ROHIT NEMA @ EUROCRYPT 2026 | MAY 13, 2026

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

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

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

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

Partial Fraction Techniques For Cryptography

Using rational functions to do cool crypto

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

Joint work with **Charanjit Jutla** and **Arnab Roy**
Paper: ia.cr/2025/2081

Contents

Overview of Our Results

Starting point: **Partial Fraction Decomposition** of rational functions with linear denominator:

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

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

(1) (Non-)Membership Testing → **Key-Value Commitments**

(2) Linear Independence → **(Dynamic) Threshold Encryption**

with security against *adaptive* corruptions!

Overview of Our Results

Instantiation

Fractions can be **instantiated cryptographically using bilinear groups**. For this talk, we'll leave them as fractions, and only focus on the techniques

Assumption

We prove security using **new q-type assumptions that hold in the generic (bilinear) group model.**

Security

For threshold encryption, we obtain **CCA-2 security against adaptive corruptions** (even if corruptions are made after the public parameters are sampled)

Partial Fraction Decomposition

the mathematical foundation of our work

Recall from high school mathematics:

We can **rewrite product of rational functions as a sum** (with easily computable coefficients)

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

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 $i \in [n]$.

Property (1): (Non-)Membership Testing

Represent set $S = \{a_1, \ldots, a_n\}$ as a sum of fractions

$$a_i \longmapsto \frac{1}{X + a_i} \qquad S \longmapsto \sum_{a_i \in S} \frac{1}{X + a_i}$$

We can test non-membership by decomposing a product of fractions

For $b \in \mathbb{F}$,

$$P = \left(\sum_{a_i \in S} \frac{1}{X + a_i} \right) \left(\frac{1}{X + b} \right)$$

If $b \notin S$,

There exists z_i such that,

$$P = \sum_{a_i \in S} \frac{z_i}{X + a_i} + \frac{z_{n+1}}{X + b}$$

If $b \in S$,

No such decomposition exists

Property (2): Linear Independence

Let's look at the same product again but with distinct values of b: say b_j for $j \in [m]$

$$P_j = \left(\sum_{a_i \in S} \frac{1}{X + a_i} \right) \left(\frac{1}{X + b_j} \right)$$

Each of these products decompose like before: (if $b_j \notin S$)

$$P_j = \sum_{a_i \in S} \frac{z_i^{(j)}}{X + a_i} + \frac{z_{n+1}^{(j)}}{X + b_j}$$

Rearranging to have only the a_i terms on the right:

$$Q_j = P_j - \frac{z_{n+1}^{(j)}}{X + b_j} = \sum_{a_i \in S} \frac{z_i^{(j)}}{X + a_i}$$

The Q_j's are linearly-independent!

(over $\mathbb{F}$ and for $m \leq n$)

Think of each Q_j as a linear relation over the $\frac{1}{X + a_i}$ fractions

Property (2): Linear Independence

$$Q_j = P_j - \frac{z_{n+1}^{(j)}}{X + b_j} = \sum_{a_i \in S} \frac{z_i^{(j)}}{X + a_i}$$

Why?

Using the decomposition theorem, $z_i^{(j)} = \frac{1}{b_j - a_i}$

These coefficients form a full-rank *Cauchy matrix!*

one can show this is full-rank

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

For $\vec{a} \in \mathbb{F}^n, \vec{b} \in \mathbb{F}^m$, we define the Cauchy matrix

$$\mathscr{C}(\vec{a}, \vec{b}) = \left(c_{ij}\right)_{i \in [n], j \in [m]} \text{ as } c_{ij} = \frac{1}{b_j - a_i} .$$

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

Viewing (2) As A Cauchy Matrix

$$Q_j = P_j - \frac{z_{n+1}^{(j)}}{X + b_j} = \sum_{i \in [n]} \frac{z_i^{(j)}}{X + a_i} \quad \text{where} \quad z_i^{(j)} = \frac{1}{b_j - a_i}$$

$$\mathscr{C}(\vec{a}, \vec{b})^T \begin{bmatrix} \frac{1}{X + a_1} \\ \vdots \\ \frac{1}{X + a_n} \end{bmatrix} = \begin{bmatrix} \frac{1}{b_1 - a_1} & \frac{1}{b_1 - a_2} & \cdots & \frac{1}{b_1 - a_n} \\ \frac{1}{b_2 - a_1} & \frac{1}{b_2 - a_2} & \cdots & \frac{1}{b_2 - a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{b_m - a_1} & \frac{1}{b_m - a_2} & \cdots & \frac{1}{b_m - a_n} \end{bmatrix} \begin{bmatrix} \frac{1}{X + a_1} \\ \vdots \\ \frac{1}{X + a_n} \end{bmatrix} = \begin{bmatrix} Q_1 \\ \vdots \\ Q_m \end{bmatrix}$$

First construction:
Key-Value Commitments

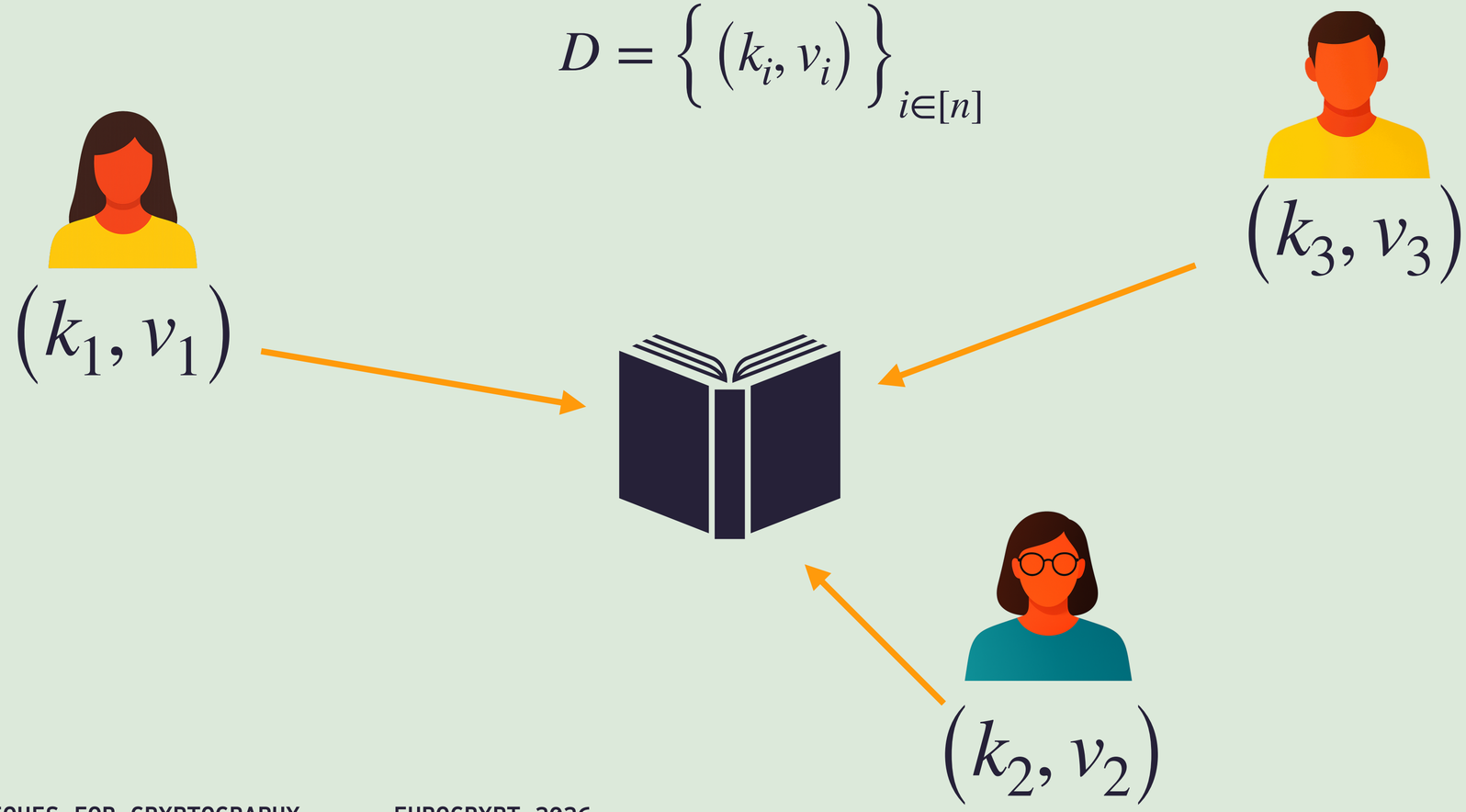
Using (1) Non-Membership Testing

Key-Value Commitments (KVCs)

introduced by [AR20]

Goal: Commit to a dictionary, a set of key-value pairs. Efficient (non-)membership proofs and updates.

$$D = \left\{ (k_i, v_i) \right\}_{i \in [n]}$$



Key-Value Commitments (KVCs)

Solution: Commit to set using linear combination of “partial fractions”

$$C_D(X) = \sum_{i \in [n]} \frac{v_i}{X + k_i}$$

$$\frac{(k, v) \in D}{\exists z_i \text{ such that, } \left(C_D(X) - \frac{v}{X+k}\right)\left(\frac{1}{X+k}\right) = \sum_{i \in [n]} \frac{z_i}{X + k_i}}$$

$$\frac{(k, v) \notin D}{\textit{no such decomposition exists} \text{ because of quadratic fraction, } \frac{1}{(X+k)^2}}$$

Using this, we construct a new primitive called **Credential-based KVCs.**

See paper for details

Efficiency & Comparison With Prior Work

with RSA-based “KVaC” [AR20]

✓ **Smaller groups**

Prime-order instead of composite-order groups. **Much smaller in practice** $\Longrightarrow$ **less communication**

✓ **Computationally efficient**

No expensive calculation of Bezout coefficients

with Cuckoo Commitments [FKdP23]

✓ **Purely algebraic**

Our construction is purely algebraic **avoiding non-algebraic operations such as cuckoo hashing**

✓ **Stateless updates**

Supports stateless updates while Cuckoo commitments does not

Second construction:
Dynamic Threshold Encryption

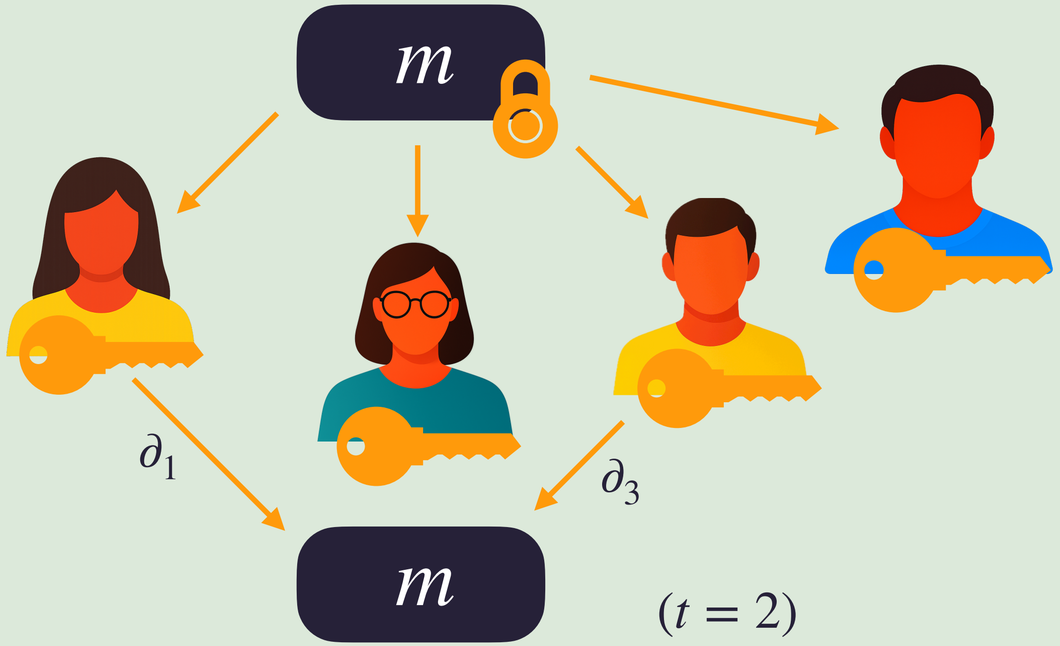
Using (2) Linear Independence

Dynamic Threshold Encryption (DTE)

introduced by [DP08]

Public-key encryption **with threshold decryption** [DF89]

Given ciphertext, users can **generate decryption shares**, which can be **combined to recover plaintext**



Additional property

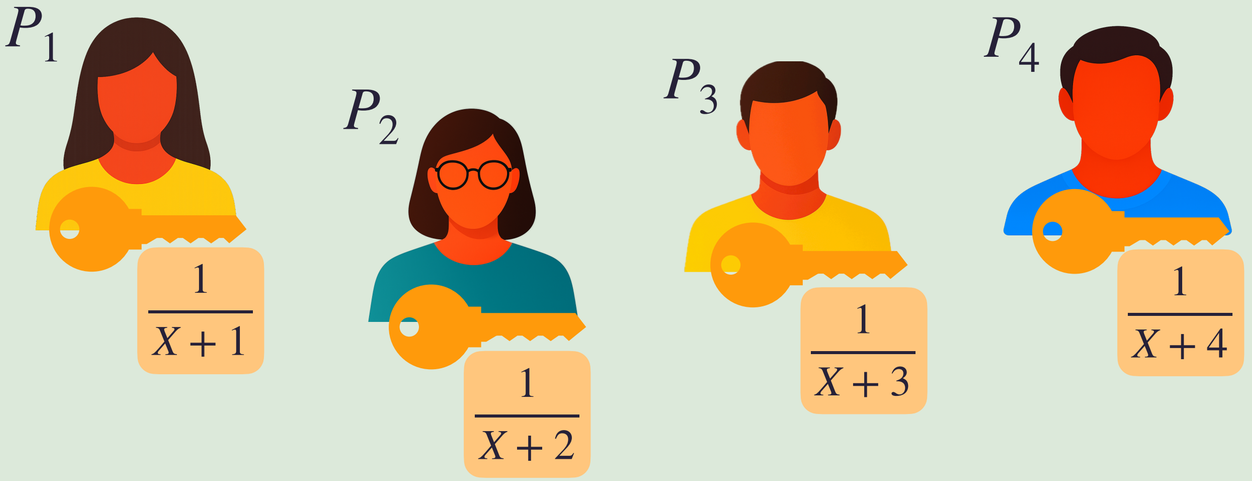
Dynamic selection of threshold and authorized set at encryption time

Use Partial Fractions as Keys and One-Time Pad

Encode public and secret keys using partial fractions

$$m \quad = \quad \sum_{i \notin A} \frac{1}{X+i} \;,\; \frac{1}{X} + m$$

Encode authorized set A as sum of partial fractions. For technical reasons, **we encode the complement** of the set, the *unauthorized set*



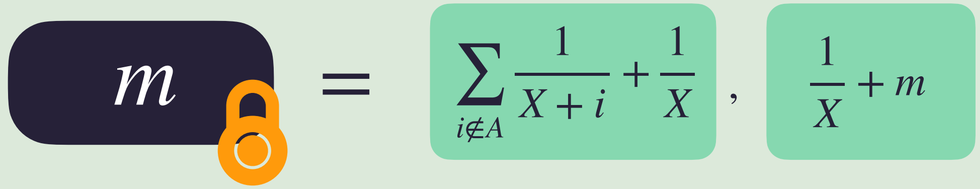
Mask the message using a “target” partial fraction as a one-time pad

$$A = \{1,3\}$$
$$(t = 2)$$

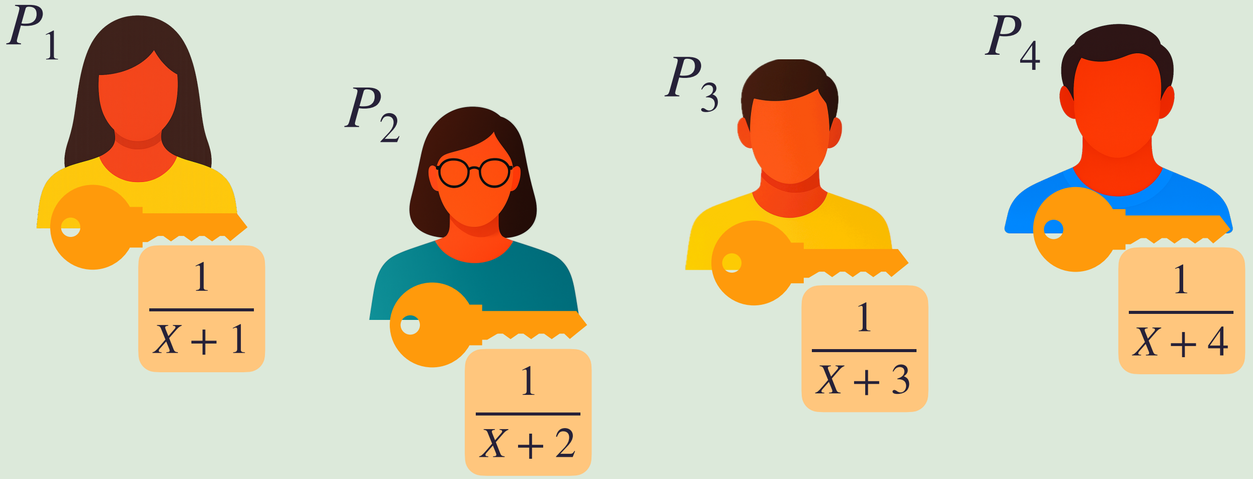
For simplicity, assume users are in $[n]$ but can extend to arbitrary identities using a hash function…

Use Partial Fractions as Keys and One-Time Pad

Encode public and secret keys using partial fractions



Encode authorized set A as sum of partial fractions. For technical reasons, **we encode the complement** of the set, the *unauthorized set*



Mask the message using a “target” partial fraction as a one-time pad

We’ll put this target in the sum as well

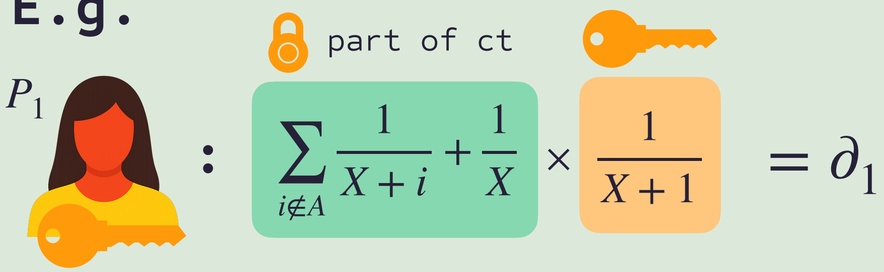
$$A = \{1,3\}$$
$$(t = 2)$$

For simplicity, assume users are in $[n]$ but can extend to arbitrary identities using a hash function…

Partial Decryption By Computing Product

Parties can **compute a product using their secret key**

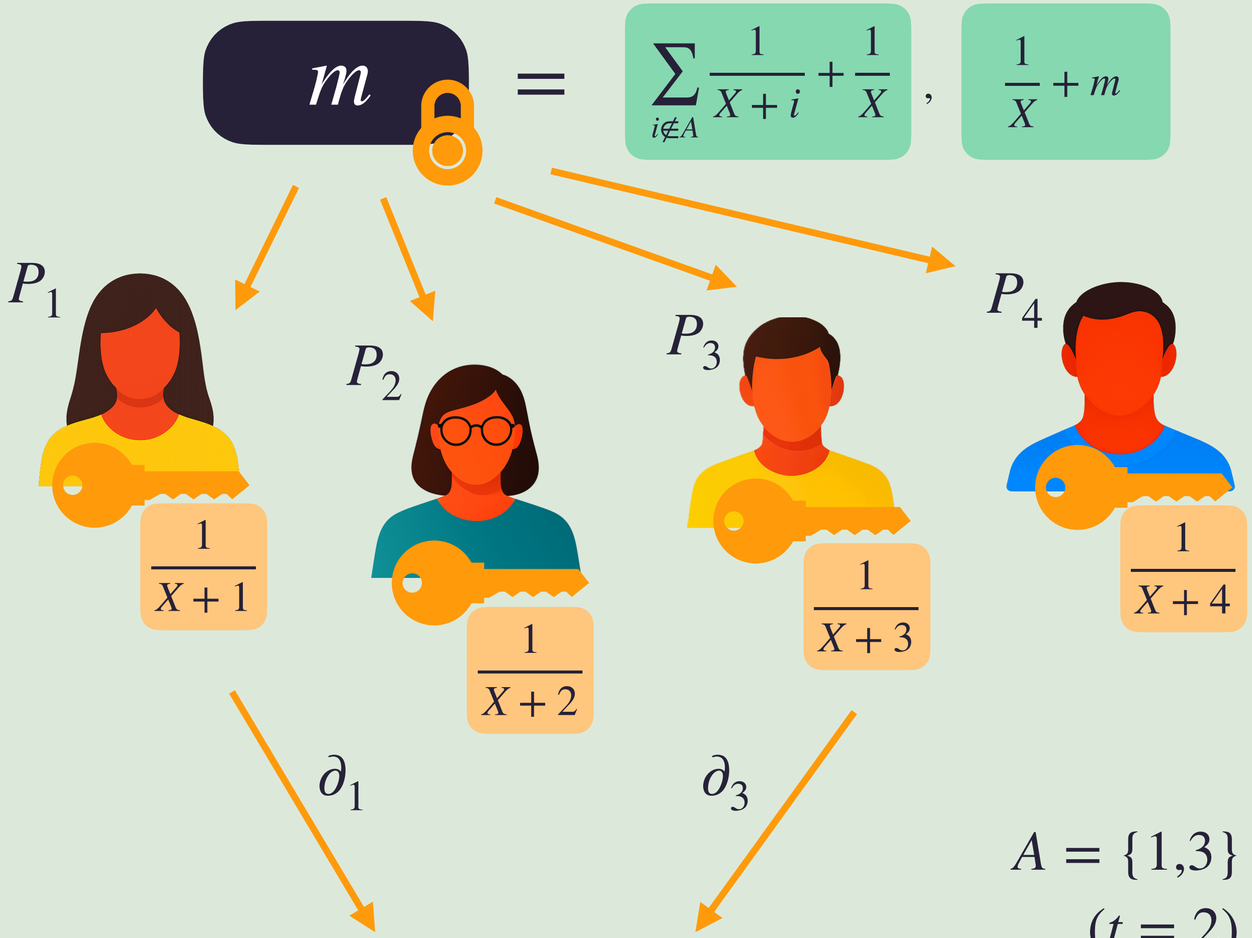
E.g.



If they are authorized, **the product decomposes!**

And if they are not, **it does not** (there is a quadratic term)

The ∂_i form a linear relation over the fractions (including the target)



$A = \{1,3\}$
$(t = 2)$

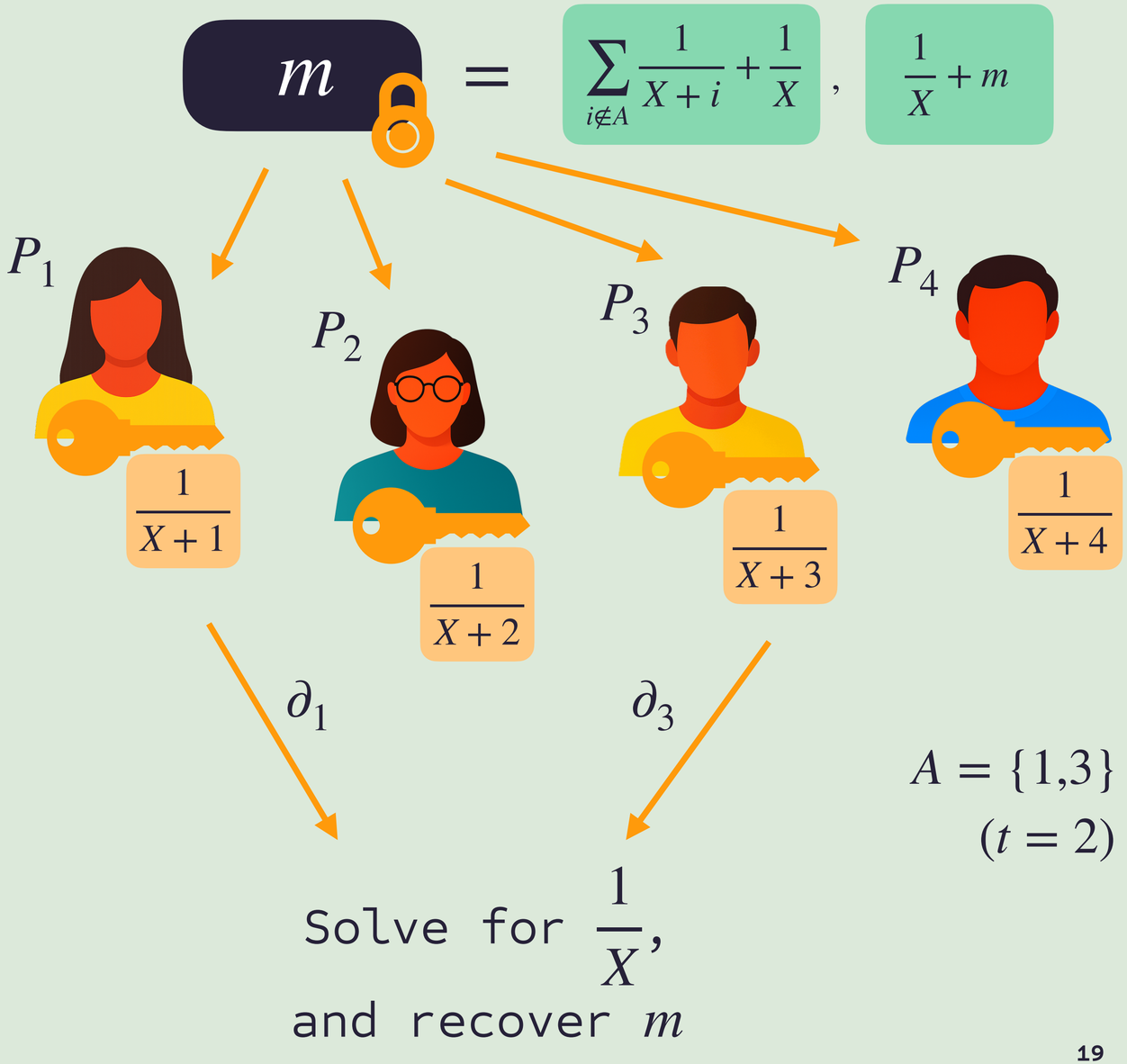
In this example, P_1 and P_3 can contribute partial decryptions

Using (2) Linear Independence to Combine

By Property (2), **the ∂_i are linearly-independent**

If we obtain enough of them (in this case 2),

we can solve for the target $\frac{1}{X}$



Additional Details

Reusing the partial fractions

Multiply entire system by a random r at encryption time (ElGamal-like ct)

Arbitrary Dynamic Threshold

Add dummy fractions to the *unauthorized set*

Preprocessing

If authorized set is known in advance, encryption key can be preprocessed into a single group element

Comparison With Prior Work

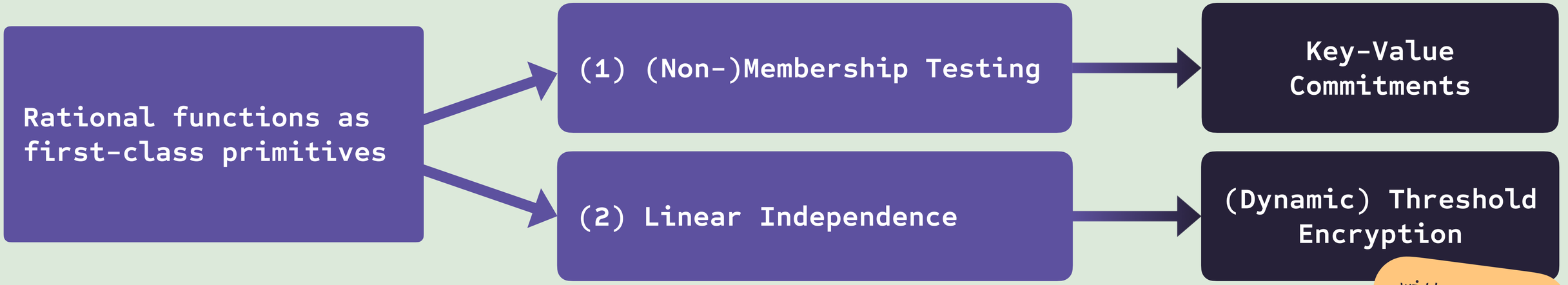
(including *silent* setup schemes)

does not support preprocessing

Construction	|ek|	|pp|	|pk|	|ct|	$\|\partial\|$	|ck|
DP08 [7]	$O(n)$	–	$O(1)$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$3\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$O(n)$
GKPW24 [11]	$O(1)$	$O(n)$	$O(n)$	$2\|\mathbb{G}_1\| + 7\|\mathbb{G}_2\| + \|G_T\|$	$1\|\mathbb{G}_2\|$	$O(s)$
WW25 [20]	$O(\log n)$	$O(n \log n)$	$O(n)$	$3\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$2\|\mathbb{G}\|$	$O(s)$
This work	$O(\log s)$	$O(\log n)$	$O(1)$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$ $(+3\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$ for CCA)	$2(\|G_1\| + \|\mathbb{G}_2\|)$	$O(n)$

Construction	ct validation?	# pairings in decrypt	# pairings in verifying share	No RO?	Adaptive?	Assumption/Model
DP08 [7]	✓	$O(t)$	2	X	X	(static) q-type
GKPW24 [11]	X	0	2	X	✓	generic group
WW25 [20]	X	3	2	✓	X	(static) q-type
This work	✓	$O(t)$	3	✓	✓	(adaptive) q-type

Takeaways & Future Work



see paper

possible patterns in other primitives: Hadamard product proofs, registration-based encryption,... (see paper's appendix)

subsequent work: Efficient batch encryption using these techniques

to appear in CRYPTO 2026

Thank you! Paper: ia.cr/2025/2081