Private Smart Contracts

Steven Goldfeder
Princeton University
stevenag@cs.princeton.edu

Arvind Narayanan
Princeton University
arvindn@cs.princeton.edu

1. INTRODUCTION

There are two main directions in which successors to Bitcoin differentiate themselves – increased functionality and increased privacy. Ethereum [4] is a prime example of the former, whereas Zcash [2] and Monero focus on the latter. There has been much interest recently in merging these two paths to create a unified system with the privacy guarantees of Zcash and the expressability of Ethereum.

Smart contract, originally proposed by Nick Szabo [3] are mechanisms by which people can agree to a set of rules that will be automatically enforced. When built on top of a cryptocurrency, smart contracts can govern monetary relationships between people and also pay out rewards or enforce penalties. Ethereum allows users to specify smart contracts, but with the caveat that all of the code for these contracts must be public.

While the vision of private smart contracts is easy to connect to, subtle issues arise. Firstly, there's a good reason why Ethereum compromised on privacy and Zcash compromised on functionality: there are major technical hurdles to combining them. Perhaps more importantly, though, upon closer examination it's not even clear what properties we would want from a private smart contract system – what exactly would we like to keep private and whom are we hiding it from?

In this talk, we will categorize various privacy modes for private smart contracts. We will also differentiate between several computational models for smart contracts and discuss the technical feasibility of each model. We will present recent and ongoing work–both ours and that of others–towards building private smart contracts, and we will lay out the technical barriers that we still need to cross.

We intend to provide the necessary background in the first part of the talk to stimulate an engaging discussion in the latter. We will conclude by opening up a conversation on the applications and tradeoffs of private smart contracts. What will they be useful for? In particular, what novel privacy enhancing technologies can we build using private smart contracts as a primitive?

Lastly, we will discuss the tradeoffs of private smart contracts and the difficulty they pose for law enforcement. As the PETS community has faced these issues before in the context of TOR, they might have useful experience in thinking about these questions, and we welcome a candid discussion on both the benefits and tradeoffs of this technology.

2. PRIVACY MODEL

To motivate the various privacy concerns, we begin with an example. Alice loves Sudoku puzzles, but she's stumped on a particularly difficult one. She decides to buy a solution from Bob, and they set up a smart contract to facilitate the payment.

Alice encodes the puzzle in a smart contract. She deposits some money to the contract, and the code specifies that if Bob provides a solution, he can collect the reward. When Bob proposes a solution, the contract verifies its correctness before paying out.

In this example, where might one desire privacy? The most obvious place is hiding the Sudoku solution. Alice is paying Bob for the solution, but if they use Ethereum, Bob must post it publicly for the whole world to see. Is there a way to design a system such that Alice learns the solution but the rest of the world does not?

A second less intuitive place in which privacy might also be important is hiding the code itself. If the code is public, everyone can see that Alice is attempting to buy a Sudoku solution from Bob. Even absent the actual solution, this still reveals a lot of information about the nature of the transaction.

This example motivates the following privacy categories:
Input hiding. A smart contract is *input hiding* if one can provide inputs to the contract that will be kept private.
Program hiding. A smart contract is *program hiding* if the code running the contract is not publicly revealed.

We note a key difference between these two notions of privacy. Program hiding only makes sense in the context of *external* parties. The people that are themselves agreeing to the contract (i.e. the *internal* parties), however, must know the code that governs their relationship.

Input hiding is well-defined both with respect to external parties as well as internally. In our example, Bob wants Alice to learn the solution (no internal privacy), but does not want anyone else to (external privacy). But say that we modify the example so that Bob doesn't want to sell Alice the Sudoku solution, but just wants to prove to her that he knows it. In this case, Bob would want the contract to be internally input hiding so that even Alice, who is a party to the agreement, does not learn his inputs.

3. COMPUTATIONAL MODEL

There are several different possible computational models for private smart contracts, and the technical solutions differ widely.

Contracts can be *stateful* or *stateless*. In a stateless contract, a set of conditions are described such that anyone meeting these conditions can collect the money. The Sudoku

example represents a stateless contract. Bob just needs to present the solution to claim the reward.

A stateful contract on the other hand can govern more complex relationships that involve multiple rounds of interaction. An example of a stateful smart contract is a bet on a game of chess. Alice and Bob can encode the rules of chess in a contract and specify that the winner collects a reward. Whenever Alice and Bob make a move, the contract verifies that it was a legal move and updates its state to reflect the new board.

Moreover, contracts may or may not perform computation on joint inputs. In the Sudoku example, only Bob has an input whereas Alice has no input. But consider a contract representing an election or an auction. In these examples, the contract needs to compute a joint result over all the inputs to determine who the winner is.

4. TECHNICAL PATHS AND CHALLENGES

Depending on which mode of privacy and computational model one wants, there are several technical tools that may prove helpful for designing smart contract systems. We survey the approaches.

4.1 Zero Knowledge proofs

Zero knowledge proofs enable parties to prove properties about data they hold without leaking information about the data. This makes them an ideal candidate for certain types of private smart contracts –particularly where *input hiding* is desired.

Unfortunately, zero knowledge proof systems have severe limitations, and the technology is not mature enough yet to realize private smart contracts. Firstly, the most efficient proof systems require a costly trusted setup. Even worse, the trusted setup is specific to the code, so each contract would require its own trusted setup. Secondly, even the most efficient zero knowledge protocols are prohibitively expensive for the prover. Thirdly, zero knowledge proofs generally require code to be written in circuits. This represents a difficult and unfamiliar development for most people.

4.2 Multiparty computation

Even if zero-knowledge proofs did not have the limitations mentioned above, they would only be useful when the contract does not perform joint computation over inputs from different users. If such computation is required, then we would need to use secure multiparty computation (MPC). Auctions and elections are prime examples of contracts which require computation on private inputs.

MPC has come a long way and can be practical for many applications. The key setback of using secure multiparty computation is that it requires multiple rounds of interaction. If some users participate in the first round but do not return for the latter rounds, they could potentially ruin the result of the computation.

4.3 Off-chain smart contracts

An off-chain protocol is one in which parties engage with each other off-chain and only use the blockchain as a dispute resolution layer. These protocol are designed such that absent a dispute, there is little or no data posted to the blockchain. But participants are guaranteed that if something goes wrong, they can resolve it using the blockchain.

While much of the recent surge in off-chain protocol development stems from the desire to increase scalability of blockchains, there are also significant privacy benefits. We describe Arbitrum, a system we developed for private off-chain contracts. Arbitrum, and other off-chain system are *program hiding* so the outside world does not learn the contract's code. However, Arbitrum does not have any mechanism for internal parties of the contracts to hide their inputs from one another.

4.4 Using trusted hardware

Another approach to building private smart contracts is using trusted hardware such as Intel SGX [1]. These systems rely on trusted hardware to certify results of computations.

Smart contract systems built using trusted hardware can be quite efficient. But this model is not without controversy: Can we really put our trust in hardware that users have physical access to? Are we putting too much trust in the manufacturers of this hardware? What would happen if say, Intel, were malicious?

5. APPLICATIONS AND TRADEOFFS

Two closely related questions are what will smart contracts be used for and how can we mitigate nefarious uses. Like all privacy enhancing technologies, private smart contracts is a double-edged sword. On the one hand, it has the potential to enable fascinating applications, but it could also be used by criminals for engaging in nefarious activity without oversight from law enforcement.

Hopefully, the talk will give the audience the tools and necessary background to evaluate potential research opportunities associated with private smart contracts. Discussion topics include:

1. What other privacy enhancing technologies can we build using private smart contracts as a primitive?
2. How can private smart contracts be combined and integrated with existing PETS?
3. What challenges will law enforcement face and how should they adapt?
4. What insights can we learn from TOR about the interplay of positive and negative applications of private smart contracts?

6. REFERENCES

[1] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *arXiv preprint arXiv:1804.05141*, 2018.

[2] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014.

[3] N. Szabo. Smart contracts. *Unpublished manuscript*, 1994.

[4] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151:1–32, 2014.