Lecture 19: Privacy for Smart Contracts

Principles of Blockchains, University of Illinois,
Professor: Pramod Viswanath
Scribe: Soubhik Deb

April 1, 2021

Abstract

The previous lecture showed how to bring privacy to the data in the blockchain. Using cryptographic tools of zero knowledge, privacy was afforded to transactions, and yet readily allowing for verification of the validity of the transaction. In this lecture we study the problem of how to bring such strong privacy to the data in a more general state management system: an account based system managing smart contracts (e.g., Ethereum). The privacy construction of the previous lecture was specific to UTXO and it is especially challenging to generalize it to the smart contract platform; this is the focus of this lecture.

Introduction

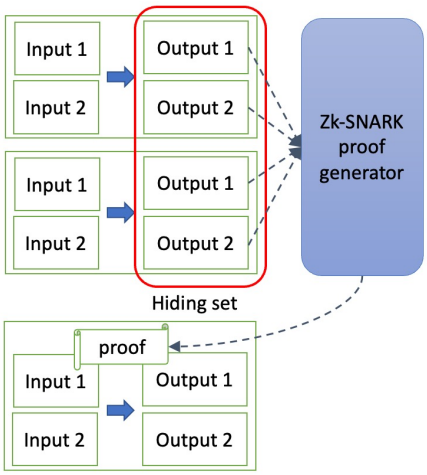


Figure 1: UTXO model was introduced in Bitcoin. However, it leads to information leakage. Zcash employs zk-SNARKs to shield the information on the payment's origin, destination, or amount in a payment transaction.

In the previous lecture, we learnt how Zcash can be used to provide privacy in an UTXO state management system. The UTXO model of state/ledger management system consists of transactions (which contain a list of inputs and outputs): outputs specify the recipients' public keys and inputs refer to previous output. However, simply using the UTXO model leads to information leakage via which one can link transactions into a graph. Zcash provides an architectural framework to remove

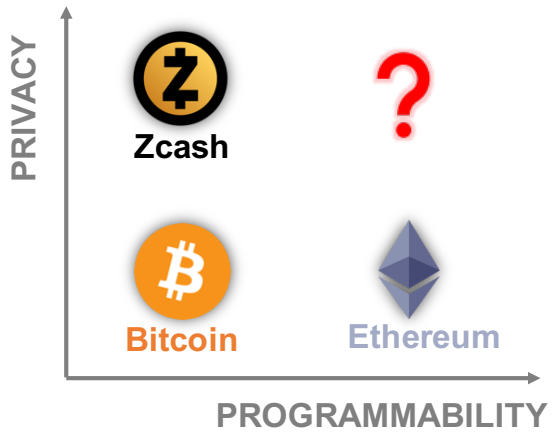


Figure 2: Bitcoin offers neither privacy nor programmability. It avails simple scripts that can be used for writing only the payment transactions. On the other hand, Zcash offers complete privacy but no programmability. In contrast to that, Ethereum offers high degree of programmability but no privacy. Currently, there is no deployed blockchain ecosystem that offers both privacy and programmability.

this information leakage by ensuring that a payment transaction doesn't reveal the payment's origin, destination, or the amount; see Figure 1 for an illustration. The simple-state environment in the UTXO model makes it difficult, if not impossible, for programming on-chain complex computation that requires some state information or requires multiple parties (see Figure 2). This lack of programmability in UTXO model leads to the introduction of account-based model in Ethereum. To facilitate this programmability in Ethereum, applications involving complex computations are defined in smart contracts which are then executed in Ethereum Virtual Machine (EVM). This feature has enabled numerous financial instruments like decentralized exchanges, arbitrage, flash loans in decentralized finance (DeFi), tokenization and games in Ethereum. However, in contemporary account-based blockchain ecosystems, all details inside a transaction are public and thus, afford no privacy. For example, suppose an account buys Ether from a cryptocurrency exchange, like Coinbase, and uses it to execute an arbitrage opportunity (see Figure 3). Next the account converts the Ether to a fiat currency such as USD. Since all the details inside the transactions in Ethereum are public, therefore, the real-world identity of the entity that obtained the arbitrage opportunity is known to the cryptocurrency exchange.

Having studied Zcash in previous lecture, a natural question is whether we can use zk-SNARKs to shield the internal details of the transactions in an account-based ecosystem like Ethereum. A successful design will then provide *both* privacy and programmability. Before answering this question, we first need to delineate what is meant by the privacy of a transaction. Internally, a transaction is composed of different components: sender address, recipient address, value of the transaction, tokens exchanged, liquidity pool whose service is availed, timestamps, etc. Therefore, privacy of a transaction can imply that a certain combination of above components like senders' and recipients' addresses is shielded whereas rest are visible to the public (see Figure 4). In this lecture, we explore a construction that guarantees the aforementioned level of privacy, termed as **user privacy**. This construction would still reveal the information regarding the value of the transaction and tokens exchanged but shield the identities of the accounts involved in the transaction.

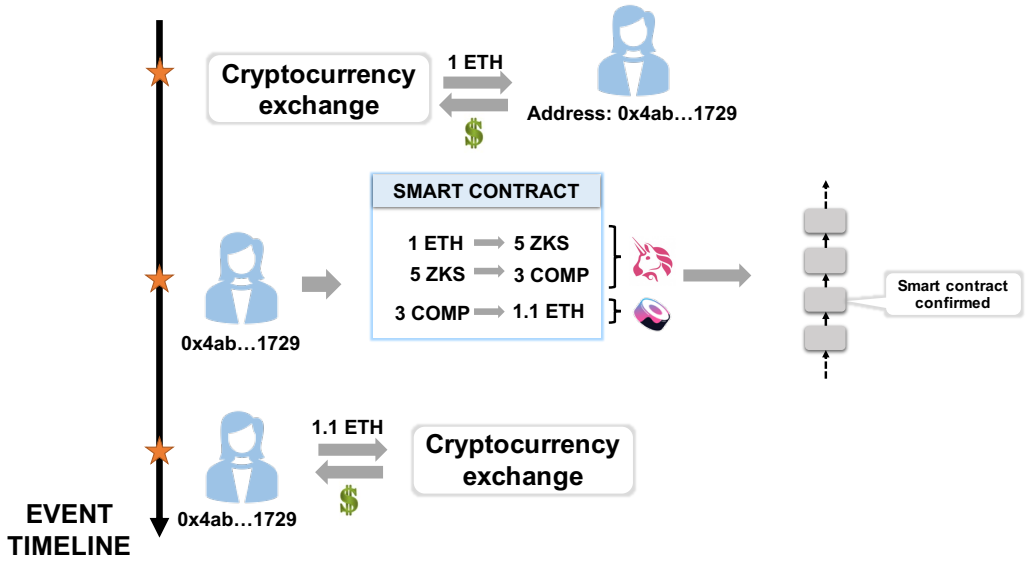


Figure 3: An entity buys Ether from a cryptocurrency exchange such as Coinbase. Then, using smart contract, it successfully executes an arbitrage opportunity that involves exchanging Ether for ZKS and ZKS for COMP in Uniswap and then, COMP for Ether in Sushiswap. After that, the entity exchanges the Ether for USD in Coinbase. However, Coinbase has full access to the real-world identity of this entity (through the "know your customer" (KYC) process).

Key Idea

The key idea to achieve user privacy in an account-based blockchain ecosystem is to use the privacy-preserving blockchain like Zcash as an anchor. Specifically, a participating node must have an account in Ethereum to engage in payment transactions, trading activities in DeFi, etc. There are two possible ways an account can avail of to transact in a privacy-preserving manner:

- the node can go to a cryptoexchange and exchange fiat for the token in the privacy-preserving blockchain, say Zcash. Then, the node can employ the hiding set in Zcash to insert liquidity in the account in Ethereum. Using the hiding set removes any linkage between the account in Zcash that obtained token from the cryptoexchange and the account in Ethereum.
- the node can avail of a flash loan from liquidity pools like Aave in one account in Ethereum. Then, the node can move this asset to an account in Zcash and then, use the *hiding set* in Zcash to again move this asset back to another account in Ethereum.

In both cases, the trades engaged on Ethereum are visible to either the cryptoexchange or the liquidity pool. However, these entities have no way to link the trades with the accounts that were used by the node for obtaining liquidity. In fact, if a node doesn't want other nodes to link the transactions done from its accounts, the node can use the hiding set in Zcash to break those links in the same manner as above (see Figure 5). Note that other parties (cryptoexchanges, liquidity pools) can observe the transactions conducted in Ethereum but have no way to determine the actual identity of the node that participated in these transactions. This provides user privacy.

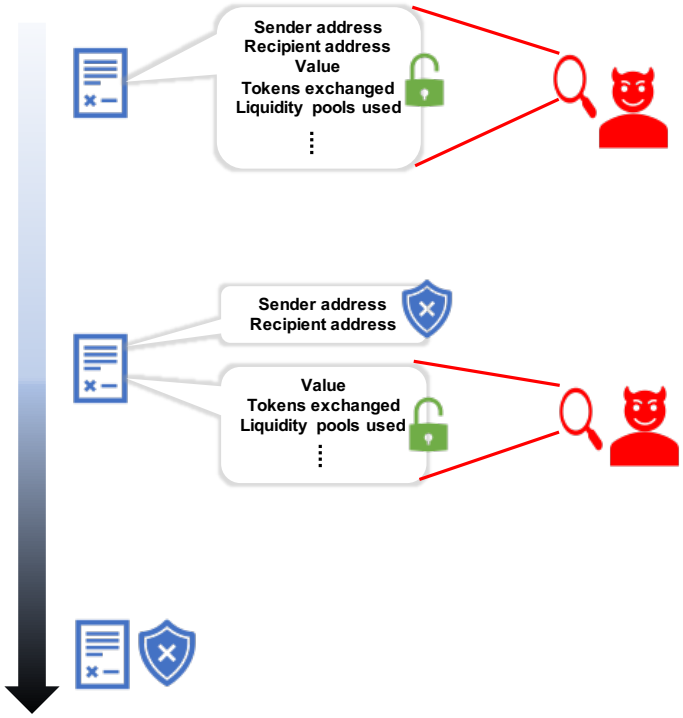


Figure 4: In the first case, every detail about the transaction is revealed to the public. This is the current state of privacy in Ethereum and other smart-contract based blockchain ecosystems. A limited amount of privacy can be achieved by shielding only the senders' address and recipients' address in the transaction. Complete privacy is obtained when all the components of a transaction are shielded yet this transaction is still verifiable.

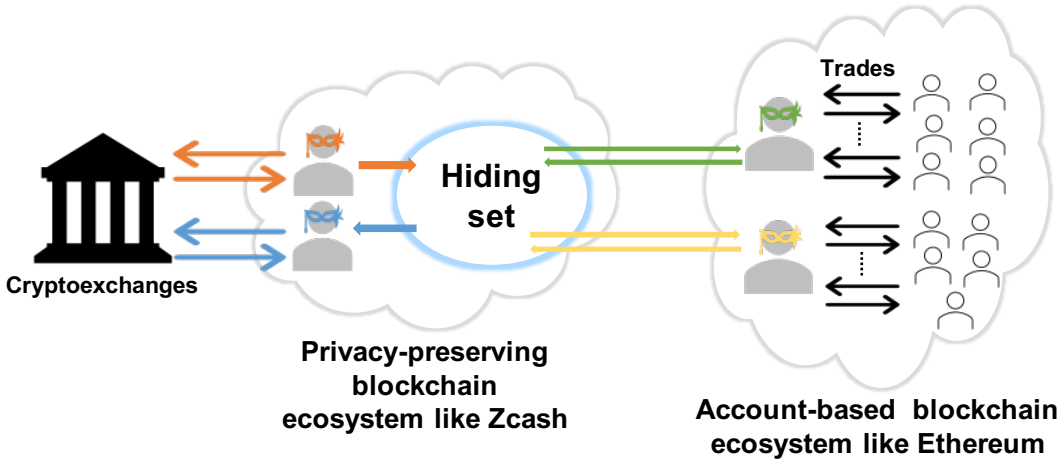


Figure 5: A node makes an account in a privacy-preserving blockchain like Zcash (with limited programmability) and exchanges fiat currency for some tokens. Then, the node utilizes the hiding set to create an account in the account-based ecosystem. For example, in case of Ethereum, hiding set can be utilized to open a contract account for a smart contract. Using the hiding set, masks the linking between the account in Zcash and the contract account in Ethereum. Now, the node can use this smart contract to engage in various trades like arbitrage, swaps, payment transactions, etc., in Ethereum. When the node wants to convert the tokens in its possession in Ethereum back to fiat currency like USD, it can first use the hiding set to transfer its assets to another account in Zcash. Note that using hiding set breaks any link between the account in Zcash and the contract account in Ethereum. Then, the node can exchange the tokens for fiat currency in a cryptoexchange. A similar strategy can be employed by the node for masking the links between different transactions done from its accounts in Ethereum.

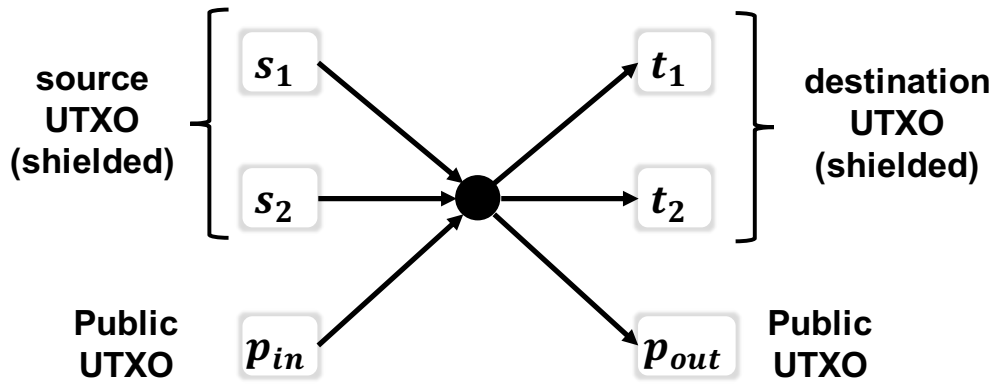
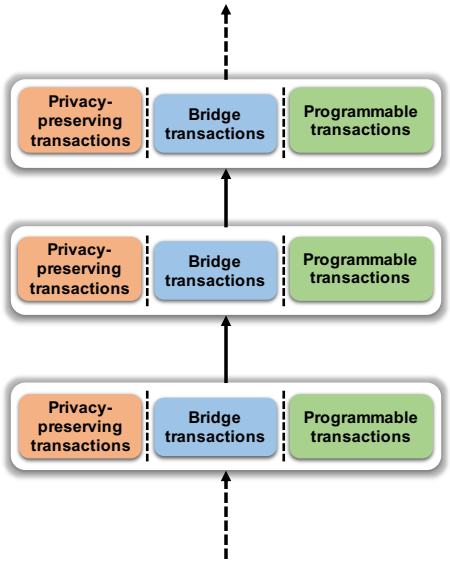


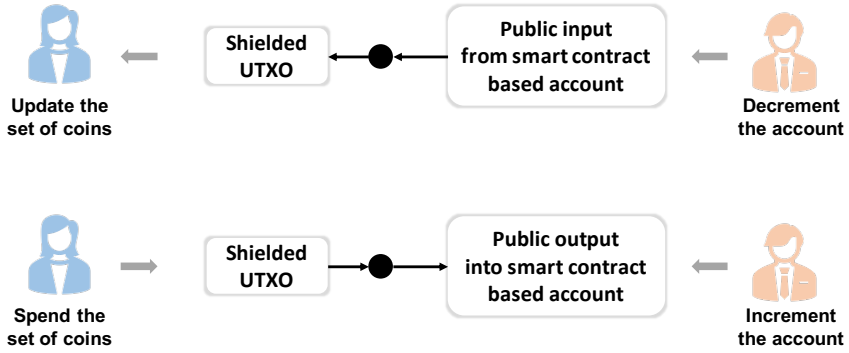
Figure 6: In a privacy-preserving blockchain like Zcash, some of the input and output UTXOs in a transaction can be shielded and some of the other input and output UTXOs can be public.

Construction

In Zcash, the addresses and values of some of the input UTXOs and output UTXOs in a transaction can be "shielded" and thus, their privacy is maintained. However, in the same transactions, some of the other input and output UTXOs can be public. We will employ this feature for constructing a *privacy bridge* to smart contracts (see Figure 6).



(a) Each block can contain three types of transactions - privacy-preserving transactions, bridging transactions and programmable transactions.



(b) Pictorial illustration of the two types of bridging transactions: the first one decreases the assets/tokens in the account of the smart contract and atomically outputs shielded UTXOs in the privacy-preserving side, the second one involves spending unspent UTXOs in the privacy -preserving side and outputting a public output that increments the account of the smart contract in the programmability-side. Both these transactions must be executed atomically.

Figure 7: A privacy bridge based blockchain ecosystem.

The privacy bridge comprises of a blockchain that has a privacy-preserving side and a programmability-side. A node has an address in the privacy-preserving side and follows a UTXO model. Transactions within the privacy-preserving side, that is, all senders and recipients of the transaction being addresses in the UTXO model of the privacy-preserving side, are guaranteed privacy. We refer to such transactions as "privacy-preserving transactions" (see Figure 7a). On the other hand, if a node

wants to employ programmability for executing complex transactions, then, it can transfer funds from the privacy-preserving side to smart contracts in the programmability-side using "bridging transactions". Now, within the programmability-side, the node can use smart contracts to create "programmable transactions" for executing complex strategies. Note that these programmable transactions do not hide details such as the addresses of the smart contracts that are in its senders or recipients list, the tokens that are exchanged or the value, etc. There are two types of bridging transactions (see Figure 7b):

- the first type decrements the token in the account in the programmability-side and outputs a shielded UTXO in the privacy-preserving side which updates the coins held by the transaction. Note that the source account of the privacy-preserving transaction is kept private.
- the second type involves spending UTXO in the privacy-preserving side and incrementing the tokens in the smart contract on the programmability-side that is specified in the output of the bridging transaction.

These bridging transactions are implemented in the same way as described at the beginning of this section (see Figure 6). Therefore, even if the transactions executed by the smart account in the programmability-side are all public, the accounts in the privacy-preserving side to which the assets have been transferred to are not publicly revealed by the bridging transaction. Note that the bridging transactions are executed atomically. Consequently, in each block, miners can include three types of transactions: privacy-preserving transactions, bridging transactions and programmable transactions (see Figure 7a).

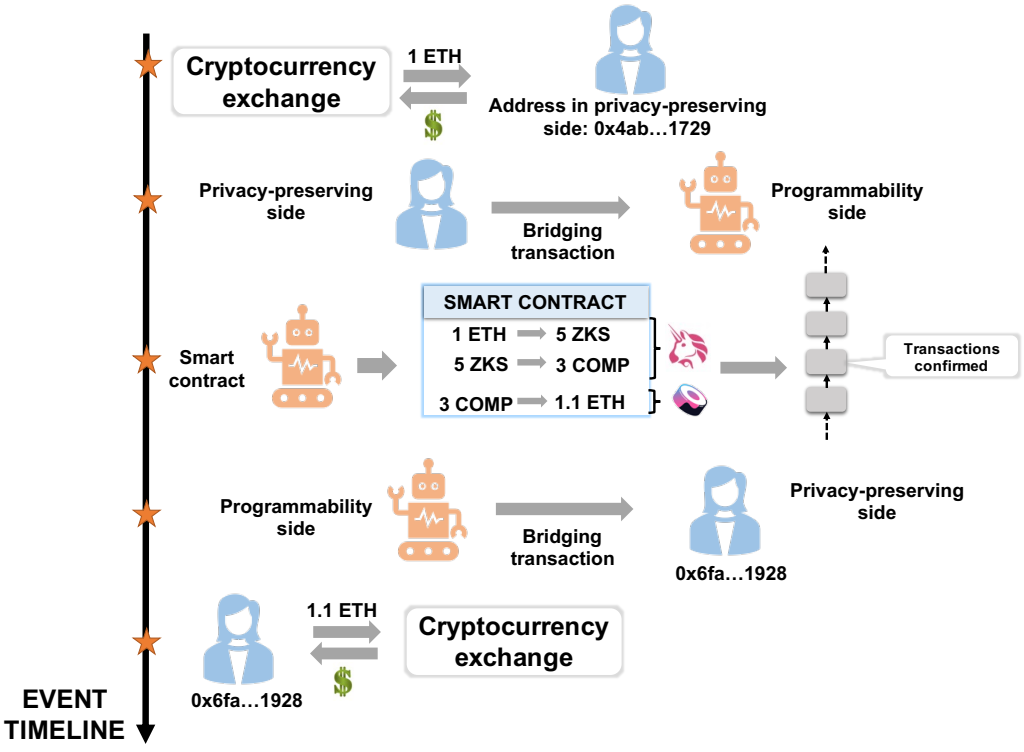


Figure 8: Privacy bridge preserves the privacy of the addresses in the privacy-preserving side that funded and received assets from the smart contract in the programmability-side.

In summary, the smart contract can execute any arbitrage strategy in the programmability side,

but the bridging transactions do not reveal the addresses of the account in the privacy-preserving side that deposited funds in the smart contract or the addresses in the privacy-preserving side to which the smart contract transferred its assets (see Figure 8). Hence, nobody, not even the cryptocurrency exchanges like Coinbase, can link the accounts in the privacy-preserving side to the transactions in the programmability-side.

References

In this lecture we saw a simple technique to harness the Zcash architecture on UTXOs to smart contract platforms. This bridging technique brings user privacy, but the actual contents of the transaction (such as the amounts) are public. A more general technique would shield such data; even more generally, perhaps all data on the blockchain could be encrypted and yet be validated by the participants. Such a broad goal is tantamount to true homomorphic encryption, which refers to computing on encrypted data without the need for any secret key, a grand goal of cryptography. Several recent works have attempted to build such general privacy preserving architectures: zexe, Zether, zkay, kachina, and this is an active area of research and development.

zexe is a ledger-based system where users can execute offline computations while hiding all information about the computations and subsequently produce transactions, attesting to the correctness of these computations which can be validated in constant time. Zether is a smart contract in Ethereum that keeps the account balances encrypted and exposes methods to deposit, transfer and withdraw funds to/from accounts through cryptographic proofs. zkay is a language formalism which introduces privacy types for Solidity. kachina is a framework for deploying privacy-preserving smart contracts under the Universal Composition (UC) model.