铭文科普｜了解铭文协议用例、实现方式与资产安全

2月1日，币安Web3钱包正式推出铭文市场，支持BRC-20、EVM等多种铭文协议。几天前，OKX也宣布支持ARC-20、Runes、Doginals等铭文协议，引发整个市场对于铭文的关注。在此背景下，由于铭文协议的复杂性和新颖性，各种安全问题频出。这不仅威胁到用户的资产安全，也对整个铭文生态的健康发展产生了负面影响。

针对此，Beosin安全团队对主流铭文协议进行了梳理，帮助用户了解铭文协议的用途、实现方式以及如何保护铭文资产。

### 铭文简介

区块链上所谓的铭文，就是通过区块链的某些特性，在区块链上记录一些特定的且具有意义的信息。该信息一旦记录到区块链之上，便将永久保存在区块链上，并且难以篡改。记录到区块链的信息可以是多种类型，例如简单的文本信息，复杂的代码、图像等都可以写入区块链，这样一来，我们便可以使用一套标准来实现数字资产的功能。

### 铭文现状

从最初BRC-20等比特币公链铭文的出现，到现在铭文生态中几乎每天都有层出不穷的铭文新协议以及新项目出现，铭文的发展可以说是突飞猛进。各个常见的公链也都加入了铭文生态圈，例如ETH公链上的Ethscription协议、BTC公链上的ARC-20协议、BSC公链上的BSC-20等协议、Polygon公链上的PRC-20等协议......。这些协议都是为了在其公链上发布铭文所产生的，接下来的内容我们将介绍各种协议的实现方式以及用例。

### 铭文详解

我们来介绍一下目前市场关注度较高的协议，来比较一下各个公链的铭文协议到底有何共同点与不同点。

### 1. BRC-20

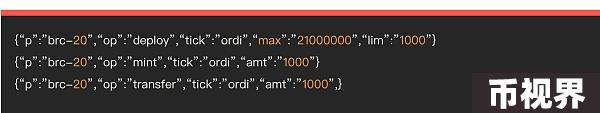
要讲清楚BRC-20，首先要介绍一下UTXO与Ordinals。

BTC采用的是UTXO 模型，交易是以 UTXO 为单位进行转账的。UTXO 是 Unspent Transaction Output 的缩写，意思是未花费的交易输出。UTXO 模型不同于以太坊等公链的账户模型，它记录交易事件，而不记录最终状态。计算某个用户有多少比特币，就需要对其地址的所有 UTXO 求和，得到结果就是该用户的持币数量。

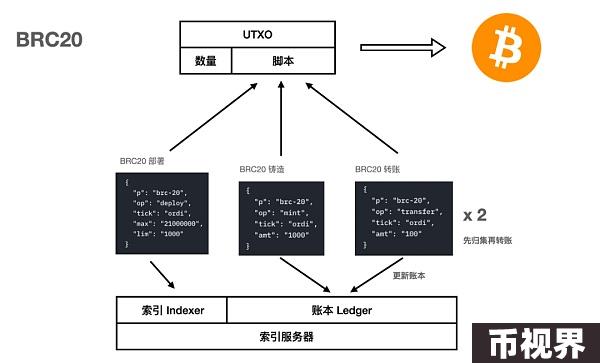
Ordinals是一个为比特币最小单位聪（sats）进行编号的一个系统协议，可以为每个UTXO里（包含了若干个聪）的每一个聪分配一个唯一的编号。Ordinals还支持文字、图片、音频、视频等写入聪的功能，从而使得每个聪都具有独特性，就类似于大家熟悉的以太坊非同质化代币NFT，而我们将其称之为比特币NFT。

而BRC-20创始者基于Ordinals协议，想出了另外一套理念。既然Ordinals协议可以通过给每个聪赋予不同的“属性”来创造比特币NFT，那么也可以通过给定一个统一的“格式”以及“属性”来创造比特币FT，也就是同质化代币。

BRC-20通过Ordinals协议，将统一的JSON格式的文本数据写入聪，该文本数据便是BRC-20代币的记账本，根据该文本数据可以解析出代币持有以及转移情况。主要包含以下内容：



以上是BRC-20的三种标准，其中，op字段表示的是需要执行的操作，包括deploy（部署）、mint（铸造）以及transfer（转移），tick表示的是需要执行操作的代币名称，max表示代币发行总量，lim表示每份代币最大铸币数量，amt表示需要操作的代币数量，在transfer标准中，还存在“to”等字段，但这不是必须的，transfer是通过将该铭文发送给目标地址来实现余额变化，如下图所示：

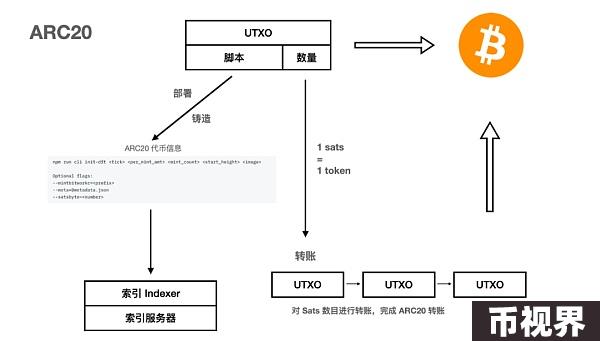


link:https://twitter.com/blockpunk2077/status/1725513817982136617

### 2. ARC-20

ARC-20依然是比特币公链上的铭文协议，和BRC-20协议一样，都是在UTXO里面写入标准的数据来实现，但不同的是ARC-20协议不用在数据中指定ARC-20代币的数量，而是使用该UTXO中的sats（聪，比特币最小单位）来表示ARC-20代币的数量，规则是1 sat=1 ARC-20 token。

ARC-20协议与BRC-20协议一样，也分为部署、铸造、转移三个步骤，其中部署阶段，需要向UTXO中填入标准的代币名称、代币总量、铸造限制、区块信息、图像信息等；铸造阶段，需要用户将代币的名称填入UTXO，而该UTXO的sats数量便为ARC-20代币的铸造数量，并非和代币名称一起填入UTXO；当用户铸造了ARC-20代币，便可以将代币发送给其他地址，在发送代币时，用户不需要再向UTXO里面填入任何数据，而是直接将持有该代币的UTXO转移给其他地址。



link:https://twitter.com/blockpunk2077/status/1725513817982136617

查询ARC-20代币时，只需要一个索引，线下索引服务器便可以读取代币注册信息以及铸造和转移交易，不需要服务器去计算资金转移关系，查询地址所拥有的ARC-20代币数量，直接读取持有该代币的UTXO的sats数量便可以得到。

了解了BRC-20和ARC-20之后，大家应该知道为什么有些会误将铭文资产转到其它地址或是“燃烧”掉了。

由于BRC-20和ARC-20这类BTC铭文协议是基于UTXO交易的，因此铭文交易实际上是附加在BTC交易中的，用户可能会在不完全理解铭文的情况下进行普通的BTC转账操作，将其现在的UTXO和其他UTXO进行融合拆分后发送给非预想的地址，从而导致铭文资产被误转或是被“燃烧”，造成不可逆转的损失。

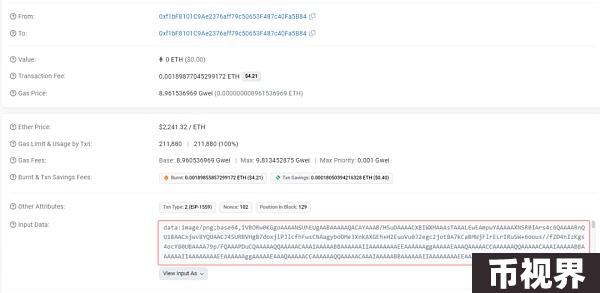
### 3. Ethscription

Ethscription是以太坊上创建和共享数据的协议，某些铭文便是使用该协议从而替代智能合约实现代币发行的方案，使用铭文可以将用户的成本降至极低。

以太坊在发送交易时，提供了一个calldata的数据块，一般情况下普通ETH转账该数据块会留白，如果是调用智能合约，则该数据块将指定为调用函数的签名以及各个参数数据。Ethscription协议便是利用了calldata这一数据块，在发送普通ETH转账的情况下，添加一些标准的数据，从而赋予相关的含义。

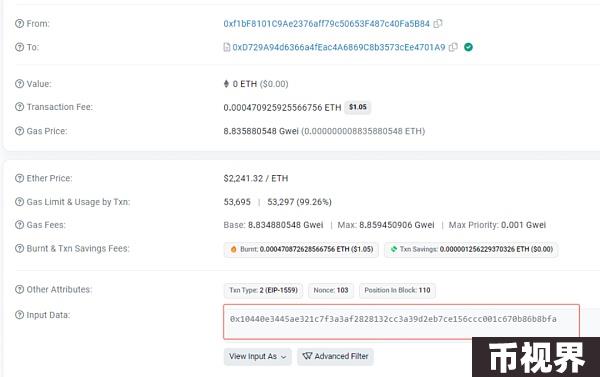
Ethscription是如何规定这些标准数据的呢？

首先如果想要创建一个Ethscription，其内容为图像数据，需要将图像（图像大小限制在96KB）转换为Base64编码数据的URI，格式为(data:image/png;base64,...)；接下来将该URI转换为16进制字符串；通过以太坊向目标地址发送一笔普通转账交易，并且将上述的16进制字符串填入calldata，如下图：



这样一来，0xf1bf地址便拥有了该Ethscription，并且在之后创建的相同calldata的Ethscription将被视为无效。

如果想要转移该Ethscription，就需要Ethscription拥有者向接收地址发送一笔普通转账，并且在calldata中填入创建该Ethscription的交易哈希，则接收地址便拥有了该Ethscription，如下图：

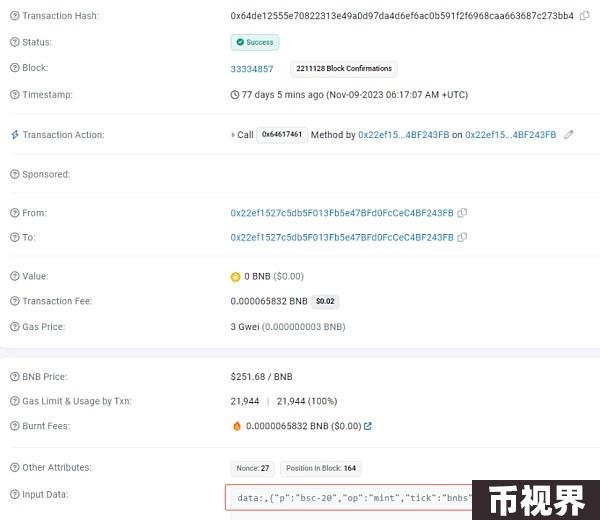


### 4. EVM区块链的铭文

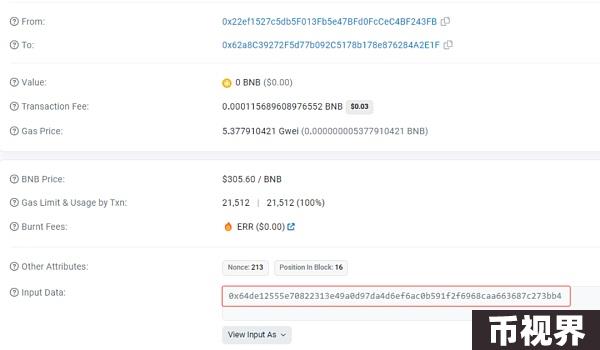
对于BSCChain、以太坊、polygon等EVM区块链，有一套共有的铭文刻录方法，就是利用calldata数据块存储固定的格式数据，与上述保存图像数据不同，该方式是向calldata中写入标准格式的文本数据。

在BSC Chain上刻录铭文，其铭刻格式与BRC20铭刻格式类似，例如铭刻格式为：data:,{"p":"\_","op":"\_","tick":"\_","amt":"\_"}，则其中p字段所代表的是协议名称，如bsc-20、bnbs-20、ltc-20、bep-20、drc-20、nrc-20、src-20等等；op字段所代表的是操作，一般为”mint”；tick字段所代表的是代币名称；amt字段所代表的是代币数量。

这里用以bnbs代币为例，我们可以看到，只要向目标地址发送一笔普通转账，在calldata中填入data:,{"p":"bsc-20","op":"mint","tick":"bnbs","amt":"1000"}便完成了bnbs代币铸造操作，如下图。此时，0x22ef地址便拥有了1000枚bnbs代币。



接下来需要转移代币，和上述一样，需要向接收地址发送一笔普通转账，并将创建该bnbs代币的交易哈希填入calldata，则接收地址便拥有了bnbs代币，如下图：



以太坊、polygon等链上也基本相同，但需要注意一点，上述BSC Chain的内容并不是evm链上创建铭文的唯一情况，不同evm链或不同协议之间填入的文本数据字段可能存在区别，转移代币的方式也可能存在区别。但对于这类方式来说，都是利用了EVM链中的calldata属性来实现的，也就显得大同小异。

### 总结

本篇文章我们讨论了多条链上的铭文实现原理。总结来说，介绍的铭文都是利用一些公链系统特性，将线下信息按照规定的标准保存在区块链，并通过线下服务器进行识别展示的过程。介绍的这些铭文都未使用智能合约，用户在参与的时候可以减少大量的交易额外费用，但用户需充分理解铭文协议的实现方式，避免误转账或误燃烧铭文，造成资产损失。