INPUT | OUTPUT

IOHK | BLOG

A solution for scalable randomness

The SCRAPE protocol can handle far more users than previous solutions and can be applied to blockchain protocols, electronic voting, anonymous instant messaging and more

JUNE 06, 2017 3 MIN READ

It is well known that randomness is a fundamental resource for secure cryptographic schemes. All common encryption and authentication schemes are only as secure as their randomness sources are good. Using a poor randomness source can basically render insecure every cryptographic scheme that would otherwise be secure.

In many cases, it is sufficient to have a trustworthy local source of fresh randomness (e.g. your operating system's randomness pool or a Random Number Generator (RNG) embedded in your CPU). However, in many distributed applications, it is necessary to use a publicly accessible source of randomness that allows users (and external observers) to make sure they are all getting the same random values. With such applications in mind, in 1983 Michael Rabin introduced the concept of a randomness beacon, a publicly accessible entity that provides a periodically refreshed random value to its clients. More recently, NIST, the National Institute of Standards and Technology in the United States, started offering a randomness beacon service that derives its

randomness from quantum physical processes.

When users are willing to trust a centralised randomness beacon they can simply choose one of them to perform this role or use a publicly available beacon, such as the one from NIST. However, standard randomness beacons provide no means for users to verify that the output values are indeed random and not skewed in a way that gives an advantage to an attacker.

As we know, cryptographic schemes have suffered both from poor implementations and from governmental agencies (e.g. the NSA) that actively work to undermine existing and future standards. In this context, a centralised randomness beacon could be providing bad randomness because of simple implementation mistake or in order to do the bidding of third party attacker who has undermined the central randomness generator or simply presented the people running the service with a subpoena that compels them to do so.

Using a coin tossing protocol (a concept also introduced by Rabin in 1981 [insert link to) is an obvious approach to (partially) solve this problem. Coin tossing allows several users to come together and generate an output that is guaranteed to be uniformly random without having to trust each other or any third party.

In a coin tossing protocol, good randomness is obtained if at least one of the parties is honest. If you run a coin tossing among many servers, somebody who trusts at least one of the servers can be assured that the output is truly random. In many applications (such as cryptocurrencies) we assume that at least half of all parties are honest by design.

Even though using coin tossing protocols solves the trust problem, it still does not make it any easier to ensure that all users will receive the random output. For example, a user with a faulty internet connection or an attacker who wants to mount a Denial-of-Service attack against the protocol (and consequently the application using the final randomness) can simply abort (i.e. stop sending protocol messages) before the output is obtained, keeping all the other users from receiving randomness. In fact, a malicious user can do even worse, he can execute the protocol up to the point where he learns the random output but still not send the final message that would enable the other users to also learn this output.

Tal Rabin and Ben-Or introduced a classical method for making sure all users receive the proper protocol output – assuming at least a majority of them is honest – in a seminal paper in 1989. The main idea is to use "secret sharing" to split each user's input into "shares" that do not reveal any information about the input by themselves but allow for total recovery of the input if a sufficient number of them are put together. If each user is given a share of every other user's input – such that the inputs can be recovered if more than 50% of the shares are pooled – malicious users still cannot learn anything because we assume that more than 50% of the users are honest. However, if a malicious user aborts the protocol at some point, the honest users can pool their shares to recover the malicious user's input and finish the protocol by themselves.

This general approach can be used to make sure that all the users running a coin tossing protocol will receive the final random value. However, this still does not allow users and external observers to make sure that the output is being generated correctly. In order to do that, the users must use a kind of secret sharing that is publicly verifiable, meaning that anyone can make sure that the shares of inputs are correctly generated. This prevents malicious users from posting invalid shares and then aborting in such a way that the honest users cannot reconstruct their inputs. This approach has been used for generating publicly verifiable randomness in the Ouroboros Proof-of-Stake based, permissionless consensus protocol and for constructing a stand-alone randomness beacon in a paper by Syta et al. that will be presented at the IEEE Symposium on Security and Privacy 2017.

The main ingredient in building randomness beacons through this approach is Publicly Verifiable Secret Sharing (PVSS), a type of secret sharing scheme introduced by Stadler that allows users to split their inputs in shares whose validity can be promptly verified by any third party. Using such schemes, it is possible to construct coin tossing protocols with Guaranteed Output Delivery through the approach of Rabin and Ben-or, meaning that every user is guaranteed to receive the final randomness produced by the protocol. However, the PVSS scheme from Schoenmakers, which until recently was the most efficient PVSS scheme, still requires a number of modular exponentiations that is quadratic in the number of users, meaning that if there are n users running the protocol, verifying the n shares produced by one of them requires O(n^2) exponentiations. Note that this poses serious scalability issues since this quadratic overhead means that the number of expensive modular exponentiations required for verifying n shares grows quite a lot given even a small increase in the number of users.

In a recent joint paper with Ignacio Cascudo, SCRAPE: Scalable Randomness Attested by Public Entities that will presented at ACNS 2017 in Japan this July, we have solved this scalability problem by introducing a PVSS scheme that only requires O(n) modular exponentiations to verify n shares (5n modular exponentiations to be precise) while maintaining efficiency of all other protocol components. The main idea behind our protocol is to use a cheap information theoretical trick to verify the validity of shares instead of performing expensive modular exponentiations. As a result, our protocol requires only 5n modular exponentiations in the verification phase and 4n modular exponentiations in the distribution phase (where shares are generated), while the protocol by Schoenmakers requires 4n+(n^2)/2 exponentiations for verification and 4n+t exponentiations for distribution, where n is the number of users/shares and when n/2 shares are required for reconstructing the secret. Our efficient PVSS protocol can be used to improve the randomness generation component of Ouroboros and the recent result by Syta et al.