about the state of the blockchain: if a deposit transaction was not really confirmed on chain, then the settlement transaction eventually trying to redeem it will be malformed and any real deposits burned.) As N grows larger, however, limits on transaction size

[10] For the purposes of remote attestation, access to some public-key infrastructure of the CPU vendor may be needed.

[11] When treating Hash as a random oracle, choosing d_i uniformly by each player is a Nash equilibrium: any bias in d_i only makes it easier for other players to guess g and, e.g., gain custody over the funds escrowed with the TEEs.

[12] Before the cloud, Intel SGX was intended for client devices to, e.g., enforce DRM policies. Hence, anonymity of end users was a priority.

[13] See, e.g., `https://cc-enabling.trustedservices.intel.com/intel-tdx-enabling-guide/02/infrastructure_setup/` (accessed: 11 September 2025).

become a problem.[14] This paper proposes that this be resolved using a *chaining* technique. If a player benefits from an output of a transaction t, their TEE can force them to publish another transaction t' by adding an output to t' that is then spent in t. The two transactions are then "chained," namely, t will not be appended to the blockchain unless t' has been appended first. This is reminiscent of a well-known *child-pays-for-parent* technique from Bitcoin, whereby users incentivize miners to mine a transaction by chaining a high-fee transaction to it.

Commodity Escrow and Proof of Settlement. If player i is to receive a commodity x in a contract, they must host a TEE and *not* be issued the settlement transaction t. Indeed, x must be escrowed with the TEE; otherwise, counterparties could abort after being issued t and leave i without x. Also, if i knew t before it was broadcast, there would be nothing they could do to convince their TEE that the contract was settled and x should be released.[15] Otherwise, i can present t to the TEE thereby convincing it that i read t, signed under the key of the TEE(s), from the blockchain (or the mempool).

Verifiable Delay and Refunds. For a similar reason as above, if player i makes a deposit and hopes to be refunded in case their counterparties default on their obligations and abort the contract, they must host a TEE that will issue a refund transaction redeeming the deposit once convinced that a contract was aborted. A standard way to ensure (eventual) fairness is to design contracts with *timeout* [1, 2, 13, 15, 32], where players can exit the contract after some time has passed without settlement. With TEEs not having any meaningful notion of time [11, 12], their hosts must prove timeout using, e.g., verifiable delay functions or, a more practical approach, Proof-of-Work puzzles [12, 20]. If timeout is expressed in a number of blocks τ, then i can present a subchain of length τ appended to a *checkpoint block* [13] agreed at setup to convince their TEE that the contract was aborted. The Proof-of-Work puzzles function as verifiable (probabilistic) delay with the extra property that an honest player does not need to invest any computational resources to produce the proof: it is instead a by-product of the miners maintaining the blockchain and proves that *at least* the time needed by honest miners to produce τ blocks have elapsed.[16] It does not, importantly, protect against player i's selfish mining: given enough time, i can and *will* produce a subchain of length τ, all while their TEE is isolated from the others and, say, escrowing some asset s to be released in case of contract timeout. It is therefore crucial that s have an *expiration date*: by the time malicious player i falsely convinces their TEE of timeout and s is released, it must have lost its value. This is the case if s is currency, and other players, being

[14] Also, transactions with a large number of outputs "stand out" affording a distinguisher between contract execution and regular blockchain activity.

[15] It would be in the best interest of other players to deny that the contract was settled.

[16] Assume no coalition of players can ever outpace honest miners, which one must assume anyway to discount the threat of blockchain reorganization and for the notion of fairness (or, indeed, *paying* anyone) to be meaningful.

honest, have already redeemed it concurrently: i's TEE may issue a transaction to them that spends s but this transaction, when broadcast, will be rejected as double-spending.

Theorem 1 (Informal). *There exist correct and fair unobservable contracts for sealed-bid auctions, lotteries, and contingent payments.*

Proof. Consider the protocol of Section 1.2. The auctioneer will not lie to their TEE about bids having been included in the blockchain, as this would invalidate the settlement transaction t. Once the auction has been cleared by their TEE, they may still drop messages at egress and prevent the bidders from obtaining t, but, at this point, the auctioneer does not yet know the closing price, so the decision to abort at this point is not informed, and they may as well have not engaged in the contract altogether. The losers will be motivated to broadcast the settlement transaction, and, as soon as one of them does, the winner will be able to obtain the auction subject x escrowed with their TEE.[17] For a lottery, suppose the TEE of player 1 clears the contract (decides the lottery), analogously to the auction contract above, but refuses to accept proofs of timeout after player 1 has confirmed all deposits. If it is 1 who won, they are given a settlement transaction t to broadcast and, if it is another player, then 1 has already lost their deposit (as claims for refund will be denied by their TEE) and are therefore indifferent between settling and aborting.[18] To incentivize 1 to do the former, one may modify the protocol to have 1 stake extra collateral at the start that eventually gets refunded to them in the settlement transaction. (Note that, even without this extra collateral, the fairness property is satisfied, as an aborting player 1 incurs a penalty through burn, while other players stay financially neutral.) For fair exchange (contingent payment), let only the buyer host a TEE which will ask the seller to confirm the deposit of a payment. The seller will escrow the commodity x with the TEE and get a settlement transaction t, which, once broadcast, can be used by the buyer to convince their TEE to release x. □

3 Conclusions and Future Work

The hybrid of TEEs and Zerocash offers an elegant solution that simultaneously addresses fairness, privacy, fungibility, as well as frontrunning and bribery prevention. It also lends itself to a game-theoretic treatment with privacy guarantees of Zerocash ensuring players gain no insight during contract execution reducing many sequential-move games to simultaneous-move ones. Important directions for future work include formal game-theoretic analysis of unobservable contracts and studying their instantiations on top of a cryptocurrency offering more limited privacy guarantees, such as Monero [21, 31].

[17] If the number of bidders is greater than 2, one may even drop the requirement of the auctioneer hosting a TEE.

[18] Since a lottery is a zero-sum game, nobody but the winner will ever want to settle.

References

1. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, : Secure multiparty computations on Bitcoin. Cryptology ePrint Archive **2013**, 784 (2013), https://eprint.iacr.org/2013/784
2. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, : Fair two-party computations via Bitcoin deposits. In: Proceedings of the 2014 Financial Cryptography and Data Security. pp. 105–121. Springer (2014), https://www.ifca.ai/fc14/bitcoin/papers/bitcoin14_submission_10.pdf
3. Austgen, J., Fabrega, A., Kelkar, M., Vilardell, D., Allen, S., Babel, K., Yu, J., Juels, A.: Liquefaction: Privately Liquefying Blockchain Assets . In: 2025 IEEE Symposium on Security and Privacy (SP). pp. 1493–1511. IEEE Computer Society, Los Alamitos, CA, USA (May 2025). https://doi.org/10.1109/SP61157.2025.00156, https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00156
4. Avizheh, S., Haffey, P., Safavi-Naini, R.: Privacy-preserving FairSwap: Fairness and privacy interplay. Proceedings on Privacy Enhancing Technologies **2022**(1), 417–439 (2022). https://doi.org/10.2478/popets-2022-0021
5. Ben Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from Bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474 (2014). https://doi.org/10.1109/SP.2014.36
6. Bentov, I., Ji, Y., Zhang, F., Breidenbach, L., Daian, P., Juels, A.: Tesseract: Real-time cryptocurrency exchange using trusted hardware. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 1521–1538. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363221, https://doi.org/10.1145/3319535.3363221
7. Bentov, I., Kumaresan, R.: How to use Bitcoin to design fair protocols. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014. pp. 421–439. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
8. Brickell, E., Li, J.: Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. IEEE Transactions on Dependable and Secure Computing **9**(3), 345–360 (2012). https://doi.org/10.1109/TDSC.2011.63
9. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. In: Bonneau, J., Heninger, N. (eds.) Financial Cryptography and Data Security. pp. 423–443. Springer International Publishing, Cham (2020)
10. Castejon-Molina, D., Vasilopoulos, D., Moreno-Sanchez, P.: MixBuy: Contingent payment in the presence of coin mixers. Proceedings on Privacy Enhancing Technologies **2025**(1), 671–706 (2025). https://doi.org/10.56553/popets-2025-0036
11. Cheng, R., Zhang, F., Kos, J., He, W., Hynes, N., Johnson, N., Juels, A., Miller, A., Song, D.: Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 185–200. IEEE (2019)
12. Choudhuri, A.R., Green, M., Jain, A., Kaptchuk, G., Miers, I.: Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 719–728. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134092, https://doi.org/10.1145/3133956.3134092

13. Das, P., Eckey, L., Frassetto, T., Gens, D., Hostáková, K., Jauernig, P., Faust, S., Sadeghi, A.R.: FastKitten: Practical smart contracts on Bitcoin. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 801–818. USENIX Association, Santa Clara, CA (Aug 2019), https://www.usenix.org/conference/usenixsecurity19/presentation/das
14. Dotan, M., Tochner, S., Zohar, A., Gilad, Y.: Twilight: A differentially private payment channel network. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 555–570. USENIX Association, Boston, MA (Aug 2022), https://www.usenix.org/conference/usenixsecurity22/presentation/dotan
15. Dziembowski, S., Eckey, L., Faust, S.: FairSwap: How to fairly exchange digital goods. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 967–984. CCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243857, https://doi.org/10.1145/3243734.3243857
16. Erwig, A., Faust, S., Riahi, S., Stöcker, T.: CommiTEE : An efficient and secure commit-chain protocol using tees. In: 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). pp. 429–448 (2023). https://doi.org/10.1109/EuroSP57164.2023.00033
17. Frassetto, T., Jauernig, P., Koisser, D., Kretzler, D., Schlosser, B., Faust, S., Sadeghi, A.: POSE: Practical off-chain smart contract execution. In: 30th Annual Network and Distributed System Security Symposium (NDSS). The Internet Society (2023). https://doi.org/10.14722/ndss.2023.23118
18. Hopwood, D.E., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification, version [NU5] (2024), https://zips.z.cash/protocol/nu5.pdf
19. Juels, A., Kosba, A., Shi, E.: The Ring of Gyges: Investigating the future of criminal smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 283–295. CCS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978362, https://doi.org/10.1145/2976749.2978362
20. Kaptchuk, G., Miers, I., Green, M.: Giving state to the stateless: Augmenting trustworthy computation with ledgers. In: Network and Distributed Systems Security (NDSS) Symposium. Internet Society (2019)
21. Koe, Alonso, K.M., Noether, S.: Zero to Monero: Second edition (2020), https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf
22. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 839–858 (2016). https://doi.org/10.1109/SP.2016.55
23. Kumaresan, R., Moran, T., Bentov, I.: How to use Bitcoin to play decentralized poker. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. p. 195–206. CCS '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813712, https://doi.org/10.1145/2810103.2813712
24. Lee, J., Kim, S., Park, S., Moon, S.M.: RouTEE: Secure, scalable, and efficient off-chain payments using trusted execution environments. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC) (2024), https://www.acsac.org/2024/program/final/s161.html
25. Lind, J., Eyal, I., Pietzuch, P., Sirer, E.G.: Teechan: Payment channels using trusted execution environments (2017), https://arxiv.org/abs/1612.07766

26. Lind, J., Naor, O., Eyal, I., Kelbert, F., Sirer, E.G., Pietzuch, P.: Teechain: a secure payment network with asynchronous blockchain access. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles. p. 63–79. SOSP '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341301.3359627, https://doi.org/10.1145/3341301.3359627
27. Möser, M., Böhme, R., Breuker, D.: Towards risk scoring of Bitcoin transactions. In: Financial Cryptography and Data Security. pp. 16–32. Springer (2014)
28. Möser, M., Narayanan, A.: Effective cryptocurrency regulation through blacklisting (2019), https://maltemoeser.de/paper/blacklisting-regulation.pdf, manuscript
29. Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology, pp. 1–9. Springer Berlin Heidelberg, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44702-4_1
30. PwC: PwC global crypto regulation report 2025 (2025), https://legal.pwc.de/en/services/pwc-legals-eu-regulatory-compliance-operations/pwcs-global-crypto-regulation-report
31. van Saberhagen, N.: CryptoNote v2.0 (2013), https://www.getmonero.org/resources/research-lab/pubs/cryptonote-whitepaper.pdf
32. Thyagarajan, S.A., Malavolta, G., Moreno-Sanchez, P.: Universal atomic swaps: Secure exchange of coins across all blockchains. In: IEEE Symposium on Security and Privacy (SP). IEEE (2022)
33. Thyagarajan, S.A., Malavolta, G., Schmid, F., Schröder, D.: Verifiable timed linkable ring signatures for scalable payments for Monero. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) Computer Security – ESORICS 2022. pp. 467–486. Springer Nature Switzerland, Cham (2022)
34. Tramèr, F., Zhang, F., Lin, H., Hubaux, J.P., Juels, A., Shi, E.: Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 19–34 (2017). https://doi.org/10.1109/EuroSP.2017.28
35. Wüst, K., Matetic, S., Schneider, M., Miers, I., Kostiainen, K., Čapkun, S.: ZLiTE: Lightweight clients for shielded Zcash transactions using trusted execution. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security. pp. 179–198. Springer International Publishing, Cham (2019)