$$\frac{1}{X+a}$$

$$\sum \frac{1}{X+a_i}$$

$$\frac{1}{X+b}$$

Multi-User Cryptography Using Rational Functions

$$\frac{1}{X+a}\cdot\frac{1}{X+b}=\frac{1}{b-a}\left(\frac{1}{X+a}-\frac{1}{X+b}\right)$$

$$\mathscr{C}=\begin{bmatrix} \frac{1}{a_1-b_1} & \frac{1}{a_1-b_2} & \cdots & \frac{1}{a_1-b_n} \\ \frac{1}{a_2-b_1} & \frac{1}{a_2-b_2} & \cdots & \frac{1}{a_2-b_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{a_m-b_1} & \frac{1}{a_m-b_2} & \cdots & \frac{1}{a_m-b_n} \end{bmatrix}$$

Rohit Nema (Stanford) | Computer Forum Annual Affiliates Meeting | Apr. 20, 2026

based on joint works with

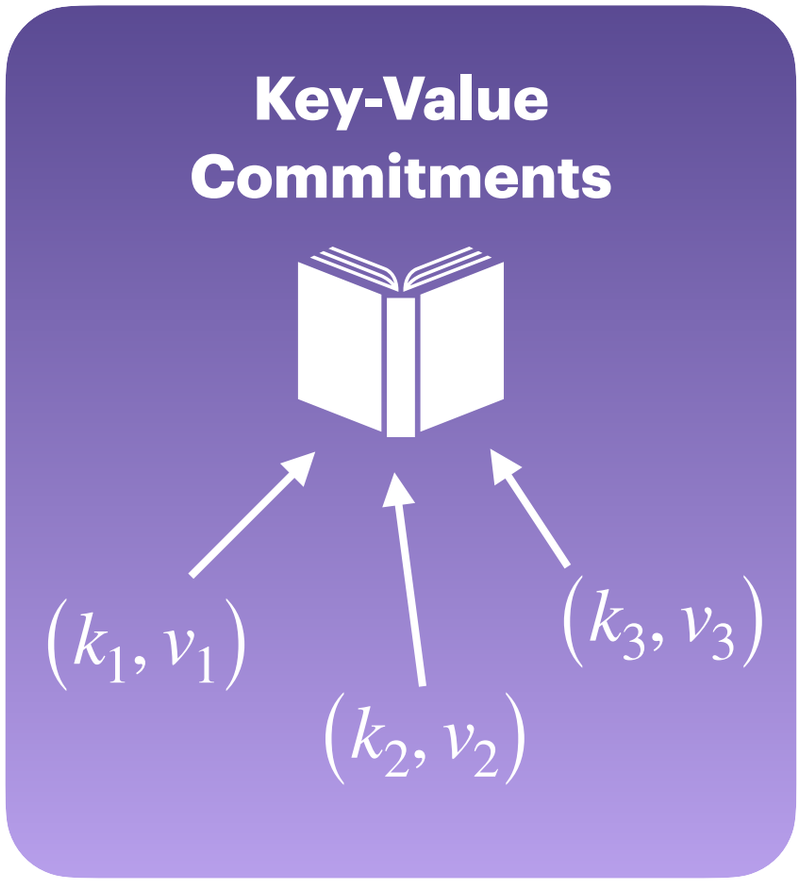
Dan Boneh (Stanford), Charanjit Jutla (IBM), Arnab Roy (Mysten Labs), and Nusret Ertem Tas (a16z crypto)

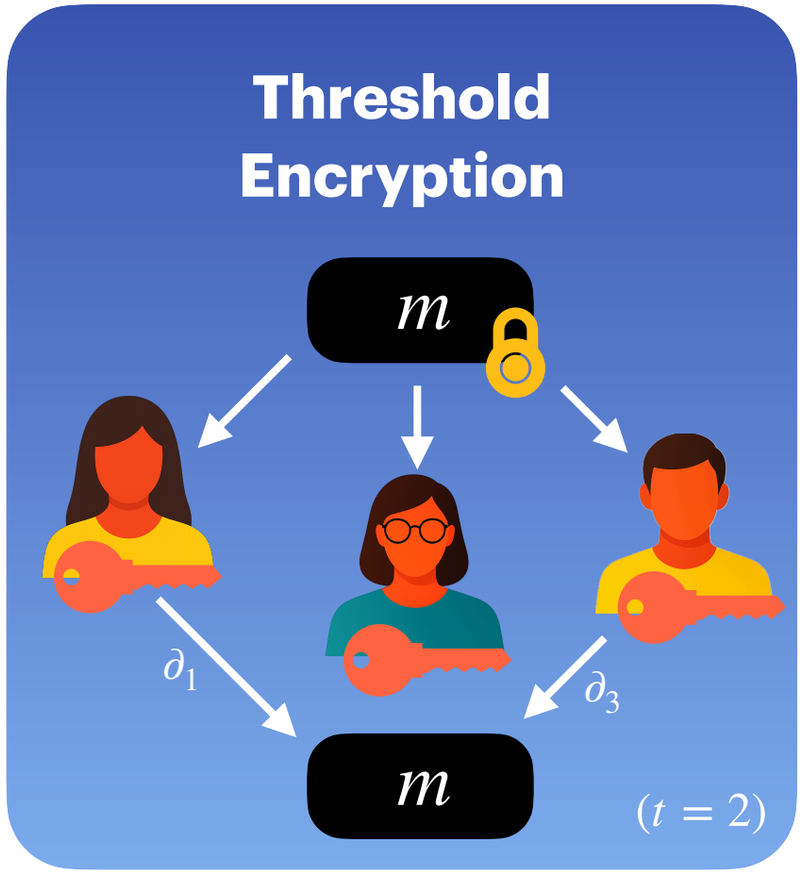
Multi-User Cryptography

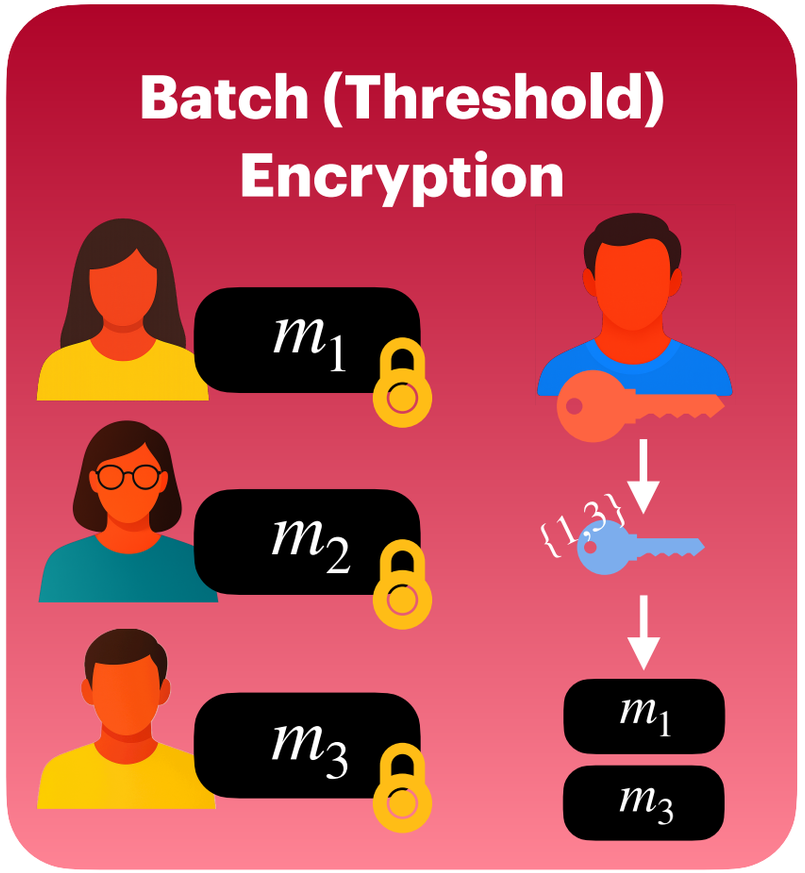
What do we mean?

multiple users in the system with data (e.g. messages) + cryptographic security guarantee

recently worked on

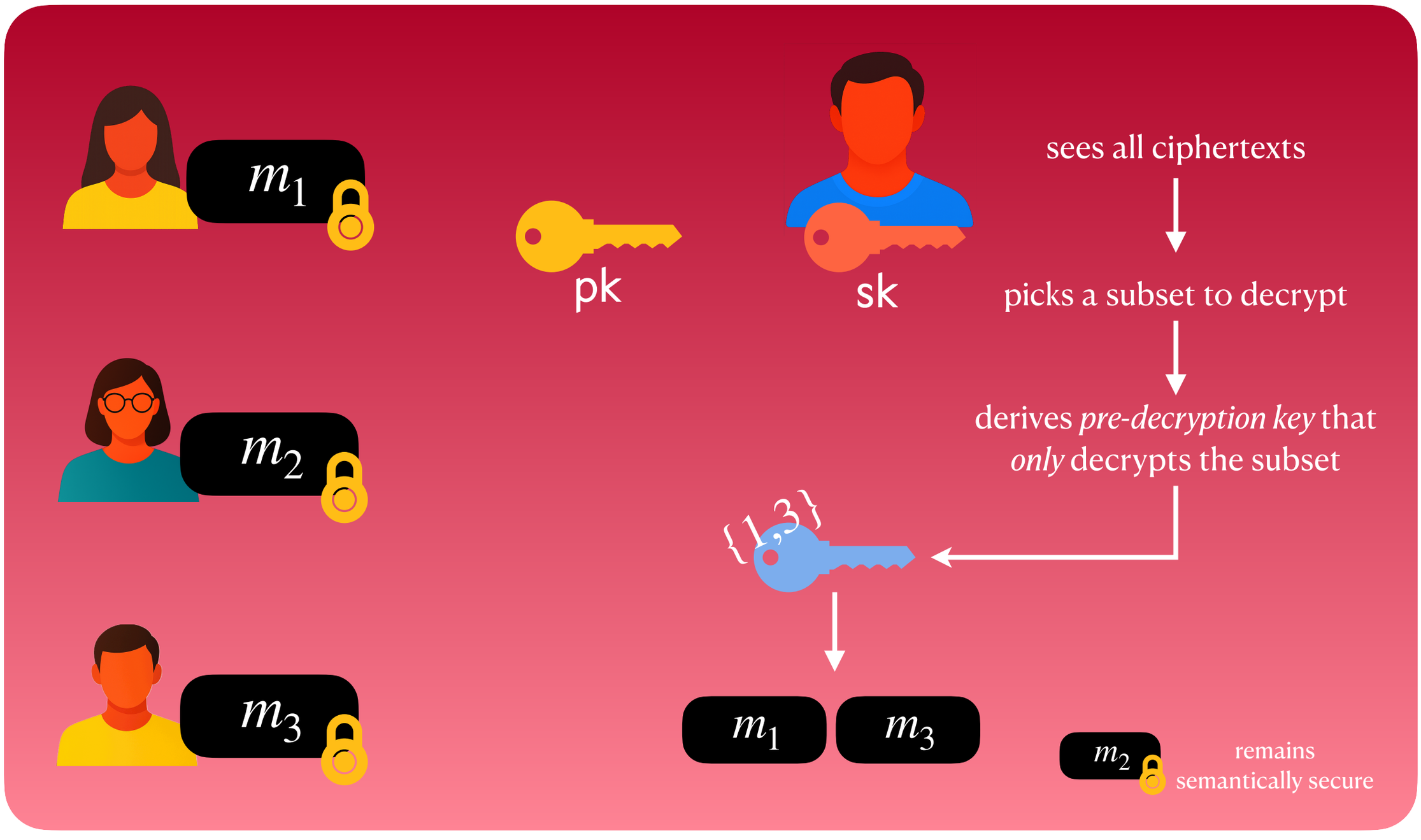






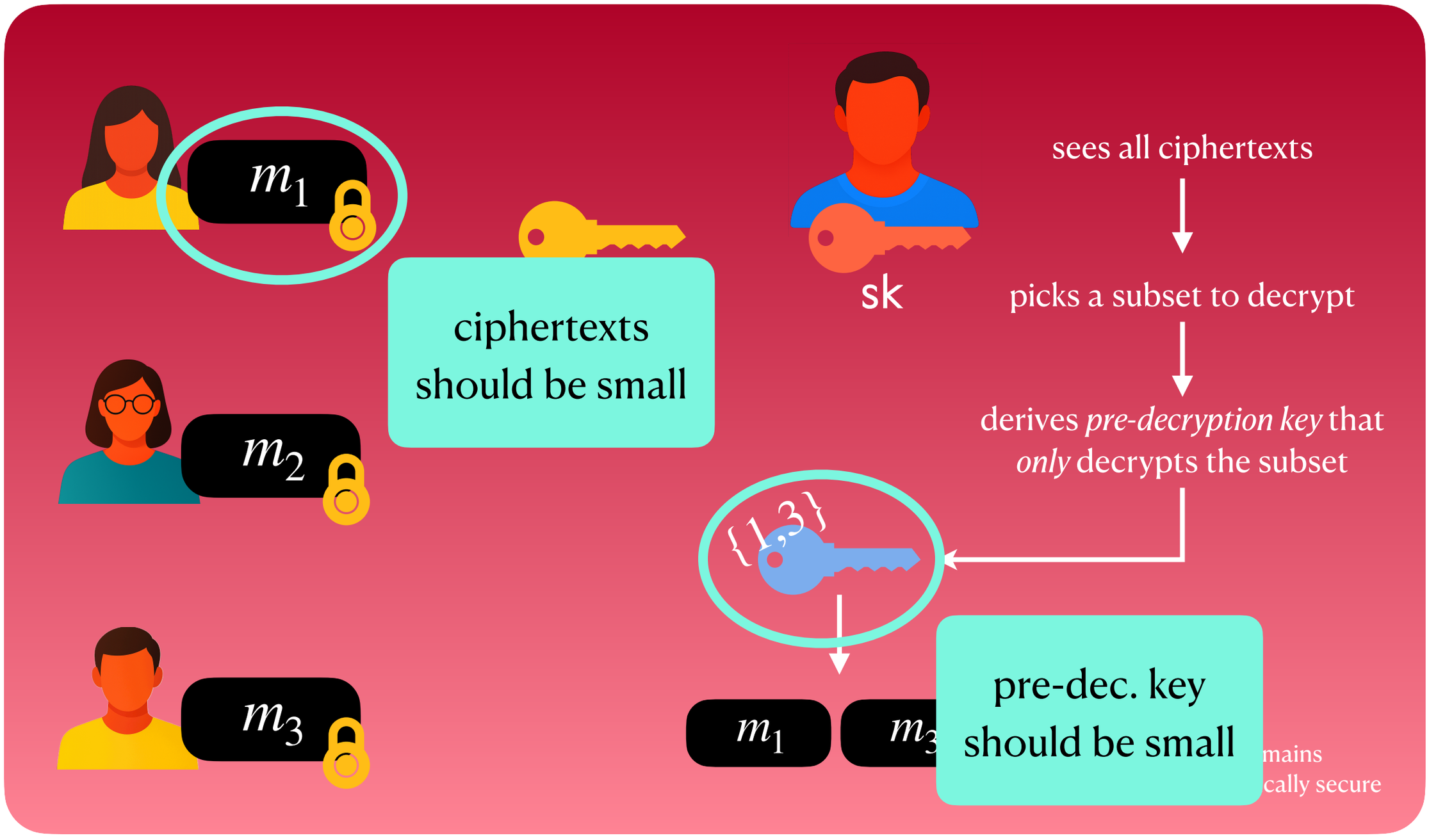
Today: Batch (Threshold) Encryption [CGPP24]

Introduction



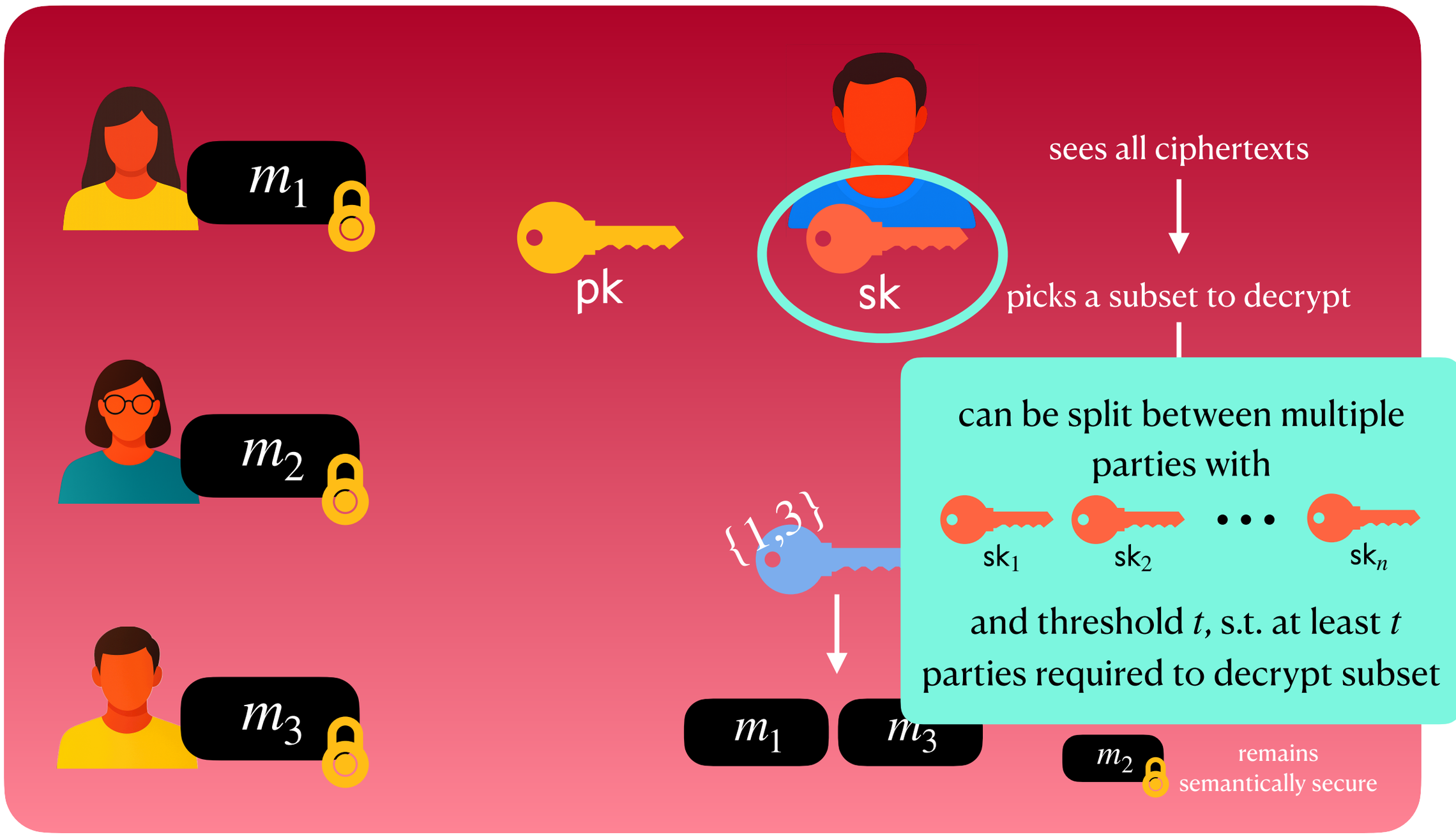
Today: Batch (Threshold) Encryption [CGPP24]

Efficiency



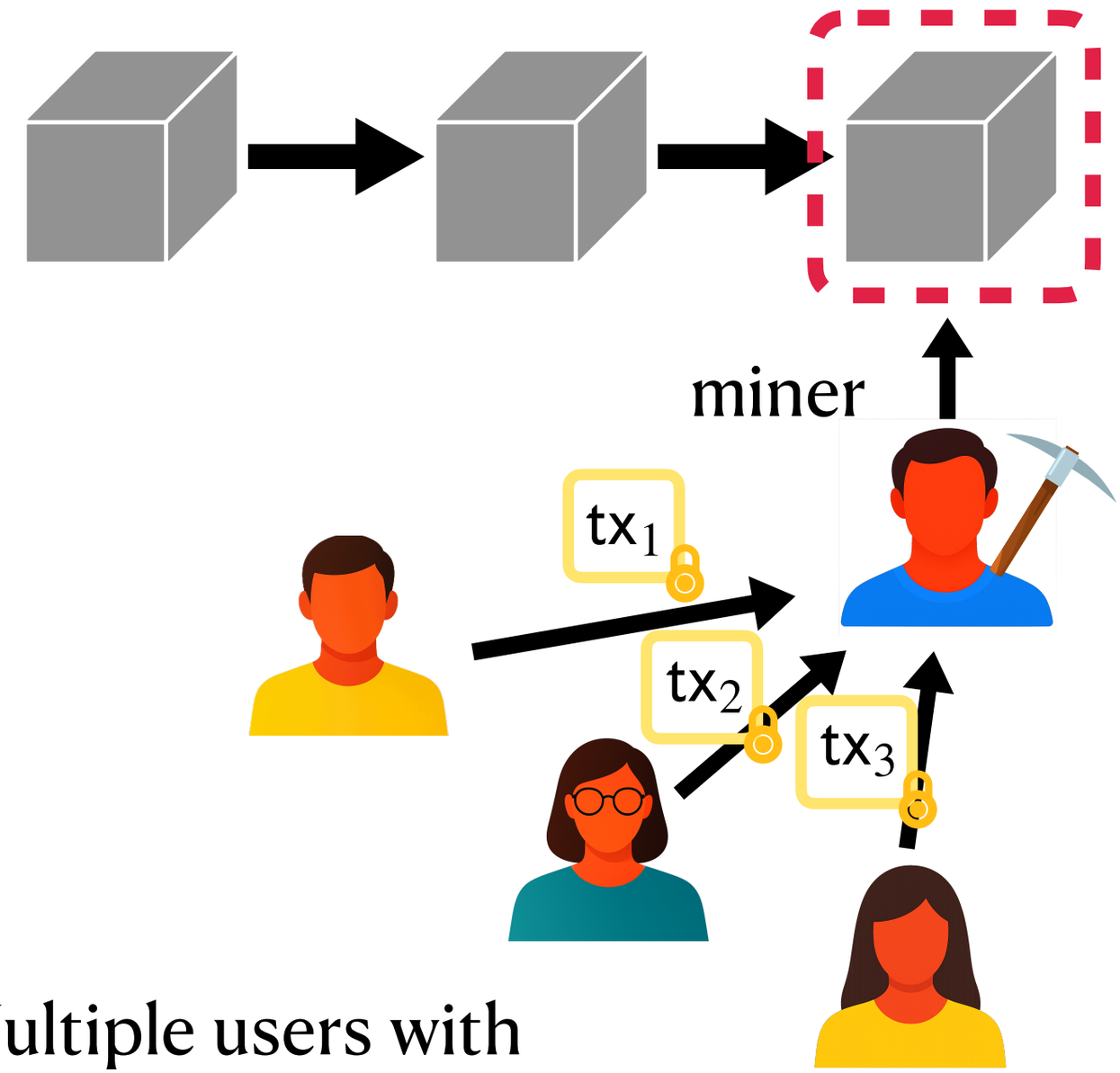
Today: Batch (Threshold) Encryption [CGPP24]

"Thresholdizing" the Secret Key



Today: Batch (Threshold) Encryption [CGPP24]

Application: Encrypted "*Mempools*" for Blockchains



Multiple users with transactions submitted to "memory pool" or *mempool*

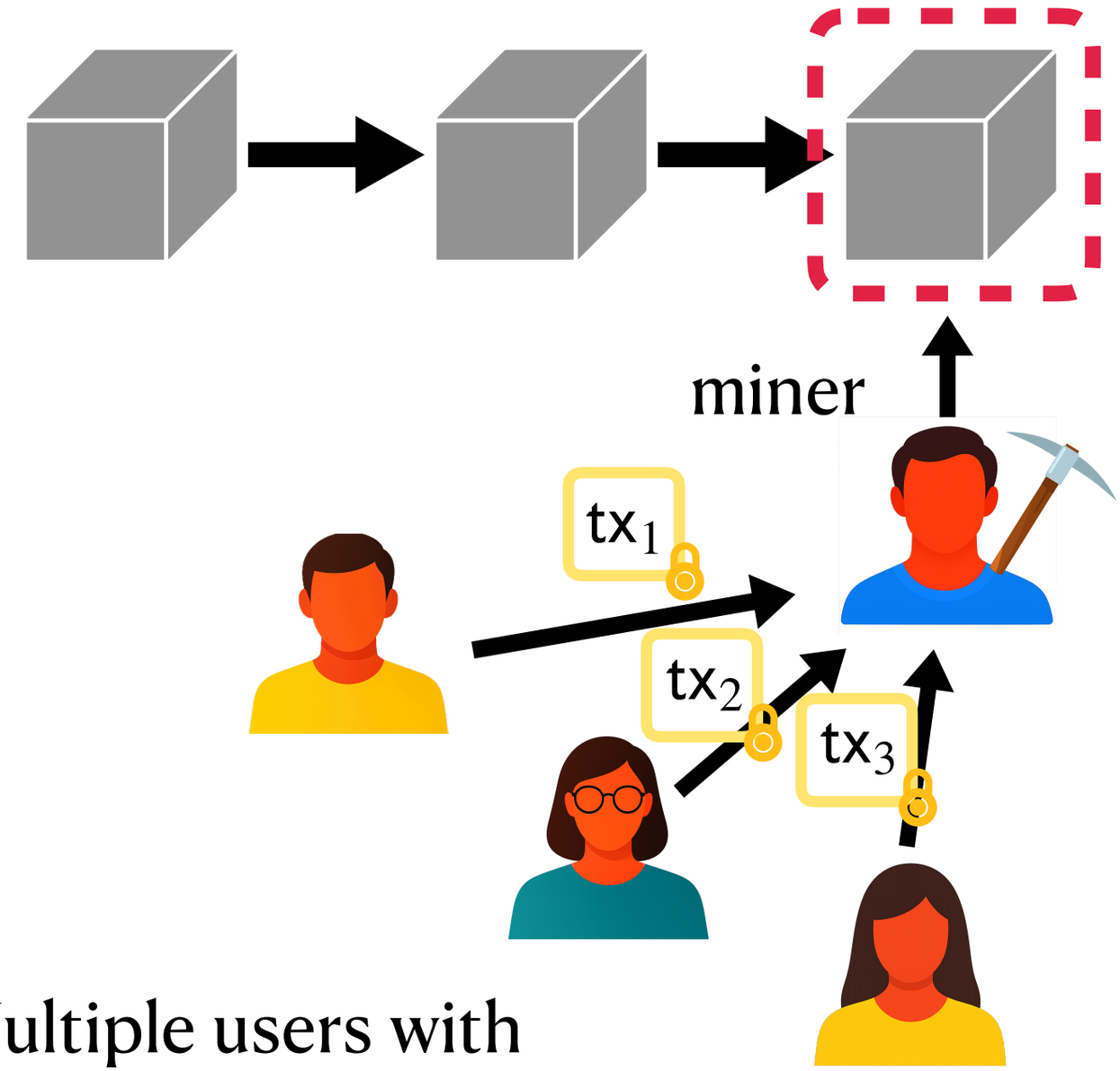
Transactions executed sequentially. Order matters!

Block proposer can order in a way that's beneficial to them (MEV attacks e.g. front-running)

Can we hide them?
Yes, we can encrypt...

Today: Batch (Threshold) Encryption [CGPP24]

Application: Encrypted *Mempools* for Blockchains



Multiple users with transactions submitted to "memory pool" or *mempool*

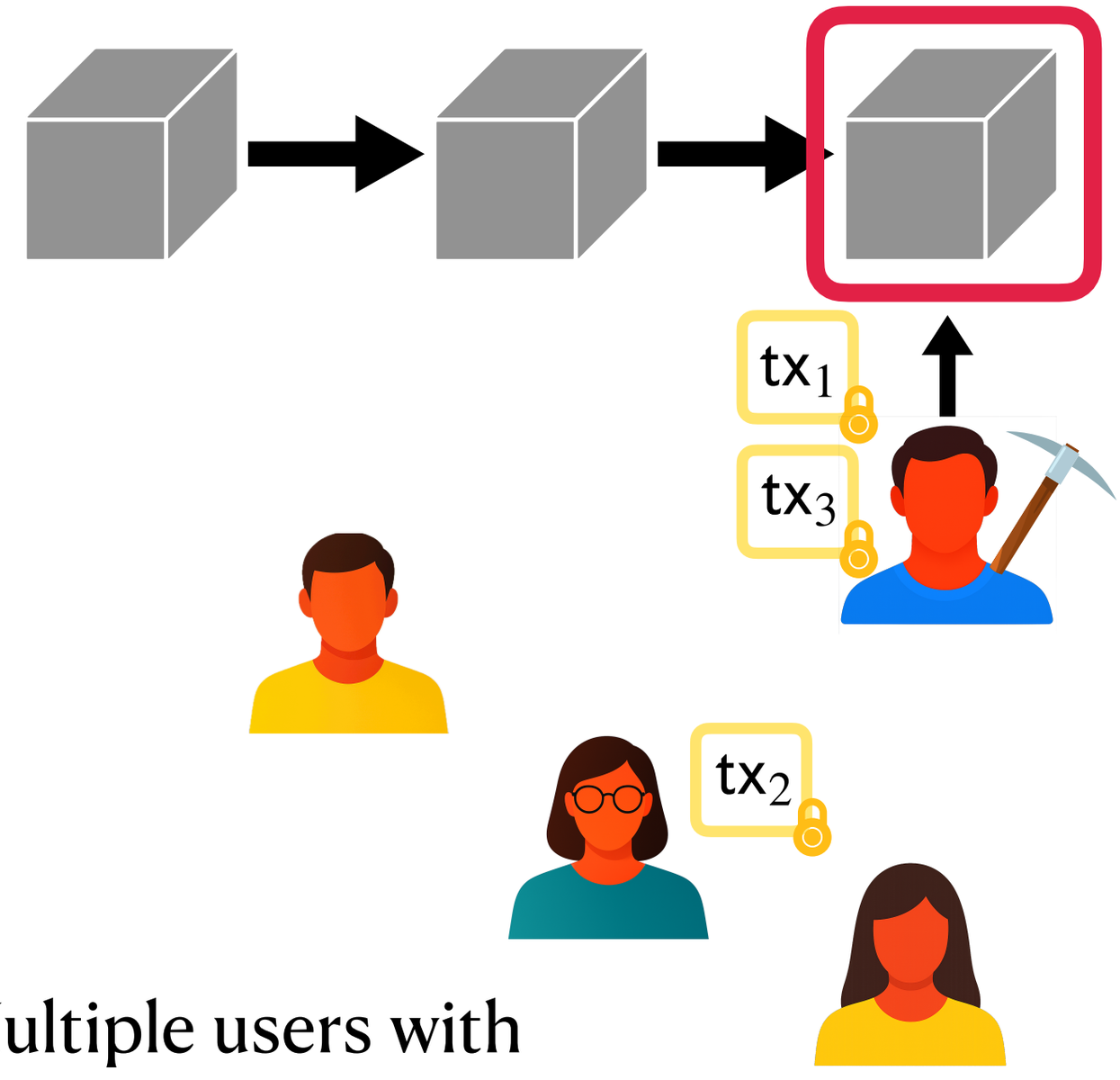
Transactions executed sequentially. Order matters!

Block proposer can order in a way that's beneficial to them (MEV attacks e.g. front-running)

Can we hide them?
Yes, we can encrypt...

Today: Batch (Threshold) Encryption [CGPP24]

Application: Encrypted *Mempools* for Blockchains



Multiple users with transactions submitted to "memory pool" or *mempool*

Block is confirmed. How to decrypt?

What about transactions that were not included?

Batch Encryption enables decryption of *only* tx_1 and tx_3

while keeping tx_2 secret (semantically secure)

Our Results

Almost Optimal | The Best of All Worlds

✓ Shortest ciphertext

Only three group elements! ~700 bytes

✓ Short keys

Encryption and pre-decryption keys are short

✓ No coordination

Encryptors don't need to coordinate

✓ Algebraically simple

Simple algebraic construction with easy proof of security

✓ *Epoch-less*

Ciphertexts are not tied to any epoch (block)

✓ Censorship-resistant

Adversary cannot force exclusion from batch

− Secret key is linear

linear in the size of the batch

Our Techniques

Rational Functions as *first-class* primitives

Rational functions with linear denominator

$$\frac{1}{X+c},\ c \in \mathbb{F}_p$$

finite field of prime order p

We will see how to encode keys and ciphertexts using these...

NOTE

Not the first to use rational functions (e.g. [BB04b,DY05,JL09,Wee16, Hab22,GV22])

BUT...

First to systematically study their properties to build crypto

Our Techniques

Partial Fraction Decomposition

A trick from high school mathematics!

Write product of rational functions as a sum

$$\frac{1}{X+3} \cdot \frac{1}{X+5} = \frac{1}{2}\left(\frac{1}{X+3} - \frac{1}{X+5}\right)$$

coefficients of sum can be easily calculated

well-known

Theorem. Suppose $n \geq 2$. Let $a_1, \ldots, a_n \in \mathbb{F}$ be distinct. Then, there exists unique coefficients $c_1, \ldots, c_n \in \mathbb{F}$ such that,

$$\prod_{i \in [n]} \frac{1}{X + a_i} = \sum_{i \in [n]} \frac{c_i}{X + a_i}$$

where $c_i = \prod_{j \neq i} \frac{1}{a_j - a_i}$ for each $i \in [n]$.

Partial Fraction Products

Linear Independence [JNR25]

- We observed that certain partial fraction products satisfy an intriguing property. Consider

$$\left(\sum_{i\in[n]} \frac{1}{X+a_i}\right)\left(\frac{1}{X+b}\right) = \sum_{i\in[n]} \frac{z_i}{X+a_i} + \frac{z_{n+1}}{X+b}$$

- Using **Theorem 1**, it is easy to show that,

$$z_i = \frac{1}{b-a_i} \text{ and } z_{n+1} = -\sum_{i\in[n]} \frac{1}{b-a_i}$$

- For $m \leq n$ distinct $b_1, \ldots, b_m \notin S$, the following are $\mathbb{F}$-linearly independent,

$$\left\{\left(\sum_{i\in[n]} \frac{1}{X+a_i} + \sum_{i\in[n]} \frac{1}{b_j-a_i}\right)\left(\frac{1}{X+b_j}\right)\right\}_{j\in[m]} = \left\{\sum_{i\in[n]} \frac{1}{b_j-a_i}\cdot\frac{1}{X+a_i}\right\}_{j\in[m]}$$

Partial Fraction Products

Linear Independence [JNR25]

- For $m \leq n$ distinct $b_1, \ldots, b_m \notin S$, the following are $\mathbb{F}$-linearly independent,

$$\left\{ \left(\sum_{i \in [n]} \frac{1}{X + a_i} + \sum_{i \in [n]} \frac{1}{b_j - a_i} \right) \left(\frac{1}{X + b_j} \right) \right\}_{j \in [m]} = \left\{ \sum_{i \in [n]} \frac{1}{b_j - a_i} \cdot \frac{1}{X + a_i} \right\}_{j \in [m]}$$

- **Why?**
  - the coefficients $\left(\frac{1}{b_j - a_i} \right)$ form a full-rank C*auchy* matrix*!*

Cauchy Matrices

- For $\vec{a} \in \mathbb{F}^n, \vec{b} \in \mathbb{F}^m$, we define the Cauchy matrix $\mathscr{C}(\vec{a}, \vec{b}) = \left(c_{ij}\right)_{i \in [n], j \in [m]}$ as,

$$c_{ij} = \frac{1}{b_j - a_i}$$

- So,

$$\mathscr{C}(\vec{a}, \vec{b}) = \begin{bmatrix} \frac{1}{b_1 - a_1} & \frac{1}{b_2 - a_1} & \cdots & \frac{1}{b_m - a_1} \\ \frac{1}{b_1 - a_2} & \frac{1}{b_2 - a_2} & \cdots & \frac{1}{b_m - a_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{b_1 - a_n} & \frac{1}{b_2 - a_n} & \cdots & \frac{1}{b_m - a_n} \end{bmatrix}$$

one can show this is full-rank

if $(a_1, \ldots, a_n, b_1, \ldots, b_m)$ are all distinct

Product of Partial Fractions as Cauchy Matrix

$$\left\{\left(\sum_{i\in[n]}\frac{1}{X+a_i}\right)\left(\frac{1}{X+b_j}\right)+\sum_{i\in[n]}\frac{1}{b_j-a_i}\right\}_{j\in[m]}=\left\{\sum_{i\in[n]}\frac{1}{b_j-a_i}\cdot\frac{1}{X+a_i}\right\}_{j\in[m]}$$

$$\mathscr{C}(\vec{a},\vec{b})^T\begin{bmatrix}\frac{1}{X+a_1}\\ \vdots\\ \frac{1}{X+a_n}\end{bmatrix}=\begin{bmatrix}\frac{1}{b_1-a_1} & \frac{1}{b_1-a_2} & \cdots & \frac{1}{b_1-a_n}\\ \frac{1}{b_2-a_1} & \frac{1}{b_2-a_2} & \cdots & \frac{1}{b_2-a_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{1}{b_m-a_1} & \frac{1}{b_m-a_2} & \cdots & \frac{1}{b_m-a_n}\end{bmatrix}\begin{bmatrix}\frac{1}{X+a_1}\\ \vdots\\ \frac{1}{X+a_n}\end{bmatrix}$$

Product of Partial Fractions as Cauchy Matrix

A 2 x 2 Example

Let $n = 2,\ m = 2.$

Suppose $a_1 = 1, a_2 = 4, b_1 = 3,$ and $b_2 = 8.$

$$\left(\frac{1}{X+1}+\frac{1}{X+4}\right)\left(\frac{1}{X+3}\right)+\left(\frac{1}{3-1}+\frac{1}{3-4}\right)\frac{1}{X+3}=\frac{1}{3-1}\cdot\frac{1}{X+1}+\frac{1}{3-4}\cdot\frac{1}{X+4}$$

$$\left(\frac{1}{X+1}+\frac{1}{X+4}\right)\left(\frac{1}{X+8}\right)+\left(\frac{1}{8-1}+\frac{1}{8-4}\right)\frac{1}{X+8}=\frac{1}{8-1}\cdot\frac{1}{X+1}+\frac{1}{8-4}\cdot\frac{1}{X+4}$$

$$\begin{bmatrix} \frac{1}{3-1} & \frac{1}{3-4} \\ \frac{1}{8-1} & \frac{1}{8-4} \end{bmatrix} \begin{bmatrix} \frac{1}{X+1} \\ \frac{1}{X+4} \end{bmatrix}$$

Partial Fraction Decomposition

Linear Independence [JNR25]

TAKEAWAY

Certain partial fraction products are linearly independent. They form a linear system over the individual partial fractions $\frac{1}{X + a_i}$.

Follows from a neat observation using Partial Fraction Decomposition and Cauchy Matrices

We can use this to do cryptography!

Building Batch Encryption

Using Partial Fractions [BNRT26]

Encode keys using partial fractions: Let $p_j(X) = \frac{1}{X + j}$

pk

$p_0(X)$ and $\sum_{j=1}^{\ell} p_j(X)$

"target"

one for each index in the batch

ℓ is the maximum number of transactions possible in a block

ct_i

(1) r_i

(2) $r_i \times \left(\sum_{j=1}^{\ell} p_j(X) + p_0(X) \right)$

(3) $r_i \times p_0(X) + m_i$

Building Batch Encryption

Using Partial Fractions [BNRT26]

Encode keys using partial fractions: Let $p_j(X) = \dfrac{1}{X + j}$

pk

$p_0(X)$ and $\sum_{j=1}^{\ell} p_j(X)$

"target"

one for each index in the batch

ℓ is the maximum number of transactions possible in a block

ct_i

(1) r_i

(2) $r_i \times \left(\sum_{j=1}^{\ell} p_j(X) + p_0(X) \right)$

(3) $r_i \times p_0(X) + m_i$

Building Batch Encryption

Using Partial Fractions [BNRT26]

ct_i

(1) r_i

(2) $r_i \times \left(\sum_{j=1}^{\ell} p_j(X) + p_0(X) \right)$

(3) $r_i \times p_0(X) + m_i$

Batch Decrypter assigns one ciphertext per index, and outputs

(ct_i is assigned to index i)

$$\mathsf{sbk} = \sum_{i=1}^{\ell} r_i \; p_i(X)$$

this is a sum of partial fractions

Building Batch Encryption

Using Partial Fractions [BNRT26]

Given ciphertexts $\{\mathsf{ct}_i\}_{i\in[\ell]}$ and sbk, how do we decrypt?

$$r_i \times \left(\sum_{j=1}^{\ell} p_j(X) + p_0(X) \right), \quad \mathsf{sbk} = \sum_{i=1}^{\ell} r_i \; p_i(X) \;,$$

2nd part of ct_i

$$p_1(X), p_2(X), \cdots, p_\ell(X)$$

We can recreate linearly-independent partial fraction products!

NOTE
cannot solve without sbk

solve linear system for each $i \in [\ell]$

$$r_i \quad p_0(X)$$

NOTE
see paper for details

Building Batch Encryption

Using Partial Fractions [BNRT26]

Subtract solved r_i $p_0(X)$ from

ct_i

---

(1) r_i

(2) $r_i \times \left(\sum_{j=1}^{\ell} p_j(X) + p_0(X) \right)$

(3) $r_i \times p_0(X) + m_i$

to recover m_i

Building Batch Encryption

Implementing Partial Fractions Cryptographically

Encode partial fractions in a **Bilinear group** $\left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\right)$

Can encode partial fractions **using just one group element**!

Bilinear groups allow us to **add partial fractions easily**, and **compute exactly one multiplication** ← exactly what we need for partial fraction products

Sample $x \leftarrow \mathbb{F}_p$ uniformly at random

Represent partial fraction $p_i(X)$ as,

$$g_1^{p_i(x)} \in \mathbb{G}_1, \text{ and } g_2^{p_i(x)} \in \mathbb{G}_2$$

Building Batch Encryption

Takeaways

① Encode keys using partial fractions in a bilinear group

② Each msg. is one-time padded using target partial fraction

③ For each ciphertext, solve linear system using the linear independence of partial fraction products

④ Without pre-decryptor, we cannot solve for the one-time pad so other ciphertexts remain secure

Building Batch Encryption

Additional Requirements

Malicious Ciphertext Injection

Adversary can inject malformed ciphertext to learn the value of a ciphertext not included in the batch.

We prevent this using NIZKs to prove correctness of ciphertext

Thresholdize Secret Key

Our secret key consists of partial fractions evaluated at random x.

Can be easily split using Shamir secret sharing

Introduction to partial fraction technique

Key-Value Commitments and Threshold Encryption

ePrint 2025/2081

ia.cr/2025/2081

presenting at EUROCRYPT in less than a month

What we saw today: Batch Threshold Encryption

ePrint 2026/674

ia.cr/2026/674

References

CGPP24. Arka Rai Choudhuri, Sanjam Garg, Julien Piet, Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: attacks and defenses.

BB04b. Dan Boneh and Xavier Boyen. Short signatures without random oracles.

DY05. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys.

JL09. Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection.

Wee16. Hoeteck Wee. Déjà q: Encore! un petit ibe.

Hab22. Ulrich Haböck. Multivariate lookups based on logarithmic derivatives.

GV22. Rishab Goyal and Vinod Vaikuntanathan. Locally verifiable signature and key aggregation.

JNR25. Charanjit Jutla, **Rohit Nema**, Arnab Roy. Partial fraction techniques for cryptography.

BNRT26. Dan Boneh, **Rohit Nema**, Arnab Roy, Ertem Nusret Tas. Efficient batch threshold encryption using partial fraction techniques.