Absentia: Submit-and-go secure function evaluation

Nan Yang and Jeremy Clark

Concordia University
nan@encs.concordia.ca
j.clark@concordia.ca

Abstract. This paper describes a blockchain-based approach for secure function evaluation in the setting where multiple participants have private inputs that no other individual should learn. The emphasis of Absentia is reducing the participants' work to a bare minimum, where they can effectively have the computation performed in their absence and they can trust the result. Our use of Ethereum is not necessary — the underlying SFE protocol, based on Mix and Match, can operate perfectly well without a blockchain. However Ethereum does add value in the three following ways: (1) SFE requires a secure bulletin board and blockchains are the most widely deployed data structure with bulletin board properties (immutability and non-equivocation). (2) Ethereum provides a built-in mechanism to financially compensate participants for the work they perform. (3) A publicly verifiable SFE protocol can be checked by the Ethereum network itself, absolving the user (and participants in the computation) of having to verify that the function was executed correctly.

1 Introductory Remarks

Consider the now classic Millionaire's problem where Alice and Bob both have a secret value that represents their wealth, $[\![a]\!]$ and $[\![b]\!]$ respectively. They wish to learn which of them is wealthier without disclosing a or b to each other.

The approach of the literature is to have Alice and Bob run a joint protocol on their private inputs that requires them to coordinate and complete a few rounds of communication. In general, it is possible to have n parties jointly compute a deterministic function such that each party's input remains private. This is multi-party computation (MPC).

In general, MPC requires that all parties involved be online, since they must send many message between them. A Byzantine fault, such as a party going offline, brings the MPC to a halt until it is resolved.

Our goal in this paper is to explore a model where Alice can leave an encryption of a, $[\![a]\!]$, in a public place, where Bob can similarly leave $[\![b]\!]$. We defer a discussion of what encryption key is used. Later they can return to receive $[\![z]\!]_a$ and $[\![z]\!]_b$ where z is the output of their function, and is encrypted under their individual keys. When they fetch this value, they are confident z is correct.

Between leaving their private inputs and retrieving the output, the computation itself is performed — not by Alice and Bob, but by some workers they have outsourced the computation to. These workers should not learn a, b, z or any intermediary values of the computation under some reasonable assumption (we rely on non-collusion between them).

1.1 Threat Model

Security enters the discussion of function evaluation in two basic ways. The first is that the computation should be performed correctly. The second is that the function inputs (and potentially outputs) should be kept confidential from some participants. Without these security constraints, we are in the domain of distributed computing.

Verifiable computing studies the issue of a computation being executed correctly. If Alice performs the computation herself, she does not need a correctness guarantee but there are two basic reasons she might not compute it herself. The first is that it might be cheaper to outsource the computation to the cloud and verify a proof of correctness than to compute it (with or without privacy of her inputs/outputs from the cloud). The second is that the computation might involve the private input of someone else, for example Bob, and so Alice has no way to perform it herself (and likewise Bob does not either if Alice has a private input).

We concern ourselves with the latter case. Alice and Bob can run a joint protocol, typically one based on garbled circuits for a two-party computation (2PC) or one based on verifiable secret sharing for the multi-party case (MPC). Alternatively a protocol might be based on homomorphic encryption, whether fully homomorphic or partially homomorphic. The most modern approach is to blend techniques, for example using FHE for a pre-computation round and MPC for online execution of the circuit (SPDZ).

If Alice and Bob care not to evaluate the correctness of the computation (they trust each other or someone they delegated the computation to), we can operate in the honest-but-curious model largely using the protocols established in the 1970s and 80s with a few modernizations. However when they care about correctness, a trickier malicious model must be considered that brings additional complexity to the protocols. Typically, a secure function evaluation (SFE) that is secure in the malicious model involves interactive proofs of correctness between the participants that are verifiable by the participants (and only the participants). A higher-reaching goal is public verifiability, where proofs are written to a public transcript and can be used by anyone to verify the computation was performed correctly.

Public verifiability enables two interesting properties. The first is that Alice and Bob can delegate the computation to someone else to compute and they do not have to trust the delegatees for correctness (they do have to trust them not to collude and reveal their private inputs). Even stronger, someone beyond Alice, Bob, and the delegatees can verify the correctness.

While it is not framed as publicly verifiable SFE, the Mix and Match protocol from Jakobsson and Juels provides this property. It works in the two-party and multi-party case (the constraints are on the number of delegatees not private inputs).[1]

1.2 Deployability

Multi users with secrets

Users to not have to be online at the same time (excludes GC, MPC, etc) - one round of communication to submit inputs

they will have to submit somehow: minimize what they need to know at submission time. encrypt it but need to know a key - to know key, need to know computational trustees

minimize privacy assumptions around submission - submit to a public broadcast channel (as opposed to private channels with trustees)

at the end, they have to receive the output - public channel, asynchronous, one round of communication,

at the end, they have to check a proof - great to delegate proof to others so they don't have to check (public verifiability vs normal malicious model)

1.3 Mix and Match

There is only one SFE protocol that seems to work this way out of the box: Mix and Match.

Users, Trustees, Miners

1.4 Trustees

Why did mix and match fail? Who are the trustees going to be? IBM, Microsoft, Google in 2-out-of-3 is great for papers but not realistic

Anyone can be with some human verification on top - Tor model

Plus, unlike Tor, we can pay them

1.5 Coordination

Bulletin board: append-only (immutable) broadcast (non-equivocating) - blockchain
Payments: blockchain Verification for free: blockchain

Which blockchain? Ethereum Other related projects: Enigma

2 Future Work

Applying this to one-time programs and short-lived signatures

Ephemeral storage

Sending a black box

[1] For what it is worth, it works in the one-party case as well but without further optimization, the proofs are far more expensive than the computation being performed. However since it is publicly verifiable, verification could also be delegated.

References