Flexible Anonymous Transactions (FLAX): Towards Privacy-preserving and Composable Decentralized Finance

Wei Dai
Bain Capital Crypto

February 7th, 2022

Media Attention on DeFi, Sept 2021

The New York Times

Crypto Banking and Decentralized Finance, Explained

The revolution in digital money is now moving into banking, as cryptocurrency starts to reshape the way people borrow and save.

Sept. 05, 2021

THE WALL STREET JOURNAL

SUBSCRIBE | SIGN IN

MARKETS | STREETWISE

DeFi Is Crypto's Wall Street, Without a Safety Net

Decentralized finance allows crypto enthusiasts a do-it-yourself version of investment banking, bringing high rewards and huge risks

Sept. 14, 2021

Forbes

Sep 9, 2021, 11:45pm EDT | 4,850 views

DeFi Innovates At A Blistering Pace As Regulators Take Action

The Economist

Subscribe | Menu

Briefing

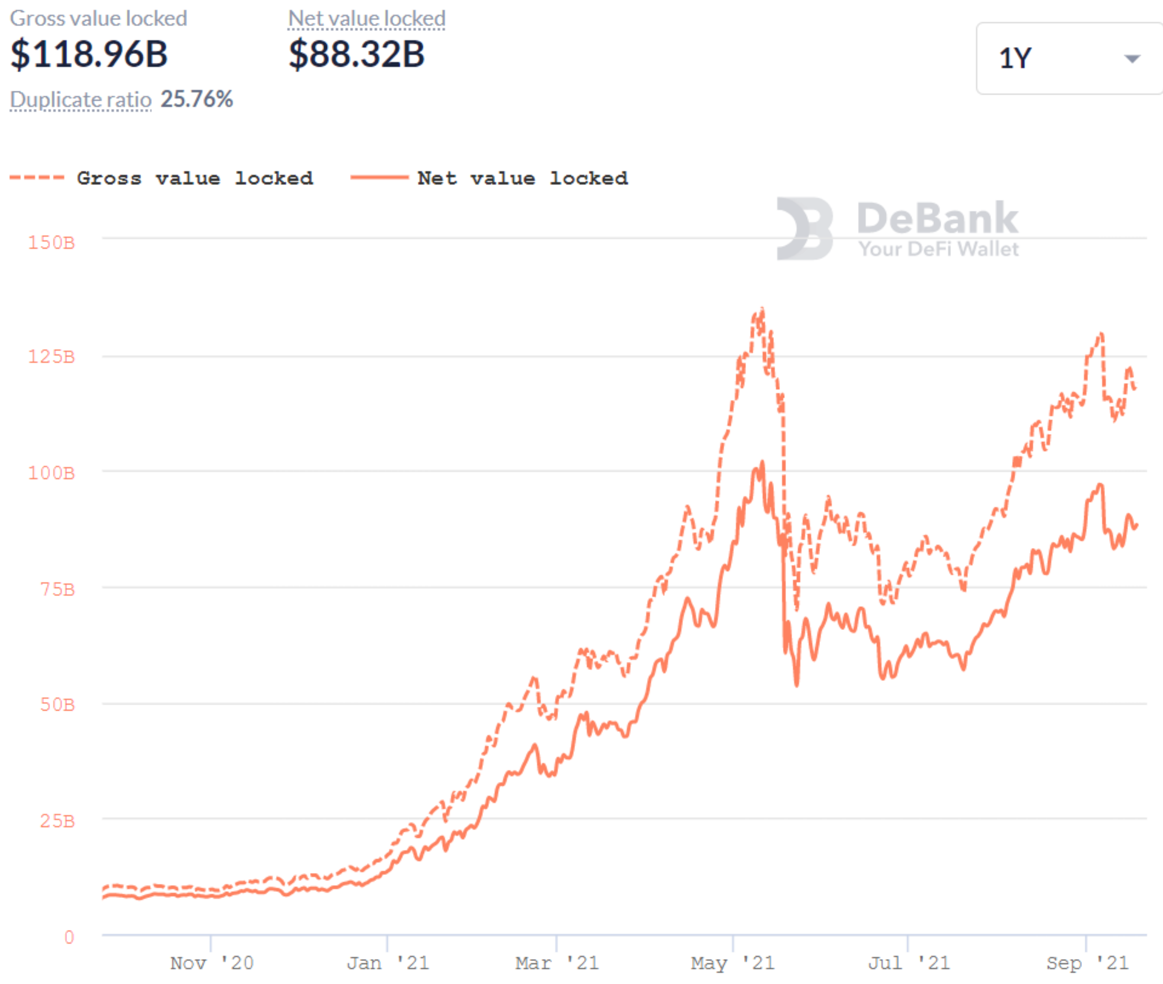
Sep 18th 2021 edition ›

Curiouser and curiouser

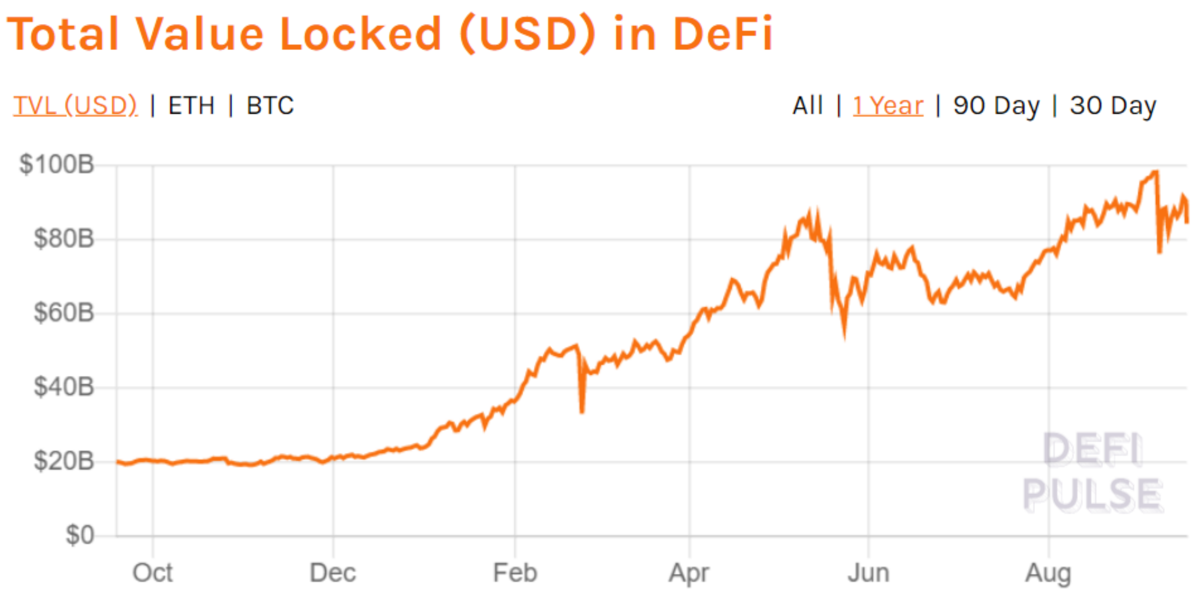
Adventures in DeFi-land

Can decentralised finance lay the foundations for an open digital economy?

Explosion of (Ethereum) DeFi in '20-'21



DeBank



DeFi Pulse

TVL: Total Value Locked ~ Assets under management (AUM) in finance

What is DeFi?

Smart-contract applications operating on distributed ledgers offering financial services beyond payments, such as **asset management**, **trading**, **lending**, and **financial derivates**.

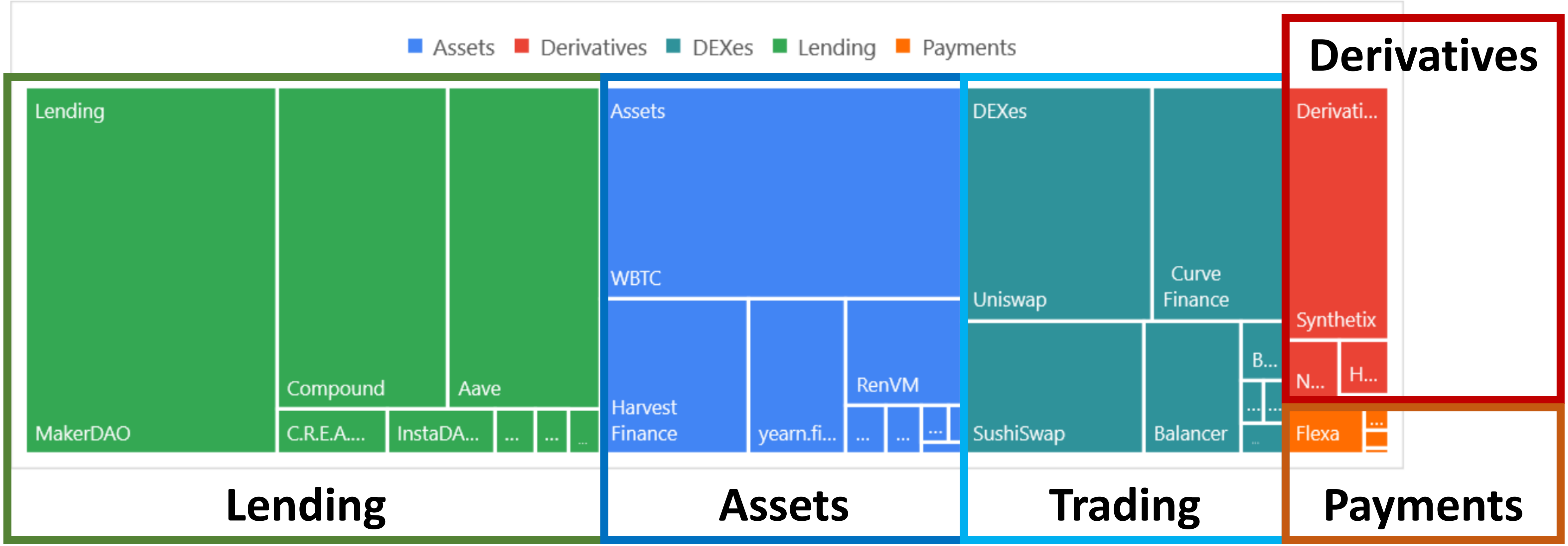
Central banks

Stock exchanges

Investment banks

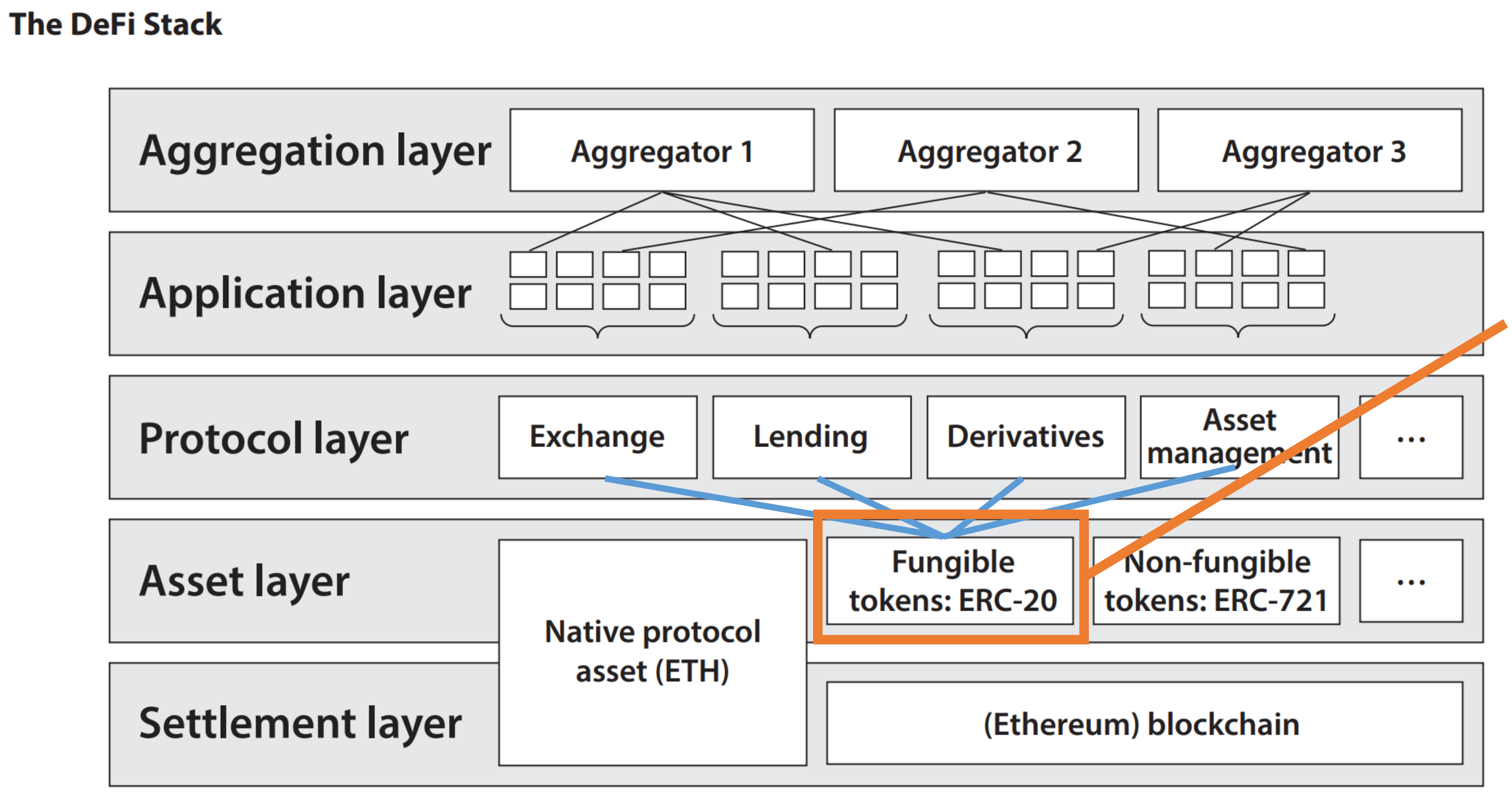
Commercial banks

DeFi on Ethereum, by TVL



[AmlerEckeyFaustKaierSandnerScholsser21]
Data from defipulse.com

DeFi on Ethereum



Interface standard for **user and contracts** to own and use tokens.

[Schär21]

More details on Ethereum DeFi

Aggregators

DEXes

Lending

Assets

Derivatives

ERC20 Token Standard

“Wrapped” tokens

Stablecoins

Project tokens

DeFi Challenges

Scalability

Ethereum	NASDAQ	Visa
~20 tx/s	~1,000 tx/s	~7,000 tx/s

Vulnerabilities

Flash Boys 2.0:
Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability

Philip Daian, *Cornell Tech*, phil@cs.cornell.edu
Steven Goldfeder, *Cornell Tech*, goldfeder@cornell.edu
Tyler Kell, *Cornell Tech*, sk3259@cornell.edu
Yunqi Li, *UIUC*, yunqil3@illinois.edu
Xueyuan Zhao, *CMU*, xyzhao@cmu.edu
Iddo Bentov, *Cornell Tech*, ib327@cornell.edu
Lorenz Breidenbach, *ETH Zürich*, lorenz.breidenbach@inf.ethz.ch
Ari Juels, *Cornell Tech*, juels@cornell.edu

Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit

Kaihua Qin, *Imperial College London*
Liyi Zhou, *Imperial College London*
Benjamin Livshits, *Imperial College London*
Arthur Gervais, *Imperial College London*

Protocol	**Loss**	**Audit**	**Attack**	**Date**	**Ref.**
bZx	0.35m	✓	TX sandwich	Feb-15-2020	[122]
bZx	0.63m	✓	TX sandwich	Feb-18-2020	[123]
Uniswap	0.30m	✓	Reentrancy	Apr-18-2020	[124]
dForce	25.00m	✗	Reentrancy	Apr-19-2020	[98]
Hegic	0.05m	✗	Logical bug	Apr-25-2020	[125]
Balancer	0.50m	✓	TX sandwich	Jun-28-2020	[126]
Opyn	0.37m	✓	Logical bug	Aug-04-2020	[127]
Yam	0.75m	✗	Logical bug	Aug-12-2020	[105]
bZx	8.10m	✓	Logical bug	Sep-14-2020	[7]
Eminence	15.00m	✗	TX sandwich	Sep-29-2020	[128]
MakerDAO	-	✓	Governance	Oct-26-2020	[111]
Harvest	33.80m	✓	TX sandwich	Oct-26-2020	[10]
Percent	0.97m	✓	Logical bug	Nov-04-2020	[129]
Cheese Bank	3.3m	✓	TX sandwich	Nov-06-2020	[130]
Akropolis	2.00m	✓	Reentrancy	Nov-12-2020	[8]
Value DeFi	7.00m	✗	TX sandwich	Nov-14-2020	[112]
Origin	7.00m	✓	Reentrancy	Nov-17-2020	[11]
88mph	0.01m	✓	Logical bug	Nov-17-2020	[131]
Pickle	19.70m	✗	Logical bug	Nov-21-2020	[132]
Compounder	10.80m	✓	Logical bug	Dec-02-2020	[133]
Warp Finance	7.80m	✓	TX sandwich	Dec-18-2020	[134]
Cover	9.40m	✓	Logical bug	Dec-28-2020	[9]
Yearn	11.00m	✗	TX sandwich	Feb-05-2021	[135]
Growth DeFi	1.30m	✓	Logical bug	Feb-09-2021	[136]
Meerkat Finance	32.00m	✗	Logical bug	Mar-04-2021	[137]
Paid Network	27.00m	✗	Logical bug	Mar-05-2021	[138]
DODO	2.00m	✗	Logical bug	Mar-09-2021	[139]

[WPGKHK21]

DeFi Challenges

Legal and regulation

Gary Gensler, Chair of SEC, Aug. 2021


Forbes

EDITORS' PICK | Aug 27, 2021, 09:01am EDT | 19,525 views

SEC Signs Deal To Investigate DeFi Transactions

*"This asset class is **rife with fraud, scams and abuse** in certain applications … We need additional **congressional authorities** to prevent transactions, products and platforms from falling between regulatory cracks."*

Privacy

*".. the decentralization, openness and integrity protection of blockchain technologies pose challenges for **compliance with privacy regulations**.."* [AEFKSS21]

*"The **anonymity and privacy** of DeFi protocols is at present a <u>significantly understudied</u> area."* [WPGKHK21]

Privacy in Blockchains

Privacy-preserving payments
(Decentralized Anonymous Payments, DAPs)

- Zerocash [BCGGMTV14]
- RingCT [Noether15,SALY17,YSLAEZG20]
- Mimblewimble [Jedusor16,Poelstra16,FOS19]
- Quisquis [FMMO19]
- Zether [BAZB20,Diamond21]

Payments, but towards DeFi

- Zether [BAZB20,Diamond21]
  Seal-bid auction, privacy-preserving PoS
- Manta [CXZ21]
  Zerocash w/ token swap
- SwapCT [EMPKB21]
  RingCT w/ token swap
- Penumbra

Privacy-preserving smart contracts

- Hawk [KMSWP16]
- Arbitrum [KGCWF18]
- Ekiden [CZKHHJJMS19]

Trust assumptions

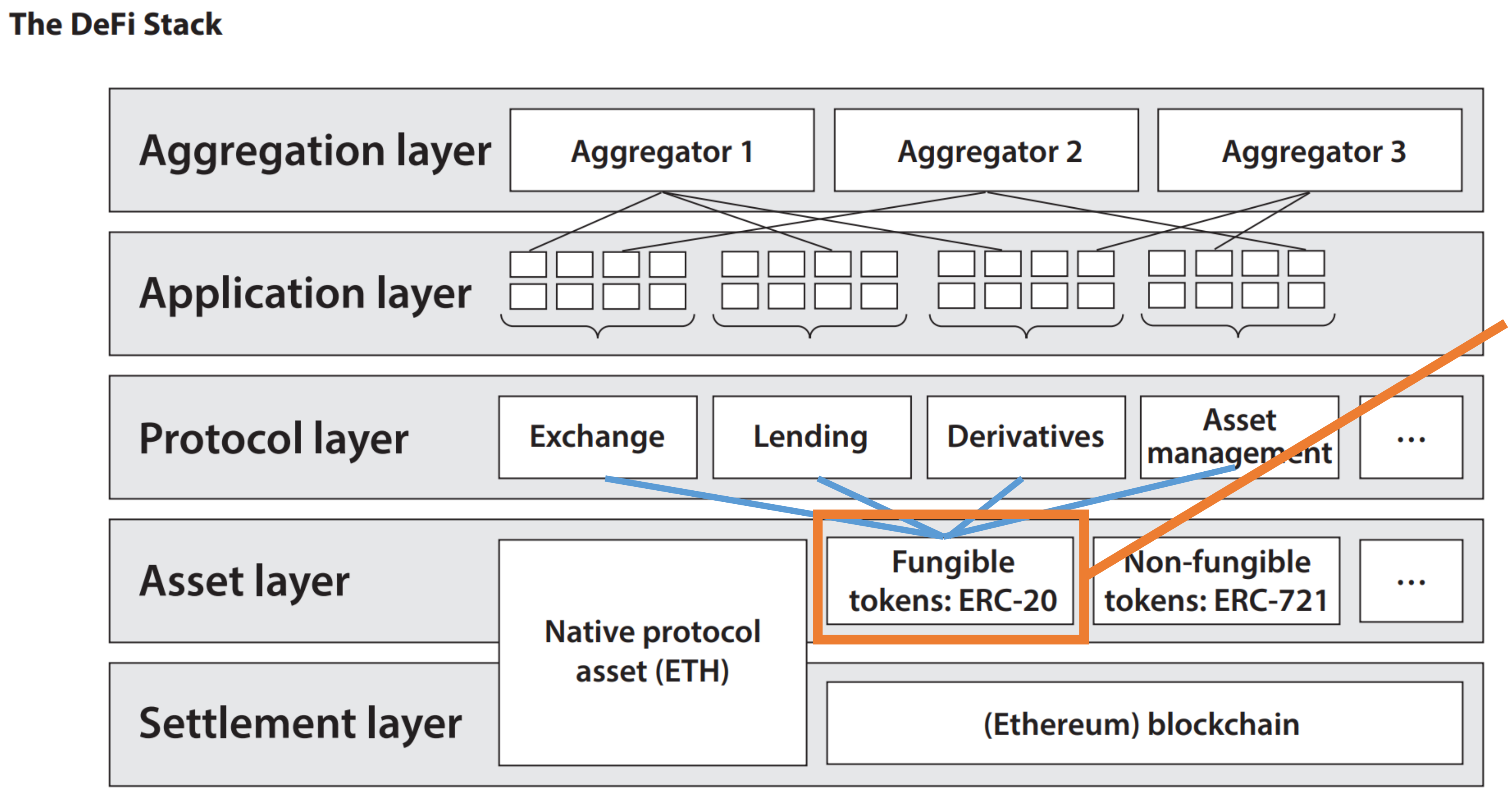
- Zkay [SBGMT19]
- Zexe [BCGMMW20]
- Kachina [KKK21]

Limited composability

*"The **anonymity and privacy** of DeFi protocols is at present a significantly understudied area."* [WPGKHK21]

No known privacy-preserving solutions for the current DeFi ecosystem.

Recall: ERC20 is Central to Ethereum DeFi



Interface standard for **user and contracts** to own and use tokens.

[Schär21]

Our work

Q: *Can we design an*

*ERC20-like **privacy-preserving** token standard?*

Our answer: Yes.

Current Defi		Proposal
Computation: **Public**	→	Computation: **Public**
Accounting: **Public**		*Accounting*: **Privacy-preserving**

Outline

Existing DAPs

Zerocash [BCGGMTV14]

RingCT [Noether15,SALY17,YSLAEZG20]

Quisquis [FMMO19]

Zether [BAZB20,Diamond21]

Flexible Anonymous Transactions (FLAX) System

Cryptographic building block

FLAX Token Standard

"Privacy-preserving ERC20"

Privacy-preserving DeFi

Asset pools

Trading / DEXes

Lending

Outline

Existing DAPs
Zerocash [BCGGMTV14]
RingCT [Noether15,SALY17,YSLAEZG20]
Quisquis [FMMO19]
Zether [BAZB20,Diamond21]

Flexible Anonymous Transactions (FLAX) System
Cryptographic building block

FLAX Token Standard
"Privacy-preserving ERC20"

Privacy-preserving DeFi
Asset pools
Trading / DEXes
Lending

Flexible Anonymous Transaction System

Parameter generation: param $\twoheadleftarrow$ ParamGen()

Ledger algorithms

st $\twoheadleftarrow$ Setup(lid)

st' / $\perp$ $\twoheadleftarrow$ Process(st, tx)

User algorithms

(pk, sk) $\twoheadleftarrow$ Keygen()

[0, MAX] $\ni$ bal / $\perp$ $\twoheadleftarrow$ Read(st, sk)

tx / $\perp$ $\twoheadleftarrow$ CreateTx(st, sk, (pk', amt), val, AD)

st – current ledger state
sk – secret key of tx originator
pk' – public key of recipient
amt $\in$ [0, MAX] – to be hidden transfer amount
val $\in$ [-MAX, MAX] – **publicly declared** net value change, accessible as tx.val
AD $\in \{0, 1\}^*$ – **publicly declared** associated data, a key-value store, accessible as tx.AD

- Account-based syntax: st maintains "encrypted" balances for each pk
- Spending from account pk is authenticated via corresponding sk
- A tx modifies account balances, in a prescribed manner (to be discussed next)
- Notation: tx.XX := tx.AD.XX

Transaction Types

ctx / dtx / ttx / $\perp \twoheadleftarrow$ CreateTx(st, sk, (pk', amt), val, AD)

sk – secret key of tx originator
pk' – public key of recipient
amt $\in$ [0, MAX] – to be hidden transfer amount
val $\in$ [-MAX, MAX] – publicly declared net change
AD – associated data, a key-value store

Credit Tx

ctx $\twoheadleftarrow$ CreateTx(st, sk, val = 1, AD)
Parameterizable with k $\in$ [0, MAX], i.e. ctx[k]
"Credit account of sk by k"
Valid for any ledger state, ctx[k].val := k

Debit Tx

dtx $\twoheadleftarrow$ CreateTx(st, sk, val, AD) // val < 0
Parameterizable with k $\in$ [val, 0], i.e. dtx[k]
"Debit account of sk by k", dtx[k].val := k

Transfer Tx

ttx $\twoheadleftarrow$ CreateTx(st, sk, (pk', amt), val, AD)
"Debit (tx.val - amt) from account of sk,
credit account of pk' by amt", ttx.val := val

- All txs declare public net change as tx.val
- tx.AD is authenticated by tx originator
- Txs type is public
- tx is *valid* wrt st if Process(st, tx) $\neq \perp$
- Parameters for dtx and ctx can be determined at processing time
- ctx should be valid for any st

Main changes from previous syntax (DAPs):

1. Associated data AD
2. Credit and debit transactions

Correctness and Security, Briefly

Correctness
If user has balance b, then she can spend it.

Consistency, security for the ledger
Transactions do not overdraft and declare the correct net value change in tx.val.

Transaction integrity, security for the user, like UF-CMA and INT-CTXT.
Adversary, **even with transaction oracle access**, cannot forge *new* tx that decreases balance of honest users.

Transaction privacy, security for the user
Anonymity for tx originator, and confidentiality of transfer amt and recipient

Replay protection
Each tx can only be applied once among a set of honest ledgers.

Properties of FLAX tx (Informal):

- tx.val declares net value change.
- If $\perp \neq st' \longleftarrow \text{Process}(st, tx)$, no overdrafts occur.
- Entire tx (esp. tx.AD) is “signed” by sk-holder.
- tx can be processed exactly once.

Outline

Existing DAPs
Zerocash [BCGGMTV14]
RingCT [Noether15,SALY17,YSLAEZG20]
Quisquis [FMMO19]
Zether [BAZB20,Diamond21]

Privacy-preserving DeFi
Asset pools
Trading / DEXes
Lending

Flexible Anonymous Transactions (FLAX) System
Cryptographic building block

FLAX Token Standard
"Privacy-preserving ERC20"

FLAX Token Standard

<u>Contract TokenStandard extends ERC20</u>

global bal, st

Constructor:

  st $\twoheadleftarrow$ FLAX.Setup(this)

FTransfer(TX: Set(tx)):

  netval $\longleftarrow \Sigma_{tx \in TX}$ tx.val

  If (netval $\neq$ 0) then

    bal[caller] $\longleftarrow$ bal[caller] + netval

    Require (bal[caller] $\geq$ 0)

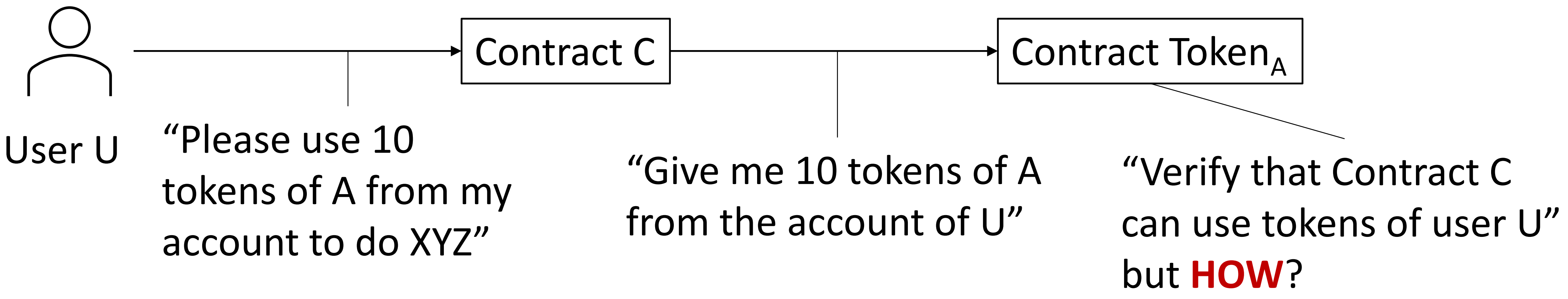
  For tx $\in$ TX:

    VerifyIntent(tx.intent)

    st $\twoheadleftarrow$ FLAX.Process(st, tx)

- *Privacy-preserving accounting* for end-users.
- Contracts use ERC20-interface to use their tokens. **No privacy for contract accounts**.
- "FTransfer" provides **anonymity for end-user transactions**.
- Limited tx confidentiality.
- Anonymity of tx => Privacy of user.
- VerifyIntent to be explain.

Delegation of Token-use



- ERC20 enables delegation of token-use via “allowance”.
  - Users tell $Token_A$ who (contracts) can use their tokens and how much (allowance)
- Our proposal (ticket approach): txs specify their intended usage, act as “ticket”.
- User U give dtx_A to Contract C. $Token_A$ only process dtx_A if certain conditions are satisfied.

Automated Market Maker (AMM), aka Liquidity Pool

<u>Contract AMM extends Pool</u>

cptAtoB(int) $\longrightarrow$ int / $\perp$

SwapAtoB(dtx$_A$, ctx$_B$, minOut):

  out $\longleftarrow$ cptAtoB(dtx$_A$.val) ; require (out $\geq$ minOut)

  TokenA.FTransfer(dtx$_A$)

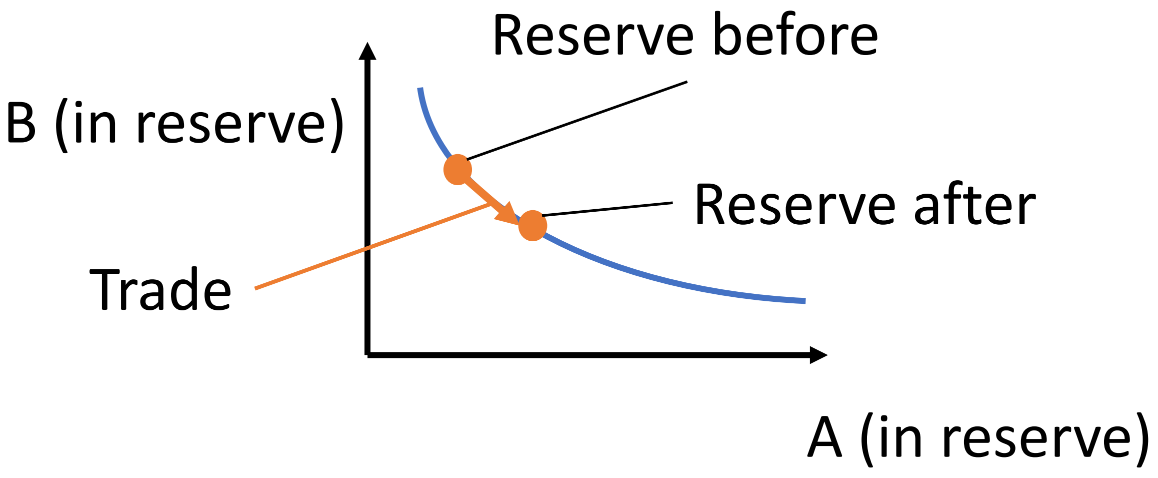
  TokenB.FTransfer(ctx$_B$[out])

- User spends A in dtx$_A$
- User obtains B with ctx$_B$
- Txs sent to a third contract!

Sidenote on AMM:

cptAtoB – many implementations

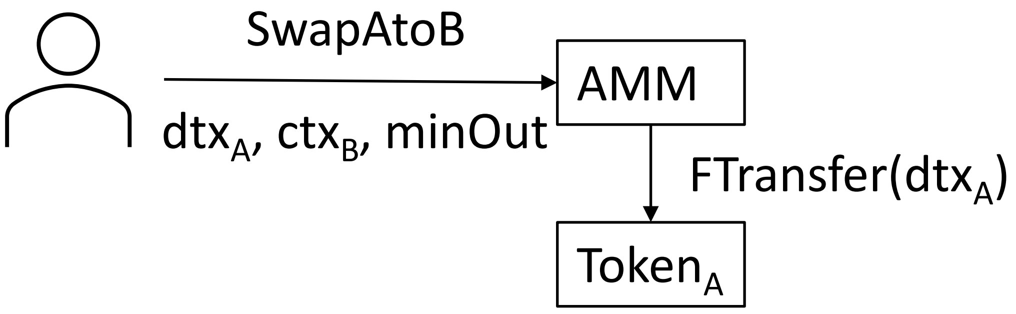
- Constant product (Uniswap)
- Constant sum (Curve)
- Other variants [AngEvaChi21]

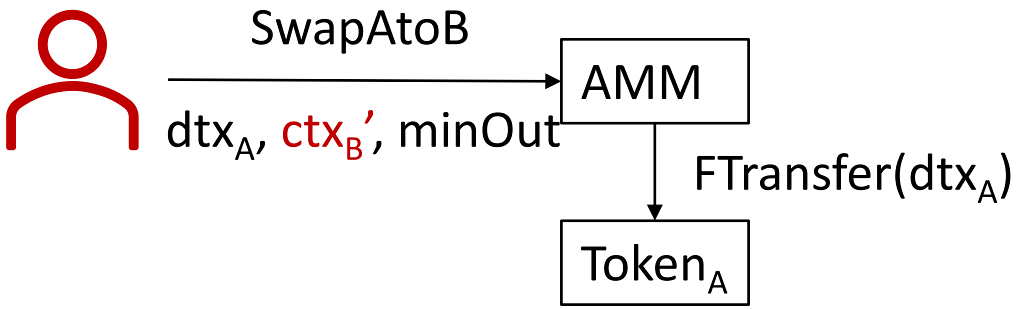


Tx Substitution Attack and Inter-contract Call Stack

Honest user constructs "AMM.SwapAtoB(dtx_A, ctx_B, minOut)"

Adversary intercepts it, submits the call "AMM.SwapAtoB(dtx_A, ctx_B', minOut)", where ctx_B' benefit the attacker instead.





Honest inter-contract call stack (ICCS)

AMM.SwapAtoB(dtx_A, ctx_B, minOut)

$Token_A$.FTransfer(dtx_A)

Malicious ICCS

AMM.SwapAtoB(dtx_A, ctx_B', minOut)

$Token_A$.FTransfer(dtx_A)

Authenticating Tx Intent via ICCS

VerifyIntent(*Intent*)

1. Input *Intent* is an inter-contract call stack (ICCS) pattern
2. Compares current ICCS with *Intent*
3. **Throws error** if matching **fails**
4. **Returns true** if matching **succeeds**

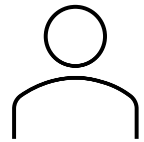
Users must construct dtx and ttx specifying the **exact contract-call intent for usage of their funds**.

More on ICCS:

- EVM only exposes tx.origin, msg.caller, and current calldata.
- Feasible if a smart-contract call is carried out “at one place”
- Potentially useful in heterogenous contract interactions
- Example: can be used to prevent re-entry attacks

Constructing Smart Contract Calls

tx – debit or transfer
ctx – credit



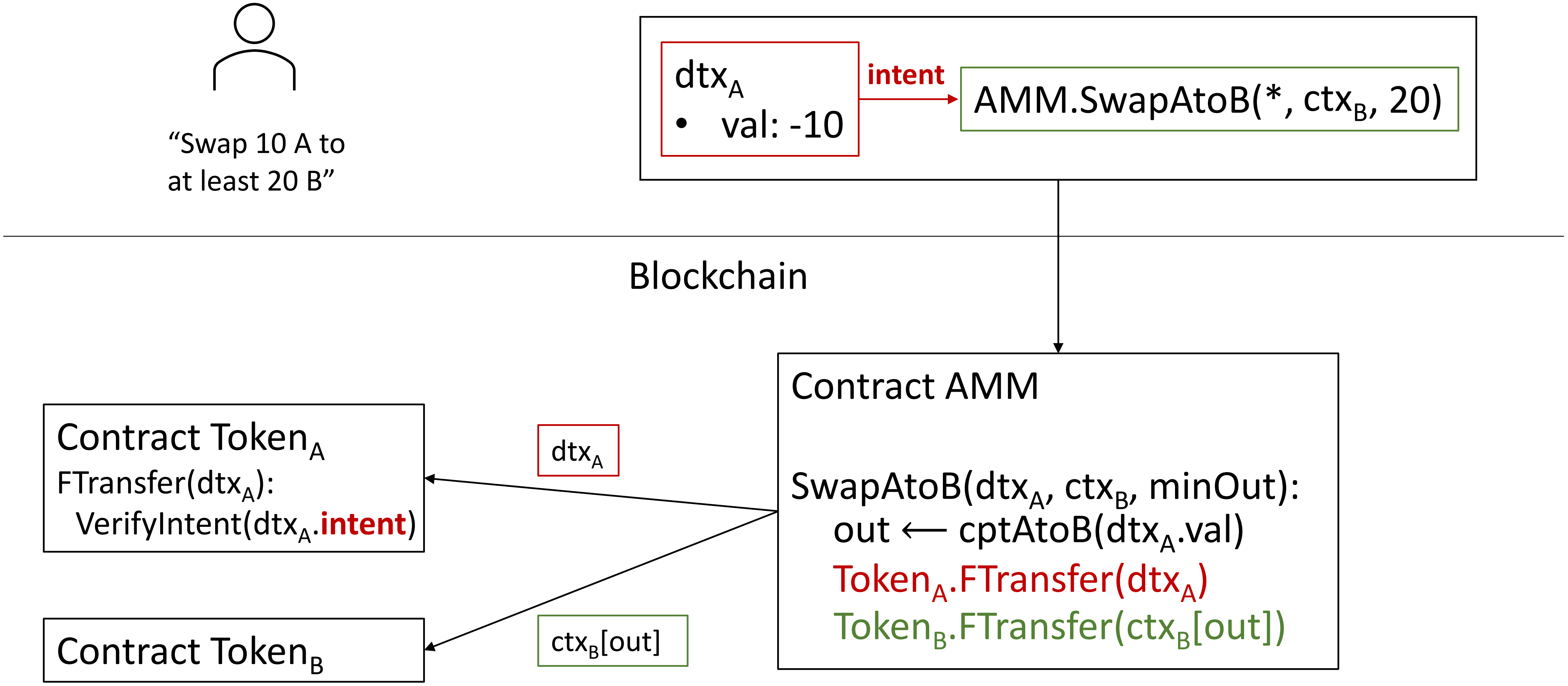
$\mathrm{Func}(tx_1, \ldots, tx_n, ctx_1, \ldots, ctx_m, arg) \longrightarrow$ Contract

$tx_1, \ldots, tx_n \xrightarrow{\textbf{intent}} \mathrm{Contract.Func}(*, ctx_1, \ldots, ctx_m, arg)$

1. Construct all credit transactions, $ctx_1, \ldots, ctx_m$.

2. Construct $tx_1, \ldots, tx_n$, tx_i.intent := "Contract.Func(*, $ctx_1, \ldots, ctx_m$, arg)".

Full Example w/ AMM Swap



FLAX Gas Token

Fix a distinguised token, $Token_{Gas}$.

dtx_{call}
- val // gas paid limit
- gasprice
- intent

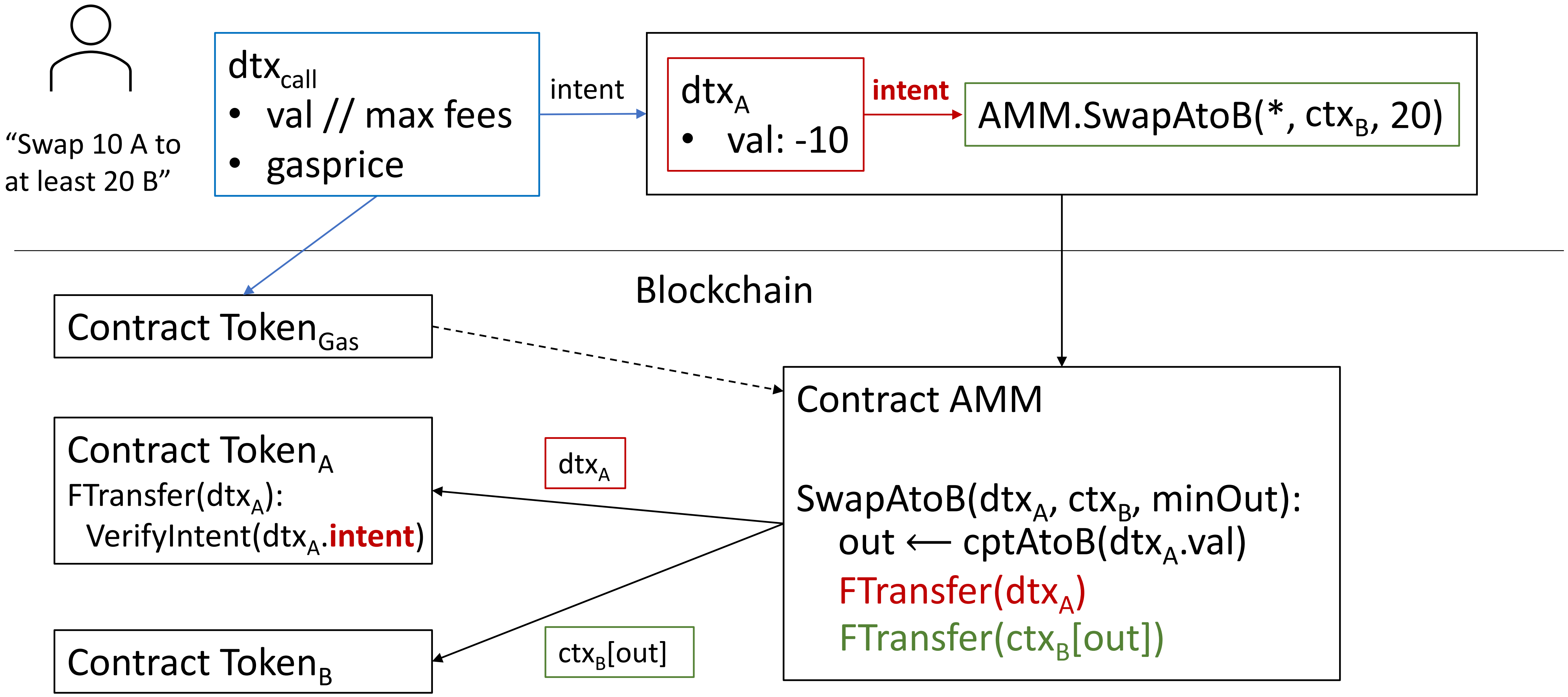
Full smart contract call intent

$tx_1, \ldots, tx_n \rightarrow$ Contract.Func(*, $ctx_1, \ldots, ctx_m$, arg)

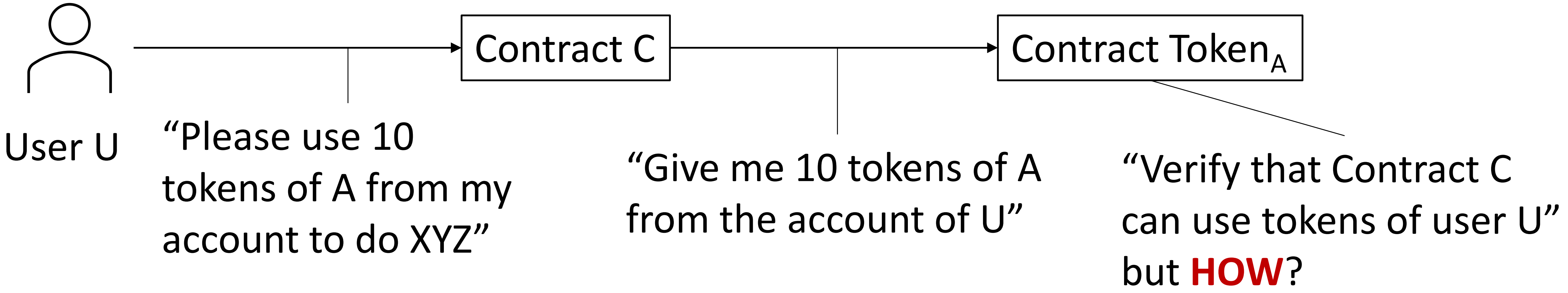
- A blockchain transaction is a single debit transaction, paying upto dtx.val for gas.
- dtx_{call}.intent specifies full contract call, i.e "Contract.Func($tx_1, \ldots, tx_n, ctx_1, \ldots, ctx_m$, arg)".

Consequence: "tx.origin" no longer available, as well as "caller" during the initial call.

Extended Example



Delegation of Token-use



- tx.intent delegate token-use to a particular partial smart contract invocation.
- A non-anonymous FLAX system can be achieved w/ only signatures:
  - User simply sign tx = {val: -10, token: A, intent: …}
- Compare to “transfer and call” ERC223: preserve currently used “top-down” approach, easier to use multiple types of tokens in one call (e.g EnterPool).
- Downside: require read-access to ICCS, not supported on current systems.

Outline

Existing DAPs

Zerocash [BCGGMTV14]

RingCT [Noether15,SALY17,YSLAEZG20]

Quisquis [FMMO19]

Zether [BAZB20,Diamond21]

Flexible Anonymous Transactions (FLAX) System

Cryptographic building block

FLAX Token Standard

“Privacy-preserving ERC20”

Privacy-preserving DeFi

Asset pools

Trading / DEXes

Lending

Token-denominated Funds (aka pools)

<u>Contract Pool extends TokenStandard</u>

$\text{cptEnter}(\text{val}_A, \text{val}_B) \longrightarrow (\text{in}_A, \text{in}_B, \text{out}_P)$

$\text{cptExit}(\text{val}_P) \longrightarrow (\text{out}_A, \text{out}_B)$

$\text{EnterPool}(\text{dtx}_A, \text{dtx}_B, \text{ctx}_P)$:

  $(\text{in}_A, \text{in}_B, \text{out}_C) \longleftarrow \text{cptEnter}(\text{dtx}_A.\text{val}, \text{dtx}_B.\text{val})$

  $\text{Token}_A.\text{FTransfer}(\text{dtx}_A[\text{in}_A])$

  $\text{Token}_B.\text{FTransfer}(\text{dtx}_B[\text{in}_B])$

  $\text{bal}[\text{this}] \longleftarrow \text{bal}[\text{this}] + \text{out}_P$

  $\text{this.FTransfer}(\text{ctx}_P[\text{out}_P], \text{this})$

$\text{ExitPool}(\text{dtx}_P, \text{ctx}_A, \text{ctx}_B)$:

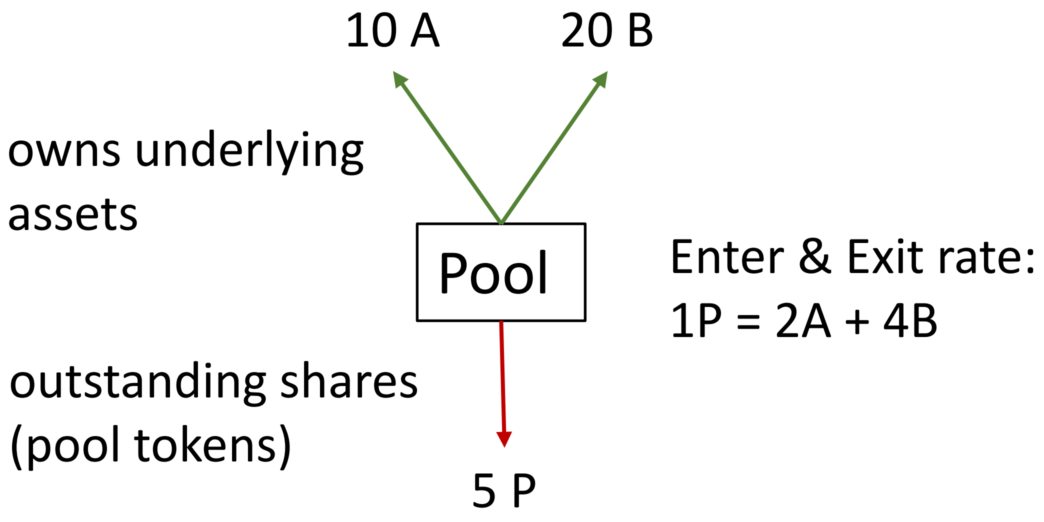
  $(\text{out}_A, \text{out}_B) \longleftarrow \text{cptExit}(\text{dtx}_P.\text{val})$

  $\text{Token}_A.\text{FTransfer}(\text{ctx}_A[\text{out}_A])$

  $\text{Token}_B.\text{FTransfer}(\text{ctx}_B[\text{out}_B])$

  $\text{this.FTransfer}(\text{dtx}_P)$

  $\text{bal}[\text{this}] \longleftarrow \text{bal}[\text{this}] - \text{dtx}_P.\text{val}$



- AMM, shown previous, is a pool that expose additionally SwapAtoB.
- Contract can manage its own assets arbitrarily.

Collateralized Debt Positions (CDP)

Vault

- Contains collateral, $(val_{collateral}, ctx_{refund})$
- Records outstanding debt, val_{debt}

Contract CDP
global vaults

<u>OpenVault(dtx_A, ctx_{rfd}, ctx_B, borrow):</u>
… // checks
$Token_A$.FTransfer(dtx_A)
$Token_B$.FTransfer(ctx_B[borrow])
vid $\longleftarrow$ vaults.push((dtx_A.val, borrow, ctx_{rfd}))
Return vid

<u>Repay(vid, dtx_B):</u>
… // checks
(coll, debt, ctx_{rfd}) $\longleftarrow$ vaults[vid]
$Token_B$.FTransfer(dtx_B)
$Token_A$.FTransfer(ctx_{rfd}[coll])

<u>Liquidate(vid, dtx_B, ctx_A):</u>
… // checks
(coll, debt, *) $\longleftarrow$ vaults[vid]
$Token_B$.FTransfer(dtx_B)
$Token_A$.FTransfer(ctx_A[coll])

Life-cycle of a vault

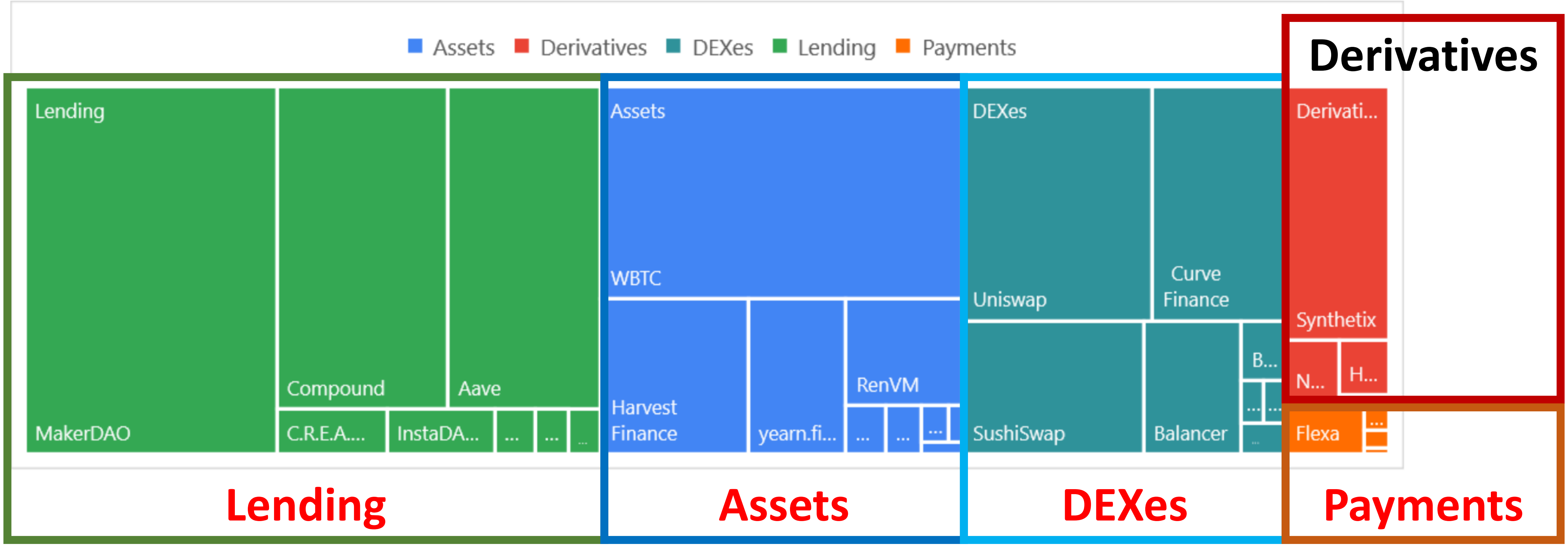
OpenVault
Deposit 10 A
Borrow 20 B

Repay
Repay 20 B
Withdraw 10 A

Liquidate
Pay 20 B
Buys ~10A

- Collateralized stablecoins (Dai stablecoin)
- Extendable to multi-asset lending w/ interest rates (Aave, Compound)

DeFi on Ethereum, by TVL



[AmlerEckeyFaustKaierSandnerScholsser21]
Data from defipulse.com

Outline

Existing DAPs
Zerocash [BCGGMTV14]
RingCT [Noether15,SALY17,YSLAEZG20]
Quisquis [FMMO19]
Zether [BAZB20,Diamond21]

Flexible Anonymous Transactions (FLAX) System
Cryptographic building block

FLAX Token Standard
“Privacy-preserving ERC20”

Privacy-preserving DeFi
Asset pools
Trading / DEXes
Lending

Instantiations

Recall: two main changes in syntax

1. Authentication of associated data.
2. Flexible debit and credit, latter of which can be applied to any state.

1. Adding associated data:

- Tx structure: tx.body and tx.π
- tx.π PoK for **R** = { (st, tx.body ; sk, …) | “prover knows sk such that ..” }
- tx.AD is authenticated via inclusion in the statement (in particular tx.body)
- Need (weakly) simulation extractability

2. Flexible debit and credit

- Balances hidden via *homomorphic* commitment / encryption
- Flexibility comes for free

From UTXO-based DAPs

Coin
- Encodes owner pk
- Encodes value (committed, encrypted)
- Spendable knowing sk + (coin secret)

Ledger state (UTXO-set)

Coin	Coin	…	Coin

Tx (spend some coins, create some coins)
- Spending info: serial numbers, key images
- New coins: $\text{NewCoin}_1, \ldots, \text{NewCoin}_m$
- Proof

Covers Zerocash, RingCT, Quisquis, (and MimbleWimble).

ctx (create a coin)
- New coins: NewCoin
- Proof

dtx (spend some coins, refund some coins)
- Spending info: ..
- New coins: NewCoin
- Proof

From Zether

- Balance of pk is stored as an ElGammal ciphertext, $(g^r, g^b pk^r)$
- Ledger state is acc: pk $\longrightarrow \mathbf{G}^2$

	Account State **st**	Transfer ttx	Debit dtx[-c]	Credit ctx[c]
Alice	$(g^{r_A}, \mathsf{pk}_A^{r_A} g^{b_A})$	$(g^r, \mathsf{pk}_A^r g^{-c})$	$(g^r, \mathsf{pk}_A^r g^{-c})$	$(g^r, \mathsf{pk}_A^r g^{c})$
Bob	$(g^{r_B}, \mathsf{pk}_B^{r_B} g^{b_B})$	$(g^r, \mathsf{pk}_B^r g^{c})$	$(g^r, \mathsf{pk}_B^r g^{0})$	$(g^r, \mathsf{pk}_B^r g^{0})$
Charlie	$(g^{r_C}, \mathsf{pk}_C^{r_C} g^{b_C})$	$(g^r, \mathsf{pk}_C^r g^{0})$	$(g^r, \mathsf{pk}_C^r g^{0})$	$(g^r, \mathsf{pk}_C^r g^{0})$
		+ Proof π	+ Proof π	+ Proof π

Comparisons of Instantiations

FLAX from	Ledger		User		Authentication	
	Setup	$\lvert\mathtt{st}\rvert$	Read	Privacy	Mechanism	Structure size
Zerocash	Trusted	$\mathcal{O}(m)$	$\mathcal{O}(\lvert\mathtt{st}\rvert)$	Full	SNARK	$\mathcal{O}(c \cdot \log(m))$
RingCT	Pub. coin	$\mathcal{O}(m)$	$\mathcal{O}(\lvert\mathtt{st}\rvert)$	Tx ring	Ring Sig.	$\mathcal{O}(c \cdot r)$
Quisquis	Pub. coin	$\mathcal{O}(u)$	$\mathcal{O}(\lvert\mathtt{st}\rvert)$	Tx ring	NIZK	$\mathcal{O}(c \cdot r)$
Zether	Pub. coin	$\mathcal{O}(n)$	$\mathcal{O}(1)$	Acc ring	NIZK	$\mathcal{O}(r)$

c – coin overhead

r – ring size

m – history size

u – UTXO size

n – accounts

$n \ll u \ll m$

Zether-based instantiation give the best user efficiency, but the worst privacy.

Summary

- DeFi has emerged as a key application area for blockchains.
- Privacy is a fundamental challenge for DeFi.
- General purpose privacy-preserving smart contracts are not needed for DeFi.
- FLAX and associated token standard bridge the gap between payments and DeFi.

Known techniques for **DAPs** → **FLAX** Cryptographic building block → **FLAX Token Standard** → **Privacy-preserving** and **Composable DeFi**

Inter-contract Call Stack

*ePrint:2021/1249