A Mathematical Approach on the use of Integer Partitions for Smurfing in Cryptocurrencies

No Author Given

No Institute Given

Abstract. In this paper, we propose the modelling of patterns of financial transactions - with a focus on the domain of cryptocurrencies - as splittings and present a method for generating such splittings utilizing integer partitions. We further exemplify that, by having these partitions fulfill certain criteria derived from financial policies, the splittings generated from them can be used for modelling illicit transactional behavior such as is shown by smurfing.

Keywords: Money laundering · Smurfing · Integer Partition.

1 Introduction

Money laundering is a global problem that affects all countries to various degrees. Criminal organizations and terrorist groups move billions of dollars each year through the international banking system to hide the source of their funds. The term *money laundering* describes the process by which the proceeds of crime and the true ownership of those proceeds are concealed or made opaque so that the proceeds appear to come from a legitimate origin [11]. Criminals do this by disguising the sources, changing the form, or moving the funds to a place where they are less likely to attract attention. In the money laundering process the following three stages are commonly identified [18], as visualized in Figure 1.

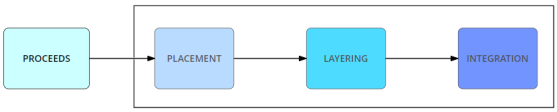

Fig. 1. Process of ML.

In the placement stage, illegal funds are introduced into the financial system, which can be done in various ways, such as depositing small cash amounts or purchasing high-value assets. In the layering stage, launderers attempt to separate the illicit funds from their original source by creating complex financial transactions that make it difficult to trace the money back to its origin. The

integration stage involves reintroducing the laundered funds into the economy, making the money legally available in the financial system [17].

With the emergence of digital currencies, the use of cryptocurrencies in money laundering has become more attractive than fiat currency due to increased anonymity in financial transactions. This has led to the development of a concept known as crypto laundering, which utilizes the features of cryptocurrencies, such as anonymity, decentralization, and mixing services to conceal the proceeds of illegal activities [23]. The user-friendly nature of crypto exchanges, which provide anonymity and operate 24/7, lowers the entry level barrier for laundering compared to fiat currencies [19]. However, there are several challenges associated with monitoring and catching illegal activities in crypto exchanges, including user identities/anonymity, transaction speed, and structured deposits.

The act of *smurfing* in finance is a term describing a common technique *within the layering stage* used by money launderers to conceal illicit activities by breaking up noticeable transactions into smaller inconspicuous ones in order to avoid detection. An example visualizing smurfing is shown in Figure 2: black money is separated into different transactions from account A to 4 different recipients B, C, D and E and finally collected in a single account F again.

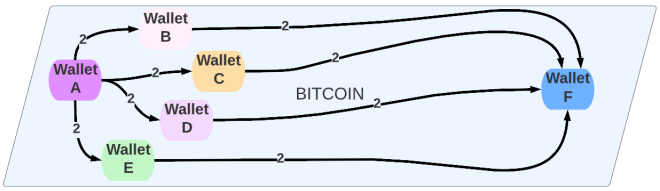

Fig. 2. Smurfing.

Smurfing is considered to be a major threat to the integrity of financial systems as it makes the detection of illicit financial transactions a significant challenge for authorities. The increase in these concerns has also led to an increase in the importance of anti-money laundering transactions monitoring systems. The primary purpose of the Anti-Money Laundering transactions monitoring system is to identify potential suspicious behavior embedded in legitimate transactions.

In this work, we will present a modelling approach for smurfing in the case of cryptocurrencies using combinatorics, more specifically, integer partitions. Derived from this model, we provide a constructive approach for splitting up large and conspicuous financial transactions, such that the resulting set of smaller transactions offer enhanced capabilities for avoiding detection in a real world financial setting. Our approach will consider as input a specific transaction amount and output one or all possible splittings of this amount while taking into account a set of criteria derived from regulations, such as fixed thresholds indicating suspicious transactions, as well as the currency used. The aim is to construct a set of

splittings respecting certain criteria with the intention to give rise to transactions able to bypass regulations.

This paper is structured as follows. In Section 2 we provide an overview of the related work. We then continue in Section 3 where we present the details of our proposed mathematical modelling for smurfing. We further provide illustrative examples for a better understanding on how our proposed model can be used in Section 3.3. Finally, we conclude this work in Section 4.

2 Related work

Several studies have explored transaction monitoring for AML efforts, including the work of [24], in which the authors developed a novel approach for detecting and disrupting money laundering activities using a combination of network-based clustering and classification techniques.

Multiple works [7,6,10,8] considered monitoring system to identify potential suspicious behaviors embedded in legitimate transactions.

In [13] a framework is proposed that focuses on revealing the networks of launderers involved in a money laundering act using social network analysis. A framework was created that uses sequence matching, case-based analysis, social network analysis, and complex event processing to detect the evolution of money laundering schemes (MLS) and a system was implemented to include social network analysis for detecting and linking related money laundering scheme networks.

In [20], a novel and efficient method for identifying suspicious smurf-like subgraphs was presented, which is based on the velocity characteristics of smurfing and uses a standard database join to bypass computational complexity. The approach was tested on a real-world transaction graph, containing a massive number of transactions and bank accounts, and was found to be highly effective. Additionally, the study provides valuable insights into the different ways that money launderers exploit geography and other factors in their illegal activities.

The work presented in [21] describes a visualization system for analyzing Bitcoin wallets which allows investigators to monitor the flow of bitcoin and point to potential criminal activities such as money laundering. The system provides valuable information about transactions and addresses involved in illegal activities. That work also includes case studies that demonstrate its effectiveness in real-world investigations, making it a valuable tool for law enforcement battling digital crime.

With regard to the rapid global digitalization and the emanating growth of the digital economy, the adaptation of information and communication technology (ICT) networks for illicit financial transactions is becoming one of the key challenges for authorities [22], with the emergence of new tools and methods, making illicit transactions increasingly easy to hide and thus lowering the entry barrier for criminals. In addition, the decentralized architecture of the internet also provides a big challenge for authorities in the detection and tracking of illicit financial flows [22]. This development is reflected in the 2022 European Union

Agency for Criminal Justice Cooperation (EUROJUST) report on the subject of money laundering [5], which reports an increasing misuse of cryptocurrencies for money laundering purposes.

In response to the increasing relevance of the digital economy in the financial system, the European Union introduced a new regulatory framework for Markets in crypto assets (MiCA) [4], which also addresses the risks and challenges posed by the now widespread use of crypto assets in the digital economy and includes measures against market manipulation, money laundering, terrorist financing and other criminal activities [15]. The framework encompasses not only crypto assets themselves, but also their issuers, as well as service providers, bringing them under a comprehensive regime for the first time.

3 Using Integer Partitions to create Patterns for Smurfing

In this section, we present the details of our proposed combinatorial modelling approach for certain transactional behaviours with potential subsequent use in money laundering. More precisely, we consider how to initially determine and then subsequently construct a complete split of a given amount of money (possibly from illicit sources) into multiple, smaller transactions which may avoid detection.

Integer partitions and mathematical notation First, as we will extensively use integer partitions for modelling certain transaction behaviours with potential links to money laundering, we consider the mathematical definition of an integer partition as follows:

Definition 1 *[cf. [3]] A* partition *of a positive integer n is a finite non-increasing sequence of positive integers $(\lambda_1, \lambda_2, \ldots, \lambda_r)$ such that $\sum_{i=1}^{r} \lambda_i = n$. The λ_i are called the* parts *of the partition.*

To illustrate the concept of integer partitions and the corresponding notation, we give an easy example.

Example 1. The integer 5 has the following 7 partitions: $(1,1,1,1,1)$, $(2,1,1,1)$, $(2,2,1)$, $(3,1,1)$, $(3,2)$, $(4,1)$ and (5).

Several modern computer algebra systems (e.g. Maple [12], R [1], WoframAlpha [25] or SageMath [16]) offer the functionality to compute integer partitions.

Now we introduce further mathematical notation we will use: We write $\mathbb{N}^{<\mathbb{N}}$ to denote the set of finite sequences of natural numbers, i.e.,

$$\mathbb{N}^{<\mathbb{N}} = \{p \mid p \text{ is a finite sequence of natural numbers}\}$$

and for a set X we write $[X]^{<\mathbb{N}}$ for the set of finite subsets of X, i.e.,

$$[X]^{<\mathbb{N}} = \{A \subseteq X \mid A \text{ is finite}\},$$

in particular we are interested in finite sets of sequence of natural numbers,

$$[\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} = \{A \mid A \text{ is finite and } \forall p \in A\ p \text{ is a finite sequences of natural numbers}\}.$$

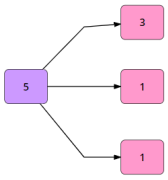

Fig. 3. One of the integer partitions of 5.

3.1 Problem Formulation

To model the financial activities, we make use of a standard account-based view on transactions. We express the financial transactions as a weighted directed graph with the nodes of the graph representing wallets or accounts in the considered currency, while transactions are directed edges between them and the respective amount of the transactions are the weights of the edges. The transaction patterns caused by smurfing are the consequence of splitting up an initial (large) amount of money via multiple transactions. Therefore, integer partitions (see Subsection 3) as basis for a mathematical model of smurfing are a natural choice. To make the connection between integer partitions and our considered directed graph model of financial transactions, note that in a weighted directed graph each node admits an integer partition of the sum of the weights of its out-edges and the parts of the partition appear as weights of the out-edges.

> **Problem:** Given an amount of money in some currency and some regulations, find all integer partitions of the corresponding integer with properties such that the resulting transactions are not affected by the regulations.

The input to this problem contains, for the scope of this work, the initial total amount of money expressed in the smallest unit of the used currency[1], applicable monitoring regulations on transactions in the corresponding currency as well as additional properties of the desired splittings like the number of (or bounds on) the total number of resulting transactions or the sizes of the occurring individual parts. We consider as solution, for the scope of this work, a splitting of the given amount fulfilling the listed requirements given in the problem input.

To tackle this problem, we need to find a function which takes as input an amount of money (in an arbitrary currency) and gives as output possibilities to transfer the money split up in one or more transactions.

$$\text{amount} \mapsto \text{possible splitting(s) for smurfing of the amount}$$

Each amount (expressed in the indivisible unit of the currency) gives a natural number and each possible splitting can be expressed as a finite sequence of

[1] For the purposes of this work, we assume the existence of a smallest currency unit; in the real world, this assumption holds – in particular – for the US-Dollar with Dollar cents, for the Euro with Euro cents and for BTC with *Satoshi*.

natural numbers summing up to the original amount. This means we are looking for a function

$$S\colon \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}}$$

such that[2]

$$\text{for each } n \in \mathbb{N} \text{ and each } p \in S(n) \quad \sum_{i=1}^{\text{length}(p)} p(i) = n. \tag{1}$$

Since (1) holds for S if and only if for each $n \in \mathbb{N}$ each $p \in S(n)$ is an integer partition of n clearly the following function fulfills (1):

$$\begin{aligned} S\colon\ & \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} \\ & n \mapsto \{p \mid p \text{ is an integer partition of } n\} \end{aligned}$$

To get functions which offer more modelling capabilities with potential advantages for smurfing additional requirements on $S(n)$ can be added.

The impact of a specific given integer partition on the corresponding smurfing is that its encoded *splitting* determines the size and number of transactions involved. Several practical aspects have to be taken into account, such as on the one hand if the size of the individual parts of the integer partition is too small, then it may result in too many transactions, making it difficult to actually perform them. On the other hand, if the parts are too large, they may attract unwanted attention and increase the risk of detection. Hence, considering various properties of integer partitions gives precise modelling capabilities of the resulting transactions. In Table 1, we provide a mapping of these properties to the application domain of smurfing, followed by a detailed explanation.

Table 1. Translation table for smurfing and discrete mathematical model via integer partitions.

Smurfing	**Integer partition**
total money value	integer
number of recipients	number of parts in the integer partition
transaction amounts	parts of the integer partition
maximal/minimal value of transaction amounts	maximal/minimal size of parts of integer partition
difference between transaction amounts	difference between parts of integer partition

Total money value: For layering via smurfing, the amount of money has to be split into smaller amounts. The total money value is expressed in terms of the underlying indivisible unit of the currency and all possible splittings can be found by computing all integer partitions of this integer.

[2] Each element p of $S(n)$ is a finite sequence, we write $p = (p(1), p(2), \ldots, p(\text{length}(p))$

Number of recipients: The given amount of money is split into smaller parts and each part gets transferred to a different recipient, which leads to one of the main parameters of smurfing: the number of recipients. Assuming that each recipient receives exactly one part, the number of recipients is equal to the number of parts in the corresponding integer partition.

Transaction amounts: Recall that each recipient gets a part of the initial total amount of currency. The individual amounts can all be the same or may differ from each other, the only thing which is invariant is that they have to sum up to the total initial currency value. Hence, the amounts sent in the individual transactions during the smurfing process, again expressed in the underlying indivisible unit of the currency, map to the individual parts of the corresponding integer partition.

maximal/minimal value of transaction amounts: This denotes the maximal value a transaction is allowed to have to satisfy the respective constraints derived from regulatory policies and therefore not to raise suspicion. Fixating the minimal amount of a transaction ensures that the number of splittings generated stays within a viable range.

difference between transaction amounts: This values provides a range for the difference between subsequent transactions. This prevents generating equally large transactions and provides some variance between the transactions with the goal of making them seem independent of each other.

3.2 Possible Implications of the Conceptualization of Smurfing via Integer Partitions

Let us now discuss some of the consequences of our proposed mathematical model. Since there are only finitely many integer partitions for each given integer, this implies there only exist finitely many ways for the smurfing of a given initial amount of currency. Moreover, there are restrictions on the usable integer partitions for smurfing to ensure that the resulting smurfing encoded in the integer partition is *very likely to go undetected* by any *controlling entities*. In particular, the number of parts, the respective size of the parts, and the similarity between the parts are parameters to consider when trying to avoid detection, but all of these additional requirements on the admissible integer partitions reduce the number of possible ways for the smurfing process of a fixed amount.

In other words, the concealment of smurfing depends on balancing the two factors of the number and the sizes of the parts of the corresponding integer partition. If transactions above some value have to be reported to some *oversight entitiy*, then this can be avoided by using integer partitions with smaller parts than this threshold. This, however, results in a larger number of transactions which could also be conspicuous.

Stating these conditions in form of the function we are looking for, gives the following requirements for the function. Restricting the size of the partitions to at most ℓ, yields the following:

$$\begin{aligned} S^{\ell}\colon\ & \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} \\ & n \mapsto \{p \mid p \text{ is an integer partition of } n \text{ of size at most } \ell\} \end{aligned}$$

Restricting instead the size of the elements of the partitions to at most k, yields the following:

$$\begin{aligned} S_k\colon\ & \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} \\ & n \mapsto \{p \mid p \text{ is an integer partition of } n \text{ and for each } i \leq \text{length}(p)\ p(i) \leq k\} \end{aligned}$$

Combining the two restrictions on the size of the partitions and the size of the parts of the partitions:

$$\begin{aligned} S_k^\ell\colon\ & \mathbb{N} \to [\mathbb{N}^{<\mathbb{N}}]^{<\mathbb{N}} \\ & n \mapsto \{p \mid p \text{ is an integer partition of } n,\ \text{length}(p) \leq \ell \text{ and} \\ & \qquad\qquad \text{for each } i \leq \text{length}(p)\ p(i) \leq k\} \end{aligned}$$

Using our proposed mathematical formulation, our models are *constructive*, in the sense that mathematical software can be used to actually construct integer partitions respecting desired properties, when given corresponding parameter values, and hence compute the selected version of the function S.

In our proposed methodology for modelling smurfing with integer partitions, the steps of the corresponding procedure to use the advantages of integer partitions for potential smurfing is as follows:

1. Select the amount in some currency that should be split in the smurfing and express it in the indivisible unit of the used currency, yielding an integer n.
2. Determine the requirements for the smurfing to circumvent controls to avoid detection based on applicable regulations.
3. Translate identified requirements from the previous step into properties of integer partitions according to Table 1.
4. Use a suitable computer algebra system to compute the integer partitions of n satisfying the required properties.
5. If there are no integer partitions of n with the needed properties, try to weaken some of the requirements, and repeat the process from step 3 onwards.
6. If an integer partition of n fulfilling the requirements has been computed, select the recipients of the individual parts and then arrange the transactions.

In subsection 3.3, we present a detailed example walkthrough for steps 1 to 5 of this process.

3.3 Examples for Integer partitions and Smurfing

In this section, we give examples illustrating how mathematical software can be used to compute integer partitions for the modelling of smurfing. All presented

computations were carried out[3] in SageMath [2] in version 9.3, which is a free and open-source mathematical software system able to handle large and complex amounts of data. The computations have been performed on a machine with an Intel i5-10210U CPU with 1.60 GHz base clock and 2.11 GHz boost clock and 16GB of RAM. In SageMath, integer partitions can be computed by the function `Partitions()`, which has various parameters on which we rely on extensively in the course of this section. Furthermore, in this section, we adopt the notation from SageMath for sequences and denote them like $[\lambda_1, \lambda_2, \ldots, \lambda_r]$.

Recall from Section 3 that in the process of smurfing, a fixed total amount of money is split into smaller amounts, corresponding to an integer partition of the total amount. To make this precise, we have to express the given total amount in the underlying smallest, indivisible unit of the corresponding currency. In the examples that we give in this section, we work with the BTC cryptocurrency. For BTC, the underlying indivisible unit is called Satoshi, where an amount of 1 BTC is equal to $100,000,000$ Satoshis [14]. Hence, each smurfing process of n BTC corresponds to an integer partition of the natural number $n \cdot 100,000,000$.

In the remainder of this section, we give different examples illustrating our approach highlighting various aspects pointed out before. Specifically, in Section 3.3, we give a first walk through example of how mathematical software can be used to compute integer partitions for smurfing. In Section 3.3, we detail a more advanced example for smurfing with more practically useful assumptions, making use of more implemented options in SageMath to compute integer partitions with more restrictions.

First basic example for integer partitions and smurfing In this first example, we consider an amount of 0.00001BTC for splitting via integer partitions for subsequent potential use in smurfing. Taking into account that the smallest unit of BTC is *one Satoshi* and their ratio, we obtain that $0.00001\text{BTC} = 1,000$ Satoshis. Hence, we are interested in integer partitions of $1,000 \in \mathbb{N}$. To this end, applying the function `Partitions(1000).cardinality()` in SageMath, yields that in total there are

$$24,061,467,864,032,622,473,692,149,727,991$$

integer partitions of $1,000$.

For the purpose of smurfing, not all integer partitions are equally well suited, due to various applicable regulations on transactions. Assume that in order to bypass certain regulations, it was determined by an analysis that the total number of transactions should be between 5 and 20 and that each transaction should be between 10 and 200 Satoshis.

Depending on the chosen values and the available computational resources it might not be possible, or at least not the best option, to compute all integer

[3] Since the functionality to compute integer partitions is implemented in multiple mathematical software systems, our specific choice of computer algebra system is not important and others could be used equally well to produce the content of this section.

partitions with the given restrictions. Therefore, instead of computing all integer partitions which fulfill the restrictions, we compute them one by one and look only at the first 10.

To obtain only partitions of 1000 respecting these two constraints, in SageMath we make now use of more parameters of the function `Partitions`:

```
x=iter(Partitions(1000, min_part=10, max_part=200, max_length=20,
                                                   min_length=5))
```

Following up this command by the function `next(x)`, gives the first integer partition: $[200, 200, 200, 200, 200]$. Repeated invocation of the function `next(x)` gives the following integer partitions:

$$[200, 200, 200, 200, 200], [200, 200, 200, 200, 190, 10],$$
$$[200, 200, 200, 200, 189, 11], [200, 200, 200, 200, 188, 12],$$
$$[200, 200, 200, 200, 187, 13], [200, 200, 200, 200, 186, 14],$$
$$[200, 200, 200, 200, 185, 15], [200, 200, 200, 200, 184, 16],$$
$$[200, 200, 200, 200, 183, 17], [200, 200, 200, 200, 182, 18].$$

Any one of the above generated partitions can be used as a template for one **specific** money distribution, e.g. by using any of the above partitions for smurfing.

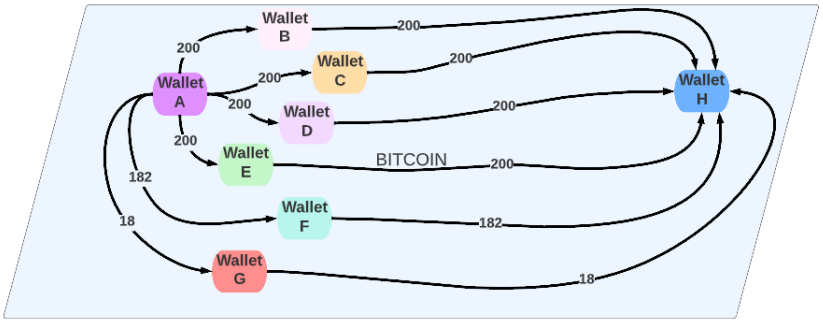

Fig. 4. One of the integer partitions used for smurfing.

In Figure 4 it is shown how one of the integer partition from the example can be used for smurfing. An amount of 0.00001 BTC, which is $1,000$ *Satoshi*, is transferred from Wallet A. The shown smurfing uses the last integer partition of the above list of integer partitions. The amounts 18, 182 and four times 200 *Satoshi* are transferred to 6 different recipients (Wallet B, Wallet C, Wallet D, Wallet E, Wallet F, and Wallet G), and then collected in one wallet (Wallet H) again.

Example using constrained integer partitions with more restrictions for smurfing Continuing with examples for the application of *specific* integer partitions for smurfing, we now consider additional requirements on the possible transactions, yielding further restrictions on the admissible integer partitions. Specifically, we assume that the analysis of regulations let to the requirement that all individual transaction amounts in the generated splitting have roughly the same size, but are still all different from each other. SageMath provides the parameters `max_slope` and `min_slope` for the function `Partitions`, which determine the maximal and minimal difference between two consecutive parts of the integer partition (when ordered non-increasingly[4]). More precisely, for a given integer partition $(\lambda_1, \lambda_2, \ldots, \lambda_r)$, the maximum and the minimum of the values $\lambda_{i+1} - \lambda_i$, for all $1 \leq i < r$, are restricted[5].

Assume we now also require that the difference between two transaction amounts is at least 10 and at most 50 Satoshi. As before, the total amount is $1,000$ Satoshi and we want at least 5 and at most 20 transactions, each having a value between 10 and 200 Satoshi. We can get the possible transaction amounts by using SageMath with the following command:

```
x=iter(Partitions(1000, min_part=10, max_part=200, max_length=20,
                     min_length=5, max_slope=-10, min_slope=-50))
```

followed by repeated invocations of `next(x)`. We list the first 10 computed integer partitions below:

$$[200, 190, 180, 170, 155, 105], [200, 190, 180, 170, 154, 106],$$
$$[200, 190, 180, 170, 153, 107], [200, 190, 180, 170, 152, 108],$$
$$[200, 190, 180, 170, 151, 109], [200, 190, 180, 170, 150, 110],$$
$$[200, 190, 180, 170, 149, 111], [200, 190, 180, 170, 148, 112],$$
$$[200, 190, 180, 170, 147, 113], [200, 190, 180, 170, 146, 114].$$

[4] This is the defined order of the parts in an integer partition according to Definition 1.
[5] Note that it follows from Definition 1 that we have $\lambda_{i+1} - \lambda_i \leq 0$, for all $1 \leq i < r$.

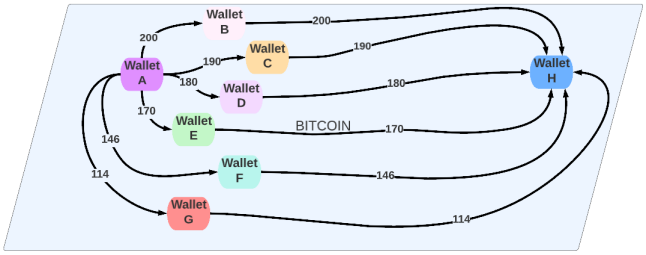

Fig. 5. One of the integer partitions used for smurfing.

In Figure 5, it is shown how an integer partition fulfilling the requirements of the considered example could be used for smurfing in the bitcoin network. Specifically, an amount of 0.00001 BTC, which is equivalent to $1,000$ *Satoshi*, is transferred from Wallet A. The shown smurfing uses the last integer partition of the above list of integer partitions. The amounts 200, 190, 180, 170, 146 and 114 *Satoshi* are transferred to 6 different recipients (Wallet B, Wallet C, Wallet D, Wallet E, Wallet F, and Wallet G), and then collected in one wallet (Wallet H) again.

Further, we visualize in the Table below, how the resulting patterns could appear in the blockchain.

Table 2. Example for transactions in smurfing.

TXID	Blockno.	From	To	Date	Time	Amount
T1	787,076	Wallet A	Wallet B	22.10.2022	16:00	200 Satoshi
T2	787,076	Wallet A	Wallet C	22.10.2022	16:00	190 Satoshi
T3	787,076	Wallet A	Wallet D	22.10.2022	16:00	180 Satoshi
T4	787,076	Wallet A	Wallet E	22.10.2022	16:00	170 Satoshi
T5	787,081	Wallet A	Wallet F	22.10.2022	16:01	146 Satoshi
T6	787,090	Wallet A	Wallet G	22.10.2022	16:02	114 Satoshi
T7	787,102	Wallet B	Wallet H	22.10.2022	16:08	200 Satoshi
T8	787,103	Wallet C	Wallet H	22.10.2022	16:08	190 Satoshi
T9	787,104	Wallet D	Wallet H	22.10.2022	16:08	180 Satoshi
T10	787,105	Wallet E	Wallet H	22.10.2022	16:08	170 Satoshi
T11	787,107	Wallet F	Wallet H	22.10.2022	16:08	146 Satoshi
T12	787,111	Wallet G	Wallet H	22.10.2022	16:08	114 Satoshi

4 Conclusion and Future Work

In this paper we presented a mathematical approach on how to split illicit financial transactions into smaller transactions in order to avoid detection in

cryptocurrencies using integer partitions. In order to advance further our initial mathematical approach towards a constructive methodology, some questions in regards of the instantiation of parameters have to be studied; such as the implication of the order in which the transactions happen or how the respective recipients are chosen.

References

1. partitions: Additive Partitions of Integers. https://cran.r-project.org/web/packages/partitions/index.html, accessed March 1, 2023
2. Sagemath. https://github.com/sagemath/sage-windows/releases/tag/0.6.3-9.3, note = Accessed March 1, 2023
3. Andrews, G.E.: The Elementary Theory of Partitions, p. 1–15. Encyclopedia of Mathematics and its Applications, Cambridge University Press (1984). https://doi.org/10.1017/CBO9780511608650.004
4. Council of the EU: Digital finance: Markets in Crypto-assets (2022), https://www.consilium.europa.eu/en/press/press-releases/2022/06/30/digital-finance-agreement-reached-on-european-crypto-assets-regulation-mica/, accessed March 1, 2023
5. for Criminal Justice Cooperation (EUROJUST), E.U.A.: Eurojust Report on Money Laundering. Tech. rep. (2020), https://www.eurojust.europa.eu/publication/eurojust-report-money-laundering, accessed March 1, 2023
6. Demetis, D.S.: Fighting money laundering with technology: A case study of Bank X in the UK. Decision Support Systems **105**, 96–107 (2018). https://doi.org/https://doi.org/10.1016/j.dss.2017.11.005
7. Gao, S., Xu, D., Wang, H., Wang, Y.: Intelligent Anti-Money Laundering System. In: 2006 IEEE International Conference on Service Operations and Logistics, and Informatics. pp. 851–856 (2006). https://doi.org/10.1109/SOLI.2006.328967
8. Han, J., Huang, Y., Liu, S., Towey, K.: Artificial intelligence for anti-money laundering: a review and extension. Digital Finance **2**, 211–239 (2020)
9. International Monetary Fund, n..A.: Anti-money laundering/combating the financing of terrorism (aml/cft), https://www.imf.org/external/np/leg/amlcft/eng/
10. Kingdon, J.: AI fights money laundering. IEEE Intelligent Systems **19**(3), 87–89 (2004). https://doi.org/10.1109/MIS.2004.1
11. Levi, M.: Money laundering and its regulation. The ANNALS of the American Academy of Political and Social Science **582**(1), 181–194 (2002). https://doi.org/10.1177/000271620258200113, https://doi.org/10.1177/000271620258200113
12. Maplesoft: Partition, https://www.maplesoft.com/support/help/maple/view.aspx?path=Iterator%2FPartition, accessed March 1, 2023
13. Mehmet, M., Wijesekera, D., Fuentes, M.: Money laundering detection framework to link the disparate and evolving schemes. Journal of Digital Forensics, Security and Law (JDFSL) 1558-7223 **8**, 41 (01 2013). https://doi.org/10.15394/jdfsl.2013.1150
14. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (Dec 2008), https://bitcoin.org/bitcoin.pdf, accessed: 2023-04-21
15. Releases, E.P.P.: Crypto-assets: Green light to new rules for tracing transfers in the EU: News: European parliament (Apr 2023), https://www.europarl.europa.eu/news/en/press-room/20230414IPR80133/

crypto-assets-green-light-to-new-rules-for-tracing-transfers-in-the-eu, accessed March 1, 2023
16. SageMath: Integer partitions, https://doc.sagemath.org/html/en/reference/combinat/sage/combinat/partition.html,note={AccessedMarch1,2023}
17. Schneider, F., Niederländer, U.: Money laundering: Some facts. European Journal of Law and Economics **26**, 387–404 (02 2008). https://doi.org/10.1007/s10657-008-9070-x
18. Schneider, F., Windischbauer, U.: Money laundering: some facts. European Journal of Law and Economics **26**, 387–404 (2008)
19. Schwarz, N., Chen, K., Markevych, M.: Virtual assets and anti-money laundering and combating the financing of terrorism (1): Some legal and practical considerations. FinTech Notes **2021**(002), A001 (2021). https://doi.org/10.5089/9781513593760.063.A001, https://www.elibrary.imf.org/view/journals/063/2021/002/article-A001-en.xml
20. Starnini, M., Tsourakakis, C.E., Zamanipour, M., Panisson, A., Allasia, W., Fornasiero, M., Puma, L.L., Ricci, V., Ronchiadin, S., Ugrinoska, A., Varetto, M., Moncalvo, D.: Smurf-based anti-money laundering in time-evolving transaction networks. In: Dong, Y., Kourtellis, N., Hammer, B., Lozano, J.A. (eds.) Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track. pp. 171–186. Springer International Publishing, Cham (2021)
21. Sun, Y., Xiong, H., Yiu, S.M., Lam, K.Y.: Bitanalysis: A visualization system for bitcoin wallet investigation. IEEE Transactions on Big Data **9**(2), 621–636 (2023). https://doi.org/10.1109/TBDATA.2022.3188660
22. Tropina, T.: Do digital technologies facilitate illicit financial flows? (2016)
23. Wang, H.M., Hsieh, M.L.: Cryptocurrency is new vogue: a reflection on money laundering prevention. Security Journal (01 2023). https://doi.org/10.1057/s41284-023-00366-5
24. Weber, M., Domeniconi, G., Chen, J., Weidele, D.K.I., Bellei, C., Robinson, T., Leiserson, C.E.: Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics (2019)
25. Wolfram: Integerpartitions, https://reference.wolfram.com/language/ref/IntegerPartitions.html, accessed March 1, 2023