Speichern von grafischen Daten für NFTs auf der Blockchain

Marianne Poser
Hochschule Mittweida, poser@hs-mittweida.de

In dieser Forschungsarbeit wird ein Überblick darüber gegeben, wie Grafikdaten eines NFT auf der Blockchain gespeichert werden können. Es werden verschiedene Ansätze untersucht und vorhandene Projekte analysiert. Dabei werden vor allem die Aspekte Sicherheit, Ressourcen und Anwendbarkeit betrachtet. Mithilfe einer Testumgebung werden die recherchierte Ansätze vergleichbar, wobei sich in der Arbeit auf skalierbare Vektorgrafiken (SVG) konzentriert wird. Letztendlich zeigt sich, dass es für simple SVG sinnvoll ist, ihren Code als String oder auch in Base64 codiert im NFT selbst abzulegen. Für komplexere Grafiken wird ein Ansatz mit einem Smart Contract empfohlen, um die Kosten pro NFT zu reduzieren. Die Vorgehensweise, die Grafikdaten durch eine Funktion wiederherzustellen, eignet sich außerdem auch für Ansätze, die nicht auf Vektor Grafiken bauen. Es zeigt sich, dass durch einen gewissen Mehraufwand durchaus NFT und Grafikdaten auf der Blockchain abgelegt werden können und kein Risiko durch die Trennung zwischen On- und Off-Chain eingegangen werden muss.

1. Einleitung

Der Begriff NFT Kunst beschreibt die Verbindung von (digitaler) Kunst und einem NFT, welcher als Besitzurkunde über das Kunstwerk fungiert. Es wird als die Revolution und Zukunft der Kunstbranche bezeichnet und wird so bei Kunstsammler:innen, Spekulant:innen und Künstler:innen immer bekannter. [1]

Doch bei der Kaufentscheidung sollte nicht nur auf die Reputation der Künstler:innen oder die Rarität des Kunstwerks geachtet werden, sondern es sollte auch ein Blick auf die Technik dahinter geworfen werden. Wo ist der NFT und wo ist das dazugehörige Kunstwerk gespeichert? Denn auch wenn der NFT sicher auf einer Blockchain wie Ethereum abgelegt ist, so muss sich dort nicht auch das Kunstwerk befinden. Viel wahrscheinlicher ist es, dass im NFT nur ein Link zu dem Kunstwerk hinterlegt wurde.

In dieser Arbeit werden die damit verbundenen Probleme gezeigt und mögliche Lösungen dafür vorgestellt und miteinander verglichen. In den folgenden Kapiteln wird die Problemstellung erörtert, die Literaturrecherche vorgestellt und daraus abgeleitete Lösungen und Varianten verglichen.

Es wird gezeigt, wie grafische Daten auf der Blockchain sicher, kostensparend und anwenderfreundlich abgelegt werden können. Denn nur so ist das Wertgebende des NFT – die Kunst, genauso gut verwahrt wie der NFT selbst. Und dies sollte genauso im Interesse der Künstler:innen sein, wenn sie ihre Kunst für teilweise insgesamt 69 Millionen USD versteigern, wie auch der Besitzer:innen von NFT. [1]

2. Problemstellung

Für die digitale Ablage des Kunstwerks bzw. dessen grafischen Daten bestehen grundsätzlich drei Möglichkeiten. Das Bild kann auf einem zentralen Server abgelegt werden oder in einem dezentralen Serversystem. Die letzte Möglichkeit ist, dass das nicht nur der NFT sich auf der Blockchain befindet, sondern auch die grafischen Daten selbst. Mit diesem Ansatz soll sich in diesem Paper tiefergehend auseinandergesetzt werden. Zunächst sollen aber die beiden auf serverbasierenden Ansätze betrachtet werden.

Bei dem Ansatz, dass das Bildmaterial auf einem zentralen Server abgelegt wird, ist folgendes Szenario vorstellbar. Ein Start-up entwickelt verschiedene Bilddateien und erzeugt zu jedem Bild einen NFT. Die Bilddateien legt es auf einem Server ab und der NFT enthält einen Zeiger mit der URL auf die dazugehörige Datei. Mit dieser Ausgangslage würde das Unternehmen nun einen Launch ankündigen und die NFT versteigern. Die Person, welche den NFT erwirbt, wird von da an auch als Besitzer:in der Bilddatei angesehen werden.

Allerdings kann es schnell zu Problemen kommen, da die einzige Verbindung zwischen NFT und Bilddatei ein Link auf einen zentralen Server des Unternehmens ist. Dieser Single Point of Failure kann in verschiedenen Szenarien angegriffen werden, ohne dass die Person selbst etwas dagegen vornehmen kann. Zum einen kann der Server abstürzen oder generell abgeschaltet werden (bspw. bei Insolvenz des Unternehmens). Das Unternehmen kann die Bilddateien aber auch im Nachhinein abwandeln, austauschen, verschieben oder löschen. Auch könnte der Zugriff auf diese Bilddatei verwehrt werden. In jedem Fall wäre der Besitz des NFTs wertlos, da das wertgebende Bild nicht mehr erreichbar oder nicht mehr damit verbunden ist. [2]

Selbst wenn zuvor eine Kopie des Bildes gesichert wurde, kann dieses nicht mehr mit dem NFT verbunden werden. In diesem Fall wäre es hilfreich, wenn zusätzlich Hash über die Bilddatei in den NFT integriert wird. So kann bei Besitz einer Kopie zumindest nachgewiesen werden, dass der NFT sich tatsächlich auf dieses Bild bezieht. [2] Auch für den Fall, dass der Server (bspw. durch das Unternehmen) gewechselt werden muss und sich

dadurch die URL ändert, gibt es bereits eine Lösung. In dem Tokenstandard ERC1155 ist die Funktion integriert, dass die URI gewechselt werden kann und sich die einzelnen URL dann aus der NFT ID ergeben. [3]

Beide Ansätze sind aber keine Lösung für das Problem, dass eine Unterbrechung zwischen On-Chain NF und Off-Chain Bilddatei den NFT wertlos macht. Denn auch die eigene Kopie kann verloren gehen und das Unternehmen vielleicht gar kein Interesse mehr daran, die URL aktuell zu halten.

Dem Problem von zentralen Servern sind sich Blockchain-affine Menschen meist bewusst. Daher ist die populäre Alternative eine Art dezentrales Serversystem. Im InterPlanetary File System (IPFS) können mehrere Kopien einer Datei abgelegt werden und so die Absicherung gegen Ausfälle erhöht werden. Allerdings speichern IPFS Server Daten nur so lange, wie ein Node des Netzwerkes dies fordert. Überlässt man die Sicherung der Bilddateien wieder nur dem Herausgeber der NFT, so kann dieser sie zu einem späteren Zeitpunkt von seinem Node entfernen und damit entsteht die Gefahr, dass es komplett vom IPFS gelöscht wird. [4, 5] Dass es nicht ungewöhnlich ist, dass Dateien im IPFS nicht mehr gefunden werden, zeigen verschiedene Twitter-Posts des Dienstes CheckMyNFT. [6]

Die Betrachtung der beiden Server-Speichermöglichkeiten zeigt, dass beide nicht ausreichend sind, um die Bilddaten eines NFT sicher aufzubewahren. Aus Sicherheitsgründen wäre es also am sinnvollsten, Kunst und NFT am gleichen Ort zu speichern - auf einer Blockchain. [7] Diese Möglichkeit und welche Varianten sie besitzt soll im folgenden Kapitel betrachtet werden.

3. On-Chain Speichermöglichkeiten

Der Ansatz, neben dem NFT auch die eigentliche Kunst auf der Blockchain zu speichern, wurde vor allem aufgrund der Kosten erst von wenigen Projekten umgesetzt. [7] Doch vor allem aus Aspekten der Sicherheit sollte sich intensiver mit dieser Möglichkeit befasst werden. Grundlegend gibt es zwei Varianten, wie das Kunstwerk auf der Blockchain abgelegt werden kann. Beide werden im Folgenden vorgestellt.

Die erste Möglichkeit ist, dass die Kunst durch eine Funktion in einem Smart Contract generiert wird. Als Beispiel für diese Variante eignet sich das Projekt Autoglyphs von LarvaLabs. Auf ihrer Webseite bezeichnen sie sich als Herausgeber der ersten On-Chain generierten Kunst. [8] Die Kunstwerke basieren auf einer Auswahl von Symbolen, wobei der ID eines NFT jeweils ein Symbol zugewiesen wird. Mit der draw-Funktion im Smart Contract wird dieses Symbol ausgelesen und basieren auf einem Seed bzw. dessen Hash geplottet. Also wird für jede Stelle des Kunstwerks entschieden, ob ein Symbol gesetzt wird oder ein Leerzeichen entsteht. Das Ergebnis wird als Base64 codiert ausgegeben und kann mit dem Präfix "data:text/plain;charset=utf-8," von allen gängigen Browsern interpretiert und ausgegeben werden. [9] Die dadurch entstandenen Kunstwerke sind einzigartig und werden so lange bestehen, wie es die Ethereum Blockchain geben wird.

Die Kosten für das Erzeugen eines solchen NFT sind recht gering. Betrachtet man die dazugehörige Transaktion, so hat diese laut Etherscan zum damaligen Zeitpunkt knapp $0.90 (0.0053 Eth) gekostet. [10] Und auch das Erzeugen des Smart Contract selbst kostete nur etwa 0.012 Ether. [11] Um das Bild aus dem Seed erneut erstellen zu lassen, reicht ein Funktionsaufruf, der draw-Funktion, welche pure ist. Der Aufruf verursacht dementsprechend keine Kosten. Diese Variante ist also nicht mit horrenden Kosten verbunden und bietet dennoch die komplette Sicherheit für NFT und Bild. Die günstigen Preise hängen allerdings auch mit der simplen Art der Kunst und Symbole zusammen. Für komplexere Bilder oder Bilder, welche nicht (komplett) mit einem Algorithmus erzeugt werden sollen, bestehen andere Möglichkeiten.

Eine andere Möglichkeit, welche auch von verschiedenen Projekten umgesetzt wurde, ist, die Bilddaten in den Metadaten des NFT zu speichern. Dabei handelt es sich zumeist um eine Vektorgraphik (SVG), welche anschließend in einen Base64-String codiert wurde. Dieser wiederum muss nur mit dem entsprechenden Präfix in einen Browser eingefügt werden und wird von diesem in die Grafik umgewandelt. [12] Ein Projekt, welches darauf basiert, ist CardanoTrees, welches ebenfalls generative Kunsttechniken nutzt und das Ergebnis dann in dem NFT auf der Cardano-Blockchain ablegt. [13]

Neben den genannten Projekten gibt es einige weitere NFT Projekte, welche komplett On-Chain arbeiten. Dabei unterscheiden sie sich in verschiedenen Faktoren. Wie bereits vorgestellt, können die Daten, welche in der Blockchain abgelegt wurden, auch in der Blockchain gerendert werden. Das Ergebnis kann dann von einem Browser als Bild interpretiert werden. Oder aber die gespeicherten Grafikdaten müssen durch ein externes Skript bearbeitet und gerendert werden. Dies ermöglicht ein komplexeres Vorgehen und kann neben grafischen Daten auch Musik verarbeiten. [14]

Neben dem Ablegen in den Metadaten und dem Generieren der Kunst On-Chain gibt es also weitere Zwischenlösungen, welche teilweise ein externes Skript benötigen. Dieses Skript kann beispielsweise auf der Blockchain gespeichert werden, ohne dass es dort ausgeführt werden kann. Dieses Skript entspricht also eher einer Anleitung, was Nutzer:innen Off-Chain durchführen müssen, um aus den Informationen auf der Blockchain ihr Kunstwerk zum NFT wiederherstellen zu können. Wichtig ist dabei, dass die Anleitung tatsächlich On-Chain abgelegt wird. Schreibt man die Anleitung hingegen in die Kommentare, so wird sie in den Metadaten des Contracts abgelegt, welche in einem automatisch generierten JSON gespeichert werden. Diese Datei wiederum ist dafür gedacht, auf IPFS abgelegt zu werden, der Hash der Datei wird am Ende des Bytecodes angehängt.

Auf diese Weise kann die Korrektheit authentifiziert werden, allerdings liegt die Anleitung dadurch nicht auf der Blockchain, sondern im IPFS. Um auch die Anleitung auf der Blockchain abzulegen, muss sie bspw. in einer Variablen im Contract abgelegt werden. Um Kosten zu sparen, kann diese Variable als Konstante definiert werden.

Je nach Art der Anleitung sollte geprüft werden, ob eine Base64 codierte Ablage Sinn macht oder nicht. Für die Anwendbarkeit kann es sinnvoller sein, die Daten ohne Codierung abzulegen, sodass leichter erkennbar ist, was in der Anleitung erklärt wird und nicht erst in lesbare Form gebracht werden muss. Sollen die Daten hingegen direkt von einem Browser in ein Bild übersetzt werden können, so ist der Base64 codierte String leichter händelbar als bspw. der Code einer SVG. Für eine nicht codierte Ablage spricht, wenn Informationen in der ID selbst oder in Form eines Seeds abgelegt werden und dann in eine Art Maske eingesetzt werden müssen. Das Ablegen von Informationen in der ID ist besonders sinnvoll, um eh bezahlten Speicherplatz effektiv zu nutzen. In den 256 Bit der ID können bereits Informationen über die Bilddaten abgelegt werden. [15]

Eine weitere Möglichkeit, um bei der Ablage On-Chain, Kosten zu sparen, wird im EIP4883 vorgestellt. Dabei wird eine neue SVG für ein NFT erzeugt, indem bereits bestehende SVGs anderer NFT verkettet werden. Dadurch könnten neue On-Chain Bilder für NFT kostensparend erzeugt werden, wenn sie auf bereits hinterlegtes Bildmaterial zurückgreifen. Dieser EIP befindet sich allerdings noch in Bearbeitung und klärt beispielsweise noch nicht, unter welchen Bedingungen die Grafikdaten anderer NFT genutzt werden dürfen. [16]

4. Vergleich der Möglichkeiten

Für eine bessere Entscheidungsgrundlage wurden verschiedene Ansätze in einem Test umgesetzt. Die Ausgangslage war dabei, dass auf der Ethereum-Blockchain sowohl NFT als auch Bilddaten abgelegt werden sollten. Die Bilddaten sollten nicht durch eine Funktion im Smart Contract erzeugt werden müssen, sondern es kann auf ein externes Skript zurückgegriffen werden. Allerdings ist die Anleitung für das Erstellen der Grafiken mit im Smart Contract abgelegt. Geprüft werden sollte nun, welches die optimale Lösung für das Ablegen der SVG auf der Blockchain sein könnte. Als Grundlage für den Token wurde der Umsetzung des Standards ERC1155 durch Open Zeppelin genutzt. [3]

Als Umgebung wurde die Truffle Suite mit einer lokalen Blockchain mit Ganache genutzt. Und für die Entwicklung selbst wurde Visual Studio Code verwendet. Für den ersten Vergleich wurde eine SVG mit dem Bild einer Fackel genutzt. Sowie der SVG Code in Base64 codiert.

Abbildung 1: Fackel

```
<?xml version="1.0" encoding="UTF-8"?>
<svg width="190.2mm" height="155.2mm" version="1.1" viewBox="0 0 190.2 155.2" xmlns="http://www.w3.org/2000/svg">
 <g transform="matrix(.3533 0 0 .3533 -.1957 -.1402)" stroke="#000" stroke-miterlimit="11.34" stroke-width="5.7">
  <rect x="403.7" y="98.3" width="20.4" height="160.4" fill="#630" stroke-linecap="round"/>
  <circle cx="419" cy="223" r="26.9" fill="#f69c7f"/>
  <g stroke-linecap="round">
   <path d="m413.9 16.1s28.8 21.6 28.8 47.1-12.9 46.1-28.8 46.1-28.8-20.6-28.8-46.1 28.8-47.1 28.8-47.1z" fill="#f63"/>
   <path d="m413.9 41.6s21.1 15.1 21.1 33.8-9.4 33.8-21.1 33.8-21.1-15.2-21.1-33.9 21.1-33.7 21.1-33.7z" fill="#f93"/>
   <path d="m413.9 78.3s9.1 6.9 9.1 15.5-4.1 15.5-9.1 15.5-9.1-6.9-9.1-15.5 9.1-15.5 9.1-15.5z" fill="#ff6"/>
  </g>
 </g>
</svg>
```

Abbildung 2: SVG Coder der Fackel

PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHN2ZyB3aWR0aD0iMTkwLjJtbSIgaGVpZ2h0PSIxNTUuMm1tIiB2ZXJzaW9uPSIxLjEiIHZpZXdCb3g9IjAgMCAxOTAuMiAxNTUuMiIgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3ZnIj4KIDxnIHRyYW5zZm9ybT0ibWF0cml4KC4zNTMzIDAgMCAuMzUzMyAtLjE5NTcgLS4xNDAyKSIgc3Ryb2tlPSIjMDAwIiBzdHJva2UtbWl0ZXJsaW1pdD0iMTEuMzQiIHN0cm9rZS13aWR0aD0iNS43Ij4KICA8cmVjdCB4PSI0MDMuNyIgeT0iOTguMyIgd2lkdGg9IjIwLjQiIGhlaWdodD0iMTYwLjQiIGZpbGw9IiM2MzAiIHN0cm9rZS1saW5lY2FwPSJyb3VuZCIvPgogIDxjaXJjbGUgY3g9IjQxOSIgY3k9IjIyMyIgcj0iMjYuOSIgZmlsbD0iI2Y2OWM3ZiIvPgogIDxnIHN0cm9rZS1saW5lY2FwPSJyb3VuZCI+CiAgIDxwYXRoIGQ9Im00MTMuOSAxNi4xczI4LjggMjEuNiAyOC44IDQ3LjEtMTIuOSA0Ni4xLTI4LjggNDYuMS0yOC44LTIwLjYtMjguOC00Ni4xIDI4LjgtNDcuMSAyOC44LTQ3LjF6IiBmaWxsPSIjZjYzIi8+CiAgIDxwYXRoIGQ9Im00MTMuOSA0MS42czIxLjEgMTUuMSAyMS4xIDMzLjgtOS40IDMzLjgtMjEuMSAzMy44LTIxLjEtMTUuMi0yMS4xLTMzLjkgMjEuMS0zMy43IDIxLjEtMzMuN3oiIGZpbGw9IiNmOTMiLz4KICAgPHBhdGggZD0ibTQxMy45IDc4LjNzOS4xIDYuOSA5LjEgMTUuNS00LjEgMTUuNS05LjEgMTUuNS05LjEtNi45LTkuMS0xNS41IDkuMS0xNS41IDkuMS0xNS41eiIgZmlsbD0iI2ZmNiIvPgogIDwvZz4KIDwvZz4KPC9zdmc+Cg==

Abbildung 3: Code der Fackel in Base64

Bereits der Vergleich der Dateigrößen zeigt, dass die Base64-Codierung mehr Speicherplatz benötigt. Für einen Vergleich auf der Blockchain bzgl. der Gas-Kosten wurde zweimal der gleiche Contract aufgesetzt. Dabei wurde die öffentliche Konstante „torch“ einmal codiert und einmal als SVG Code eingefügt. Dabei wurde der SVG Code insofern verändert, dass die doppelten Anführungszeichen (“) durch einfache Anführungszeichen (‘) ersetzt wurden und alle Zeilenumbrüche entfernt wurden. Beide Contracts wurden auf der lokalen Blockchain deployed und die Gaskosten konnten verglichen werden. Dabei entstanden die folgenden Werte: Bei der Base64 Variante wurden 2.728.024 Gas verbraucht und bei der SVG Code Variante 2.665.545 Gas. Der Unterschied beträgt also etwa 60.000 Gas zwischen codiert und nicht codiert.

Je nach Anwendung kann die Codierung mit Base64 weitere Nachteile oder auch Vorteile in der Anwendung mit sich bringen. Wird zu jedem einzelnen NFT in den Metadaten das Bild mit abgelegt, ist es für den Anwender leichter, einen Base64 codierten String zu kopieren und von einem Browser (mit entsprechendem Präfix) interpretieren zu lassen. Sollen jedoch im SVG-Code selbst noch Anpassungen oder Personalisierungen vorgenommen werden, so ist dies in nicht codierten Form leichter.

Zusätzlich könnten in verschiedenen Grafiken mehrfach vorkommende gleiche Codezeilen als Overhead extrahiert werden und müssten nicht redundant, sondern nur einmal in einer weiteren Konstante abgelegt werden. Verzichtet man in dem genutzten Beispiel auf die ersten beiden Zeilen des SVG Code, verringern sich die Gas-Kosten beim Deployen des Contracts um etwa 39.000 Gas. Diese müssten zwar dennoch im Contract hinterlegt werden und in der Anleitung das Zusammensetzen erläutert werden, dennoch spart es für jede weitere zu speichernde SVG diese Menge an Gas. Zusätzlich ermöglicht die nicht codierte Ablage eine Weiterverarbeitung durch andere Programme/Skripte, ohne dass diese erneute decodieren müssen.

Eine weitere genannte Variante, um Gas-Kosten zu sparen, ist das Ablegen von Informationen in der ID. In den NFT Tokenstandards ERC721 und ERC1155 bekommen die einzelnen NFT eine ID zugewiesen. Diese ist 256 Bit groß und könnte dafür genutzt werden, bereits Informationen über die Gestalt des NFT zu enthalten. Dabei muss beachtet werden, wie viele NFT es geben wird und wie viele Stellen der ID noch frei sind, sodass die IDs dennoch eindeutig bleiben. Die größte (Dezimal-)Zahl, welche mit dem Datentyp uint256 dargestellt werden kann, ist
‘115.792.089.237.316.195.423.570.985.008.687.907.853.269.984.665.640.564.039.457.584.007.913.129.639.935‘. Diese Zahl hat 78 Ziffern, zieht man eine davon ab, hat man 77 Ziffern, welche man mit Informationen füllen kann. Ist beispielsweise geplant, 120 NFT zu erzeugen, so benötigt man nur drei dieser 77 Stellen für die eindeutige Zuordnung. Die restlichen Stellen können für andere Informationen genutzt werden.

Ausgehend vom Beispiel der Fackel könnte die Farbe des Fackelstabs variiert werden und diese Information mit in der ID abgelegt werden. Würde man den RGB-Farbcode dafür nutzen, würden für diese Information dreimal drei Stellen, also 9 insgesamt benötigt werden. Auf diese Weise könnten auch die Farbcodes für die verschiedenen Schattierungen der Flamme und der Hand in der ID hinterlegt werden. Insgesamt wären dann fünf Farbcodes mit je neun Stellen in der ID hinterlegt. Und selbst dann wären immer noch 29 Ziffern der ID ungenutzt.

Doch auch die Nachteile werden im inspirierenden Twitter-Post genannt. So kann ausgehend von der ID nicht so einfach abgelesen werden, wie viele NFT es gibt und auch URL, welche die ID enthalten, werden sperriger. Auch wenn über verschiede NFTs diskutiert oder die ID anderweitig angegeben werden soll, muss so mit großen Zahlen hantiert werden. Eleganter kann es daher sein, das Gas für ein zusätzliches Mapping id=>seed zu investieren. Dabei werden die genannten Metadaten (Farbinformationen) im Seed (ebenfalls uint256) abgelegt und den IDs zugeordnet. Hier müssen die Seeds nicht eineindeutig sein, sondern nur die IDs.

In der Testumgebung ergab das Einsparen des Mappings und der Funktion zur Abfrage des Seeds zur gegebenen ID einen Unterschied von 28.000 Gas. In der mint-Funktion zum Kreieren eines neuen NFT ergab sich ein Unterschied von 6000 Gas, da keine Information im Mapping abgelegt werden musste. In beiden Fällen müsste in der Anleitung erklärt werden, welche Stellen was codieren und inwiefern sich daraus Anpassungen des SVG-Codes ergeben. Im Fall der Fackel könnten die Farbcodes für die Füllung der einzelnen Elemente im SVG durch Platzhalter ersetzt werden, wobei Platzhalter 1 durch den Farbcode der ersten neun Stellen des Seeds bzw. der ID ersetzt werden soll.

Als weitere Möglichkeit, Gas-Kosten zu sparen, wurde sich mit den SVG auseinandergesetzt. SVG sind im Allgemeinen recht platzsparend, da nicht einzelne Punkte definiert werden, sondern geometrische Elemente. Je näher man bei diesen Elementen bleibt, umso kleiner die SVG-Datei. Diesen Effekt erkennt man beim Vergleich der beiden Versionen der Fackel.

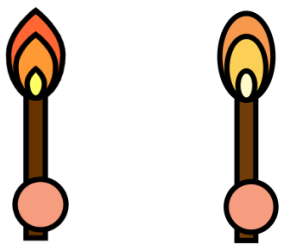

Abbildung 4: Vergleich Fackel Version 1 und Version 2

Der Unterschied zwischen den Fackeln liegt in der Gestaltung der Flammen, bei der Version 1 wurden diese durch Pfadelemente erzeugt und in der Version 2 durch Ellipsen. Durch den vermehrten Einsatz von geometrischen Elementen (Ellipsen statt zu definierende Pfade) unterscheiden sich die beiden Versionen bereits in der Dateigröße um fast 200 Bytes. Beim Speichern im Smart Contract als Konstanten ergab sich für die Version 2 eine Einsparung von etwa 26.000 Gas gegenüber der Version 1.

Neben diesen offensichtlichen Unterschied in Kosten und Gestalt konnten außerdem Verbesserungen geschafft werden, in dem die Anzahl der Nachkommastellen reduziert wurde. Eine weitere Vereinfachung des Codes der SVG kann durch automatisierte Optimierung mit Programmen wie Inkscape erreicht werden. Zusätzlich gibt es weitere Tools wie SVGminify, welche unnötige doppelte Informationen entfernt. [17] So reicht beispielsweise die einmalige Angabe der Strichbreite (stroke-width) innerhalb einer SVG und muss nicht für jedes Element neu mit angegeben werden. Neben dem Entfernen der Zeilenumbrüche könnten auch noch die Leerzeichen entfernt werden, was allerdings die Lesbarkeit verschlechtern würde. [18] Ebenso bringt die Vereinfachungen der Grafik Einbußen in Bezug auf die gestalterische Freiheit bzw. die künstlerische Finesse.

Grundsätzlich sollte beim Abwägen der verschiedenen Ansätze und Möglichkeiten nicht nur auf die Kosten geachtet werden, sondern auch auf die Anwendbarkeit. Für die Besitzer:innen der NFT ist es am komfortabelsten, wenn die Grafikdaten ihres NFT auch im NFT abgelegt werden und nicht im Smart Contract. Wenn die Ablage im Smart Contract ist es wiederum leichter aus Sicht der Anwender:innen, wenn eine Funktion aufgerufen werden kann, welche die Grafikdaten zurückgibt und nicht erst eine Anleitung gelesen, nachvollzogen und umgesetzt werden muss. Auf der anderen Seite ist die Ablage auf der Blockchain vorrangig für ein Notfallszenario gedacht und muss nicht jedes Mal beim Anzeigen des NFT vollzogen werden. Für die normale Nutzung könnten die Bilddaten eines NFT auch weiterhin auf einem Server abgelegt werden und von dort bezogen werden. Und nur in der Situation, wo dieser nicht (mehr) erreichbar ist, kann auf die Informationen im Smart Contract zurückgegriffen werden.

5. Fazit

Nachdem in diesem Paper verschiedene Speichermöglichkeiten für grafische Daten auf der Blockchain vorgestellt und zum Teil verglichen wurden, sollen die Erkenntnisse an dieser Stelle zusammengefasst werden.

Zunächst hat sich gezeigt, dass SVG, als Format für die grafischen Daten, verschiedene Optimierungsmöglichkeiten bereithält. So wären bereits bei der Entwicklung der Bilder große Einsparungen möglich. Dabei muss allerdings eine Balance zwischen Kostenreduktion und Einschränkung der künstlerischen Freiheit gefunden werden.

Ausgehend von den Vektor-Grafiken kann dann entschieden werden, ob der Code als solcher gespeichert werden soll oder ob er in Base64 codiert werden soll. Dies ist vor allem dann praktikabel, wenn die einzelnen Bilder direkt im zugeordneten NFT abgelegt werden soll. Besonders praktisch ist dies für die Nutzer:innen, welche diesen String nur in ihren Browser einfügen müssen. Der Nachteil daran ist, dass dieses Verfahren teurer ist und Anpassungen nur mit Zwischenschritten möglich sind.

Hat man viele ähnlich aufgebaute Grafiken, dann ist es sinnvoll, gleiche Codeteile zu separieren und in einer Anleitung das Zusammensetzen zur ursprünglichen Grafik zu erklären. Dieses System kann als Back-up zusätzlich zur Ablage im IPFS genutzt werden, sodass Nutzer:innen nur im Notfall die Grafik selber zusammensetzen müssen.

Auch bei der Entscheidung, wo Metadaten des NFT abgelegt werden sollen, muss zwischen Anwendbarkeit und Kosteneinsparung abgewogen werden. Speichert man Metadaten direkt in der ID des NFT, wird diese unhandlich und schwer lesbar. Das Speichern der Metadaten in einem struct für jeden NFT wiederum sollte aus Kostengründen vermieden werden. Eine gute Zwischenlösung scheint der Einsatz eines Mappings von ID zu einem Seed zu sein, welches die Metadaten enthält.

Letztendlich konnte gezeigt werden, dass die Möglichkeit, grafische Daten für NFT auf der Blockchain abzulegen, durchaus umsetzbar ist und mit verschiedenen Abwandlungen an die Gegebenheiten des eigenen NFT-Projekts angepasst werden kann. So können sichere NFT-Projekte entstehen, deren Kunst genauso lange erhalten bleibt wie der NFT selbst.

Literatur

[1] Christoph Peterson, „NFT Kunst kaufen und verkaufen 2022: So funktioniert es!“, 23. März 2022, 2022. [Online]. Verfügbar unter: https://coincierge.de/nft/nft-kunst/. Zugriff am: 1. August 2022.

[2] J. Benson, „Yes, Your NFTs Can Go Missing—Here's What You Can Do About It“, *Decrypt*, 19. März 2021, 2021. [Online]. Verfügbar unter: https://decrypt.co/62037/missing-or-stolen-nfts-how-to-protect. Zugriff am: 2. August 2022.

[3] OpenZeppelin, *ERC1155.sol.* [Online]. Verfügbar unter: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC1155/ERC1155.sol (Zugriff am: 4. August 2022).

[4] B. Dale, „It's an NFT Boom. Do You Know Where Your Digital Art Lives?“, *CoinDesk*, 23. Feb. 2021, 2021. [Online]. Verfügbar unter: https://www.coindesk.com/tech/2021/02/23/its-an-nft-boom-do-you-know-where-your-digital-art-lives/. Zugriff am: 2. August 2022.

[5] V. Tangermann, „NFTs Have a Huge Persistence Problem“, *Futurism*, 17. März 2021, 2021. [Online]. Verfügbar unter: https://futurism.com/nfts-have-huge-persistence-problem. Zugriff am: 2. August 2022.

[6] CheckMyNFT, *Check My NFT 🔎🖼 auf Twitter: „@jonty @cloudinary Btw we've been tracking this for 7 days now and most of the files we check from @niftygateway on IPFS fail“ / Twitter.* [Online]. Verfügbar unter: https://twitter.com/CheckMyNFT/status/1372253288863825925 (Zugriff am: 4. August 2022).

[7] ART HAUS, *On-chain NFTs and Why They're Better - ART HAUS.* [Online]. Verfügbar unter: https://art.haus/on-chain-nfts-and-why-theyre-better/ (Zugriff am: 1. August 2022).

[8] LarvaLabs, *Autoglyphs* (Zugriff am: 4. August 2022).

[9] Etherscan.io, *Ethereum Transaction Hash (Txhash) Details | Etherscan.* [Online]. Verfügbar unter: https://etherscan.io/tx/0x10757d45a56f93afdc78cc712553ba999e5a1a881be9139200be9f021a716712#eventlog (Zugriff am: 4. August 2022).

[10] Etherscan.io, *Ethereum Transaction Hash (Txhash) Details | Etherscan.* [Online]. Verfügbar unter: https://etherscan.io/tx/0x10757d45a56f93afdc78cc712553ba999e5a1a881be9139200be9f021a716712 (Zugriff am: 4. August 2022).

[11] Etherscan.io, *Ethereum Transaction Hash (Txhash) Details | Etherscan.* [Online]. Verfügbar unter: https://etherscan.io/tx/0x754661a46f11f62b311866a608d20034f940c3d3db6697564d26c2ad1fe9774a (Zugriff am: 4. August 2022).

[12] A. |. @Ruttkowa, „Completely “on chain” stored NFTs— what? - Alex | @ruttkowa - Medium“, *Medium*, 25. Sep. 2021, 2021. [Online]. Verfügbar unter: https://ruttkowa.medium.com/a-nft-stored-on-chain-what-fb890b6261ff. Zugriff am: 4. August 2022.

[13] *CardanoTrees.* [Online]. Verfügbar unter: https://cardanotrees.com/ (Zugriff am: 4. August 2022).

[14] 0xchain.art, *On-Chain Art.* [Online]. Verfügbar unter: https://www.0xchain.art/info (Zugriff am: 5. August 2022).

[15] w1nt3r_eth, *WINTΞR 🤍🤍 auf Twitter: „Next frontier in the NFT gas optimization game: put the data into the token id itself! The thread goes into more details ↓ https://t.co/FjlC0u98H3“ / Twitter.* [Online]. Verfügbar unter: https://twitter.com/w1nt3r_eth/status/1538229135897554944 (Zugriff am: 8. August 2022).

[16] A. Coathup, D. Martinelli, blockdev und A. Griffith, *EIP-4883 Composable SVG NFT by abcoathup · Pull Request #4888 · ethereum/EIPs.* [Online]. Verfügbar unter: https://github.com/ethereum/EIPs/pull/4888/files (Zugriff am: 8. August 2022).

[17] *SVG Minifyer.* [Online]. Verfügbar unter: https://www.svgminify.com/de.html (Zugriff am: 9. August 2022).

[18] A. Malz und I. Junghans, *Badges [internes Dokument].*

Verwendete Tools

Inkscape - https://inkscape.org/de/

Visual Studio Code - https://code.visualstudio.com/

Truffle Suite - https://trufflesuite.com/

Remix - https://remix.ethereum.org/

Base64 Encoder - https://base64.guru/converter/encode/image/svg