Malleability of the blockchain's entropy

Cécile Pierrot[1] and Benjamin Wesolowski[2]

[1] Sorbonne Universités, UPMC Univ Paris 06, LIP6, 4 place Jussieu, 75005 Paris, France
[2] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland

Abstract. Trustworthy generation of public random numbers is necessary for the security of many cryptographic applications. It was suggested to use the inherent unpredictability of blockchains as a source of public randomness. Entropy from the Bitcoin blockchain in particular has been used in lotteries and has been suggested for a number of other applications ranging from smart contracts to election auditing. In this *Arctic*le, we analyse this idea and show how an adversary could manipulate these random numbers, even with limited computational power and financial budget.

Keywords: Random number generation, Blockchain, Random Beacon, Bitcoin, Dyck language.

1 Introduction

Randomness is a key concept in cryptography. Even though random numbers are often meant to be kept secret in cryptographic protocols, a number of applications require a form of publicly available randomness which cannot be predicted or manipulated. Some straightforward examples include draws of national lotteries, sampling of assemblies or citizen juries, or tie-breaking in sports or elections. This concept also appears in more technical settings, where a secure public source of randomness can be used to provide trust between communicating parties while implementing contract signing, confidential disclosures, or certified mail in an electronic mail system. These examples were first proposed by Rabin in [Rab83], where he defines the source of randomness as a *random beacon*: an online service that broadcasts allegedly unpredictable numbers at regular intervals.

In these situations the randomness should be generated in such a way that no one can willingly bias the outcome to anyone's advantage or disadvantage. Simple methods hardly provide any strong guarantee, like a skilled prestidigitator can fool entire crowds while tossing a coin. The existing online services that make available fresh random numbers at regular intervals such as [NIS11,RAN98] can at best be considered as trusted third parties: despite extensive documentation and audits, they do not provide users with any mechanism that allows to verify the freshness and correct generation of the published numbers.

A line of research aims at developing and analysing methods to provide trustworthy public randomness. Clark and Hengartner proposed in [CH10] a beacon based on stock market prices, whose security relies on a presumed unmalleability of published financial data – hardly sufficient to convince the most skeptical users. In [LW15], Lenstra and Wesolowski constructed and proved the security of a verifiable beacon that anyone can influence, but no one can bias or predict, by feeding some public input data to a slow hash function.

In another attempt to build a decentralised beacon with no trusted parties, it has been suggested to exploit the inherent unpredictability of blockchains, in particular that of the Bitcoin protocol [Nak09]. In addition to the applications mentioned above, a source of randomness internal to Bitcoin – or any other blockchain based currency – allows that randomness to be used by the currency's scripting language. This extends the range of implementable smart contracts considerably. There is currently no possibility in the Bitcoin scripting language to allow any form of random execution. It is still not the case in cryptocurrencies with more powerful scripting languages

such as Ethereum [Woo14]. A few Bitcoin protocols [BCG15,BNM+14,GGMR14,MGGR13] are however already relying on the blockchain as a source of randomness.

This idea becoming widespread together with the lack of a security model for the blockchain's unpredictibility motivated Bonneau, Clark and Goldfeder [BCG15] to propose an analysis of the method. They consider a model where the attacker controls all the miners, and suffers a monetary penalty whenever a miner finds a valid block that the attacker does not want to see included in the blockchain (see Paragraph 2.1 for a presentation of the notions of miners and valid blocks). This penalty is meant to compensate for the loss of the reward going to a miner finding a valid block. This reward currently consists of 25 ₿ (25 bitcoins[3]) per block plus some transaction fees in the case of the Bitcoin protocol. The amount of this reward decays over time: it started at 50 ₿, and is divided by two every 210 000 blocks. In [BCG15], the authors derive that if the next block is used to define a single random bit – presumably unbiased –, an attacker can force the bit to 1 by having the miners drop the valid blocks that give a 0, and this attacker would suffer an expected cost of 50 ₿ at the moment the paper was published.

We aim in the present paper at overcoming the limitations of this model. First, we measure the adversary in terms of computational resources rather than monetary. An associated cost in bitcoins as in [BCG15] can then be derived, but is not the primary focus. It makes sense indeed to investigate computational limitations when not only money is at stake in the random number generation. Also, it allows us to analyse adversaries that do not have control over the entire pool of miners: it seems unrealistic to assume that the adversary can bribe any miner, who would systematically and silently cooperate. We also account for adversaries with limited resources: whereas in the original analysis the adversary keeps spending money in bribes as long as its goal is not achieved (e.g., obtaining a bit 1 in the example above), our analysis can measure the extent to which the adversary can bias the outcome if they have a limited budget. It also accounts for more general strategies, where the adversary can take advantage of non canonical phenomenons on the blockchain such as forking.

Whereas the authors of [BCG15] concluded from their analysis that the method is secure enough to be used in practice, we show that an attacker, even when not controlling the entire pool of miners and when limited by a budget, can still significantly bias the probability distribution of the outcome. For instance, again when the blockchain is applied to generate a single unbiased bit, we show that an adversary controlling only a quarter of the total mining power can increase the probability of having a 1 from 0.5 to 0.6 approximately, for an expected cost of around 25 ₿, i.e. US$9400. With the same parameters we also propose a cheaper strategy that permits to increase the probability from 0.5 to about 0.57, with an expected cost of 3.6 ₿ ≈ US$1320. We also show how this probability can be further improved by preparing the attack in advance: a head start of a day increases the probability to 0.74.

Also, the security properties that we define and analyse might be relevant beyond the problem of generating public random numbers, as a more general measure of the security of the blockchain construction. Namely, we study the malleability of the blockchain, a formalisation of the capacity of a party to influence the probability distribution of some bits in the chain. Even though unmalleability is not the primary goal of a secure public ledger, it comes as a desirable property of an ideal instantiation of the blockchain.

The details of the model, together with a brief introduction to blockchain protocols, are presented in Section 2. In Section 3, we study the simple case where the adversary simply wants to influence the next published block with immediate effect, as an example and a warm-up. The more complex setting is analysed in Section 4. Adversaries limited by a financial budget are analysed in Section 5 whereas adversaries preparing themselves in advance are studied in Section 6.

[3] Note that 1 ₿ roughly equals US$370 at the time of writing this article, keeping in mind that the Bitcoin exchange rate still fluctuates a lot.

2 A model for the network and the miners

2.1 Blockchains

A blockchain is a form of distributed database, introduced in [Nak09] as the backbone of the Bitcoin protocol. The protocol involves a set of active participants, called miners, distributed over a network, whose job consists in maintaining the blockchain: a global ledger of the history of the system. In the case of Bitcoin, this history is the set of all transactions that happened. At regular time intervals, a new block is added to the chain, testifying for the latest transactions or events. A block is called *valid* if its hash has a certain number d of leading zeros[4]. Only valid blocks can be added to the blockchain. The process of finding a valid block is called mining, and it simply consists in hashing a block a tremendous amount of times with slight modifications until a valid version is found.

In addition to the list of new transactions, a block also contains the hash of the previous valid block. The set of all broadcast blocks thus forms a tree, whose longest chain is the valid blockchain on which there is a consensus. In the case of Bitcoin, each block on that longest chain comes with a reward (some bitcoins) to the miner who mined it. This mechanism encourages the miners to always work on increasing the longest chain they are aware of, in order to keep a mostly linear tree: a tree that consists in the longest chain itself and a few small branches, that quickly stop growing. When such a branching occurs at the tail of the blockchain, it is called a fork. Miners might work on a different branch of the fork at a given moment, but if they keep agreeing to work on the longest one, only the branch with a majority of miners will survive, and consensus is quickly reached again.

It is justified in [BCG15, Paragraph 3.1] that assuming the underlying hash function is secure, each valid block found via the mining process contains at least d bits of computational min-entropy, and therefore $\lfloor d/2 \rfloor$ near-uniform bits can securely be extracted by a cryptographic extractor [DGH$^+$04]. An *extractor* is a function that transforms a random variable with sufficient entropy into an almost uniformly distributed variable in a smaller domain, which we call the *extract*. In this paper we consider an extractor `Ext` that maps valid blocks to elements of $\mathcal{E}$, the set of binary strings of length $\lfloor d/2 \rfloor$. Assuming the hash function used for mining is secure, we can consider the extract $\texttt{Ext}(B)$ of a freshly generated valid block B to be uniformly distributed in $\mathcal{E}$.

2.2 Game model

We consider a single instance of a blockchain protocol executed in isolation. The quite detailed abstraction proposed in [GKL15, Section 2] fits our needs, yet we will not detail it here. Indeed, such a precise network diffusion model is not necessary as we will not be dealing with network adversaries: they will not be able to tamper with messages on transit in the network, or with the network's structure itself. This weaker adversary receives and broadcasts messages over the network, and its power will only be measured in terms of its computational resources.

As in Garay et al.'s formal analysis of the Bitcoin backbone protocol [GKL15], the problem is made tractable by assuming that communications over the network are instantaneous. Honest miners are always working with the longest blockchain that they know of, and the broadcast messages are always received by all the miners. Therefore forks can appear when a valid block is found simultaneously by two miners, but the fork will disappear as soon as one of the branches will be lengthened before the other. This matches the real world situation, where the notion of simultaneity is loosened to take into account communication and processing delays: a fork only

[4] At the time of writing this article, the value of d for Bitcoin was approximately 66. This parameter is continuously adjusted so that it takes an expected time of 10 minutes for the worldwide mining effort to find a valid block.

survives as long as new blocks are added simultaneously to both branches, which cannot happen for very long. We argue that in a model where new valid blocks are never found simultaneously an adversary is not advantaged compared to a model where such accidental forks occur. Therefore we make the latter assumption for the rest of this paper. Even without accidental forks, malicious forks are still possible: an adversary can secretly work on a branch longer than the public one, and once this branch is revealed, it will completely replace the old one as the new longest chain.

The game involves a set of honest miners and an adversary $\mathcal{A}$. Honest miners simply aim at growing the blockchain, by broadcasting valid blocks to get the associated reward. Meanwhile, the goal of the adversary is to bias the probability distribution of the extract of an upcoming block in the chain, which we call the *decisive block*. For illustration, say that the extract x of that decisive block is a lottery draw: the adversary wins whenever x falls in a given subset of $\mathcal{E}$, the "winning tickets". Let $\chi : \mathcal{E} \mapsto \{0, 1\}$ be the characteristic function of that subset of the set of possible extract values. The adversary $\mathcal{A}$ wins if, at the end of the game, $\chi(x) = 1$ where x is the extract of the decisive block, which we call *pleasant* in this case. Note that it does not matter whether that decisive block was broadcast by the adversary or by a honest miner. We denote by $\mathcal{P}_{\mathcal{A}}$ the probability that $\mathcal{A}$ wins.

The game starts at the round where a fixed *initial block* indexed by 0 is received and ends when the blockchain reaches a length of $n + \Delta$, for some predefined integers n and Δ. The n-th block is the decisive block, whose extract determines whether or not the adversary wins. Since that block determines the outcome of the game, it might first seem surprising to continue for Δ additional blocks. But the value of any specific block can be changed via forking: in blockchain protocols, consensus does not go to the version of a block that arrived first, but to the version of the block that belongs to the longest chain. An adversary could try to take advantage of that by attempting to create a fork to modify an unpleasant decisive block. The positive parameter Δ takes into account the possibility that the decisive block can still be modified until the blockchain has been lengthened by Δ blocks.

Finally we denote by μ the probability for a uniformly distributed extract value $x \in \mathcal{E}$ to satisfy $\chi(x) = 1$. We clearly have $\mu = |\{\chi(x) = 1 \,|\, x \in \mathcal{E}\}|/|\mathcal{E}|$.

2.3 Adversary model

We consider a weak adversary that cannot change the content of the messages traveling around the network nor prevent them from being delivered. The adversary can mine on the blockchain and, unlike honest miners, they are not assumed to always work on extending the longest chain, nor to broadcast all the valid blocks they find. Thus, in this context, the critical property of the adversary is their computing power, or more precisely, their relative power with respect to the total power of the miners. Concretely, this power is measured as a number $s_{\mathcal{A}}$ (respectively s) of hashes per second that the adversary (respectively the total set of miners, including $\mathcal{A}$), is able to compute. Let

$$\lambda = \frac{s_{\mathcal{A}}}{s}$$

measure the relative power of $\mathcal{A}$. Such hash computations can be modelled as calls to a random oracle. The number of random oracle calls to be done before finding a valid block follows a Poisson distribution, and at any point in time, λ can be interpreted as the probability that the next oracle request giving a valid block has been done by the adversary.

Remark 1. The adversary could be a single entity as well as a set of colluding, corrupted miners. It therefore extends the model of [BCG15], where all the miners are ready to accept bribes to collaborate: this corresponds to $\lambda = 1$ in our setting. We can still account for the money it takes to corrupt the miners, via the notion of *virtual cost* (see Paragraph 3.4), yet it is not the only metric available anymore.

3 A warm-up: manipulating the next published block

3.1 Delay-free games

As a first example, let us look at the simple case where $n = 1$ and $\Delta = 0$. The adversary aims at manipulating the next broadcast block, without bothering whether it will survive potential future forks. This case occurs whenever the extracted randomness is meant to be used immediately, such as a random beacon that broadcasts the random numbers as soon as they are available, or a simple national lottery. Indeed, a random beacon is meant to be unpredictable, and its output cannot be changed in the future – unlike a blockchain where forks can "change the past". The beacon must broadcast a number as soon as it is available, which does not permit to wait for a safety delay $\Delta > 0$ before outputting the numbers.

3.2 Strategy

The adversary's strategy here is straightforward: they will focus all their computational power on mining the next block. Whenever they find an unpleasant valid block B such that $\chi(\texttt{Ext}(B)) = 0$, they throw it away and continue to mine. The game ends whenever a honest miner finds a valid block or the adversary finds one that satisfies $\chi(\texttt{Ext}(B)) = 1$, which they immediately broadcast. In this context, it is clearly an optimal strategy as valid blocks can only be found by mining (which is a classical assumption for blockchains, and can be enforced by the random oracle model), and an adversary who would themselves publish a block such that $\chi(\texttt{Ext}(B)) = 0$ would only decrease their winning probability, as well as an adversary who would not publish instantly a valid block such that $\chi(\texttt{Ext}(B)) = 1$.

3.3 The winning probability $\mathcal{P}_{\mathcal{A}}$

Let us compute the probability $\mathcal{P}_{\mathcal{A}}$ that the adversary wins that game. The event that they win the game is denoted $\mathfrak{w}$. The possible outcomes can be split into the infinite family of disjoint events $\mathfrak{e}_a$, for any $a \geq 0$, that the adversary found and threw away exactly a valid blocks. The event $\mathfrak{e}_a \wedge \mathfrak{w}$ further requires that the final block B, i.e. the $(a+1)$-th block that is found, satisfies $\chi(h_B) = 1$ (and it does not matter who found that last one). The winning probability can now be expressed as:

$$\mathcal{P}_{\mathcal{A}} = \sum_{a \geq 0} \Pr(\mathfrak{e}_a \wedge \mathfrak{w}) = \sum_{a \geq 0} (\lambda(1-\mu))^a \mu = \frac{\mu}{1 - \lambda(1-\mu)}.$$

Since $\lambda \geq 0$ and $\mu \leq 1$ this is clearly better than μ. For instance, when $\mu = 1/2$, and $\lambda = 1/4$, the probability that $\mathcal{A}$ wins is increased from 0.5 to $4/7 \approx 0.57$. Note that $\lambda = 1/4$ is a realistic assumption: for instance, the relative computational power λ of the four actual most influent mining pools Antpool, F2Pool, BitFury, and BTCChinaPool vary between 13% and 27% each, depending on sources and dates.

3.4 Virtual cost expectation for the adversary

The *virtual cost* refers to the number of blocks the adversary has to drop, thereby giving up the associated reward. We emphasise that this virtual cost does not take into account potential profits related to the outcome of the game. It is only a measure of the amount of mining power they sacrificed to play the game: namely, the difference between the expected reward they would

have had by behaving as a honest miner, and the reward resulting from playing the described strategy. For each event $\mathfrak{e}_a$, exactly a blocks are dropped, therefore the expected virtual cost E is:

$$E = \sum_{a \geq 0} a \Pr(\mathfrak{e}_a) = \sum_{a \geq 0} a(\lambda(1-\mu))^a (1 - \lambda(1-\mu)) = \frac{\lambda(1-\mu)}{1-\lambda(1-\mu)}.$$

Again, to illustrate this virtual cost we look at the example where $\mu = 1/2$ and $\lambda = 1/4$. It gives $1/7$ expected lost blocks. In terms of money, the actual reward at the time of writing is 25 ₿. Thus this strategy can be approximately evaluated to have an expected cost of US$1320. The case where $\lambda = 1$ ensures the victory as long as $\mu > 0$, for an expected cost of $(1-\mu)/\mu$. This is the situation in [BCG15] where all the miners are assumed to collude with the adversary, and they indeed found the same expected cost.

4 Reaching consensus

4.1 Games with delay

We now deal with the case where $n = 1$ and $\Delta > 0$. Delaying the end of the game later than the broadcast of the decisive block is necessary to measure the security of protocols involving several parties that would base an agreement on the extract value of this decisive block. Smart contracts are a good example for such worries. Indeed, unlike in our first naive delay-free games model, forking can interfere and change the initial execution of the contract, therefore modifying the outcome of the contract itself after its execution. Here, the random value is only considered valid when it is deep enough in the chain. More precisely, choosing a sufficiently large Δ helps to reach consensus, when it is considered impossible to create a fork of length strictly larger than Δ. Then the parties involved in the contract gain confidence that the outcome is definitive.

4.2 Relevant values for Δ

Relevant values for the delay Δ can be deduced from the analysis performed in [GKL15] where the authors evaluate the probability to have a fork of two branches of a given length. From a practical point of view, a look back to the history of broadcast Bitcoin blocks permits to find all the accidental forks that have ever been observed. For instance, from [Blo16] we see that 540 forks were noticed since the block number[5] 142 257: two branches of length 4, four of length 3, fourteen of length 2, and what remains are orphan blocks. Thus, keeping a safety margin, $\Delta = 6$ seems sufficient to make sure that no fork will alter the decisive block after the end of the game.

4.3 Strategy

The adversary's strategy begins as previously: they first focus all their computational power on mining the first block (that is also the decisive block since $n = 1$) and throw away all valid blocks B they find as long as $\chi(\texttt{Ext}(B)) = 0$. At the first broadcast of the decisive block B_d two cases can occur. If it satisfies $\chi(\texttt{Ext}(B_d)) = 1$ then $\mathcal{A}$ starts to act as a honest miner. Namely, they continue to mine on this chain and the game ends when the $(1+\Delta)$-th valid block is found, leading to the success of $\mathcal{A}$. Otherwise, if one honest miner broadcasts a decisive block satisfying $\chi(\texttt{Ext}(B_d)) = 0$, $\mathcal{A}$ continues to secretly mine on the previous block as if they received nothing. Again, they throw away all valid decisive blocks that would lead to their failure. If they find one such that $\chi(\texttt{Ext}(B_d)) = 1$ they keep it secret, create a fork and continue to mine on their new

[5] At the time of writing the last blocks are around number 400 000.

chain, hoping to develop a branch longer than the current public one containing the unpleasant decisive block. If they manage to do so, they broadcast their fork, and all miners switch to that new longest chain and proceed to mine on that one, thereby including the cheated pleasant decisive block in the new consensus chain. The game ends when a miner finds a valid final block, whether it is on the original chain, or on the branch later appeared.

4.4 Computing the probability $\mathcal{P}_{\mathcal{A}}$

To compute the probability that $\mathcal{A}$ wins we consider a partition of our universe as follows. Let $\mathfrak{w}_1$, $\mathfrak{f}_0$, $\mathfrak{f}_{01}$ and $\mathfrak{w}_{01}$ ($\mathfrak{w}$ implicitly stands for $\mathcal{A}$ *wins* and $\mathfrak{f}$ for $\mathcal{A}$ *fails*) be the events defined as:

- $\mathfrak{w}_1$: the first broadcast decisive block B_d is such that $\chi(\texttt{Ext}(B_d)) = 1$.
- $\mathfrak{f}_0$: the first broadcast decisive block B_d is such that $\chi(\texttt{Ext}(B_d)) = 0$ and $\mathcal{A}$ fails to find any decisive block B'_d such that $\chi(\texttt{Ext}(B'_d)) = 1$.
- $\mathfrak{f}_{01}$: the first broadcast decisive block B_d is such that $\chi(\texttt{Ext}(B_d)) = 0$, $\mathcal{A}$ manages to find a decisive block B'_d such that $\chi(\texttt{Ext}(B'_d)) = 1$ but not to compute a branch sufficiently long to replace the main one.
- $\mathfrak{w}_{01}$: the first broadcast decisive block B_d is such that $\chi(\texttt{Ext}(B_d)) = 0$ but the adversary manages to find one B'_d verifying $\chi(\texttt{Ext}(B'_d)) = 1$, and to have it replace the original one.

So the event that $\mathcal{A}$ wins, denoted $\mathfrak{w}$, is the union of $\mathfrak{w}_1$ and $\mathfrak{w}_{01}$. Note that the probability of $\mathfrak{w}_1$ has already been computed in Section 3, as:

$$\Pr(\mathfrak{w}_1) = \frac{\mu}{1 - \lambda(1-\mu)}. \tag{1}$$

For convenience we also define $\mathfrak{e}_0$ as $\mathfrak{e}_0 = \mathfrak{f}_0 \vee \mathfrak{f}_{01} \vee \mathfrak{w}_{01}$.

Probability of $\mathfrak{f}_0$. Since $\mathfrak{f}_0$ is a sub-event of $\mathfrak{e}_0$, its probability can be expressed as:

$$\Pr(\mathfrak{f}_0) = \Pr_{\mathfrak{e}_0}(\mathfrak{f}_0) \times \Pr(\mathfrak{e}_0) = \Pr_{\mathfrak{e}_0}(\mathfrak{f}_0) \times (1 - \Pr(\mathfrak{w}_1)),$$

where we denote by $\Pr_B(A)$ the conditional probability of event A given that event B occurred.

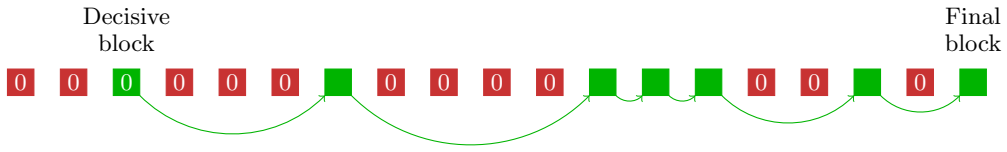

Fig. 1. Secret and public blocks mined during the event $\mathfrak{f}_0$. Green blocks represent the blockchain integrally mined by honest miners whereas red ones correspond to the k valid decisive blocks (secretly) found by $\mathcal{A}$, all such that $\chi(\texttt{Ext}(B)) = 0$. A bit inside a block indicates the value of χ for its extract. It is only indicated on blocks that are meant to be decisive: here, the first broadcast block, and the adversary's failed attempts at finding a pleasant one. In this example, $a = 12$ and $\Delta = 6$.

Figure 1 represents the situation for a fixed number a of valid blocks mined by $\mathcal{A}$. The probability that $\mathcal{A}$ fails precisely a times knowing $\mathfrak{e}_0$ is given by the product:

$$\binom{a+\Delta-1}{a}(\lambda(1-\mu))^a(1-\lambda)^{\Delta-1}(1-\lambda).$$

Thus the formula for $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0)$ is:

$$(1-\lambda)^{\Delta}\sum_{a\in\mathbf{N}}\binom{k+\Delta-1}{a}(\lambda(1-\mu))^a.$$

The summation over a can be simplified using the fact that for any positive integer h, and any $x \neq 1$:

$$\sum_{a\in\mathbf{N}}\binom{a+h}{a}x^a = \frac{1}{(1-x)^{h+1}}. \tag{2}$$

From Equation (2) we get the final value of the probability of $\mathfrak{f}_0$ knowing $\mathfrak{e}_0$:

$$\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0) = \left(\frac{1-\lambda}{1-\lambda(1-\mu)}\right)^{\Delta}. \tag{3}$$

Probability of $\mathfrak{f}_{01}$. Again we compute $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})$ the probability of $\mathfrak{f}_{01}$ knowing $\mathfrak{e}_0$ in order to find the probability of $\mathfrak{f}_{01}$ that satisfies:

$$\Pr(\mathfrak{f}_{01}) = \Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01}) \times (1 - \Pr(\mathfrak{w}_1)).$$

Figure 2 represents the situation and gives an indication of the various parameters we need to express this probability. We define:

- a_2 as the number of valid decisive blocks mined by $\mathcal{A}$ after the first decisive block is broadcast by a honest miner and before $\mathcal{A}$ finds a valid decisive block verifying $\chi(\texttt{Ext}(B_d)) = 1$.
- h_2 as the number of valid blocks mined by all honest miners in the blockchain in the meantime, namely after the first decisive block is broadcast by a honest miner, and before the adversary manages to find a decisive block verifying $\chi(\texttt{Ext}(B_d)) = 1$.
- a_3 as the number of valid blocks mined by $\mathcal{A}$ on its chain after $\mathcal{A}$ finds a valid decisive block verifying $\chi(\texttt{Ext}(B_d)) = 1$ and before the last block is mined.

With this in hands, we see that the number of valid blocks mined by all honest miners, after $\mathcal{A}$ finds a pleasant valid decisive block, and before the last block is mined, is given by $\Delta - 1 - h_2$. Besides, some parameters are upper bounded: we have $h_2 < \Delta$ and $a_3 < \Delta$.

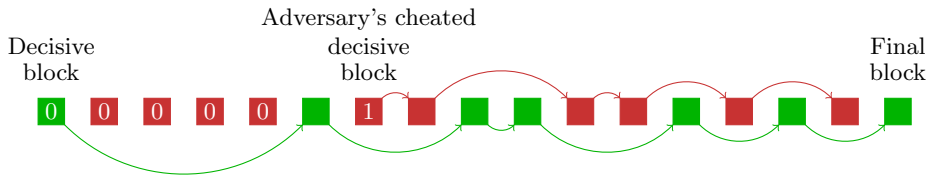

Fig. 2. Secret and public blocks mined during the event $\mathfrak{f}_0$. Green blocks represent the blockchain integrally mined by honest miners whereas red ones correspond to the $a_2 + 1 + a_3$ valid blocks (secretly) found by $\mathcal{A}$. Again a number inside a block corresponds to the value of χ for its extract, when the block is meant to be a decisive one. In this example $\Delta = 6$, $a_2 = 4$, $h_2 = 1$ and $a_3 = 5$.

The probability, knowing $\mathfrak{e}_0$, that $\mathcal{A}$ finds a pleasant valid decisive block exactly after a_2 trials that failed, and to get h_2 blocks mined by the honest miners before this new decisive

block is:

$$\binom{a_2+h_2}{a_2}\lambda^{a_2+1}(1-\lambda)^{h_2}(1-\mu)^{a_2}\mu.$$

To complete the computation of $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})$ we need to count the number of possible cases in which $\mathcal{A}$ mines a_3 valid blocks but fails to ever create a chain longer that the main one. This number in hand, it will only remain to sum over the possible values of the three variables a_2, a_3 and h_2. Fixing these parameters, an outcome where the adversary fails to replace the main blockchain with its own corresponds to a sequence where *at every point in time*, the adversary's chain is shorter that the main one. The insightful reader might have observed that this situation shares similarities with Dyck words. A word in two letters H and A is a *Dyck word* if no initial segment contains more A's than H's. Here the situation is slightly different as the honest main chain (whose blocks correspond to the letter H) starts growing as soon as a honest miner finds a first valid block, while the adversary's chain (the A's) starts growing only when the adversary finds a first valid block such that $\chi(\texttt{Ext}(B)) = 1$. Therefore we define *generalised Dyck words* as follows.

Definition 1. *Let Δ, a and h be three non-negative integers such that $a < \Delta$ and $h < \Delta$. We call* generalised Dyck words *(for the parameters Δ, a, and h) words with $a + h$ letters formed with exactly a letters A and h letters H that verify the following property:* for any prefix of the word, we have $\#A \leq \#H + \Delta - h - 1$ where $\#H$ (respectively $\#A$) represents the number of H's (respectively A's) in that prefix.

Lemma 1. *Let Δ, a and h be three non-negative integers, and suppose $a < \Delta$ and $h < \Delta$. Defining $D(\Delta, a, h)$ as the number of generalised Dyck words we exactly have:*

$$D(\Delta, a, h) = \binom{a+h}{a} - \binom{a+h}{\Delta}.$$

Proof. The proof follows a classical idea developed to count Dyck words. We first look at words with exactly a letters A and h letters H that are not generalised Dyck words. For each such word, there exists a first A that breaks the property. Changing each A into an H and each H into an A strictly after this special A leads to write a word[6] with exactly $a - \Delta + h$ letters H and Δ letters A. Thus for a fixed Δ this process describes an injection:

$$f : E \longrightarrow F$$

that sends E (the set of all the words with a letters A, h letters H that are not generalised Dyck words) into F (the set of words with $a - \Delta + h$ letters H, Δ letters A). Besides, the map f is surjective. Indeed, let us consider a word w in F. From $a < \Delta$ we see that $\Delta > (a - \Delta + h) + \Delta - h - 1$, which means that w breaks the condition $\#A \leq \#H + \Delta - h - 1$. Hence considering the first letter A in w that breaks the condition, and changing each A into an H and each H into an A strictly after this special A, we obtain a word in E such that the image of this word is exactly w. The bijection f yields $\#E = \binom{a+h}{\Delta}$, and at the end:

$$D(\Delta, a, h) = \binom{a+h}{a} - \binom{a+h}{\Delta}.$$

Summing over the possible values of the three variables a_2, a_3 and h_2, we obtain:

$$\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01}) = \lambda\mu(1-\lambda)^{\Delta} \sum_{\substack{0 \leq a_2 \\ 0 \leq a_3, h_2 < \Delta}} (\lambda(1-\mu))^{a_2} \binom{a_2+h_2}{a_2} \lambda^{a_3} D(\Delta, a_3, \Delta - 1 - h_2),$$

[6] For instance, if $h = 3$, $a = 8$ and $\Delta = 6$ the word AHHH**H**AAHHHH becomes AHHH**H**HHAAAA that has $8 - 6 + 3 = 5$ letters A and 6 letters H.

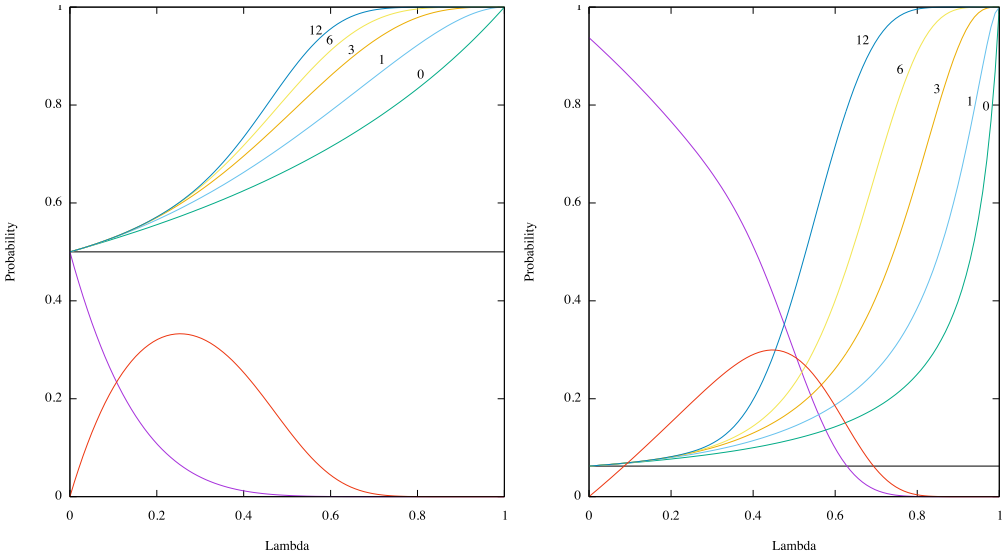

Fig. 3. Probabilities $\mathcal{P}_{\mathcal{A}}$ as functions of λ when $\mu = 1/2$ (first graph) and $\mu = 1/16$ (second graph). The corresponding values for Δ, ranging from 0 to 12, are indicated on the curves. For information purposes, the red curves correspond to $\Pr(\mathfrak{f}_{01})$ and the purple ones to $\Pr(\mathfrak{f}_0)$, both with $\Delta = 12$.

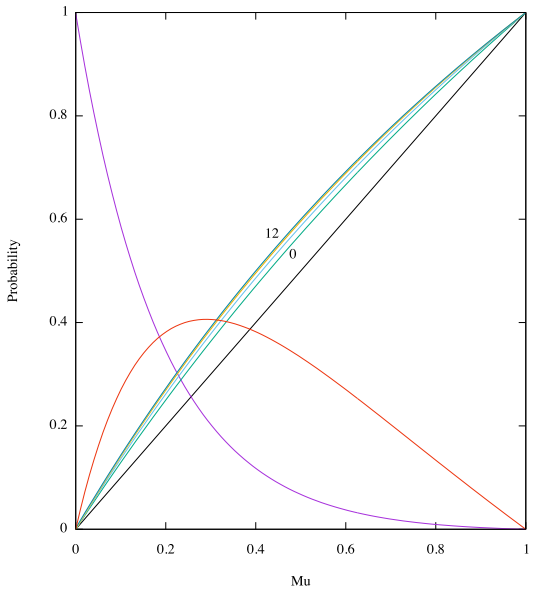

Fig. 4. Probabilities $\mathcal{P}_{\mathcal{A}}$ as functions of μ when $\lambda = 1/4$. Values for Δ are indicated on the curves. Again the red curve corresponds to $\Pr(\mathfrak{f}_{01})$ and the purple one to $\Pr(\mathfrak{f}_0)$ both with $\Delta = 12$. The black line indicates the probability that a honest player wins.

where D is the function in 3 variables defined in Lemma 1 that counts the number of possible cases in which the chain mined by $\mathcal{A}$ never overtakes the honest one. The probability $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})$ can be rewritten as:

$$\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01}) = \lambda\mu(1-\lambda)^{\Delta} \sum_{0\leq a_3,h_2<\Delta} \lambda^{a_3} D(\Delta, a_3, \Delta-1-h_2) \sum_{0\leq a_2} \binom{a_2+h_2}{a_2} (\lambda(1-\mu))^{a_2}.$$

We can simplify the inner sum using Formula (2) and get:

$$\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01}) = \frac{\lambda\mu(1-\lambda)^{\Delta}}{1-\lambda(1-\mu)} \sum_{0\leq a_3,h_2<\Delta} \frac{\lambda^{a_3}}{(1-\lambda(1-\mu))^{h_2}} D(\Delta, a_3, \Delta-1-h_2). \tag{4}$$

To conclude, we can put everything together using $\mathcal{P}_{\mathcal{A}} = 1 - \Pr(\mathfrak{f}_0) - \Pr(\mathfrak{f}_{01})$, with the formula

$$\mathcal{P}_{\mathcal{A}} = 1 - (1 - \Pr(\mathfrak{w}_1)) \cdot (\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0) + \Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})), \tag{5}$$

where $\Pr(\mathfrak{w}_1)$, $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0)$ and $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})$ are functions in λ, μ and Δ respectively given in Equations (1), (3) and (4). To compare again with μ we plot several of these probabilities. Figure 3 gives the probability that $\mathcal{A}$ wins as a function of their relative computational power λ, for a fixed value of $\mu = 1/2$ on the left or $\mu = 1/16$ on the right. For instance, if $\mu = 1/2$, $\lambda = 1/4$ and $\Delta = 6$ – which are three arguably realistic parameters – then the probability that the adversary wins is increased from 50% to approximately 60%. Figure 4 takes the other point of view and considers how the probability $\mathcal{P}_{\mathcal{A}}$ varies with μ when λ is fixed to 1/4.

4.5 Virtual cost expectation for the adversary

Again, the virtual cost refers to the number of blocks the adversary has to drop, thereby giving up the associated reward. To compute the expected virtual cost E we consider three cases. In the event $\mathfrak{w}_1$, such blocks only appear before the first decisive block is mined. Thus the expected number of dropped blocks that correspond to this case is:

$$E_1 = \sum_{a\geq 0} a \Pr(\mathfrak{e}_a \wedge \mathfrak{w}) = \sum_{a\geq 0} a(\lambda(1-\mu))^a \mu = \frac{\mu\lambda(1-\mu)}{(1-\lambda(1-\mu))^2}. \tag{6}$$

Besides, considering the union of $\mathfrak{f}_0$ and $\mathfrak{f}_{01}$, we see that in both cases the chain making the consensus at the end has entirely been mined by honest miners and consists in $\Delta+1$ valid blocks. Since the relative computing power of $\mathcal{A}$ is λ, it yields that $\mathcal{A}$ dropped in average $(\Delta+1)\lambda/(1-\lambda)$ valid blocks. So the expected number of lost blocks related to those two events is:

$$E_2 = (1 - \mathcal{P}_{\mathcal{A}}) \cdot \frac{(\Delta+1)\lambda}{1-\lambda}. \tag{7}$$

where $\mathcal{P}_{\mathcal{A}}$ is given in Equation (5). Last but not least, we can see the event $\mathfrak{w}_{01}$ as the union of all $\mathfrak{e}_a$ for $a \in \mathbf{N}$ where $\mathfrak{e}_a$ is the event: $\mathcal{A}$ finds precisely a valid blocks with an unpleasant extract, then mines a pleasant decisive block and manages to mine a chain longer than the current honest one. The expected number of dropped blocks related to this last case is:

$$E_3 = \sum_{a\geq 0} a \Pr(\mathfrak{e}_a).$$

To compute $\Pr(\mathfrak{e}_a)$ we denote by h_2 the number of honest valid blocks broadcast (strictly) between the two decisive blocks and by h_3 the number of honest valid blocks broadcast after the pleasant decisive one is mined by $\mathcal{A}$. We get:

$$\Pr(\mathfrak{e}_a) = \lambda^a(1-\mu))^{a+1}\mu \sum_{0\leq h_2<\Delta} \binom{a+h_2}{h_2} (1-\lambda)^{h_2+1} \sum_{0\leq h_3\leq\Delta} \bar{D}(h_2+h_3+1, \Delta, h_3)\lambda^{\Delta}(1-\lambda)^{h_3}$$

where $\bar{D}(h_2+h_3+1,\Delta,h_3)$ is the number of words with Δ letters A and h_3 letters H that are *not* generalised Dyck words (with the correspontding parameters). Since we have the equality $\bar{D}(h_2+h_3+1,\Delta,h_3)=\binom{h_2+h_3+1}{\Delta+h_3}$ we can write:

$$E_3=\lambda^{\Delta}(1-\lambda)(1-\mu)\mu \times\sum_{a\geq 0}a(\lambda(1-\mu))^a\sum_{0\leq h_2<\Delta}\binom{a+h_2}{h_2}(1-\lambda)^{h_2}\sum_{0\leq h_3\leq\Delta}\binom{h_2+h_3+1}{\Delta+h_3}(1-\lambda)^{h_3} \tag{8}$$

Therefore the expected virtual cost E is given by:

$$E=E_1+E_2+E_3,$$

where E_1, E_2 and E_3 are respectively given in Equations (6), (7) and (8). To illustrate this virtual cost we plot in Figure 5 the expected number of blocks that are dropped by the adversary as a function of their relative computational power λ, in the case where $\mu=1/2$. For instance, when $\lambda=1/4$ and considering $\Delta=6$ it gives roughly 1.017 expected lost blocks. Thus this strategy can be approximately evaluated to have an expected cost of US$9400.

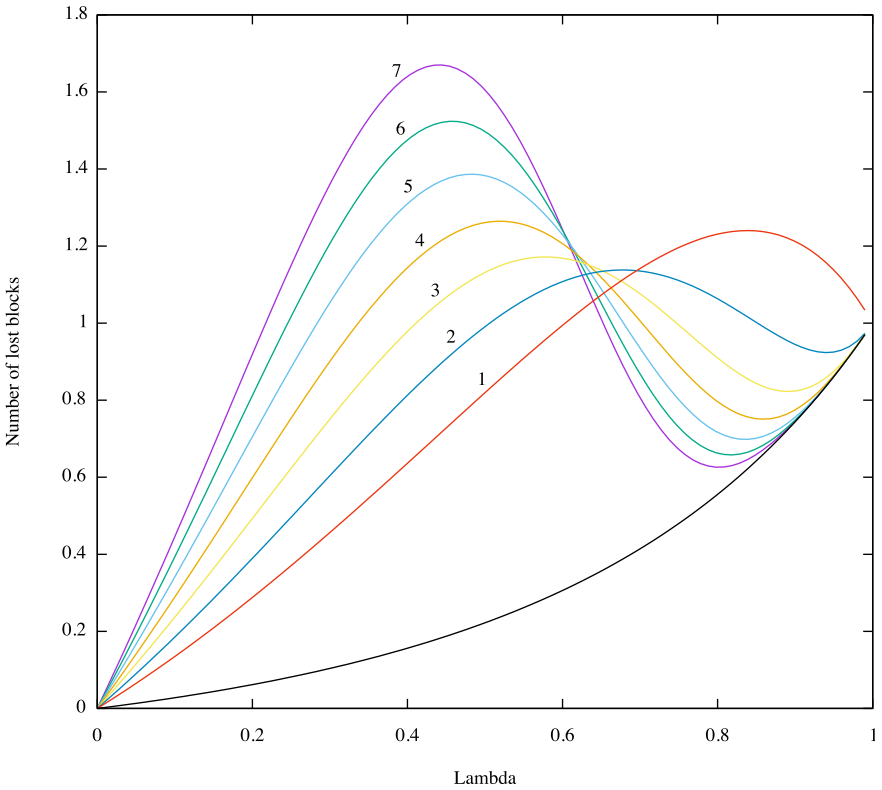

Fig. 5. Expected number of dropped blocks as functions of the relative computational power λ, when $\mu=1/2$. The black curve corresponds to the basic case with no delay, i.e. $\Delta=0$. The other curves correspond to values of Δ that vary from 1 to 7.

5 Adversaries with a virtual budget

A virtual budget is the amount of money the adversary is ready to lose by not publishing valid blocks they found. An adversary with a virtual budget enforces a bound on the virtual cost, and thereby on the number of abandoned blocks. The analysis is similar to the previous case, but now, when the adversary runs out of budget, they must stop mining blocks that do not directly contribute to the longest chain.

Let b be the maximal number of blocks the adversary can drop, i.e., its budget. Then, the probability of $\mathfrak{w}_1$ becomes:

$$\Pr(\mathfrak{w}_1) = \sum_{a=0}^{b} (\lambda(1-\mu))^a \mu = \frac{\mu(1-(\lambda(1-\mu))^{b+1})}{1-\lambda(1-\mu)}.$$

This is the probability to win when $\Delta = 0$, generalising the results of [BCG15] to adversaries with a budget. Let us compute the winning probability for $\Delta > 0$. Now, we cannot forget about what happened before the publication of the first decisive block as we did previously by conditioning over $\mathfrak{e}_0$, as we need to keep track of how many blocks the adversary dropped during that period.

Let $\mathfrak{f}$ be the event that $\mathcal{A}$ fails the game, $\mathfrak{b}$ the event that $\mathcal{A}$ used its full budget during the game, and $\mathfrak{o}$ be the event that the blocks found by the adversary within their budget are all unpleasant. The event $\mathfrak{f}$ is partitioned into four disjoint events:

$$\Pr(\mathfrak{f}) = \Pr(\mathfrak{f} \wedge \mathfrak{b} \wedge \mathfrak{o}) + \Pr(\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \mathfrak{o}) + \Pr(\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \neg\mathfrak{o}) + \Pr(\mathfrak{f} \wedge \mathfrak{b} \wedge \neg\mathfrak{o}).$$

Let us first focus on the event $\mathfrak{f} \wedge \mathfrak{b} \wedge \mathfrak{o}$. Look at the prefix of the sequence of found blocks up to the point where the adversary went out of budget. If $b = 0$, that prefix is empty and the probability of failure is $1-\mu$; otherwise, the last block of that prefix is the bth block found by $\mathcal{A}$. Let h denote the number of blocks honestly mined during that period. Then:

$$\Pr(\mathfrak{f} \wedge \mathfrak{b} \wedge \mathfrak{o}) = \begin{cases} 1-\mu & \text{if } b = 0, \\ (1-\mu)(\lambda(1-\mu))^b \sum_{h=0}^{\Delta} \binom{b+h-1}{b-1} (1-\lambda)^h & \text{otherwise.} \end{cases} \tag{9}$$

Now, let us compute the probability of $\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \mathfrak{o}$. Look at the full sequence of found blocks, $\Delta + 1$ of which are honestly mined, and $a < b$ are found by the adversary. The probability can then be written as:

$$\Pr(\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \mathfrak{o}) = (1-\mu)(1-\lambda)^{\Delta+1} \sum_{a=0}^{b-1} \binom{\Delta+a}{a} (\lambda(1-\mu))^a. \tag{10}$$

We now look at $\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \neg\mathfrak{o}$. Split the sequence of found blocks into two parts: the initial segment up to the pleasant block found by the adversary, and the rest of the sequence until the end. Let a_1 be the number of blocks found by $\mathcal{A}$ during the initial part, excluding the last one (i.e., only the unpleasant blocks), and h_1 the number of blocks honestly mined in the meantime, including the unpleasant broadcast decisive block. Let a_2 be the number of blocks found by $\mathcal{A}$ during the second phase. Since we suppose $\neg\mathfrak{b}$, we have $a_1 + 1 + a_2 < b$. The number of blocks honestly mined in the second phase is $\Delta + 1 - h_1$, including the final block. Observe that since $\mathcal{A}$ loses, the second part is a generalised Dyck word. Then, the probability can be computed as:

$$\Pr(\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \neg\mathfrak{o}) = (1-\mu)\lambda\mu(1-\lambda)^{\Delta+1} \sum_{\substack{0 \le a_1 < b-1 \\ 1 \le h_1 \le \Delta \\ 0 \le a_2 < b-a_1}} \binom{a_1+h_1}{a_1} (\lambda(1-\mu))^{a_1} D(\Delta, a_2, \Delta - h_1) \lambda^{a_1}. \tag{11}$$

The last event $\mathfrak{f} \wedge \mathfrak{b} \wedge \neg\mathfrak{o}$ needs to be further split into two events: the case $\mathfrak{u}$ where the pleasant decisive block found by $\mathcal{A}$ is the last block in there budget, and the case $\mathfrak{v}$ where $\mathcal{A}$ still has some

budget after that pleasant block was found. Then, $\Pr(\mathfrak{f} \wedge \mathfrak{b} \wedge \neg \mathfrak{o}) = \Pr(\mathfrak{u}) + \Pr(\mathfrak{v})$. We first focus on $\mathfrak{u}$. Look at the prefix of the sequence of blocks up to the pleasant decisive block. That prefix contains exactly b blocks found by $\mathcal{A}$ (including the last one), and a number $h \leq \Delta$ of blocks honestly mined (including the unpleasant decisive block). The probability is then:

$$\Pr(\mathfrak{u}) = \lambda^b \mu (1-\mu)^b \sum_{h=1}^{\Delta} \binom{b+h-1}{h} (1-\lambda)^h. \tag{12}$$

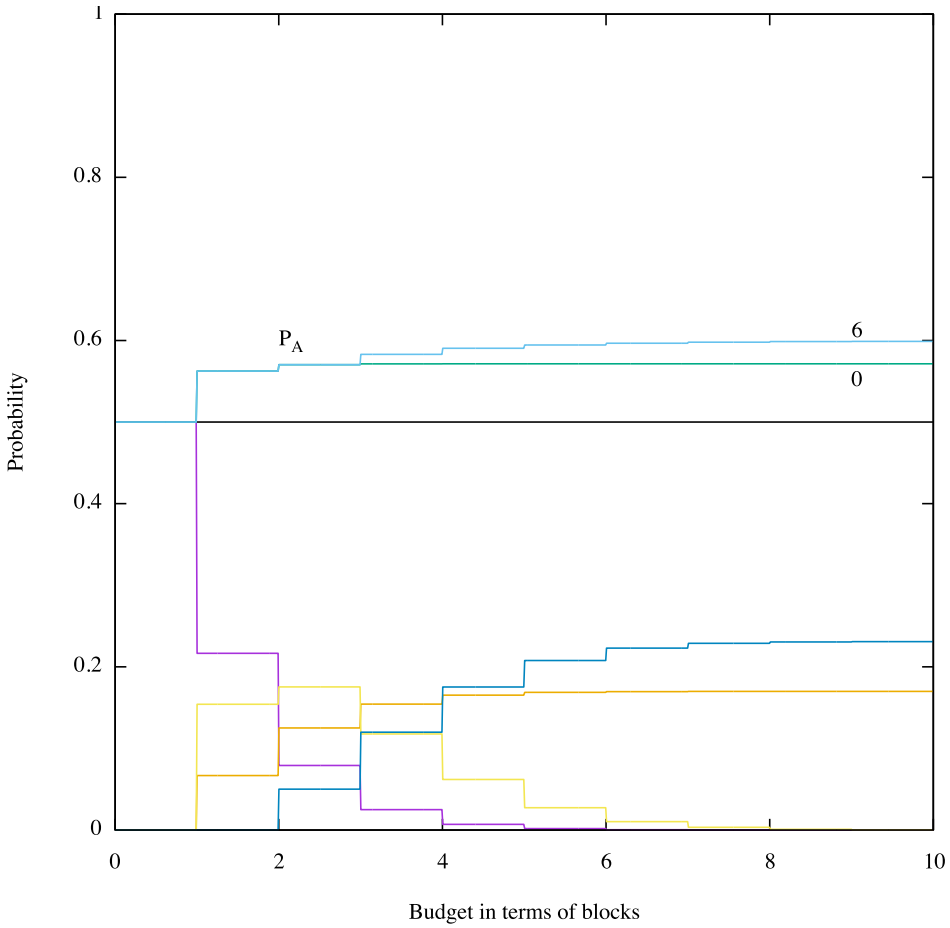

Fig. 6. Probabilities as functions of the budget (number of blocks that can be dropped), when $\lambda = 1/4$ and $\mu = 1/2$. The two highest curves correspond to $\mathcal{P}_{\mathcal{A}}$ in the delay-free case ($\Delta = 0$) and when $\Delta = 6$. The lowest curves are the four components that permit to compute $\mathcal{P}_{\mathcal{A}}$ when $\Delta = 6$: the purple curve corresponds to $\Pr(\mathfrak{f} \wedge \mathfrak{b} \wedge \mathfrak{o})$, the orange one to $\Pr(\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \mathfrak{o})$, the blue one to $\Pr(\mathfrak{f} \wedge \neg\mathfrak{b} \wedge \neg\mathfrak{o})$, and the yellow one to $\Pr(\mathfrak{f} \wedge \mathfrak{b} \wedge \neg\mathfrak{o})$.

It only remains to compute $\Pr(\mathfrak{v})$. Again, focus on the prefix of the sequence of found blocks which stops when the adversary runs out of budget, i.e. when they mine their bth block. Split that prefix into two parts: the initial part up to the pleasant decisive block found by the adversary, and the second part, from that pleasant block to the end, i.e., the bth block of $\mathcal{A}$. Let a_1 be the number of blocks found by the adversary during the first part, excluding the last one (i.e., only the unpleasant blocks), h_1 the number of blocks honestly mined in the meantime, including again the unpleasant decisive block, and h_2 the number of blocks honestly mined in the second part. The number of blocks found by $\mathcal{A}$ in that part is $b - a_1 - 1$. The probability is then:

$$\Pr(\mathfrak{v}) = (1-\mu)\lambda^b\mu \sum_{\substack{0\le a_1<b-1\\ 1\le h_1\le \Delta\\ 0\le h_2\le \Delta - h_1}} \binom{a_1+h_1}{a_1}(1-\mu)^{a_1}(1-\lambda)^{h_1+h_2} D(h_1+h_2, b-a_1-2, h_2). \quad (13)$$

Finally, the winning probability is computed by subtracting all the possible failure probabilities:

$$\mathcal{P}_{\mathcal{A}} = 1 - \Pr(\mathfrak{f}\wedge\mathfrak{b}\wedge\mathfrak{o}) - \Pr(\mathfrak{f}\wedge\neg\mathfrak{b}\wedge\mathfrak{o}) - \Pr(\mathfrak{f}\wedge\neg\mathfrak{b}\wedge\neg\mathfrak{o}) - \Pr(\mathfrak{u}) - \Pr(\mathfrak{v}),$$

where $\Pr(\mathfrak{f}\wedge\mathfrak{b}\wedge\mathfrak{o})$, $\Pr(\mathfrak{f}\wedge\neg\mathfrak{b}\wedge\mathfrak{o})$, $\Pr(\mathfrak{f}\wedge\neg\mathfrak{b}\wedge\neg\mathfrak{o})$, $\Pr(\mathfrak{u})$ and $\Pr(\mathfrak{v})$ are respectively given in Equations (9), (10), (11), (12) and (13). Figure 6 details how $\mathcal{P}_{\mathcal{A}}$ varies as a function of the number of blocks that can be dropped in the two cases $\Delta = 0$ (no delay) and $\Delta = 6$. Note that these probabilities tend to 0.57 (resp. 0.6) which were the probabilities previously found in Section 3.3 (resp. Section 4.4), when the budget was unlimited. Moreover, we can see that the winning probability converges fast enough so that even a small budget is sufficient to significantly cheat the game.

6 Provident adversaries

6.1 Games with early start and delay

Let us consider the more general case where not only there is a final delay $\Delta \ge 1$ as previously, but the game also starts early, i.e. the decisive block is not the next one, but the n-th one, where $n > 1$. This corresponds to the real world situation where an attacker knows in advance which future block will be decisive for the goal he tries to achieve. This case is relevant when a far-sighted adversary may get ahead of the game to manipulate an upcoming decisive block. In truth, nothing prevents adversaries that already cheat from preparing themselves several blocks in advance, without waiting for the beginning of the game to start mining in secret.

The analysis of this situation is essential to applications where some parties commit on an upcoming random number. In a lottery, the ticket seller should announce in advance which (future) block will determine the winner, giving the adversary a head start to manipulate that precise block. It is also crucial in electronic voting systems, where unpredictable, verifiable, public randomness is required for the security of the random auditing after the election [CCC+08].

6.2 Strategy

For the adversary, the goal will be to obtain a chain of length $n-1$ as soon as possible. If they can obtain such a branch before the honest miners – and keeps it secret –, they can start mining for a pleasant n-th block (the decisive one) while the honest miners are still working on building the initial segment of the chain. The adversary tries to build a chain longer than the honest one as follows: on one hand there is the public, honestly mined chain, and on the other hand there is the secret branch of the adversary. The adversary only mined on their own secret branch. At

any point, if the honest chain becomes longer than the adversary's branch, the latter is thrown away and replaced by the longer honest chain. With this strategy, the adversary's branch is never shorter than the public one.

When an $n-1$-th block has finally been found (either by $\mathcal{A}$, or by the honest miners if $\mathcal{A}$ has not been fast enough) the adversary starts following the same strategy as in the previous game – see Section 4.3. Again, they focus all their computational power on mining the decisive block and throw away all valid but unpleasant blocks. At the first broadcast of the decisive block B_d two cases can occur. If it satisfies $\chi(\texttt{Ext}(B_d)) = 1$ then $\mathcal{A}$ starts to act as a honest miner. Otherwise, if one honest miner broadcasts a decisive block satisfying $\chi(\texttt{Ext}(B_d)) = 0$, $\mathcal{A}$ continues to secretly mine on the previous block as if they received nothing. Again, they throw away all valid decisive blocks that would lead to their failure. If they find a pleasant one, they keep it secret, create a fork and continue to mine on their new chain, hoping to develop a branch longer than the current public one containing the unpleasant decisive block. If they manage to do so, they broadcast their fork, and all miners switch to that new longest chain and proceed to mine on that one, thereby including the cheated pleasant decisive block in the new consensus chain. The game ends when a miner finds a valid final block, whether it is on the original chain, or on the branch later appeared.

6.3 Computing the probability $\mathcal{P}_{\mathcal{A}}$

To evaluate the probability that $\mathcal{A}$ wins, we consider again the event $\mathfrak{e}_0$ that the first decisive broadcast block is an unpleasant one. As in Equation (5), $\mathcal{P}_{\mathcal{A}}$ is given by:

$$\mathcal{P}_{\mathcal{A}} = 1 - \Pr(\mathfrak{e}_0) \cdot (\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0) + \Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})), \tag{14}$$

where $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0)$ and $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})$ are functions given in Equations (3) and (4). Yet in this far-sighted adversary case, the analysis differs when computing the probability of $\mathfrak{e}_0$.

Let us consider the family of disjoint events $\mathfrak{a}_k$ that $\mathcal{A}$ finds an $n-1$-th block with an advantage of k blocks over the public branch, where k varies from 0 (meaning that the $n-1$-th block is mined by a honest miner) to $n-1$ (when $\mathcal{A}$ has computed the entire chain from blocks 1 to $n-1$ while the honest miners did not find a single valid block). This leads to:

$$\Pr(\mathfrak{e}_0) = \sum_{k=0}^{n-1} \Pr_{\mathfrak{a}_k}(\mathfrak{e}_0) \cdot \Pr(\mathfrak{a}_k).$$

If a denotes the number of blocks dropped by $\mathcal{A}$ while the honest miners broadcast all the blocks needed to complete their chain up to the unpleasant decisive block, $\Pr_{\mathfrak{a}_k}(\mathfrak{e}_0)$ is given by the sum $(1-\lambda)^{k+1}(1-\mu)\sum_{a\geq 0}\binom{a+k}{a}(\lambda(1-\mu))^a$. Using Equation (2) we find:

$$\Pr_{\mathfrak{a}_k}(\mathfrak{e}_0) = (1-\mu) \cdot \left(\frac{1-\lambda}{1-\lambda(1-\mu)}\right)^{k+1}. \tag{15}$$

Thus it suffices now to evaluate the probabilities $\Pr(\mathfrak{a}_k)$.

Probability $\Pr(\mathfrak{a}_k)$ for a fixed k. Let k be an integer between 0 and $n-1$. The different outcomes that lead the adversary to obtain an advantage of k blocks can be modelled as paths on an $(n-1)\times(n-1)$ grid of blocks from the point $(0,0)$ to the point $(n-1, n-1-k)$ as in Figure 7. Following certain constraints.

Each point (a,h) of a path corresponds to the moment at which $\mathcal{A}$ has a valid branch from block 0 to block a whereas the honest chain only goes up to the block h. The adversary's strategy enforces $h \leq a$, i.e. the path never overpasses the diagonal line. The short arrows forming the

path represent the events that a new valid block has been found. The arrow is red if the block was found by $\mathcal{A}$ and green if it was found by a honest miner. There are only three kinds of arrows:

1. Red arrows to the right: when $\mathcal{A}$ finds a block, it only increases their own secret branch.
2. Green arrows that go up, without overpassing the diagonal: when a honest miner finds a new block but $\mathcal{A}$'s chain is longer, this block only increases the honest branch.
3. Green arrows on the diagonal: when a honest miner finds a new block and $\mathcal{A}$'s branch coincides with the honest chain, this block increases both chains.

Also, the path never follows the blue line: once the blue line has been reached, the next phase of $\mathcal{A}$'s strategy begins. To determine the probability of $\mathfrak{a}_k$, the strategy will consist in computing the number of such paths to $(n-1, n-1-k)$ for any specified number of red arrows.

Let $C_{x,y,r}$ denote the number of paths from $(0,0)$ to (x,y) that follow all the previous requirements and go through r red segments precisely. $C_{x,y,r}$ can be computed thanks to the following recursive formulae:

$$C_{x,y,r} = \begin{cases} 0 & \text{if } y = 0, r \neq x \\ 1 & \text{if } y = 0, r = x \\ C_{x-1,y,r-1} & \text{if } x = n-1, y \neq n-1 \\ C_{x-1,y-1,r} & \text{if } x = n-1 = y \\ C_{x,y-1,r} + C_{x-1,y-1,r} & \text{if } x = y, y \neq n-1 \\ C_{x,y-1,r} + C_{x-1,y,r-1} & \text{otherwise.} \end{cases}$$

At the end, since the number r of red arrows to go to $(n-1, n-1-k)$ varies between k and $n-1$, the probability of $\mathfrak{a}_k$ is given by:

$$\Pr(\mathfrak{a}_k) = \sum_{r=k}^{n-1} (1-\lambda)^{n-1-k} \lambda^r C_{n-1,n-1-k,r}.$$

Together with Equation (15), we find:

$$\Pr(\mathfrak{e}_0) = \frac{(1-\mu)(1-\lambda)^n}{1-\lambda(1-\mu)} \sum_{k=0}^{n-1} \left(\frac{1}{1-\lambda(1-\mu)} \right)^k \sum_{r=k}^{n-1} \lambda^r C_{n-1,n-1-k,r}. \tag{16}$$

The probability $\mathcal{P}_{\mathcal{A}}$ that $\mathcal{A}$ wins is so computed thanks to Equation (14), using Equations (3), (4) and (16) as explicit formulae for the inner probabilities $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_0)$, $\Pr_{\mathfrak{e}_0}(\mathfrak{f}_{01})$ and $\Pr(\mathfrak{e}_0)$. Figure 8 shows how $\mathcal{P}_{\mathcal{A}}$ varies with λ, for $\mu = 1/2$ and a fixed value of $n = 144$ that roughly corresponds to an adversary that starts mining with a day in advance. Note that with a relative computational power $\lambda = 1/4$, the probability that a provident adversary wins takes off from 50% to 74% in the realistic case when $\Delta = 6$. Figure 8 also suggests that being too provident is not necessary. Indeed, $\Pr(\mathfrak{e}_0)$ is the only parameter that governs the variations of the probability $\mathcal{P}_{\mathcal{A}}$ when the adversary mines more or less in advance. We note that this probability stabilises very quickly, as mining with half-a-day, a day or a week in advance is roughly the same in terms of success.

Acknowledgement

We would like to thank Arjen Lenstra for his comments and Antoine Joux for both his technical and financial support. A special thanks goes to Direction Générale de l'Armement and CNRS for funding the first author, and to the Swiss National Science Foundation for supporting the second author via grant number 200021-156420.

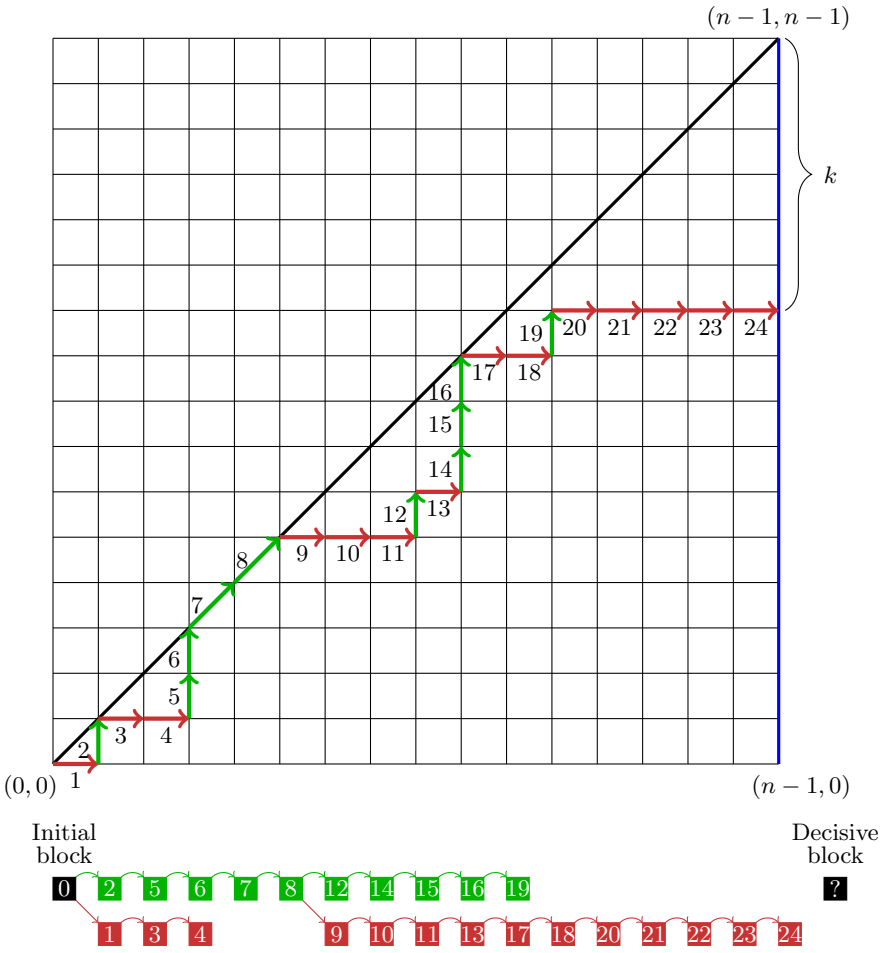

Fig. 7. Secret and public blocks mined during the event $\mathfrak{a}_k$. Green blocks (or arrows) represent the blockchain integrally mined by honest miners whereas red blocks (or arrows) are those secretly mined by $\mathcal{A}$. On this example, $n = 17$ and $\mathcal{A}$ finds the block number $n-1$ with a head start of $k = 6$ blocks. The number inside each block (or near each arrow) indicates the order in which the corresponding block was mined, irrespective of whether it was broadcast or kept secret.

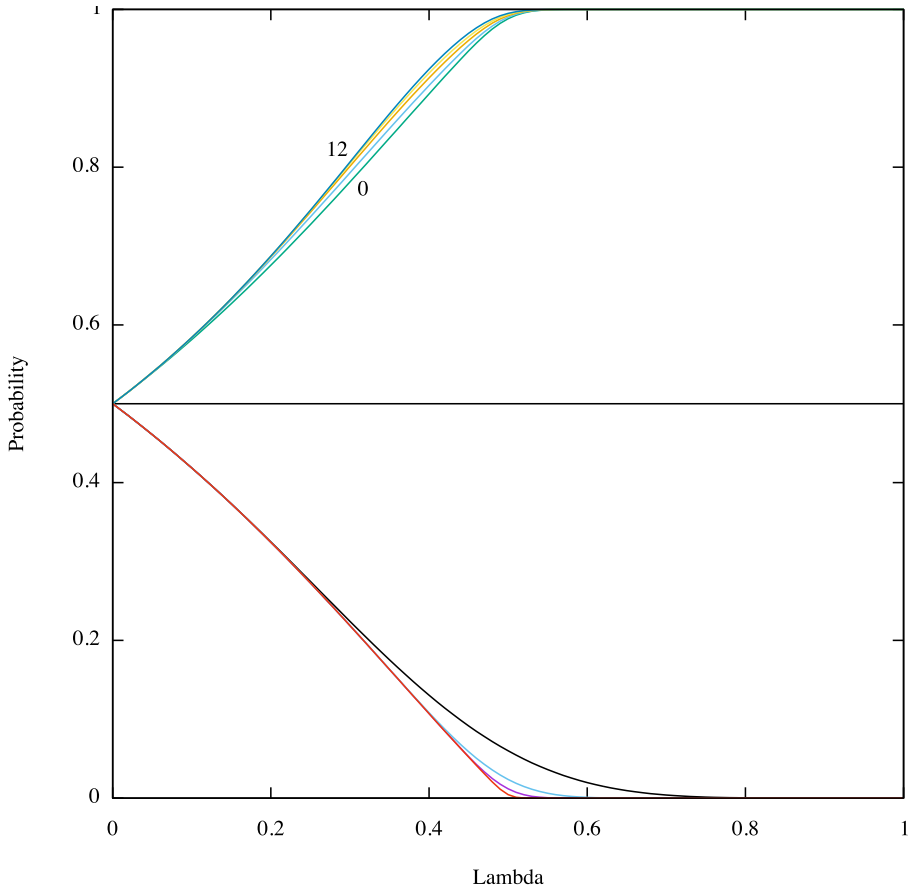

Fig. 8. Probability $\mathcal{P}_{\mathcal{A}}$ as a function of λ when $\mu = 1/2$ and $n = 144$ (in the case of Bitcoin, 144 blocks are found in an expected time of one day). The corresponding values of Δ are 0, 1, 3, 6 and 12 as suggested on the curves. The lower curves represent the inner probability $\Pr(\mathfrak{e}_0)$ as a function of λ for $\mu = 1/2$ and various choices of n. The black one corresponds to $\Pr(\mathfrak{e}_0)$ when $n = 6$ (corresponding, in Bitcoin, to an expected time of one hour), the blue one when $n = 36$ (six hours), the purple one when $n = 144$ (one day) and the red one when $n = 1008$ (one week).

References

[BCG15] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. *IACR Cryptology ePrint Archive*, 2015:1015, 2015.

[Blo16] Blockchain Luxembourg SARL. https://blockchain.info, 2016.

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014*, pages 486–504, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[CCC+08] David Chaum, Richard Carback, Jeremy Clark, Aleksander Essex, Stefan Popoveniuc, Ronald L. Rivest, Peter Y. A. Ryan, Emily Shen, and Alan T. Sherman. Scantegrity ii: End-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *USENIX/ACCURATE Electronic Voting Technology Workshop (EVT)*, 2008.

[CH10] Jeremy Clark and Urs Hengartner. On the use of financial data as a random beacon. In *USENIX EVT/WOTE*. USENIX Association, 2010.

[DGH+04] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the cbc, cascade and hmac modes. In *Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference*, pages 494–510, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[GGMR14] Christina Garman, Matthew Green, Ian Miers, and Aviel D. Rubin. Rational zero: Economic security for zerocoin with everlasting anonymity. In *Financial Cryptography and Data Security: BITCOIN*, pages 140–155. Springer Berlin Heidelberg, 2014.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 2015, Proceedings, Part II*, pages 281–310, 2015.

[LW15] Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015. http://eprint.iacr.org/2015/366.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411, May 2013.

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf, 2009.

[NIS11] NIST randomness beacon. https://beacon.nist.gov, 2011.

[Rab83] Michael O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256 – 267, 1983.

[RAN98] RANDOM.ORG. https://www.random.org, 1998.

[Woo14] Gavin Wood. Ethereum: A secure decentralized transaction ledger. http://gavwood.com/paper.pdf, 2014.