Tokens

Digital Assets - Week 3 (Lecture)

Rhys Bidder
rhys.m.bidder@kcl.ac.uk

KBS, QCGBF

Autumn 2024

Disclaimer 1: Any opinions expressed, errors or omissions should be regarded as those of the author and not necessarily those of KCL, KBS, QCGBF, QCB, CBI, or BoE.

Disclaimer 2: Any references to cryptography are heavily simplified and leave out many important details. For any real-world security applications these notes should not be relied upon. Instead you should consult appropriate security resources and reliable security professionals.

Disclaimer 3: Cryptoasset transactions are illegal in some jurisdictions. Do not violate these or any other laws. I am not promoting the use of crypto in countries where it is illegal in any form and these slides are not a promotion of crypto or an invitation to participate in crypto-related activities in such countries. They are purely for educational purposes.

Outline

Introduction

Introduction

- One of the most powerful applications of smart contracts is the creation of 'tokens'
  - SC defines the token's functionality and carries out its admin
- Many blockchains allow for elaborate smart contracts and tokens (see here)
  - Ethereum is the most dominant smart contract platform
- We will focus on the types of tokens supportable within the Ethereum protocol
  - Our high level discussion should nevertheless be quite general
- May exist without reference to the 'real world' but frequently obtain data from, and refer to, off-chain sources
  - Tokenization of real world assets (RWAs) is a prime example

Introduction

Nice quote (from OpenZeppelin documentation):

> *A token contract is simply an Ethereum smart contract.* **'Sending tokens' actually means 'calling a method on a smart contract that someone wrote and deployed'.**
>
> *At the end of the day, a token contract is not much more a mapping of addresses to balances, plus some methods to add and subtract from those balances.*

Introduction

What are tokens?

- A 'native' token for a protocol is one that is intrinsically part of the system and used in its fundamental operations
  - ETH for Ethereum

Many other non-native tokens can be created - the number of tokens out there is *dizzying*

- There are various overlapping ways to categorize them

Introduction

Fungible vs Non-fungible is a common classification

- **Fungible:** Any instance of the token is regarded as indistinguishable from, and substitutable for, any other
  - A pound in my pocket is just the same as any other
  - Same goes for BTC, or ETH, or LINK
- **Non-fungible:** Relates to a unique asset, one that is not equivalent to, or substitutable for, any other
  - A collectible painting or a house, are very different from other paintings or houses
  - Cryptokitties are the most famous NFT example but there is now a vast marketplace (see many at OpenSea)

Introduction

Other classifications may be based more on function:

- Transactional
  - Act as 'money', such as stablecoins pegged to fiat or to crypto currencies
  - Tether's USDT, Centre/Circle's USDC and Wrapped bitcoin (wBTC)

- Governance
  - Used to participate in decision-making within a protocol and/or to incentivize certain operational roles
  - Especially prominent in allowing influence within DAO's
  - MKR (influence within the MakerDAO protocol), LINK (incentivizes providing accurate off chain information) and NXM (incentivizes correctly assessing insurance claims)

Introduction

- Utility tokens
  - Entitle holder to certain services or benefits - so used to 'pay' for a particular 'utility' provided by the protocol
  - Not used to purchase anything else (though can be traded)
  - Basic Attention Token (BAT) (access to a secure and ad-blocked browser) and Golem (GLM) (access to computing resources)
- Liquidity Provider tokens (arguably subset of gov. tokens)
  - Key element of Defi: 'Automated Market Makers' (AMMs)
  - Traders exchange tokens at prices defined by formulae
  - Liquidity from agents sending asset pairs to smart contracts and in return get LP tokens
  - UNI (Uniswap), CRV (Curve DAO) and SUSHI (SushiSwap)

Introduction

For now we will defer a discussion of security tokens and tokenized securities. . .

Token standards

- Scope to create tokens is vast
  - Permissionless nature of Ethereum and other blockchains **combined with** the flexibility of SC languages
- Could result in a 'zoo' of tokens, all slightly different
  - A nightmare for users, regulators, exchanges, traders
  - Requires constant updating/checking of compatibility
  - Fractured liquidity, duplicated code checking, unpredictability...

Token standards

- Scope to create tokens is vast
  - Permissionless nature of Ethereum and other blockchains **combined with** the flexibility of SC languages
- Could result in a 'zoo' of tokens, all slightly different
  - A nightmare for users, regulators, exchanges, traders
  - Requires constant updating/checking of compatibility
  - Fractured liquidity, duplicated code checking, unpredictability...
  - A **lot** of pain and sadness... ☹

Token standards

- Token ‘standards’ have gradually emerged to enhance consistency among developers
  - A bit like how there are a small set of USB standards
  - Most electronics companies don’t use proprietary ones
- Standards offer many benefits
  - Enhanced consistency and compatibility
  - Compatibility/interoperability elicit positive network effects
  - Best practice and updates can be adopted rapidly / uniformly
  - Concentrates developer efforts, rather than fracturing it

Token standards

- Clearly, there can't be **one** ideal form of token
  - Too many applications with different costs/benefits
- A few of the key standards:
  - ERC-20: Fungible tokens (ERC-777 updates/improves)
  - ERC-721: Non-fungible tokens
  - ERC-3643: Permissioned and designed for regulated exch.
  - ERC-1400/ERC-1404: Security tokens/tokenized securities

ERC-20: Fungible tokens

- ERC-20 defines a set of rules for functionality of smart contracts
  - It doesn't say how to program the *implementation* of the rules
  - It just says that (somehow) the SC must do certain things
- Knowing this, developers can plan for their dApps to invoke such capabilities when dealing with the tokens
- ERC-20 tokens must exhibit:
  - **Interface:** Six mandatory functions and three optional
  - **Events:** Approval and Transfer

ERC-20: Fungible tokens

Six mandatory functions (ignoring tech details):

1. **totalSupply():** how many of the tokens exist
2. **balanceOf(address):** how many a certain address has
3. **transfer(address, amount):** takes tokens from total supply and sends to address (and emits Transfer event)
4. **transferFrom(address1, address2, amount):** sends tokens from address1 to address2
5. **approve(address, amount):** specifies how much a certain address can spend (and emits Approval event)
6. **allowance(owner_address, delegate_address):** how much a delegate can spend from the owner's balance

I recommend this *excellent* guide on how to write a simple contract that implements these functions (and here for an extension)

ERC-20: Fungible tokens

Three optional functions (to be used when deploying the contract):

1. **name():** what is the contract called (eg. Bidcoin)
2. **symbol():** what is the token's symbol (eg. BDC)
3. **decimals():** too boring to discuss (most adopt 18 - check it)

ERC-20: Fungible tokens

How might we use this:

- First - not obvious if you're not a programmer - we actually have to write the code to do these things!
- Let's look at some smart contract code (from here)
  - **Warning:** This is (intentionally) a very naive implementation - not for real use!

ERC-20: Fungible tokens

```solidity
pragma solidity ^0.8.0;

interface IERC20 {

    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function allowance(address owner, address spender) external view returns (uint256);

    function transfer(address recipient, uint256 amount) external returns (bool);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);


    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}


contract ERC20Basic is IERC20 {···
}
```

The programmer needs to fill in the bottom bit!

ERC-20: Fungible tokens

The first bits filled in are:

- optional bits
- what happens when contract is created (setting total amount)
- mandatory function to find out how much is in circulation

```
string public constant name = "ERC20Basic";
string public constant symbol = "ERC";
uint8 public constant decimals = 18;


mapping(address => uint256) balances;

mapping(address => mapping (address => uint256)) allowed;

uint256 totalSupply_ = 10 ether;


constructor() {
balances[msg.sender] = totalSupply_;
}

function totalSupply() public override view returns (uint256) {
return totalSupply_;
}
```

- Slightly confusing use of Solidity keyword 'ether' in the code!
- Will come back to 'balances' and 'allowed'

ERC-20: Fungible tokens

Stop. Already, we can see some decisions have been made:

- Assumed a very simple way of creating tokens
  - See here for discussion of 'minting' mechanisms
- Hardcoded how much is created, and assigned everything to the person who deployed the contract
- Hardcoded the *totalSupply* function to return same number

Also, what's this **msg.sender** stuff?

- Remember, what are transactions in Ethereum?
- They are just (signed) **messages**
- The message info is captured in **msg** and the **.** is used to extract a bit of the info (the sender)

ERC-20: Fungible tokens

```
function balanceOf(address tokenOwner) public override view returns (uint256) {
    return balances[tokenOwner];
}

function transfer(address receiver, uint256 numTokens) public override returns (bool) {
    require(numTokens <= balances[msg.sender]);
    balances[msg.sender] = balances[msg.sender]-numTokens;
    balances[receiver] = balances[receiver]+numTokens;
    emit Transfer(msg.sender, receiver, numTokens);
    return true;
}

function approve(address delegate, uint256 numTokens) public override returns (bool) {
    allowed[msg.sender][delegate] = numTokens;
    emit Approval(msg.sender, delegate, numTokens);
    return true;
}

function allowance(address owner, address delegate) public override view returns (uint) {
    return allowed[owner][delegate];
}
```

ERC-20: Fungible tokens

Now we see why we needed to create two structures called 'balances' and 'allowed'

- Recall that *Ethereum* smart contracts can remember stuff
- In our case we want it to remember:
  - who is approved to send tokens on whose behalf (+ how much)
  - how much each address possesses of the token

ERC-20: Fungible tokens

In **balanceOf**:

- Finds the token owner's address and spits out their balance

In **transfer**:

- Checks sender has enough tokens to make the transfer
- Deducts it from hers
- Adds it to the recipient's address
- Tells Ethereum blockchain that a 'Transfer' is occuring

ERC-20: Fungible tokens

In **approve**:

- Suppose an owner wants another person (or her 'delegate') to handle making transfers?
- The function adds the address of the delegate to an allowed list (for the given owner) and specifies a limit
- Tells Ethereum blockchain that an 'Approval' is occuring

In **allowance**:

- Finds how much of the permitted spending power the delegate has remaining

But how can the delegate send the money?

ERC-20: Fungible tokens

```
function transferFrom(address owner, address buyer, uint256 numTokens) public override returns (bool)
    require(numTokens <= balances[owner]);
    require(numTokens <= allowed[owner][msg.sender]);

    balances[owner] = balances[owner]-numTokens;
    allowed[owner][msg.sender] = allowed[owner][msg.sender]-numTokens;
    balances[buyer] = balances[buyer]+numTokens;
    emit Transfer(owner, buyer, numTokens);
    return true;
}
```

ERC-20: Fungible tokens

In **transferFrom**:

- Checks the owner (not the delegate) has enough for transfer
- Checks the delegate is approved to transfer enough of the owner's tokens
- Deducts the amount from the owner, adds it to the buyer's
  - Though here we haven't got real 'buying'
- Also deducts from what the delegate can spend in the future
- Tells Ethereum blockchain that a 'Transfer' is occuring

ERC-20: Fungible tokens

That's it. A naive **(don't use it)** but functional ERC20 token

- When the contract is deployed it will be associated with a contract address
- Also, initially only the deployer has tokens (remember the constructor)
- After that, tokens can be handed out and then transferred between addresses in transactions

ERC-20: Fungible tokens

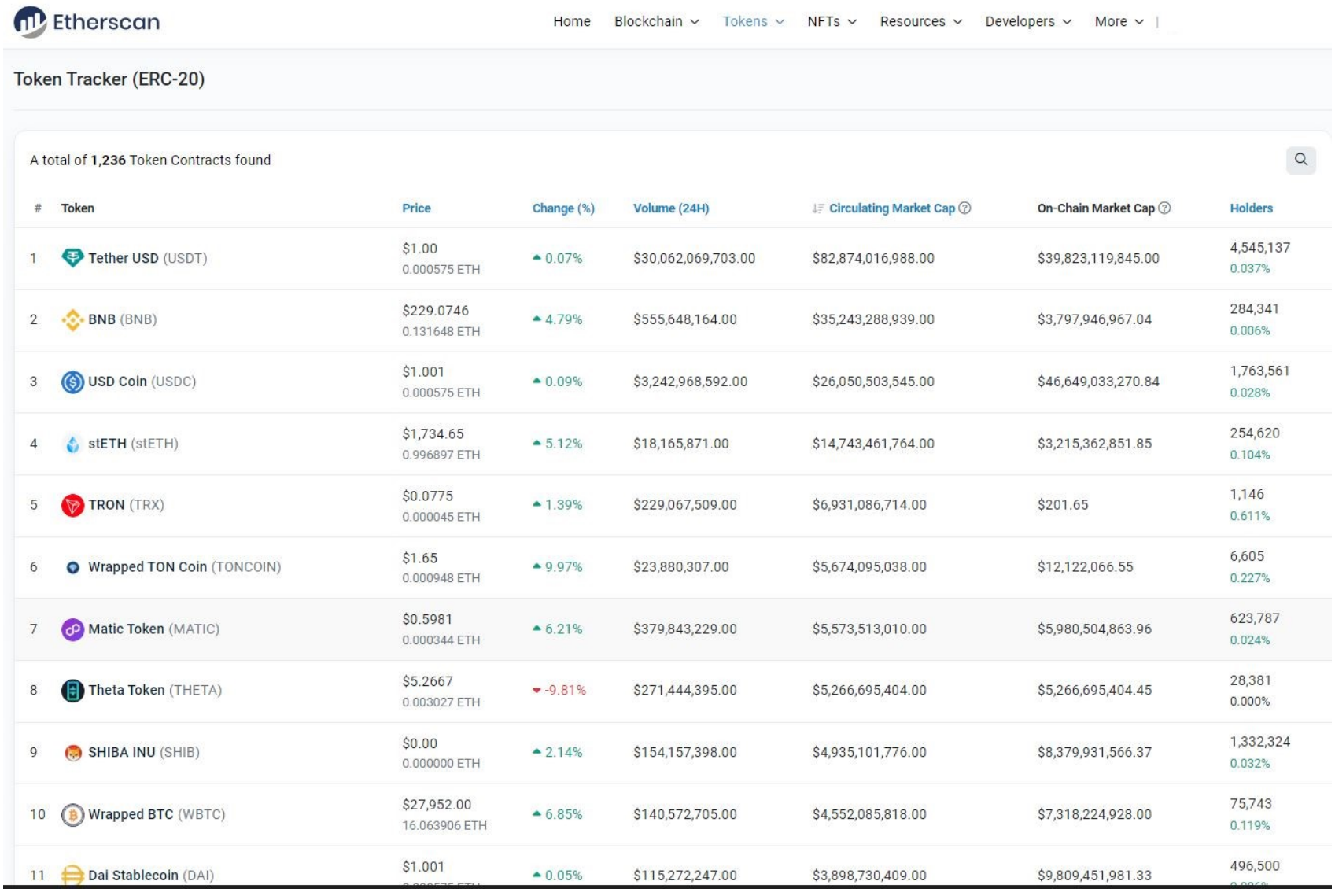
Who might the delegate be (in a richer ERC-20 token)?

- A broker?
- The admininistrator of an ICO?
- A crypto exchange?

We could make these functions far more complicated (and add others beyond those demanded by the ERC-20) but that's for another day...

- Many of the tokens you've heard of are ERC-20 tokens
- There are over $450,000$ ERC-20 tokens on Ethereum

ERC-20: Fungible tokens



Etherscan

Home | Blockchain | Tokens | NFTs | Resources | Developers | More

Token Tracker (ERC-20)

A total of **1,236** Token Contracts found

#	Token	Price	Change (%)	Volume (24H)	Circulating Market Cap	On-Chain Market Cap	Holders
1	Tether USD (USDT)	$1.00 0.000575 ETH	0.07%	$30,062,069,703.00	$82,874,016,988.00	$39,823,119,845.00	4,545,137 0.037%
2	BNB (BNB)	$229.0746 0.131648 ETH	4.79%	$555,648,164.00	$35,243,288,939.00	$3,797,946,967.04	284,341 0.006%
3	USD Coin (USDC)	$1.001 0.000575 ETH	0.09%	$3,242,968,592.00	$26,050,503,545.00	$46,649,033,270.84	1,763,561 0.028%
4	stETH (stETH)	$1,734.65 0.996897 ETH	5.12%	$18,165,871.00	$14,743,461,764.00	$3,215,362,851.85	254,620 0.104%
5	TRON (TRX)	$0.0775 0.000045 ETH	1.39%	$229,067,509.00	$6,931,086,714.00	$201.65	1,146 0.611%
6	Wrapped TON Coin (TONCOIN)	$1.65 0.000948 ETH	9.97%	$23,880,307.00	$5,674,095,038.00	$12,122,066.55	6,605 0.227%
7	Matic Token (MATIC)	$0.5981 0.000344 ETH	6.21%	$379,843,229.00	$5,573,513,010.00	$5,980,504,863.96	623,787 0.024%
8	Theta Token (THETA)	$5.2667 0.003027 ETH	-9.81%	$271,444,395.00	$5,266,695,404.00	$5,266,695,404.45	28,381 0.000%
9	SHIBA INU (SHIB)	$0.00 0.000000 ETH	2.14%	$154,157,398.00	$4,935,101,776.00	$8,379,931,566.37	1,332,324 0.032%
10	Wrapped BTC (WBTC)	$27,952.00 16.063906 ETH	6.85%	$140,572,705.00	$4,552,085,818.00	$7,318,224,928.00	75,743 0.119%
11	Dai Stablecoin (DAI)	$1.001	0.05%	$115,272,247.00	$3,898,730,409.00	$9,809,451,981.33	496,500

ERC-721: NFTs

- ERC-721 tokens can have different values, even if they are created by the same smart contract
- All the tokens are regarded as unique (like, Amethyst or Algorithmic Aurora, for example)

ERC-721: NFTs

Some of the functions in the ERC-721 interface that aren't in ERC-20:

- **ownerOf(tokenId):**
  - outputs an address
- **setApprovalForAll(operator, yes/no):**
  - doesn't output anything (though it does something)
- **getApproved(tokenId):**
  - outputs an address
- **isApprovedForAll(owner, operator):**
  - outputs yes/no

ERC-721 has an approve function:

- **approve(to, tokenId):**
  - doesn't output anything (though it does something)
  - how does this differ from the ERC-20 **approve**?

ERC-721: NFTs

ERC-721 (and ERC-20) are the most commonly discussed classes of Ethereum-base tokens

- An excellent article on these (and others) can be found here

Possible to combine **both** to create **fractional ownership** of an asset

- Very interesting for real world assets (fascinating articles here and here)
- Consider a painting - which is unique (suited to NFT) but where shares in its ownership might be useful (suited to FT)
- Expanding access to 'art' beyond the super rich - and enhances liquidity
- But. . . maybe starts to look like a *security*?

Security tokens

The definition of security tokens is *slippery*:

- Tokens that represent a claim to company profits (like stocks and bonds - but on blockchain)
- They may be 'purely' native to a blockchain protocol or they may *also* represent claims to assets yielding off-chain payoffs
  - If they relate to off-chain assets or commodities, they may be referred to as 'tokenized securities'
- In some people's eyes, many are very close to being, or were 'intended' to be, *utility* or *governance* tokens - but possibly not from regulators' perspective!
  - SEC: *'whether a digital asset is a security is not dependent on the terminology or technology used to create or sell it'*

Security tokens

- Although there are many in existence, few 'big' issuers have emerged (yet)
  - Two of the more prominent SEC-regulated ones are, EXOD and BCAP
- Owing to role of regulation, geography and jurisdiction feature more prominently as a driver of issuance
  - In Switzerland, there have been digital bond issuances (on the SDX exchange) and similar issuances in Singapore
  - Some examples have even been trialed as part of wholesale CBDC pilots (digital sovereign bonds in DvP experiments)

Security tokens

Metrics	Global financial centres		European peers				Regional (non-EU) centres			
	UK	US	Switzerland	France	Germany	Luxembourg	Singapore	Hong Kong	Brazil	UAE (Dubai)
Digital bond issuance	N/A	N/A	~$630 Mn	~$260 Mn	~$105 Mn	~$170 Mn	~$2,030 Mn	~$100 Mn	N/A	N/A

Indicative digital bond issuance activity across select jurisdictions (non-exhaustive, as of end May 2023 - and caveated). Source: UK Finance

Security tokens

Token standards and issuance protocols have emerged to accommodate this

- **Standards:** ERC-3643, ERC-1400 and ERC-1404
- **Protocols:** In the US, issuance may occur under a variety of *regulatory* classifications (helpfully listed here and discussed here) while some countries, such as Switzerland, have issued new *legislation* (recognizing *'ledger-based securities'*) or seem set to do so (as perhaps, in the UK)

There are several reasons for this emergence:

- The 'wild west' period (2017 to mid-2018) of Initial Coin Offerings (ICOs) ended badly for many

Security tokens

There were many attractions of ICOs:

- Decentralized/democratized finance - removing power from Wall Street, and bailed-out banks etc.
- IPO process is extraordinarily complex, slow and expensive, whereas ICOs could be done in an afternoon (almost)
- As we have seen, ERC-20 tokens are very simple to deploy **and you no longer have to build your own blockchain!**

Lack of regulatory requirements, KYC, AML, **and the ability to sell directly to the man or woman on the street**

$\Rightarrow$ vast amounts could be raised on the basis of a 'white paper' or even a nice idea, in return for a token with (often) dubious or no value

Security tokens

That's great - if the idea is good

- Brave browser raised $\$36m$ in...

Security tokens

That's great - if the idea is good

- Brave browser raised $\$36m$ in...25 seconds

Security tokens

That's great - if the idea is good

- Brave browser raised $\$36m$ in...25 seconds
- Some other long term successes emerged (Aave, Cosmos,...)

But...

Security tokens

But those positive examples must be cherry-picked from **a much bleaker big picture dominated by fraud, theft and failure**. *The takeaway seems to be that ICOs can be very effective fundraising tools in individual cases where founders are trustworthy and well-intentioned, but that overall they invite massive fraud.*

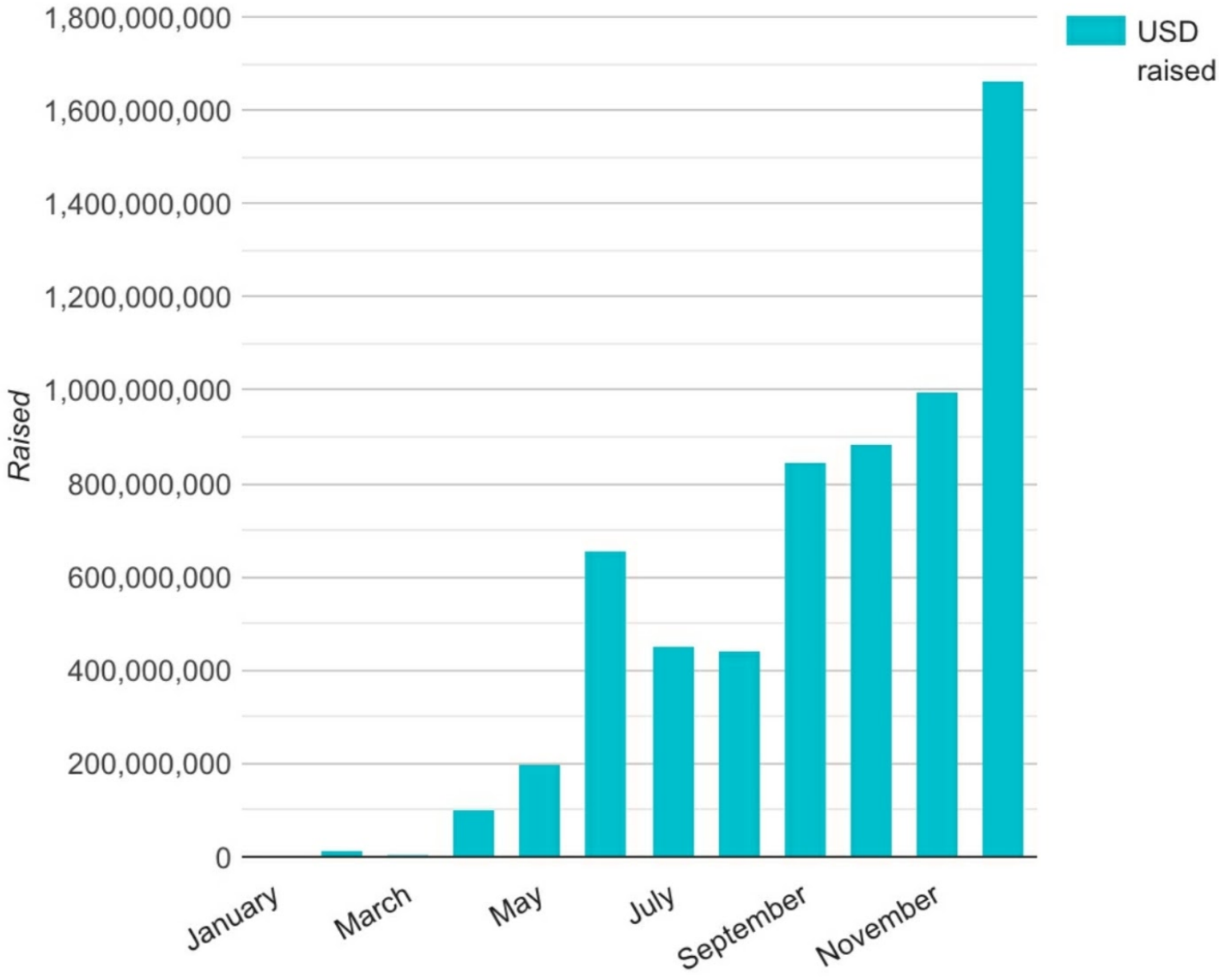
- Coindesk, June 1, 2023

Security tokens

Year	Number of ICOs	Total amount raised in USD
2014	2	16,032,802
2015	3	6,084,000
2016	29	90,250,273
2017	876	6,226,689,449
2018	1,251	7,705,825,063

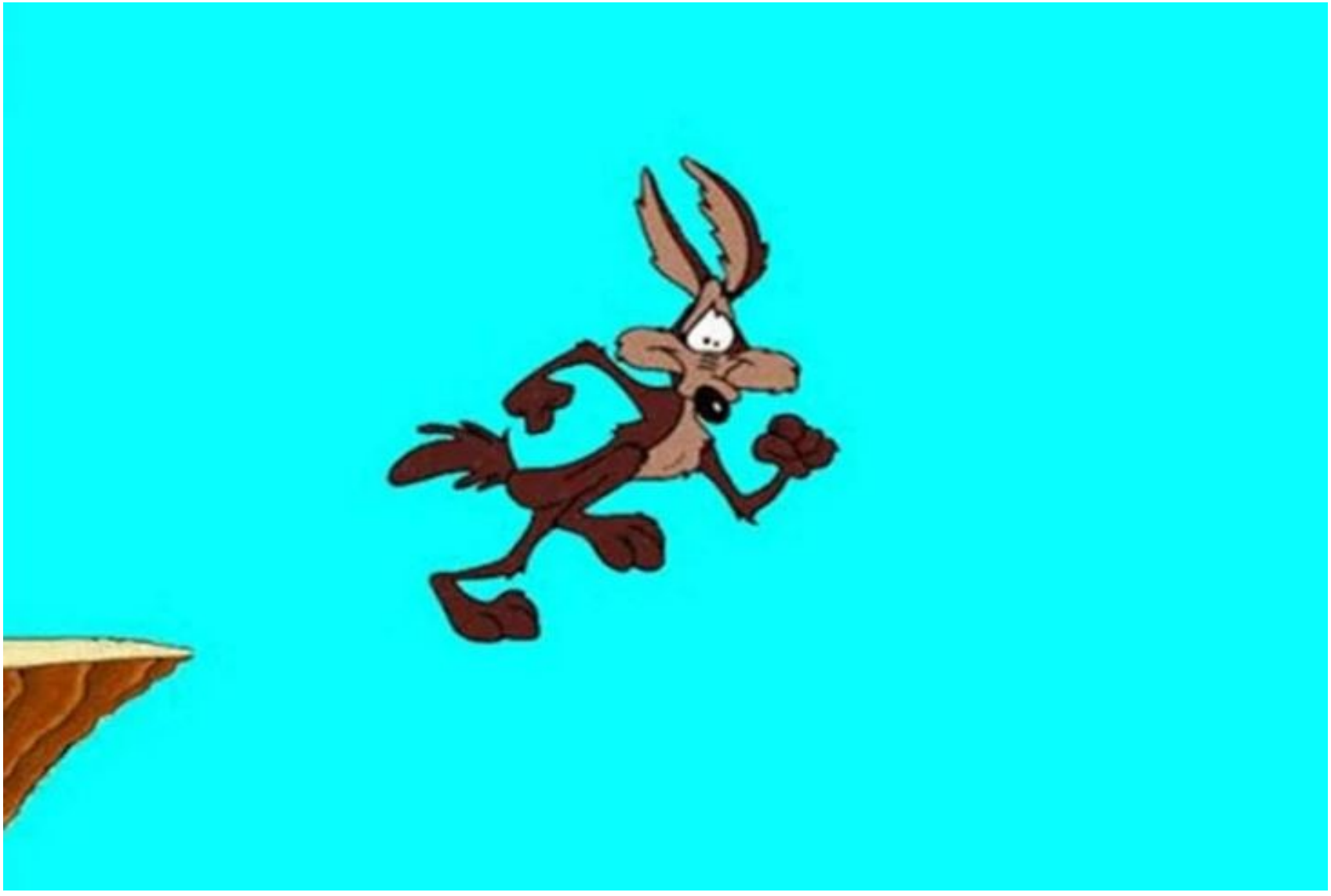
Source: Security Tokens and Stablecoins Quick Start Guide

Security tokens

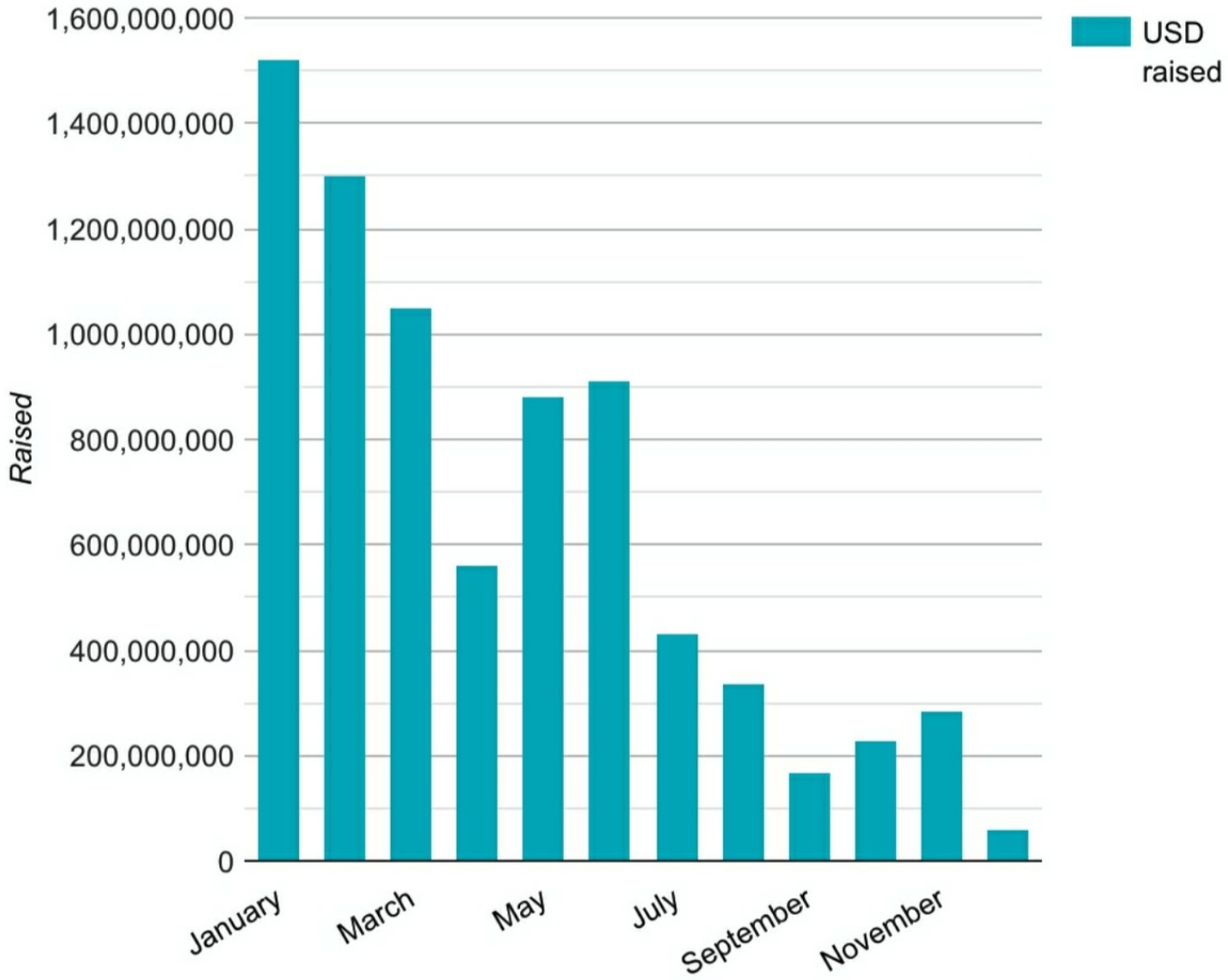


Source: Security Tokens and Stablecoins Quick Start Guide

Security tokens



Security tokens



Source: Security Tokens and Stablecoins Quick Start Guide

Security tokens

Regulators and market participants (somwehat) learnt the lesson that a more formal and measured approach was needed

- The STO emerged as a way of raising funds that retained some of the attractions of crypto and decentralization
- But it also brought regulatory compliance, governance
- Plus awareness that there should perhaps be controlled access (rather than *retail* investors signing up and getting burned)
- Intermediaries return to the process - e.g. in providing custody of underlying assets, roadshows, and conducting due dilligence

There remains the concern that legal frameworks in many/most countries are not yet completely compatible with digital assets

- These concerns also arise a lot in CBDC pilots/debates

Security tokens

An STO presumably reduces the risk of unexpected regulator objections, *ex post*

- We still are seeing ongoing debates and legal tussles over whether certain tokens are securities
- Interestingly, there is (*perhaps?*) a doctrine emerging whereby a token may be regarded as a security on issuance (say, to a narrow set of institutional investors) but not at a later stage
  - Decentralization may increase over time, through the secondary market
  - Currently debated in the SEC vs Ripple case as it relates to the commonly cited Howey test

Security tokens

Let's return to ERC standards - in particular, consider ERC-3643

> *The T-REX token is an institutional grade security token standard. This standard provides a library of* **interfaces for the management and compliant transfer** *of security tokens, using an automated on-chain validator system leveraging* **onchain identities for eligibility checks.**
>
> *These tokens cannot be permissionless like utility tokens; they must be permissioned to* **track ownership and ensure that only eligible investors** *can hold tokens.*

Obviously, much more like a traditional security

Security tokens

It aims to provide a comprehensive framework for **managing the life-cycle** *of security tokens, from issuance to transfers between* **eligible investors**, *while enforcing* **compliance rules at every stage**. *The standard also supports additional features such as token pausing and freezing, which can be used to manage the token in response to* **regulatory requirements** *or changes in the status of the token or its holders.*

Moreover, the standard is designed to work in conjunction with an on-chain identity system, allowing for the **validation of the identities and credentials of investors** *through signed attestations issued by* **trusted claim issuers**.

- In various CBDC trials (though not ERC-3643) there have already been applications controlling dividends, coupons, repo - and even tokenization of syndicated loans

Security tokens

The aim (perhaps?) is to retain *some* of the technological benefits of blockchain but also **a lot** of traditional regulatory control and centralization

- Difficult balance to strike
- Will we see decentralized shadow tokenization if regulation is too heavy handed?

Will this herald a golden era of efficient reg-tech / automated compliance?

- Recall our ERC-20 interface and flexibility
- ERC-3643 specifies interfaces for (100+) functions!
- See the 'compliance' interface

Tokenized securities

The power of blockchain and smart contracts may allow innovation and enhancements for traditional securities

- However, many securities and assets are already traded extremely efficiently
- It is in tokenizing real world assets (RWAs) where 'tokenized securities' might be transformative
- Digital representations of such assets could be put on blockchain

Tokenized securities

ERC-3643 documentation refers to assets like real estate, or funding for small businesses

- Market funding for such assets is expensive or non-existant
- Intermediation chains across different countries - with complex manual / paper-based compliance - also limit the pool of possible investors
- Settlement risks from delays and non-atomicity prevent many international and multistage transactions

Blockchain's automation, transparency, rapidity, and accessibility *could* ameliorate all these problems

- At least, that's the hope
- We will return to RWA tokenization later in the course