Optimization of a Reed-Solomon code-based protocol against blockchain data availability attacks

Paolo Santini, Giulia Rafaiani, Massimo Battaglioni, Franco Chiaraluce, and Marco Baldi
Dipartimento di Ingegneria dell'Informazione
Università Politecnica delle Marche, Ancona (60131), Italy
email:{p.santini, g.rafaiani, m.battaglioni, f.chiaraluce, m.baldi}@univpm.it

***Abstract*—ASBK (named after the authors' initials) is a recent blockchain protocol tackling data availability attacks against light nodes, employing two-dimensional Reed-Solomon codes to encode the list of transactions and a random sampling phase where adversaries are forced to reveal information. In its original formulation, only codes with rate $1/4$ are considered, and a theoretical analysis requiring computationally demanding formulas is provided. This makes ASBK difficult to optimize in situations of practical interest. In this paper, we introduce a much simpler model for such a protocol, which additionally supports the use of codes with arbitrary rate. This makes blockchains implementing ASBK much easier to design and optimize. Furthermore, disposing of a clearer view of the protocol, some general features and considerations can be derived (e.g., nodes behaviour in largely participated networks). As a concrete application of our analysis, we consider relevant blockchain parameters and find network settings that minimize the amount of data downloaded by light nodes. Our results show that the protocol benefits from the use of codes defined over large finite fields, with code rates that may be even significantly different from the originally proposed ones.**

***Index Terms*—Blockchain, Data Availability Attack, Reed-Solomon Codes.**

I. Introduction

Given the recent blooming and spreading of blockchain applications, the scalability of existing networks based on this technology represents a serious issue. Indeed, as the number of users grows, an increasingly larger number of transactions must be handled. However, the majority of existing blockchains impose some limit on the maximum dimension of each block, which translates into an upper bound on the number of transactions that can be validated per time unit.

According to the Simplified Payment Verification [1] paradigm, a light node only verifies the inclusion of specific transactions and, hence, does not need to download the entire blockchain. In such a scenario, *full nodes*, that is, nodes with more computational resources and full privileges, are the sole responsible for proposing and validating new blocks. However, this distinction affects the decentralization and security of the network. In fact, as the number of validators decreases, the probability that the majority of them collude to create invalid transactions gets higher. This makes the straightforward solution of increasing the block size limits hardly recommendable. In fact, this would cause an increase in the resources needed to store a copy of the ledger and, therefore, to run a full node capable of validating the blockchain. Consequently, users would more likely run *light nodes*, which are not able to verify the transactions correctness. Therefore, in a network with a majority of dishonest consensus-participating nodes, light nodes may not be able to detect invalid transactions included in a block by dishonest full nodes. A solution to this problem consists in requiring honest full nodes to produce evidences of the invalidity of a transaction and to broadcast it to every connected light node, so that the latter becomes aware of the fraud and eventually rejects invalid blocks.

However, networks with a majority of dishonest full nodes are vulnerable to *data availability attacks*. These attacks consist in a malicious node including invalid transactions in a block; the block header is then distributed to the network, but the malicious node withholds the part of the block containing invalid transactions. This way, a honest full node cannot validate the block and is also unable to demonstrate that a fraud attempt is occurring; in fact, in order to generate a fraud proof for an invalid block, it is necessary that all the transactions included in the block are available.

Hence, light nodes are interested to know whether all the data in a block are available to the network or not. In order to contribute to countering these attacks, light nodes could randomly ask some pieces of a block and discard the entire block if they do not receive any answer. Instead, if their request is answered, they can forward the received data to the neighbouring full nodes and wait for a valid fraud proof. This way, malicious nodes may be required to release some part of the hidden information. Still, as the block size increases, malicious nodes could hide very small parts of the block, thus reducing the probability for light nodes to successfully sample withheld data. Some protocols based on linear codes have been proposed to address this issue [2]–[4]. In this paper, we focus on the protocol in [2], called ASBK according to the authors' initials. According to ASBK, the list of transactions of a block is encoded through two-dimensional Reed-Solomon (RS) codes, so that honest full nodes can recover all the data even when a relatively small portion of the block is missing. Hence, for preventing honest full nodes from recovering missing parts through decoding, malicious nodes should increase the portion of hidden data, thus increasing the probability for light nodes to sample it.

Our contribution: We derive a simplified model for the coded blockchain protocol in [2] that allows us to deepen its analysis, with the aim of optimizing the involved parameters for minimizing the amount of data that light nodes have to download as well as the number of samples a light node should ask for. We demonstrate that significant improvements can be achieved through our approach over the original one in [2].

The paper is organized as follows. In Section II we introduce the notation we use and provide the necessary background. In Section III we describe and analyze the ASBK protocol. In Section IV we discuss our improved theoretical model, which is validated in Section V. Section VI concludes the paper.

II. Background

In this section we recall basic concepts that are fundamental to our analysis.

A. Coupon Collector's Problem

The coupon collector's problem is a well-known classical problem in probability theory, and we refer to its formulation presented in [5]. Let us consider the set of coupon indexes $V = \{1, 2, \ldots, v\}$ and a group of y persons; each person picks a subset of V containing $z \leq v$ distinct indexes. Let $W \subseteq V$ of cardinality w and $t = |W \cap (\bigcup_{i=1}^{y} J_i)|$, where J_i is the set of indexes selected by the i-th person. When each J_i is picked independently and uniformly at random, then t is a random variable with mean value given by [5]:

$$\langle t \rangle = v\left(1 - \left(1 - \frac{z}{v}\right)^y\right). \qquad (1)$$

The Cumulative Distribution Function (CDF) of t results in

$$\Pr[t < x] = \sum_{i=0}^{x-1} (-1)^{x-i+1} \binom{w}{i} \binom{w-i-1}{w-x} \left(\frac{\binom{v-w+i}{z}}{\binom{v}{z}}\right)^y. \qquad (2)$$

B. Reed-Solomon Codes

Let $\mathbb{F}_q$ denote the finite field with q elements. A linear code $\mathcal{C}$ defined over $\mathbb{F}_q$ with *length* n and *dimension* k is a k-dimensional subspace of $\mathbb{F}_q^n$. The *code rate* is $R = k/n$. Any linear code can be represented through a *generator matrix* $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, such that each information vector $\mathbf{u} \in \mathbb{F}_q^k$ is mapped into a codeword $\mathbf{c} = \mathbf{u}\mathbf{G}$. A special class of linear codes over $\mathbb{F}_q$ is that of RS codes. An RS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with length $n \leq q$ and dimension $1 \leq k < n$ is defined[1] as

$$\mathcal{C} = \left\{ \left(g(a_1), \ldots, g(a_n)\right) \mid g \in \mathbb{F}_q^k[x] \right\}, \qquad (3)$$

where $\{a_1, \ldots, a_n\}$ are distinct elements of $\mathbb{F}_q$ and $\mathbb{F}_q^k[x]$ is the set of polynomials with coefficients over $\mathbb{F}_q$ and maximum degree $k-1$. In our model, malicious parties intentionally introduce *erasures*: an erased symbol assumes an unknown value, and cannot be singularly recovered by the receiver. It can be proven that any code defined by (3) can fill up to $n-k$ erasures, *i.e.*, any codeword can be recovered from any set of k of its non-erased symbols using a conventional decoding algorithm [7]. Normalizing with respect to the code length, we denote with $\gamma = \frac{k}{n} = R$ the minimum fraction of non-erased symbols allowing for full codeword recovery.

C. Blockchain Technology

The core of the blockchain technology is a distributed ledger having the form of a chain of blocks, where the link between adjacent blocks is obtained through cryptographic functions. Each block contains an ordered list of data units called *transactions*, except for the first one, which is fixed according to the particular blockchain and initiates the ledger. The blockchain is driven by a *consensus protocol*, that sets all the rules to verify a block and append it to the chain. A validator chooses a set of valid transactions to be included in a block and proposes it to the network according to the consensus protocol. The transactions included in the block are used as leaves to build a Merkle tree. The root of the Merkle tree (*i.e.*, the Merkle root) is stored, together with some additional information, in the block header.

We distinguish between *full nodes* and *light nodes*. Full nodes participate in the blockchain with all the rights and duties. Namely, they are able to download and store the entire ledger and can actively participate in the consensus mechanism (*i.e.*, proposing and validating blocks). On the contrary, light nodes do not have enough resources to store a copy of the entire ledger and/or to participate in consensus. For this reason, light nodes have a somehow limited capacity to interact with the network. In fact, they do not participate in consensus (*i.e.*, they cannot propose and validate blocks), and store only the block headers (instead of the full blockchain). Light nodes cannot autonomously verify the validity of transactions, and are only interested in verifying its inclusion in a valid block; they accomplish this task by means of Merkle proofs.

D. Data Availability Attacks

Let us consider the situation in which a block is not available to all the full nodes participating in the network. This may be due to one of two possible reasons: either the block is valid but, due to fluctuations in the network synchronization, it has not still been received by all the network nodes, or the block is not valid, and the invalid portions have been withheld by malicious nodes. Light nodes do not have a reliable way to distinguish between these two cases. So, a malicious full node can take advantage of this situation and try to deceive the network, which happens when: i) the contents of a block are not available to the honest full nodes, and ii) at least one light node accepts the block. Also notice that, in such a case, the malicious node can put invalid transactions in the unrevealed part; so, it can ultimately make a light node accept an invalid transaction. In principle, honest full nodes could raise an alarm any time some part of a block is missing, but this could occur every time network synchronization slows down, with the consequence of flooding the network with many false alarms. Moreover, malicious full nodes could raise fake alarms, preventing light nodes from accepting valid blocks.

[1] According to some authors (see, for instance, [6, Chapter 10]), RS codes are only defined with maximum length, i.e., $n = q - 1$. According to such a narrower definition, the codes described by (3) correspond to shortened RS codes, or to a special case of Generalized RS codes. However, this little inconsistency in the nomenclature does not affect the properties of the family of codes we consider.

In order to tackle these attacks, a better solution consists in relying on *fraud proofs*, *i.e.*, objects produced by full nodes to prove that a certain header is associated to an invalid block. Upon receiving a fraud proof, light nodes become aware of the fraud and reject the block. We describe next the ASBK protocol, which has been the first one to use fraud proofs.

III. The ASBK protocol

Let us consider the same threat model and network topology as in [2]. In particular, we assume that the majority of full nodes is dishonest and produces invalid blocks. The network is supposed to be reliable and partially asynchronous (*i.e.*, the maximum delay is finite), with peer-to-peer authenticated communications. Full nodes can communicate among themselves and with light nodes; additionally, we assume that malicious nodes can collude. Light nodes cannot communicate among themselves, but can query full nodes with completely anonymized requests. Such an assumption, which in [2] is referred to as *enhanced model*, is crucial to the scheme functioning, and involves that a malicious node receives all the requests together and in mixed order, without any information about the sender. Concerning the topology, we assume that each light node is connected to at least one honest full node, while every full node is connected to m light nodes.

A. Protocol Description

We generalize the setting in [2] and consider RS codes having whichever value of code rate. The ASBK protocol works by encoding the list of transactions in each block as a codeword $\mathbf{c}$ of a 2D-RS code. In a nutshell, 2D encoding is performed through a product code, employing as component codes two identical RS codes with length n', dimension k' and rate R', defined over a finite field with $q \geq n'$ elements. Hence, the resulting product code has length $n = n'^2$, dimension $k = k'^2$, and, consequently, rate $R = R'^2$. It can be easily proven that such a construction yields a code that can recover up to $(n' - k' + 1)^2 - 1$ erasures[2], and hence

$$\gamma = \frac{n'^2 - (n' - k' + 1)^2 + 1}{n'^2} = 1 - (1 - R' + 1/n')^2 + \frac{1}{n'^2}$$
$$\approx 1 - (1 - R')^2 = R'(2 - R'). \quad (4)$$

After receiving the header of a new block, light nodes start querying the connected full nodes, asking for random symbols of $\mathbf{c}$ together with the Merkle proof. Each received symbol is then gossiped to the connected full nodes. This way, honest full nodes will receive further entries of $\mathbf{c}$ and, upon reception of enough symbols, they become able to retrieve the whole block through RS decoding: in case the retrieved block includes invalid transactions, they will produce the fraud proof and deliver it to light nodes, that will consequently reject the block. Thus, in order to prevent honest full nodes from retrieving the whole block through RS decoding, malicious nodes neglect, *i.e.*, do not reply to, some of the light nodes queries. This is the only possibility for malicious nodes to deceive the network. Indeed, the light nodes that do not receive an answer will put the block in a pending state, but all the other light nodes (*i.e.*, the ones for which all queries are replied) will accept the block, causing a fork in the blockchain.

In the enhanced network model, all queries are anonymous: malicious nodes choose the requests to be neglected only on the basis of the asked symbols, and not of the sender. This means that there is still some probability that malicious nodes are able to deceive the network. We call such a probability *adversarial error probability*, and describe in the following section how it can be computed.

B. Protocol Analysis

Let us recall the analysis in [2] to estimate the adversarial success probability. We remind that adversaries succeed whenever full nodes are unable to recover an invalid block through decoding and, at the same time, there is at least one light node that accepts the block, since all its queries have been replied.

Depending on the symbols asked by light clients, malicious nodes will behave differently, with the purpose of maximizing the probability to deceive the network. Let J_i denote the set of indexes of symbols asked by the i-th light node, and let $J = \bigcup_{i=1}^{m} J_i$. Also, let E be the set of indexes of the symbols which have been initially hidden by malicious nodes; we denote $|E| = \beta n$, where $\beta \in [\![0; 1]\!]$, *i.e.*, the set of rational numbers between 0 and 1. The fraction of symbols available to full nodes at the end of the sampling process, if the malicious node replied to all queries, is given by φn, with

$$\varphi = 1 - \beta + \frac{|J \cap E|}{n}, \quad (5)$$

which is easily obtained by summing the fraction $1 - \beta$ of symbols already known to full nodes to the fraction of symbols obtained by light nodes through sampling. The value of φ, which can be computed by the adversaries on the run, determines their behavior, namely:

A) if $\varphi < \gamma$, full nodes will not be able to decode: the malicious node will reply to all queries, and all light nodes will accept the block;

B) if $\varphi \geq \gamma$, the malicious node must avoid replying to some queries. Let d be the minimum number of indexes from $E \cap J$ which, if not revealed, would make the block undecodable. It can be easily seen that it must be $d = \delta n + 1$, where $\delta = \varphi - \gamma$. Note that, by adapting their behavior to the light nodes requests, malicious nodes maximize the probability to have at least one light node for which all queries are correctly replied.

The sampling process can be seen as an instance of the coupon's collector problem in which the variables (v, y, z, w), defined in Section II-A, take the values $(n, m, s, \beta n)$. Indeed, there is a group of m light nodes, each asking for s distinct symbols with indexes in $\{1, \ldots, n\}$; in our case, the set W corresponds to that of hidden symbols, thus $w = \beta n$. Hence, in order to estimate the probability associated to condition A),

[2] We remark that the analysis in [2] is restricted to the case of $R' = \frac{1}{2}$ and therefore, as shown in [2, Theorem 1], the code is able to fill up to $(k' + 1)^2 - 1$ erasures.

it is enough to plug $x = (\gamma + \beta - 1)n - 1$ into (2).[3] In the following, we will call $\Pr[\mathsf{NoDec}]$ the resulting probability.

Moreover, in order to assess the adversarial success probability, one should also take into account condition B), which happens with probability $1 - \Pr[\mathsf{NoDec}]$. Then, we have to consider the probability that there is at least one light node that receives all the asked symbols, which we are going to denote as $\Pr[\mathsf{Deny}]$. Putting all of this together, we finally obtain that the adversarial success probability is

$$\epsilon = \Pr[\mathsf{NoDec}] + (1 - \Pr[\mathsf{NoDec}]) \Pr[\mathsf{Deny}]. \tag{6}$$

IV. A General Analysis of the ASBK scheme

In this section, we describe how the analysis proposed in [2] can be simplified and generalized. We still consider 2D-RS codes, but with variable rate $R \in [\![0;1]\!]$ as a parameter to optimize. In addition, we get rid of hard-to-implement formulas, in favour of a simpler and more intuitive analysis.

We first notice that, in order to implement (2) in our case, a significant amount of computational resources is needed. Indeed, a rigorous computation of (2) requires a number of operations which is $O\left(n^2 + n^2 \log_2(m)\right)$, having assumed that computing a binomial $\binom{a}{b}$ costs $O(b)$ operations, and that computing a^b costs $O\big(\log_2(b)\big)$ operations. Considering that we expect to have $n \sim 10^4 \div 10^5$ and that we need to numerically search for optimal parameters, we observe that performing these computations may be not easy. Note that there exist ways to speed-up the process: for instance, one can approximate binomials and can also rely on precomputation to reduce the number of operations to execute on the run. Still, the computational burden remains significant, which motivates the need for a simpler analytical model like the one described in the next section.

A. Simple Theoretical Model

First of all, we consider $\beta = 1$, *i.e.*, we assume that malicious full nodes hide the whole content of a block. In fact, from the analysis in Section III, it is clear that the most convenient strategy for malicious nodes is to reveal symbols only when queried by light nodes. Starting from this observation, the analysis of the ASBK protocol can be significantly simplified, by employing some basic approximations.

Proposition 1 Let us consider the ASBK protocol with $\beta = 1$, employing a 2D-RS code with length n and able to fill up to $(1-\gamma)n$ erasures. Let m be the number of light nodes, each querying a malicious node by asking for s random and distinct symbols. Then, the adversarial success probability can be well approximated as

$$\epsilon = \begin{cases} 1 & \text{if } x^* < \gamma n, \\ 1 - \left(1 - \frac{\binom{\gamma n - 1}{s}}{\binom{x^*}{s}}\right)^m & \text{otherwise,} \end{cases} \tag{7}$$

where $x^* = n\left(1 - \left(1 - \frac{s}{n}\right)^m\right)$.

Proof: We assume that, in every execution of the protocol, the number of distinct symbols requested by the light nodes altogether is equal to its expected value that, recalling (1) with the appropriate notation, can be estimated as

$$x^* = \langle x \rangle = n\left(1 - \left(1 - \frac{s}{n}\right)^m\right). \tag{8}$$

We remind that at least γn symbols are necessary to decode the employed 2D-RS code. Hence, when $x^* < \gamma n$, we can set $\Pr[\mathsf{NoDec}] = 1$ and, recalling (6), we get $\epsilon = 1$. When instead $x^* \geq \gamma n$, decoding is always successful, and, consequently, we set $\Pr[\mathsf{NoDec}] = 0$. In such a case, the best possible strategy for the adversary is to ignore some queries, keeping d symbols hidden. In order to prevent decoding, it must be $d > x^* - \gamma n$; hence, in order to maximize the success probability, the adversary sets $d = x^* - \gamma n + 1$.

We know that the ensemble of light nodes asks for a total of x^* symbols; let $J \subseteq \{1, \ldots, n\}$ be the set of positions pointing at such x^* symbols. We can consider that each light node randomly selects s distinct indexes of J. Let $D \subseteq J$ be the set of indexes pointing at the d positions of the symbols that the malicious node keeps hidden: a light node will not put the block in pending state if and only if it only requests positions coming from $J \setminus D$. We now proceed by computing the probability that there is at least a light node that only requests symbols in positions indexed by $J \setminus D$, whose size is $x^* - d$. The probability that a single light node asked for a symbol in D is $p = 1 - \binom{x^*-d}{s} / \binom{x^*}{s}$. Note that p corresponds to the probability that a single light node is not misled, *i.e.*, does not end up accepting the block. Since there are m light nodes operating independently, we have that the adversarial success probability, *i.e.*, the probability that there is at least one light node that did not select positions from D, is obtained as the complementary of the probability that every light node asked at least a symbol coming from D. Then

$$\epsilon = 1 - p^m = 1 - \left(1 - \frac{\binom{x^*-d}{s}}{\binom{x^*}{s}}\right)^m.$$

Considering $d = x^* - \gamma n + 1$, we prove the thesis. ■

In order to confirm the validity of our analysis, we have run numerical simulations of the ASBK sampling process. We have measured the adversarial success probability and compared it with the ones obtained through Proposition 1; results are shown in Fig. 1. We observe a good matching between the analytical values and the simulated ones.

B. Model Introspection and Asymptotic Analysis

Let us write $x^* = n(1-\sigma)$, with $\sigma \in [\![0;1]\!]$, and obtain from (8) that $\ln \sigma = m \ln\left(1 - \frac{s}{n}\right)$. Since it is desirable to have $s \ll n$, we have $\ln\left(1 - \frac{s}{n}\right) \approx -\frac{s}{n}$, from which $ms \approx n \ln(1/\sigma)$. We want full nodes to be able to decode; hence, taking into account (4), it must be $\sigma \leq 1 - \gamma \approx 1 - R'(2 - R')$. We can then derive a lower bound on the product ms as

$$ms > n \ln\left(\frac{1}{1-\gamma}\right) \approx n \ln\left(\frac{1}{1 - R'(2-R')}\right). \tag{9}$$

[3] The authors in [2] provide a formula (see Theorem 4) which is essentially the same, apart from some rewriting. We have chosen a different formulation only to simplify the treatment in the subsequent computations.

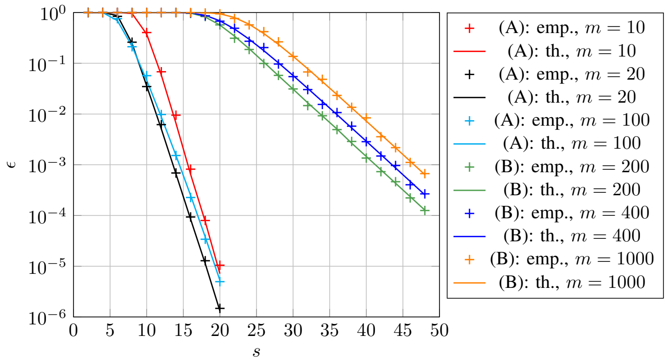

Fig. 1. Simulated values of ϵ and comparison with its theoretical values from (7). The following two settings are considered: (A) $(n, \gamma) = (100, \frac{1}{2})$, (B) $(n, \gamma) = (1000, \frac{3}{4})$.

The above condition gives a rough idea of the setting one should choose in ASBK. Indeed, we have that when (9) is not satisfied, then, with high probability, full nodes will not be able to decode. Additionally, (9) explicitly expresses the relation between the component code rate and the total number of samples which are asked for full nodes (that is, the value of ms). If one wants to keep the value of s as low as possible, assuming that the number m of light nodes is fixed, then the only possibility consists in reducing the code rate R'. Notice that, by doing this, one also reduces the value of γ, which is the reason why a smaller number of queries is required: the obtained 2D-RS code can fill a larger number of erasures.

As we know, the number of nodes participating in modern blockchain networks is rapidly increasing. This implies that the number of light nodes is increasing as well; so, it is worthwhile analyzing the ASBK protocol also in this regime, *i.e.*, when m becomes extremely large. In particular, as m grows, from (8) we see that x^* rapidly tends to n. In the limit of $x^* = n$, from Proposition 1 we obtain

$$\epsilon = 1 - \left(1 - \frac{\binom{\gamma n - 1}{s}}{\binom{n}{s}}\right)^m.$$

We observe that, whenever $s \ll (1-\gamma)n$, we can set $\binom{\gamma n-1}{s}/\binom{n}{s} \approx \gamma^s$. Exploiting such an approximation, we find

$$s \approx \frac{\ln\left(1 - (1-\epsilon)^{\frac{1}{m}}\right)}{\ln \gamma}. \tag{10}$$

Since ϵ is relatively small and m is relatively large, we can make use of the following further approximations

$$(1-\epsilon)^{\frac{1}{m}} \approx 1 + \frac{\ln(1-\epsilon)}{m}, \quad \ln(1-\epsilon) \approx -\epsilon,$$

which permit us to rewrite (10) as

$$s \approx \frac{\ln(m) + \ln\left(\frac{1}{\epsilon}\right)}{\ln\left(\frac{1}{R'(2-R')}\right)}. \tag{11}$$

We first notice that increasing the block size has basically no impact on the value of s which is required to reach a desired adversarial success probability. However, the value of s grows with the network size (which is indirectly measured by m), but the growth rate is only logarithmic.

V. Results

In this section we validate our analysis, finding optimal settings for the ASBK protocol[4]. In order to consider a case of practical interest, we choose block sizes which are similar to those used in common blockchains (such as Ethereum), and find the optimal protocol setting for different network sizes.

We first define the link between code parameters and block size, assumed to be equal to ℓ_b bits. Considering a code defined over $\mathbb{F}_q$, we need to use component RS codes with dimension

$$k' = \left\lceil \sqrt{\frac{\ell_b}{\log_2 q}} \right\rceil,$$

which guarantee that the list of transactions fits into a square block with side k' (if needed, padding is employed to fill all the matrix entries). We now proceed by determining the header size. We exclude from our analysis all the header elements which do not depend on the code design, such as the hash of the header of the previous block, since they provide a constant contribution which is the same in all the considered cases. Consequently, we assume that the header only contains the Merkle roots of the encoded block, which are $2n'$. By denoting with $\ell_{\mathcal{H}} = 256$ the binary length of the digests, we have that the header size is $2n'\ell_{\mathcal{H}}$. We now consider that the reply to each light node query is composed by an element of $\mathbb{F}_q$ and its Merkle proof. A Merkle proof has size $\lceil \log_2(n') \rceil \ell_{\mathcal{H}}$, while each symbol of $\mathbb{F}_q$ is represented by $\lceil \log_2(q) \rceil$ bits. Hence, a light node downloads a total amount of data, in bits, given by

$$\ell_D = 2\ell_{\mathcal{H}} k'/R' + s\big(\lceil \log_2(q) \rceil + \ell_{\mathcal{H}} \left\lceil \log_2(k'/R') \right\rceil\big).$$

In order to provide parameters with practical interest, we consider a block size of 75 kB[5], yielding $\ell_b = 600{,}000$.

Firstly, we consider $q = 2^{256}$ and several values of R', and for each configuration we find the minimum value of s for which the adversarial success probability is below the target 0.01 (as in [2]). The corresponding values of ℓ_D are shown in Fig. 2, where we consider different values of m and, for each curve, we highlight the value of m yielding the minimum ℓ_D. We observe how the amount of downloaded data decreases for low code rates, reaches its minimum and then increases for higher code rates. Moreover, for low rates, the number of samples (or downloaded data) decreases as m increases, while the opposite tends to occur for high rate values. As expected, the optimum working point depends on m: in a real network, the value of R' should be adjusted as the network size changes.

Note that, as another degree of freedom, one may change q. In Table I we have reported the optimal rate values, found for several values of q when $m = 1000$. To quantify the gain with

[4] The software programs used to obtain the results in this paper are available at https://github.com/secomms/blockchainRS

[5] This is the average block size of the Ethereum network in August 2021. Data are extracted from Etherscan (https://etherscan.io/chart/blocksize).

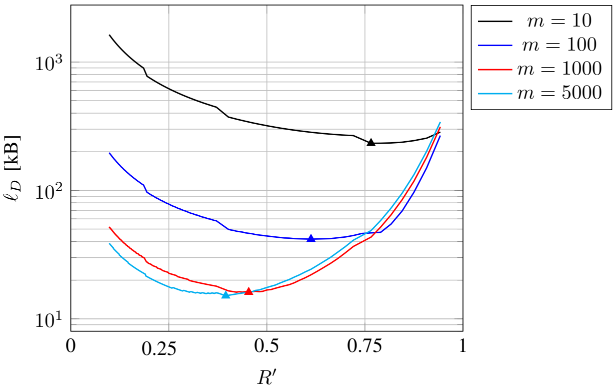

Fig. 2. Amount of downloaded data as a function of the component code rate R', when $q = 2^{256}$.

TABLE I
OPTIMAL RATES AND CORRESPONDING VALUES OF ℓ_D AND s, FOR $m = 1000$ AND $\epsilon = 10^{-2}$.

q	k'	R'	n'	ℓ_D	s	$\tilde{\ell}_D$	$\tilde{s}$
2^{16}	194	0.647	300	84.740	226	92.112	232
2^{32}	137	0.591	232	48.128	136	54.912	128
2^{64}	97	0.545	194	31.984	78	32.480	76
2^{128}	69	0.548	126	21.504	56	22.704	51
2^{256}	49	0.430	114	16.256	35	16.768	41
2^{512}	35	0.402	87	13.344	27	15.424	38
2^{1024}	25	0.321	78	11.680	19	15.040	37
2^{2048}	18	0.295	61	11.072	16	17.984	35
2^{4096}	13	0.224	58	12.160	12	23.840	33
2^{8192}	9	0.155	58	14.656	10	36.672	30

respect to the case of $R' = 0.5$, the last two columns report the values of ℓ_D and s when $R' = 0.5$ (which we have denoted, respectively, as $\tilde{\ell}_D$ and $\tilde{s}$). As we see from the table, for all the considered cases, the optimum rate never corresponds to 0.5. We further notice that the finite field size q plays a crucial role in determining the amount of data each light node downloads. Indeed, it appears that using a rather large finite field makes the value of ℓ_D decrease significantly. For example, in our scenario, ℓ_D achieves its minimum for $q = 2^{2048}$. At a first glance, such a large q may appear unpractical. However, notice that sums and multiplications in $\mathbb{F}_q$ cost $O\big(\log_2(q)\big)$ and $O\big(\log_2^2(q)\big)$, respectively. For instance, $\mathbb{F}_{2^{2048}}$ is 2^{1792} times larger than $\mathbb{F}_{2^{256}}$, but sums and multiplications in $\mathbb{F}_{2^{2048}}$ are expected to cost only 8 and 64 times as much than in $\mathbb{F}_{2^{256}}$, respectively. Furthermore, some specific choices for the finite field construction may lead to computational advantages (see [8], for instance). In any case, in a practical situation, the choice on q should be based (also) on the computational resources actually available to full nodes.

Finally, we study how the number of asked samples varies as m grows, for several values of q. The obtained results are reported in Fig. 3, where we also show the values resulting from (11). As expected, as m grows, the values of s tend to become closer and closer to those given by (11). Moreover, as correctly predicted by (11), s does not depend on q and n and, therefore, the block size has no impact. In other words, s is only a function of R' and m. This implies that, for very heavily participated networks, one can increase the amount of data in each block without any evident effect on the computational burden of each light node.

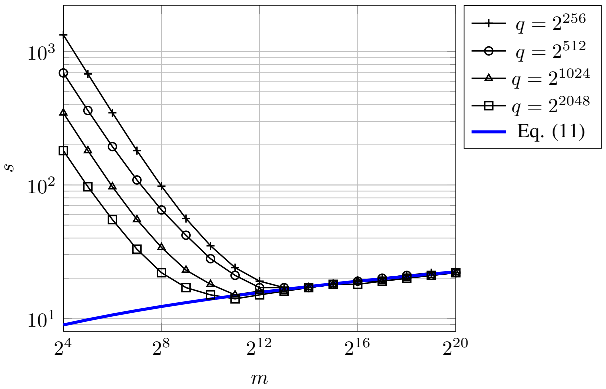

Fig. 3. Minimum number of samples to achieve $\epsilon \leq 10^{-2}$, as a function of m; the code rate is $R' = 0.25$.

VI. CONCLUSIONS

We have presented a novel mathematical model for the ASBK protocol, a recently proposed blockchain protocol which counters data availability attacks through 2D-RS codes. Differently from existing analyses, our study does not fix any code parameter and embeds easy-to-implement formulas. This allows for a deeper understanding of the protocol features, and ultimately provides a simple method to devise optimal settings. Namely, our approach allows to settle the best code parameters (e.g., rate and finite field size) to minimize the amount of data each light node downloads, given a desired adversarial success probability. Our results show that the ASBK protocol benefits from the use of component codes with rate different from the value $1/2$, fixed in the original proposal.

REFERENCES

[1] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] M. Al-Bassam, A. Sonnino, V. Buterin, and I. Khoffi. (2021) Fraud and data availability proofs: Detecting invalid blocks in light clients. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/M.AlBassam/publications/fraudproofs.pdf

[3] M. Yu, S. Sahraei, S. Li, S. Avestimehr, S. Kannan, and P. Viswanath, "Coded Merkle tree: Solving data availability attacks in blockchains," in *Financial Cryptography and Data Security, FC 2020*, ser. Lecture Notes in Computer Science, N. H. J. Bonneau, Ed., vol. 12059. Springer, Cham, 2020, pp. 114–134.

[4] D. Mitra, L. Tauz, and L. Dolecek, "Concentrated stopping set design for coded Merkle tree: Improving security against data availability attacks in blockchain systems," in *Proc. ISIT 2020*, Los Angeles, CA, USA, Jun. 2020, pp. 136–140.

[5] W. Stadje, "The collector's problem with group drawings," *Advances in Applied Probability*, vol. 22, no. 4, pp. 866–882, Dec. 1990.

[6] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. Elsevier, 1977, vol. 16.

[7] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. USA: Prentice-Hall, Inc., 2004.

[8] A. Maximov and H. Sjoberg, "On fast multiplication in binary finite fields and optimal primitive polynomials over GF(2)." *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 889, 2017.