The Theory Behind Blockchains (Spring 19)
Recitation 8

Eliad Tsfadia

1 The RSW time-lock puzzle (reminder from class)

Rivest, Shamir, and Wagner [RSW96] introduced the concept of a time-lock puzzle, and proposed the following elegant construction:
The puzzle is a tuple (N, x, T) where $N = p \cdot q$ is an RSA modulus, $x \in \mathbb{Z}_N^*$ is random and $T \in \mathbb{N}$ is a time parameter.
The solution of the puzzle is $y = x^{2^T} \mod N$. A party that knows p and q (and thus knowns the group order $\phi(N) = (p-1)(q-1)$) can just compute $e = 2^T \mod \phi(N)$ using $O(\log T)$ steps and then compute $y = x^e \mod N$. However, it is conjectured that a party that does not know p and q needs at least T sequential steps in order to compute $y = x^{2^T} \mod N$:

$$x \to x^2 \to x^{2^2} \to x^{2^3} \to \ldots \to x^{2^T}$$

2 Proofs of Sequential Work (PoSW)

Proofs of sequential work (PoSW) are closely related to time-lock puzzles. Informally, in such proof system, given a random challenge x and time parameter T, one can compute a *publicly verifiable* proof making T sequential steps, but it's hard to come up with an accepting proof in significantly less than T sequential steps, even given access to massive parallelism.

3 Interactive PoSW from RSW

The RSW time-lock puzzle looks like a promising starting point for constructing a PoSW. The main difficulty one needs to solve is achieving public verifiability: to efficiently verify that $y = x^{2^T} \mod N$, one needs p and q — the factorization of N. But the factorization cannot be public as otherwise also computing y becomes easy.

Pietrzak [Pie19] presented an elegant solution to the above problem. He constructed a protocol where a prover P can convince a verifier V it computed the correct solution $y = x^{2^T} \mod N$ without either party knowing the factorization of N. The key idea of the proof is very simple. Assume P wants to convince V that a tuple (x, y) satisfies $y = x^{2^T} \mod N$. For this, P first sends $\mu = x^{2^{T/2}}$ to V. Now $\mu = x^{2^{T/2}}$ together with $y = \mu^{2^{T/2}}$ imply $y = x^{2^T}$. The only thing we have achieved at this point is to reduce the time parameter from T to $T/2$ at the cost of having two statements to verify instead of just one. Pietrzak showed that the verifier choose a random $r \leftarrow \{0,1\}^n$ (for some security parameter n) and merge those two statements into a single statement $(x', y') = (x^r \cdot \mu, \mu^r \cdot y)$ that satisfies $y' = x'^{2^{T/2}}$ if the original statement $y = x^{2^T}$ was true (and P sends the correct μ), but is almost certainly wrong (over the choices of the random exponent r) if the original statement was wrong, no matter what μ and the malicious prover did send (i.e., the probability that the new statement is true is smaller than $\text{negl}(n)$). This subprotocol is repeated $\log(T)$ times — each time

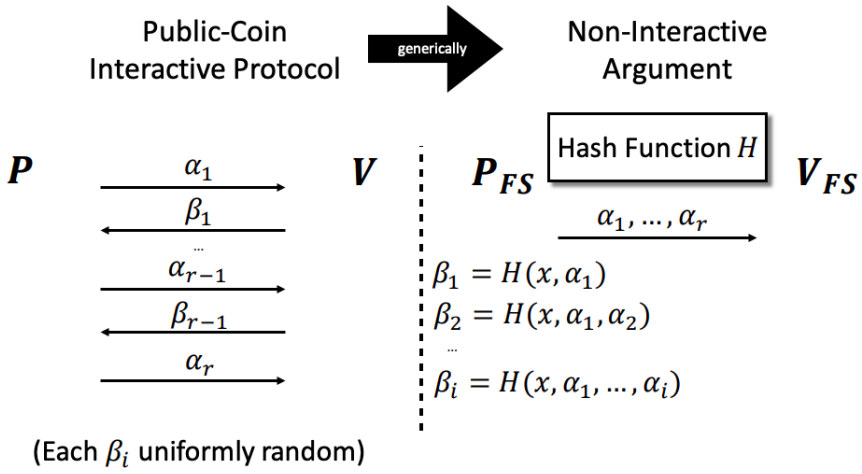
halving the time parameter T — until $T = 1$, at which point V can efficiently verify correctness of the claim itself.

4 Non-Interactive PoSW via Fiat-Shamir Transformation

Pietrzak's protocol presented above is interactive, and therefore, does not exactly feet to the definition of PoSW and cannot be used in the blockchain setting as it is. Yet, the protocol is public-coin — the verifier just sends its random coins at each round, which means that it can be easily transformed into a non-interactive proof by the Fiat-Shamir transformation using a good hash function H (modeled as a random oracle):

4.1 The Fiat-Shamir Transform

Given a public-coin interactive protocol (P, V) in which a prover P tries to convince a verifier V in the validity of some statement x, one can transform the protocol into a non-interactive one by using a good hash function H as follows:



Namely, the outcome of the transformation is a non-interactive protocol (P_{FS}, V_{FS}) where the prover P_{FS} on input x emulates an execution of $(P, V)(x)$ where each β_i — the random coins used by the verifier at round i — is set to $H(x, \alpha_1, \ldots, \alpha_i)$ rather than taken at random. The proof then is just the transcript of the execution (note that the β_i's are not really needed since they are determined by x and the α_i's). Now anyone can verify the proof by checking that the transcript $(\alpha_1, \beta_1, \ldots, \alpha_m, \beta_m)$ is indeed an accepting transcript according to (P, V). The security is assumed to hold since given a very good hash function (modeled as random oracle) and given an (unpredictable) challenge x, the β_i's from the prover's point of view P_{FS} looks random. In our case, the tuple (N, x, T, y) represents the statement: $y = x^{2^T} \bmod N$.

References

[Pie19] K. Pietrzak, "Simple verifiable delay functions," in *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, 2019, 60:1–60:15 (cit. on pp. 1, 2).

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," 1996 (cit. on p. 1).