PriDe CT: Towards Public Consensus, Private Transactions, and Forward Secrecy in Decentralized Payments

Harish Karthikeyan

Joint work with Yue Guo, Antigoni Polychroniadou, Chaddy Huussin

Accepted to IEEE Security & Privacy (S&P), 2024

AlgoCRYPT CoE

AI Research

J.P.Morgan

DISCLAIMER


The views expressed in my post represent my own opinions and not those of my employer
HumanRisk

OUR FOCUS

- Private: identity of parties are hidden
- Decentralized: smart contract-based payment mechanism deployable on blockchain
- Account-based Model
- Confidential: payload of transactions remain hidden
- Batchable: multiple receivers can receive in one posted tx message
- Concurrent: competing transactions can succeed

OUR APPROACH

Transferring Payload

- The sender chooses a set of receivers.
- ANY number of them can be decoy.
- Payload is encrypted under that user's public key.
- Balance updated by "homomorphically" adding balance ciphertext with payload ciphertext

Preventing Malicious Behavior

- Every information is encrypted.
- Use zero-knowledge proofs to prove the honest behavior of parties
- ZKPs allow proving information about a secret info, without revealing secret info.

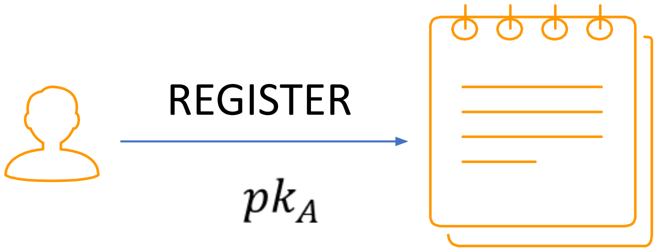
Encrypted Balances

- Each user registers with a public-key secret key pair.
- Each user's balance is stored in an encrypted format
- Balance and public key is a part of public state.

OUR SMART CONTRACT

Functionalities:

- Registering an Account

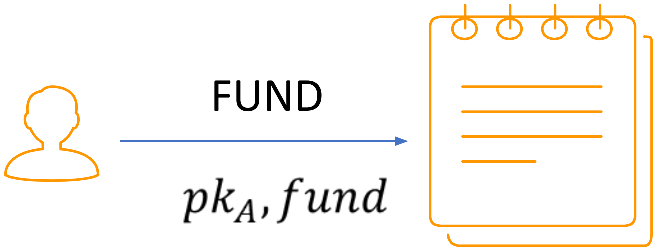


Update State:
Encryption of 0 under pk_A with randomness 0

OUR SMART CONTRACT

Functionalities:

- Registering an Account
- Funding an Account



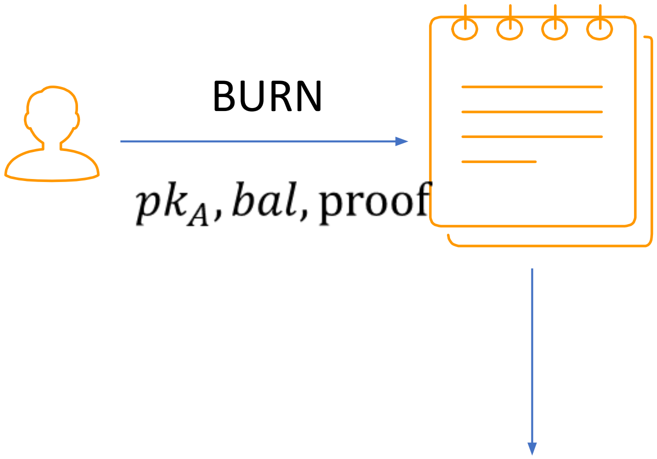
Update State:
Encryption of $fund$ under pk_A with randomness 0

Uses ERC20's "transferFrom" and "balanceOf" functions

OUR SMART CONTRACT

Functionalities:

- Registering an Account
- Funding an Account
- Burning an Account



Update State:

- Verify proof is correct (user knows secret key of account and bal is the encrypted balance)
- Update balance to 0.

Uses ERC20's "transfer" function

OUR SMART CONTRACT

Functionalities:

- Registering an Account
- Funding an Account
- Burning an Account
- Transferring to Accounts

Transfer

- pk_A
- Public keys of receivers
- Encrypted payloads under public keys of receivers
- Proof

Update State:

- Verify proof is correct
  - ❑ Sender knows sk_A consistent with pk_A
  - ❑ Each payload encryptions is correct, and each payload is ≥ 0
  - ❑ Money debited = money credited
  - ❑ No overdraft, no double spending
- Update Balances if proof succeeds.

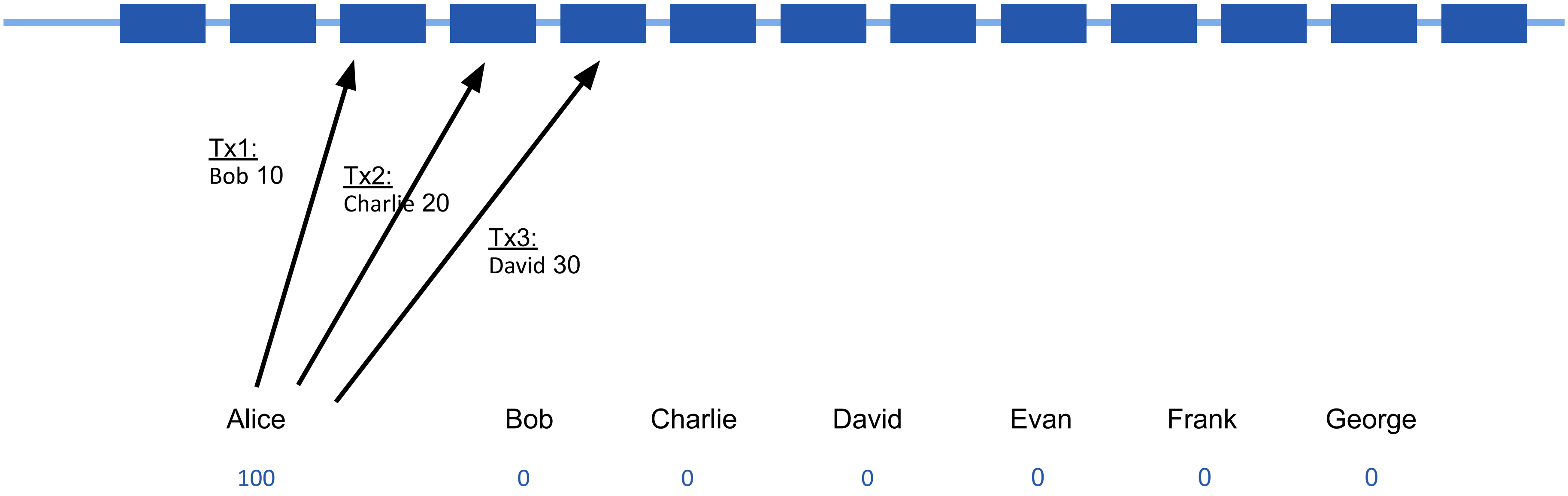
Prior Work

- Zether
  - Use ElGamal Encryption to encrypt Balances
  - For anonymity: Choose a set of N users
    - N-2 are decoys
    - 1 sender, 1 receiver
    - Sender identity is encoded as a bit string of length N such that sen[i]==1 iff i is sender
    - Similarly receiver
    - Informally suggested Two 1-out-of-N proofs
- Anonymous Zether
  - Pointed out issues with Zether's proposal
  - Introduced many-out-of-many proofs
    - Can now use one vector of length N, with 2 non-zero entries

OUR CRYPTOGRAPHIC TOOLS

- ElGamal Encryption:
  - Enc(pk_A,b)=(g^r, pk_A^r * g^b)
  - Property: Additive Homomorphism
    - Enc(pk_A,b_1)*Enc(pk_A,b_2)=Enc(pk_A,b_1+b_2)
- Commitment:
  - Any user who says C commits to b, cannot later prove that C commits to b’. And C does not leak b.
- Zero-Knowledge Proofs:
  - Prove that secret b satisfies some constraint, without leaking b.
  - Range Proofs:
    - Specifically prove that b lies in some range [0,B] - We use bulletproofs

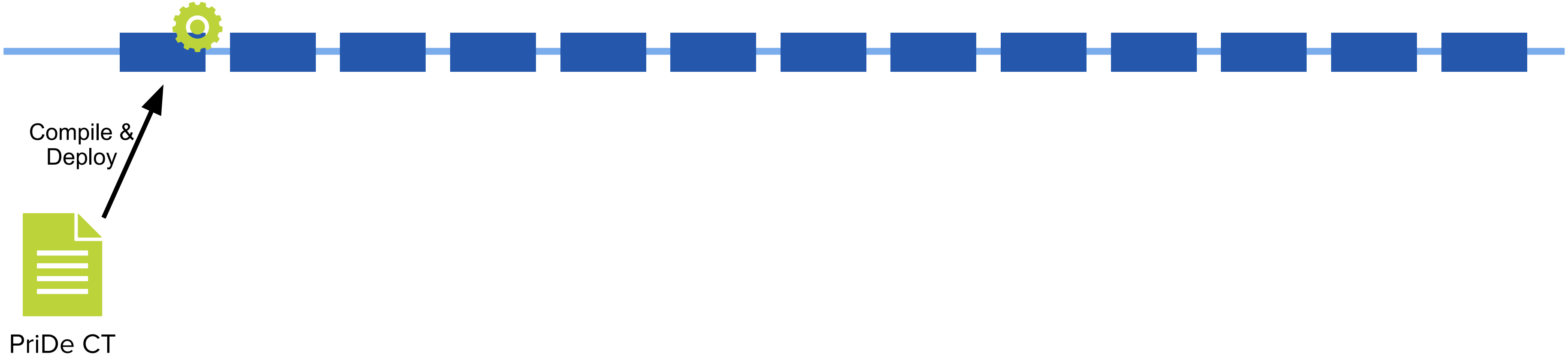
Transaction without Privacy



Alice	Bob	Charlie	David	Evan	Frank	George
100	0	0	0	0	0	0

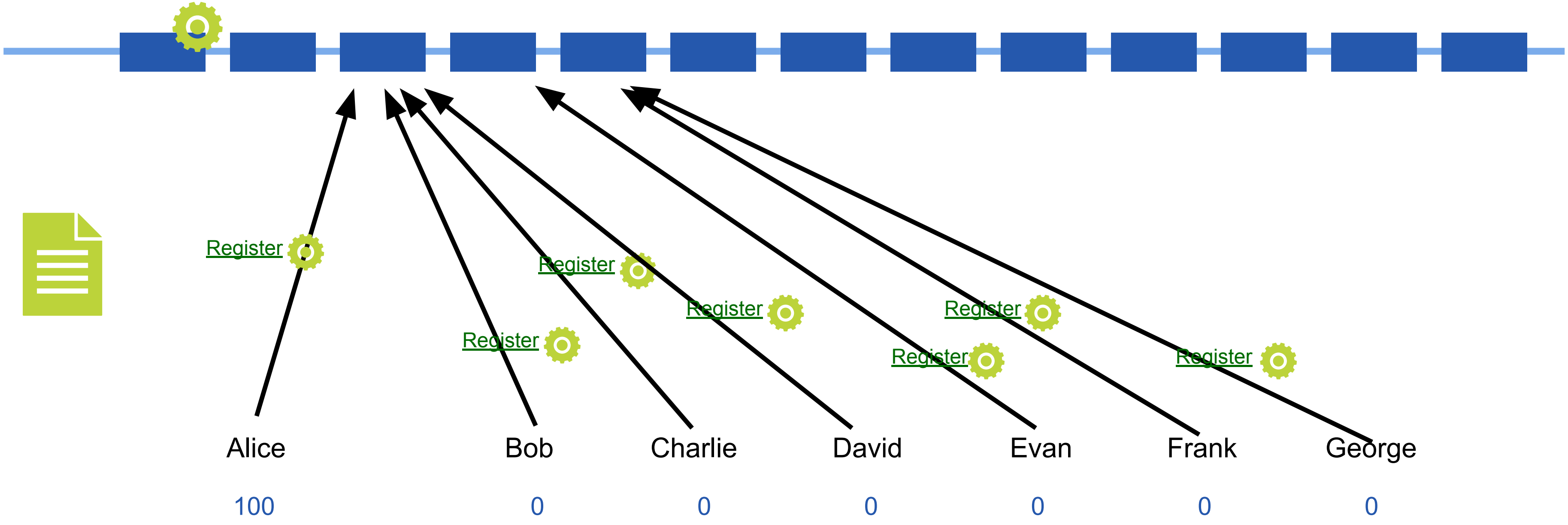
Alice	Bob	Charlie	David	Evan	Frank	George
40	**10**	**20**	**30**	0	0	0

Transaction with PriDe CT

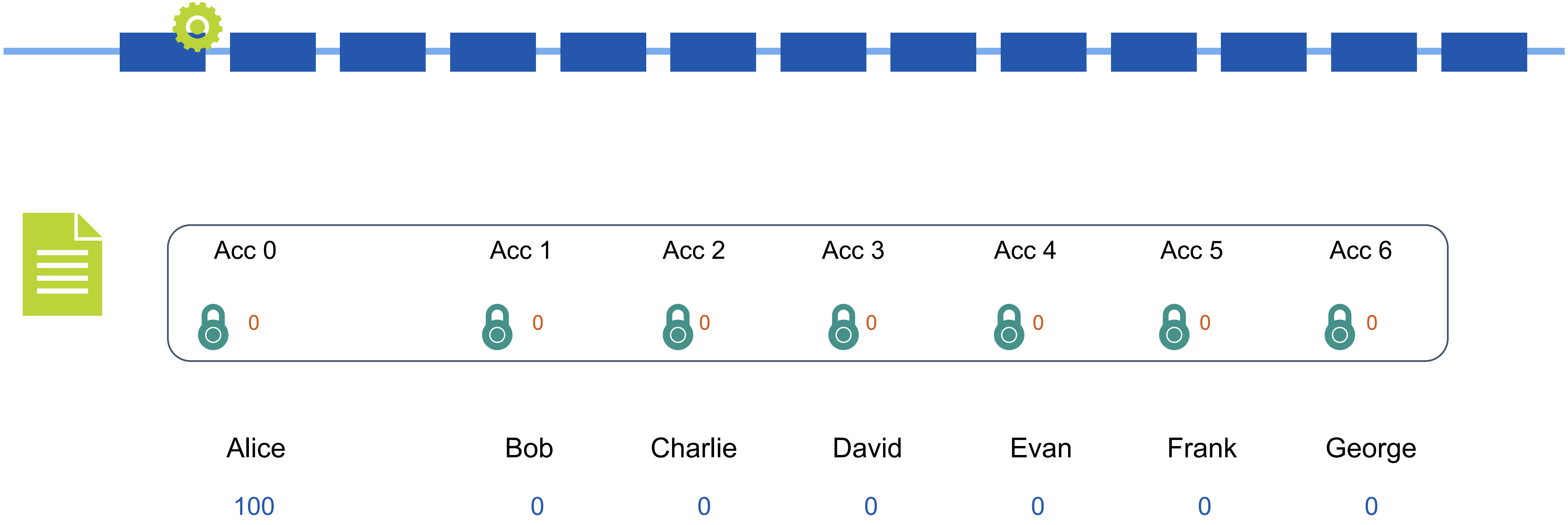


Alice	Bob	Charlie	David	Evan	Frank	George
100	0	0	0	0	0	0

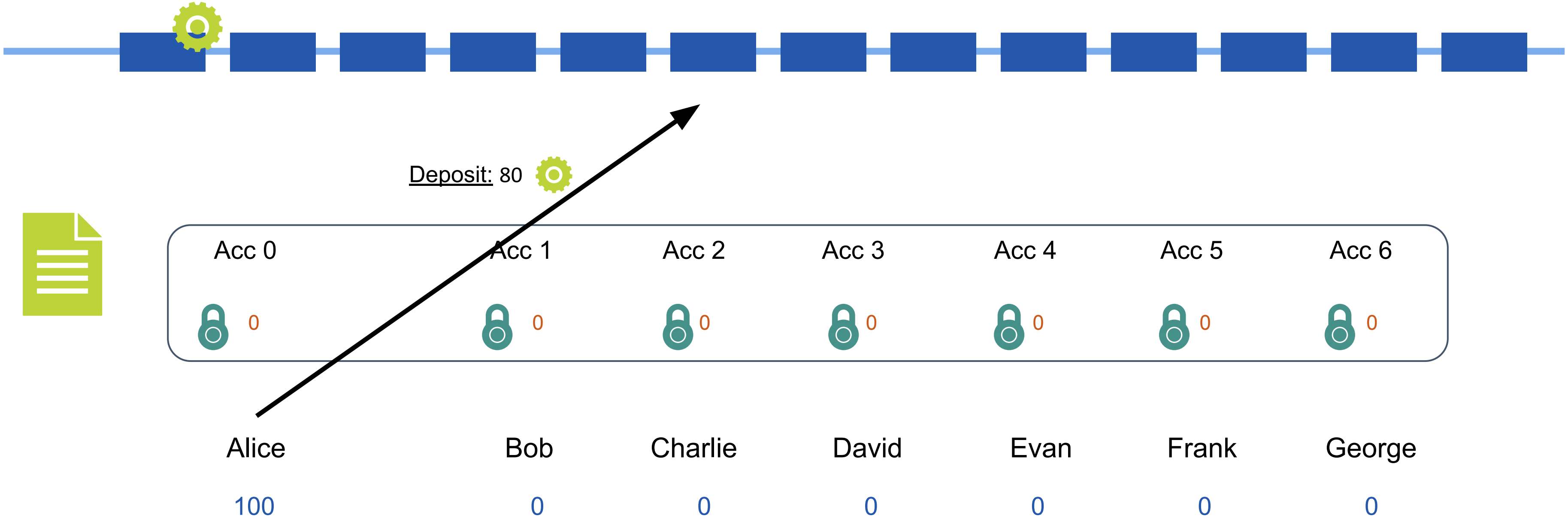
Transaction with PriDe CT



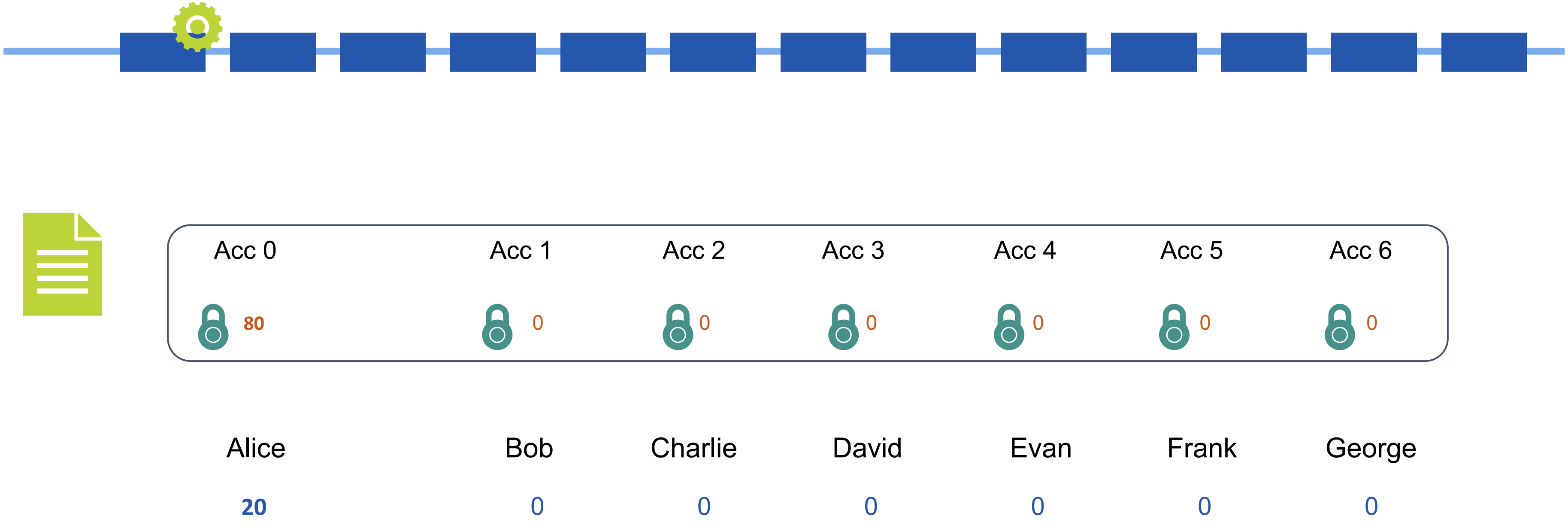
Transaction with PriDe CT



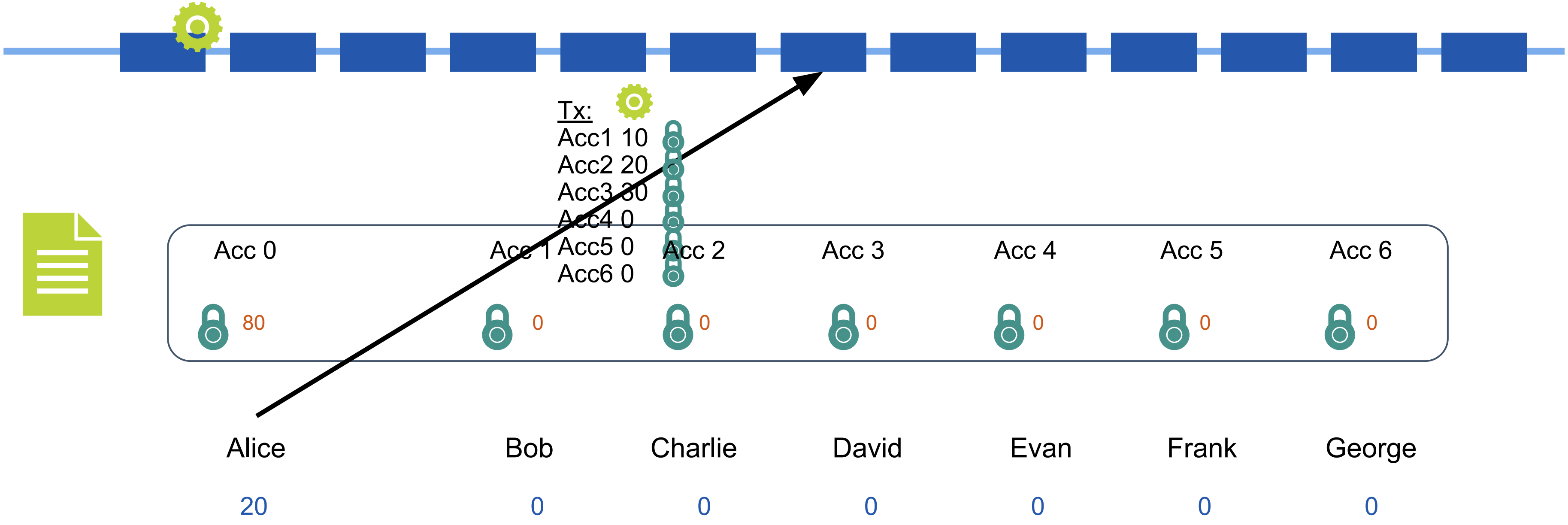
Transaction with PriDe CT



Transaction with PriDe CT



Transaction with PriDe CT

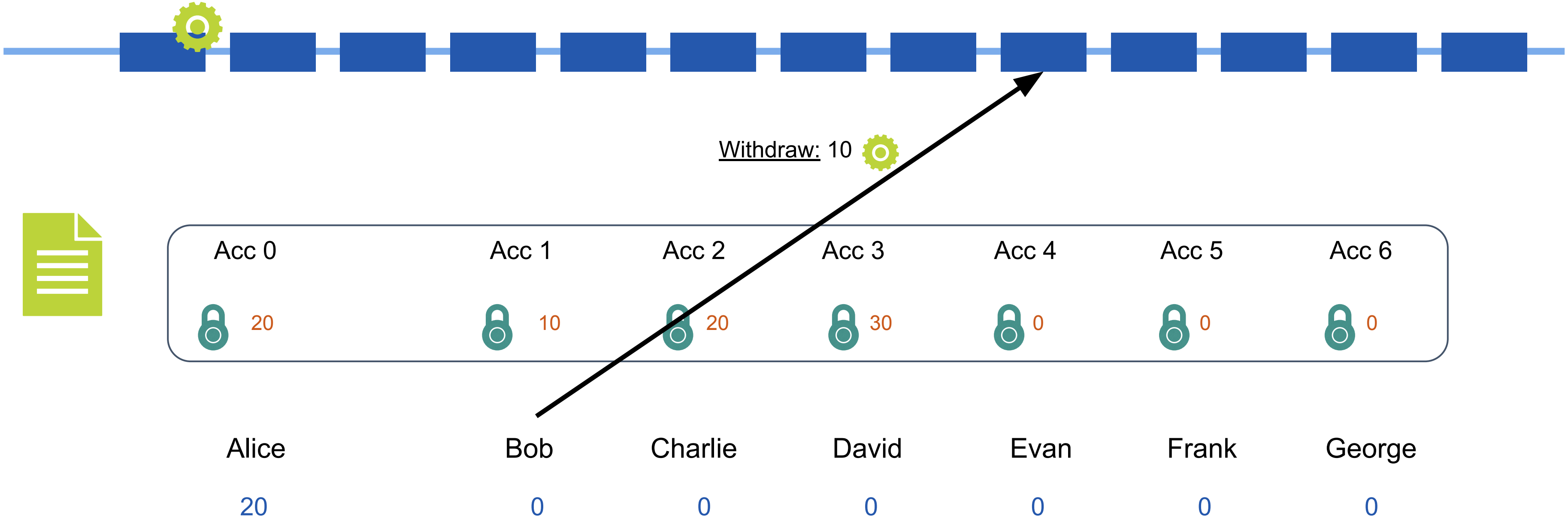


Transaction with PriDe CT

Acc 0	Acc 1	Acc 2	Acc 3	Acc 4	Acc 5	Acc 6
20	10	20	30	0	0	0

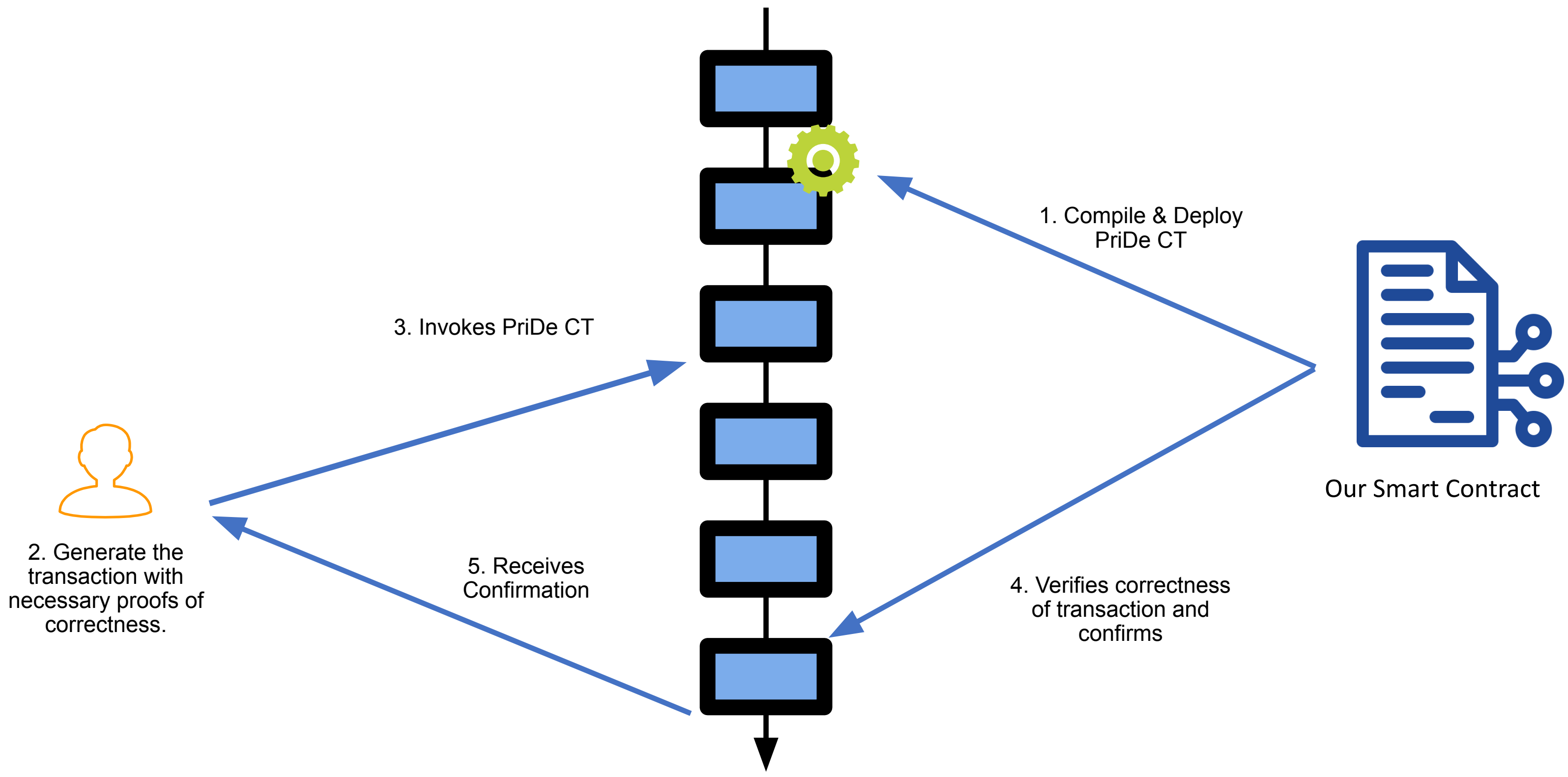
Alice	Bob	Charlie	David	Evan	Frank	George
20	0	0	0	0	0	0

Transaction with PriDe CT



Transaction with PriDe CT

Acc 0	Acc 1	Acc 2	Acc 3	Acc 4	Acc 5	Acc 6
20	**0**	20	30	0	0	0
Alice	Bob	Charlie	David	Evan	Frank	George
20	**10**	0	0	0	0	0


1. Compile & Deploy
PriDe CT
Our Smart Contract
3. Invokes PriDe CT
2. Generate the
transaction with
necessary proofs of
correctness.
5. Receives
Confirmation
4. Verifies correctness
of transaction and
confirms

OUR PERFORMANCE

- We compare performance with Anonymous Zether. Note that Anonymous Zether also uses the idea of a set of receivers, but only one receiver can actually receive a non-zero payload while others must be zero.

	Proving Time Ratio	Gas Consumption Ratio
4 Receivers	1.86	1.10
8 Receivers	3.27	1.87
16 Receivers	4.78	2.67
32 Receivers	6.88	3.27
64 Receivers	8.21	3.50

Takeaway: Batching of transactions makes our work more optimal!

ONGOING WORK

- Investigate Flashproofs (ASIACRYPT 2022) as replacement for range proof
  - Flashproofs for ZKPs are shown to be 8 times more efficient in gas consumption. Projected numbers are shown below:
    - Unfortunately, this does come at a price - aggregating does not reduce proof size.

TABLE 5. PERFORMANCE COMPRISON BETWEEN PRIVATE DECENTRALIZED CONFIDENTIAL TRANSACTIONS, ANONYMOUS ZETHER, AND PRIVATE DECENTRALIZED CONFIDENTIAL TRANSACTIONSWITH FLASHPROOFS.

	Proving Time			Gas Consumption		
	Anonymous Zether	PriDe CT	PriDe CT with Flashproofs	Anonymous Zether	PriDe CT	PriDe CT with Flashproofs
Transfer(4)	1,897	3,543	709	3,453,438	3,812,298	434,698
Transfer(8)	2,066	6,757	1,351	4,332,444	8,106,123	924,301
Transfer(16)	2,699	12,910	2,582	6,325,889	16,877,598	1,924,470
Transfer(32)	3,672	25,263	5,053	10,919,626	35,758,365	4,077,351
Transfer(64)	3,266	51,445	10,289	22,022,114	77,024,171	8,782,688

FlashProofs (ASIACRYPT 2022)

- Bit Decomposition Approach, with a twist
- Compute y=2^0 b_0+2^1 b_1 +....+2^{N-1} b_{N-1}
- Represent these terms a matrix M of dimensions L x K where N+padding=L x K

$$\begin{pmatrix} 2^0 b_0 & \cdots & 2^{K-1} b_{K-1} \\ 2^K b_K & \cdots & 2^{K+K-1} b_{K+K-1} \\ \vdots & \ddots & \vdots \\ 2^{(L-1)K} b_{(L-1)K} & \cdots & 2^{(L-1)K+K-1} b_{(L-1)K+K-1} \end{pmatrix} = \begin{pmatrix} w_0 & \cdots & w_{K-1} \\ w_K & \cdots & w_{K+K-1} \\ \vdots & \ddots & \vdots \\ w_{(L-1)K} & \cdots & w_{(L-1)K+K-1} \end{pmatrix}$$

FlashProofs (ASIACRYPT 2022)

- $$\begin{pmatrix} 2^0 b_0 & \cdots & 2^{K-1} b_{K-1} \\ 2^K b_K & \cdots & 2^{K+K-1} b_{K+K-1} \\ \vdots & \ddots & \vdots \\ 2^{(L-1)K} b_{(L-1)K} & \cdots & 2^{(L-1)K+K-1} b_{(L-1)K+K-1} \end{pmatrix} = \begin{pmatrix} w_0 & \cdots & w_{K-1} \\ w_K & \cdots & w_{K+K-1} \\ \vdots & \ddots & \vdots \\ w_{(L-1)K} & \cdots & w_{(L-1)K+K-1} \end{pmatrix}$$

- Prove that w_i is either 0 or 2^i
- Then flatten to one dimension (column vector) by adding elements along the row
- Prove that y=sum of new column vector

ONGOING WORK

- Investigate SpringProofs (S&P 2024) as replacement for range proof
  - SpringProofs solves the problem of efficiency without requiring padding.
  - When range is [0,2^N-1), same effort as Bulletproofs
  - Shows efficiency gains for Monero
    - No solidity implementation to incorporate into PriDe CT
  - However, better when range is not of a “nice form”.

Conclusion

- We build a privacy-preserving smart contract to work in account-based model
- It is modular with easy to plug in other range proofs
- We also discuss what it means to be forward secure, in the context of blockchain in our paper
- **https://eprint.iacr.org/2023/1948**

Forward Security and Private Transactions

- Forward Security: Compromise of secret key at time *i*, does not compromise the confidentiality of any prior messages.
- In Private Transactions:
  - Blockchains store all information, in perpetuity.
  - So, compromise of secret key in time period *i*, can mean loss of privacy in earlier epochs.
  - Naive Solution:
    - Regularly user creates a new key pair, moves transactions from old to new account
    - Delete old key pair
    - Problem: User actively participates in forward secrecy
    - Problem: Sender needs to synchronize on which account to send to
    - Problem: When's a good time to delete old key pair?

Forward Security and Private Transactions

- Our Solution: Key Evolution happens using Updatable Public Key Encryption (JMM19, ACDT20, D**K**W21,HLP22,HPS23,AsaWat23, **Kar**Pol24...)
  - Sender chooses how to update the receiver's key.
  - As long as there is one honest update of the receiver key, all exchanges prior to this honest update is secure.
  - We show regular ElGamal is a secure UPKE
    - Even under stronger security definitions