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

Wei Dai
Bain Capital Crypto
February 7th, 2022
**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.

**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.

**DeFi Innovates At A Blistering Pace As Regulators Take Action**

Can decentralized finance lay the foundations for an open digital economy?
Explosion of (Ethereum) DeFi in ’20–’21

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 derivatives.

Central banks

Stock exchanges  Investment banks  Commercial banks
DeFi on Ethereum, by TVL

Data from defipulse.com

[AmlerEckyFaustKaierSandnerScholsser21]
DeFi on Ethereum

The DeFi Stack

Aggregation layer
- Aggregator 1
- Aggregator 2
- Aggregator 3

Application layer

Protocol layer
- Exchange
- Lending
- Derivatives
- Asset management
- ...

Asset layer
- Native protocol tokens: ERC-20
- Non-fungible tokens: ERC-721
- ...

Settlement layer
- Native protocol asset (ETH)
- (Ethereum) blockchain

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

[Schär21]
More details on Ethereum DeFi

**Aggregators**
- [Image of aggregators]

**DEXes**
- [Image of DEXes]

**Lending**
- [Image of lending]

**Assets**
- [Image of assets]

**Derivatives**
- [Image of derivatives]

**ERC20 Token Standard**
- **“Wrapped” tokens**
- **Stablecoins**
- **Project tokens**
DeFi Challenges

Scalability

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

Vulnerabilities

Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit

Kaihua Qiu  Liyi Zhou  Benjamin Livshits  Arthur Gervais
Imperial College London  Imperial College London  Imperial College London  Imperial College London
DeFi Challenges

**Legal and regulation**

Gary Gensler, Chair of SEC, Aug. 2021

“**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 **significantly understudied 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]
- Zkay [SBGMT19]
- Zexe [BCGMMW20]
- Kachina [KKK21]

“*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

The DeFi Stack

- Aggregation layer
  - Aggregator 1
  - Aggregator 2
  - Aggregator 3

- Application layer

- Protocol layer
  - Exchange
  - Lending
  - Derivatives
  - Asset management
  - ...

- Asset layer
  - Native protocol token: ERC-20
  - Non-fungible tokens: ERC-721
  - ...

- Settlement layer

[Schär21]

Interface standard for user and contracts to own and use tokens.
Q: Can we design an ERC20-like privacy-preserving token standard?

Our answer: Yes.
Outline

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

Flexible Anonymous Transactions (FLAX) System
Cryptographic building block

Privacy-preserving DeFi
Asset pools
Trading / DEXes
Lending

FLAX Token Standard
“Privacy-preserving ERC20”
Outline

**Existing DAPs**
- Zerocash \cite{BCGGMTV14}
- RingCT \cite{Noether15,SALY17,YSLAEZG20}
- Quisquis \cite{FMMO19}
- Zether \cite{BAZB20,Diamond21}

**Flexible Anonymous Transactions (FLAX) System**
Cryptographic building block

**Privacy-preserving DeFi**
- Asset pools
- Trading / DEXes
- Lending

**FLAX Token Standard**
“Privacy-preserving ERC20”
Parameter generation: $\text{param} \leftarrow \text{ParamGen}()$

**Ledger algorithms**

$\text{st} \leftarrow \text{Setup(lid)}$

$\text{st}' \mathbin{/} \bot \leftarrow \text{Process(st, tx)}$

**User algorithms**

$(\text{pk}, \text{sk}) \leftarrow \text{Keygen}()$

$[0, \text{MAX}] \ni \text{bal} \mathbin{/} \bot \leftarrow \text{Read(st, sk)}$

$\text{tx} \mathbin{/} \bot \leftarrow \text{CreateTx(st, sk, (pk', \text{amt}), \text{val}, \text{AD})}$

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

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

\[ \text{ctx} / \text{dtx} / \text{ttx} / \bot \leftrightarrow \text{CreateTx(st, sk, (pk', amt), val, AD)} \]

**Credit Tx**

\[ \text{ctx} \leftrightarrow \text{CreateTx(st, sk, val = 1, AD)} \]
Parameterizable with \( k \in [0, \text{MAX}] \), i.e. \( \text{ctx}[k] \)
“Credit account of \( \text{sk} \) by \( k \)”
Valid for any ledger state, \( \text{ctx}[k].\text{val} := k \)

**Debit Tx**

\[ \text{dtx} \leftrightarrow \text{CreateTx(st, sk, val, AD)} \quad // \quad \text{val} < 0 \]
Parameterizable with \( k \in [\text{val}, 0] \), i.e. \( \text{dtx}[k] \)
“Debit account of \( \text{sk} \) by \( k \)”
\( \text{dtx}[k].\text{val} := k \)

**Transfer Tx**

\[ \text{ttx} \leftrightarrow \text{CreateTx(st, sk, (pk', amt), val, AD)} \]
“Debit \( (\text{tx.val} - \text{amt}) \) from account of \( \text{sk} \),
credit account of \( \text{pk}' \) by \( \text{amt} \)”
\( \text{ttx.val} := \text{val} \)

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

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

**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-CTX.
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 \( \perp \neq st' \leftarrow \text{Process}(st, tx) \), 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]

Flexible Anonymous Transactions (FLAX) System
Cryptographic building block

Privacy-preserving DeFi
Asset pools
Trading / DEXes
Lending

FLAX Token Standard
“Privacy-preserving ERC20”
Contract TokenStandard extends ERC20

global bal, st

Constructor:
   st ← FLAX.Setup(this)

FTransfer(TX: Set(tx)):
   netval ← \sum_{tx \in TX} tx.val
   If (netval \neq 0) then
      bal[caller] ← bal[caller] + netval
   Require (bal[caller] \geq 0)

For tx ∈ TX:
   VerifyIntent(tx.intent)
   st ← 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 $\text{Token}_A$ 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 $\text{dtx}_A$ to Contract C. $\text{Token}_A$ only process $\text{dtx}_A$ if certain conditions are satisfied.

![Diagram of token delegation process]
Automated Market Maker (AMM), aka Liquidity Pool

Contract AMM extends Pool

cptAtoB(int) → int / ⊥

\[
\text{SwapAtoB}(\text{dtx}_A, \text{ctx}_B, \text{minOut}): \\
\quad \text{out} \leftarrow \text{cptAtoB}(\text{dtx}_A.val); \text{require} (\text{out} \geq \text{minOut}) \\
\quad \text{TokenA.FTransfer}(\text{dtx}_A) \\
\quad \text{TokenB.FTransfer}(\text{ctx}_B[\text{out}])
\]

- User spends A in \text{dtx}_A
- User obtains B with \text{ctx}_B
- Txs sent to a third contract!

Sidenote on AMM:

cptAtoB – many implementations
- Constant product (Uniswap)
- Constant sum (Curve)
- Other variants \cite{AngEvaChi21}
Honest user constructs “AMM.SwapAtoB(dtx\textsubscript{A}, ctx\textsubscript{B}, minOut)”

Adversary intercepts it, submits the call “AMM.SwapAtoB(dtx\textsubscript{A}, ctx\textsubscript{B}', minOut)”, where ctx\textsubscript{B}' benefit the attacker instead.

**Honest inter-contract call stack (ICCS)**

AMM.SwapAtoB(dtx\textsubscript{A}, ctx\textsubscript{B}, minOut)
Token\textsubscript{A}.FTransfer(dtx\textsubscript{A})

**Malicious ICCS**

AMM.SwapAtoB(dtx\textsubscript{A}, ctx\textsubscript{B}', minOut)
Token\textsubscript{A}.FTransfer(dtx\textsubscript{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

1. Construct all credit transactions, ctx$_1$, ..., ctx$_m$.

2. Construct tx$_1$, ..., tx$_n$, tx$_i$.intent := “Contract.Func(*, ctx$_1$, ..., ctx$_m$, arg)”.
"Swap 10 A to at least 20 B"

```
Contract AMM
SwapAtoB(dtx_A, ctx_B, minOut):
out ← cptAtoB(dtx_A.val)
Token_A.FTransfer(dtx_A)
Token_B.FTransfer(ctx_B[out])
```

```
Contract Token_A
FTransfer(dtx_A):
VerifyIntent(dtx_A.intent)

Contract Token_B
ctx_B
```

```
dtx_A
• val: -10

AMM.SwapAtoB(*, ctx_B, 20)
```

Blockchain
Fix a distinguished token, Token_{Gas}.

A blockchain transaction is a single debit transaction, paying up to dtx.val for gas.

dtx\_call
• val // gas paid limit
• gasprice
• intent

Consequence: “tx.origin” no longer available, as well as “caller” during the initial call.
Extended Example

“Swap 10 A to at least 20 B”

Contract Token\textsubscript{Gas}

\texttt{dtx\textsubscript{call}}
• val // max fees
• gasprice

Contract Token\textsubscript{A}
\texttt{FTransfer(dtx\textsubscript{A})}:
\texttt{VerifyIntent(dtx\textsubscript{A}.intent)}

Contract Token\textsubscript{B}
\texttt{ctx\textsubscript{B}[out]}

Blockchain

AMM.SwapAtoB(*, ctx\textsubscript{B}, 20)

\texttt{dtx\textsubscript{A}}
• val: -10

SwapAtoB(dtx\textsubscript{A}, ctx\textsubscript{B}, minOut):
out ← cptAtoB(dtx\textsubscript{A}.val)
\texttt{FTransfer(dtx\textsubscript{A})}
\texttt{FTransfer(ctx\textsubscript{B}[out])}
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 \( tx = \{ \text{val: -10, token: A, intent: ...} \} \)
- 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

Privacy-preserving DeFi
- Asset pools
- Trading / DEXes
- Lending

FLAX Token Standard
“Privacy-preserving ERC20”
Token-denominated Funds (aka pools)

**Contract Pool extends TokenStandard**

cptEnter(val\_A, val\_B) → (in\_A, in\_B, out\_p)
cptExit(val\_p) → (out\_A, out\_B)

EnterPool(dtx\_A, dtx\_B, ctx\_P):
  (in\_A, in\_B, out\_C) ← cptEnter(dtx\_A.val, dtx\_B.val)
  Token\_A.FTransfer(dtx\_A[in\_A])
  Token\_B.FTransfer(dtx\_B[in\_B])
  bal[this] ← bal[this] + out\_p
  this.FTransfer(ctx\_P[out\_p], this)

ExitPool(dtx\_p, ctx\_A, ctx\_B):
  (out\_A, out\_B) ← cptExit(dtx\_p.val)
  Token\_A.FTransfer(ctx\_A[out\_A])
  Token\_B.FTransfer(ctx\_B[out\_B])
  this.FTransfer(dtx\_p)
  bal[this] ← bal[this] – dtx\_p.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, \((\text{val}_{\text{collateral}}, \text{ctx}_{\text{refund}})\)
• Records outstanding debt, \(\text{val}_{\text{debt}}\)

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

Data from defipulse.com
Existing DAPs
- Zerocash [BCGGMTV14]
- RingCT [Noether15, SALY17, YSLAEZG20]
- Quisquis [FMMO19]
- Zether [BAZB20, Diamond21]

Flexible Anonymous Transactions (FLAX) System
Cryptographic building block

Privacy-preserving DeFi
- Asset pools
- Trading / DEXes
- Lending

FLAX Token Standard
“Privacy-preserving ERC20”
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 \( R = \{ (st, tx.body ; sk, ...) | \text{“prover knows sk such that ..”} \} \)
   • 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 \textit{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)**

**Tx (spend some coins, create some coins)**
- Spending info: serial numbers, key images
- New coins: NewCoin\(_1\), ..., NewCoin\(_m\)
- Proof

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

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

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

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

<table>
<thead>
<tr>
<th>Account</th>
<th>State (st)</th>
<th>Transfer ttx</th>
<th>Debit dtx[-c]</th>
<th>Credit ctx[c]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>((g^r_A, pk^r_A g^{b_A}))</td>
<td>((g^r, pk^r_A g^{-c}))</td>
<td>((g^r, pk^r_A g^{-c}))</td>
<td>((g^r, pk^r_A g^c))</td>
</tr>
<tr>
<td>Bob</td>
<td>((g^r_B, pk^r_B g^{b_B}))</td>
<td>((g^r, pk^r_B g^c))</td>
<td>((g^r, pk^r_B g^0))</td>
<td>((g^r, pk^r_B g^0))</td>
</tr>
<tr>
<td>Charlie</td>
<td>((g^r_C, pk^r_C g^{b_C}))</td>
<td>((g^r, pk^r_C g^0)) + Proof (\pi)</td>
<td>((g^r, pk^r_C g^0)) + Proof (\pi)</td>
<td>((g^r, pk^r_C g^0)) + Proof (\pi)</td>
</tr>
</tbody>
</table>
Comparisons of Instantiations

<table>
<thead>
<tr>
<th>FLAX from</th>
<th>Ledger</th>
<th>User</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Setup</td>
<td>Read</td>
<td>Privacy</td>
</tr>
<tr>
<td>ZeroCash</td>
<td>Trusted</td>
<td>$O(m)$</td>
<td>Full</td>
</tr>
<tr>
<td>RingCT</td>
<td>Pub. coin $O(m)$</td>
<td>$O(</td>
<td>st</td>
</tr>
<tr>
<td>Quisquis</td>
<td>Pub. coin $O(u)$</td>
<td>$O(</td>
<td>st</td>
</tr>
<tr>
<td>Zether</td>
<td>Pub. coin $O(n)$</td>
<td>$O(1)$</td>
<td>Acc ring</td>
</tr>
</tbody>
</table>

- $m$ – history size
- $u$ – UTXO size
- $n$ – accounts
- $n \ll u \ll m$

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