CRYPTO 2026 | AUGUST 17, 2026

$$\frac{1}{X+i}\cdot\frac{1}{X+j}=\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$$

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

Batch Encryption Using “Partial Fractions”

Using rational functions to do cool crypto (part 2)
(See EUROCRYPT 2026 for part 1!)

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

Rohit Nema

joint work with

Dan Boneh
(Stanford)

Arnab Roy
(Mysten Labs)

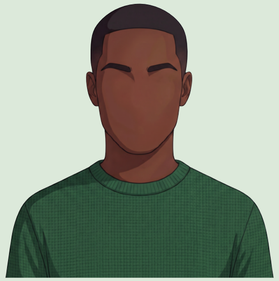
Ertem Nusret Tas
(a16z Crypto Research)

$$\frac{1}{(X+k)^2}$$

Paper: ia.cr/2026/674

Batch Encryption

introduced by [CGPP24]

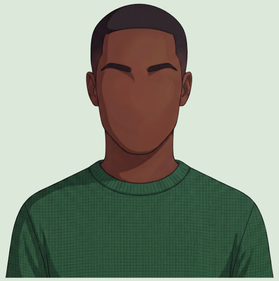


Batch Encryption

introduced by [CGPP24]

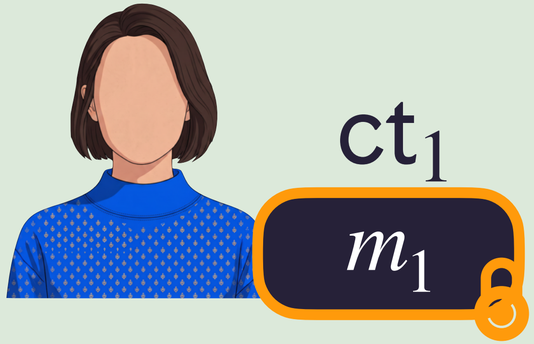
pk

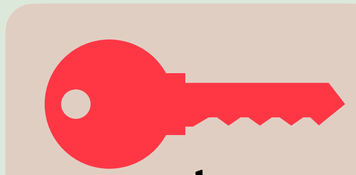
sk

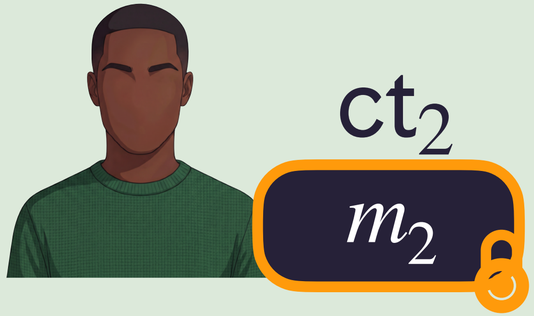


Batch Encryption

introduced by [CGPP24]

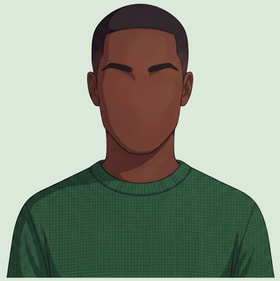




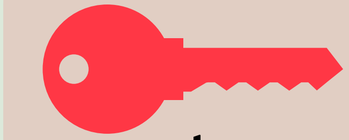


Batch Encryption

introduced by [CGPP24]



pk

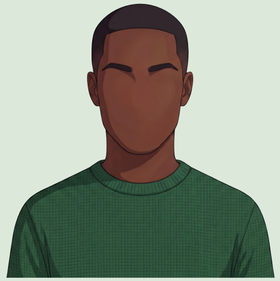


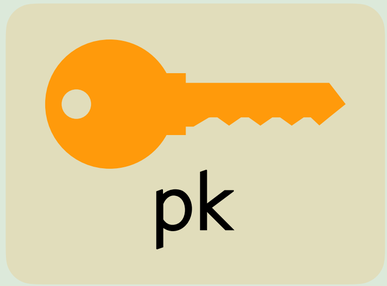
sk

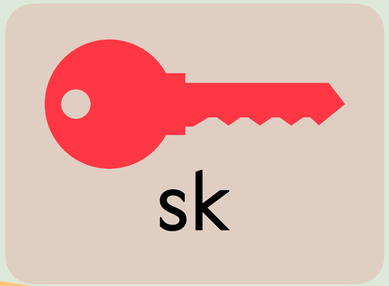
m_1 m_2 m_3

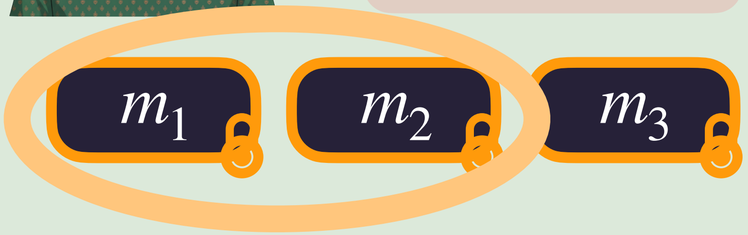
Batch Encryption

introduced by [CGPP24]



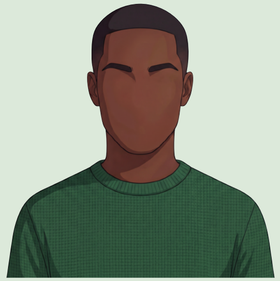


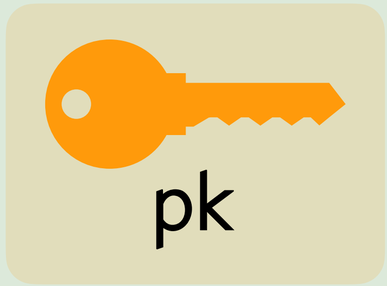


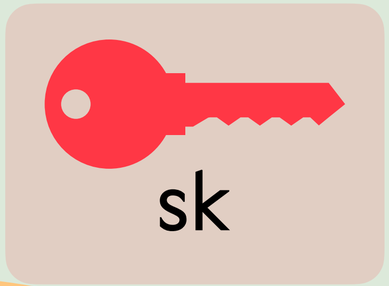


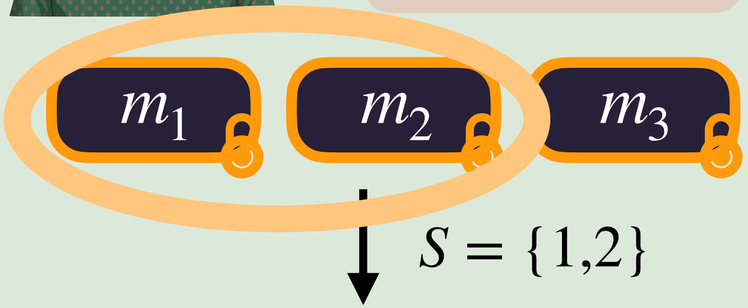
Batch Encryption

introduced by [CGPP24]



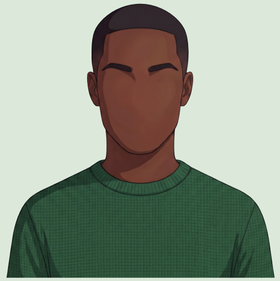


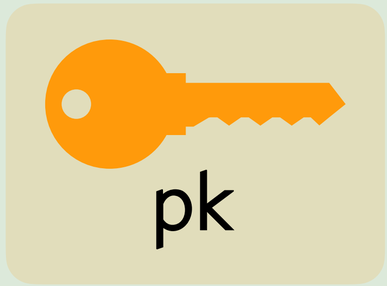


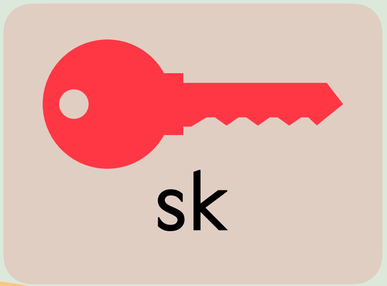


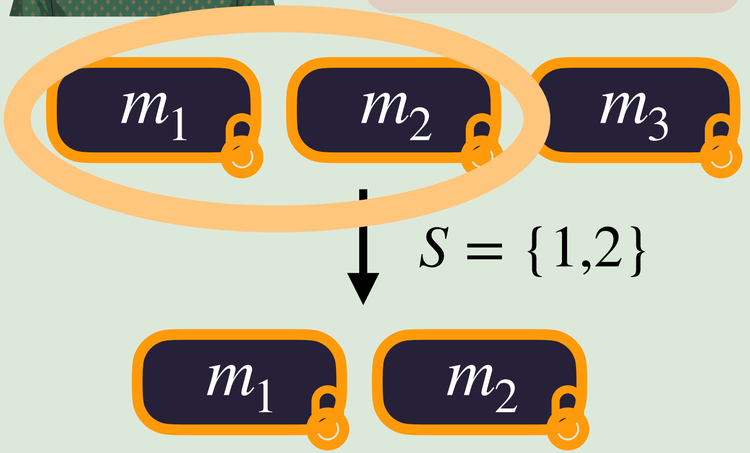
Batch Encryption

introduced by [CGPP24]



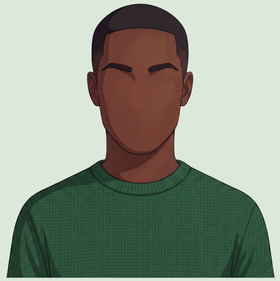


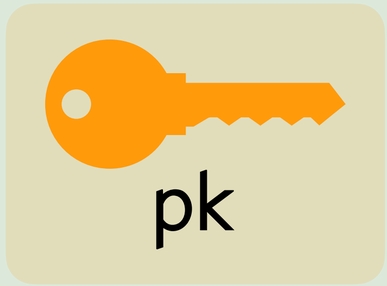


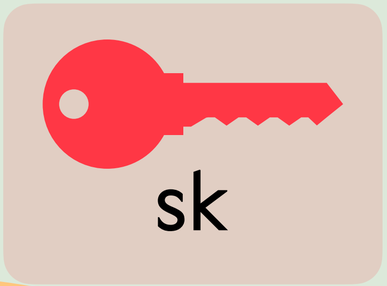


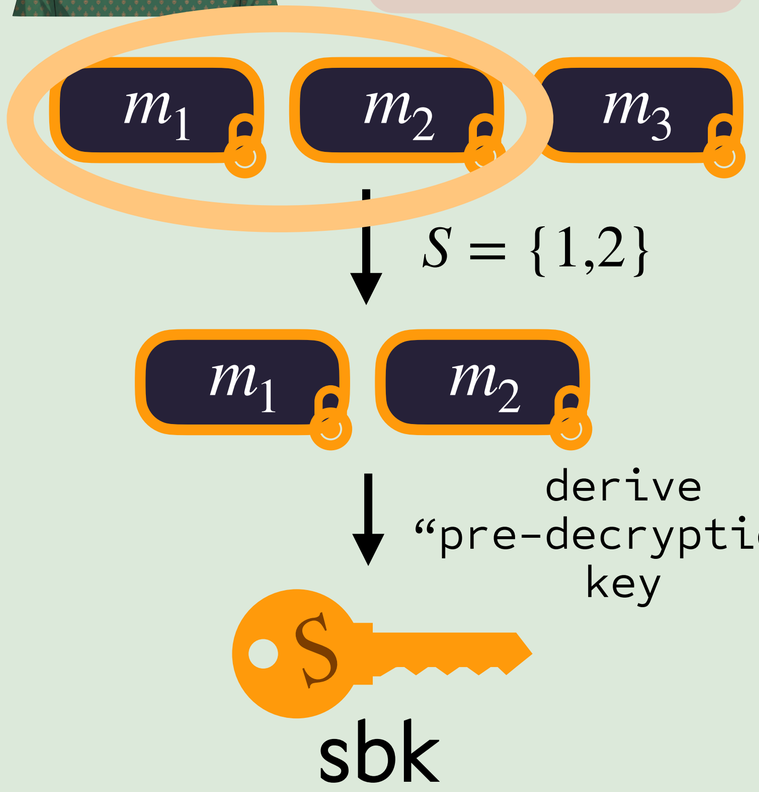
Batch Encryption

introduced by [CGPP24]



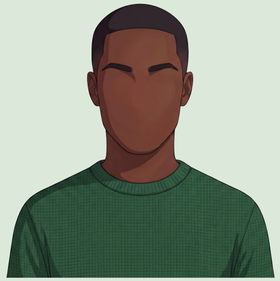


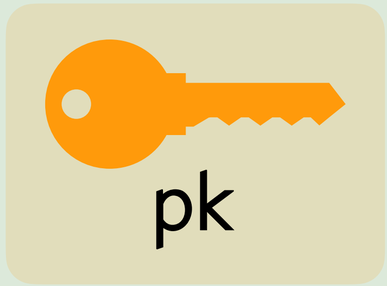


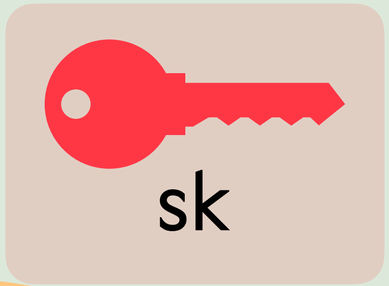


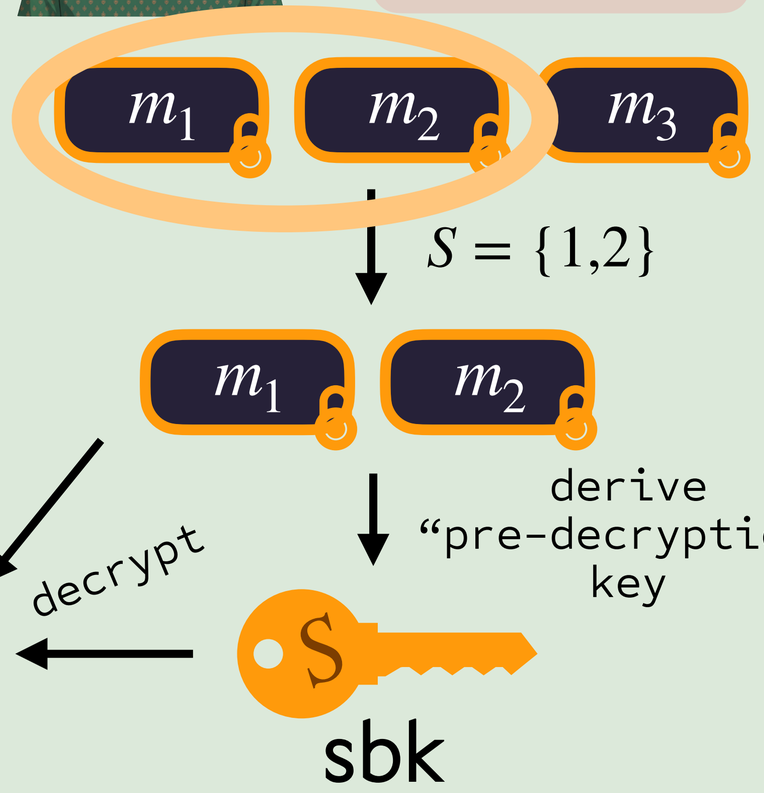
Batch Encryption

introduced by [CGPP24]



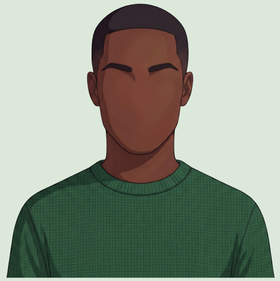




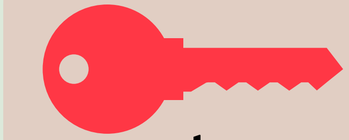


Batch Encryption

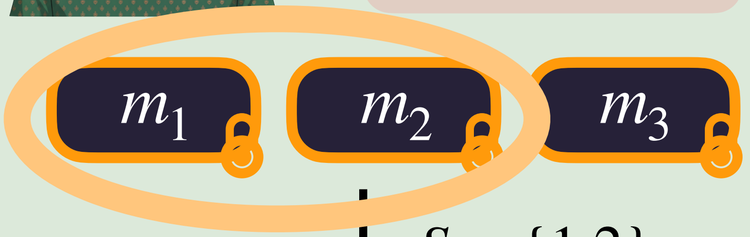
introduced by [CGPP24]



pk



sk



$S = \{1,2\}$

derive "pre-decryption" key

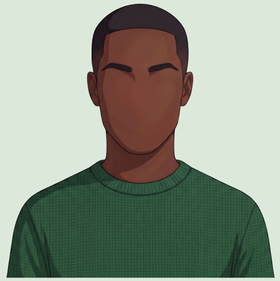
sbk

decrypt

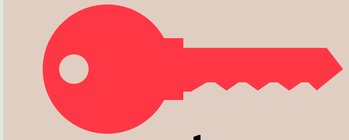
remains semantically secure

Batch Encryption

introduced by [CGPP24]



pk



sk

m_1 m_2 m_3

trivial solution:
Decryptor can simply do the decryption themself and publish the decrypted messages as sbk

decrypt

“pre-decryption” key

m_1 m_2

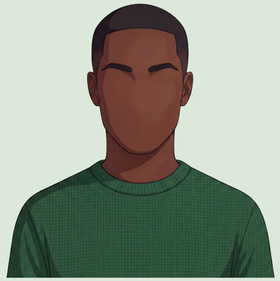
sbk

m_3

remains semantically secure

Batch Encryption

introduced by [CGPP24]



pk

sk

m_1 m_2 m_3

$S = \{1,2\}$

m_1 m_2

derive "pre-decryption" key

ecrypt

pre-decryption key should be short

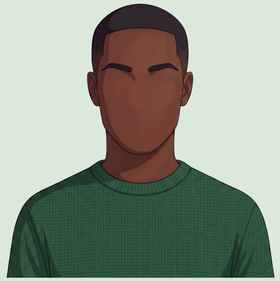
m_3

S

sbk

Batch Encryption

introduced by [CGPP24]



pk

sk

ciphertexts should also be short

m_1 m_2 m_3

$S \subseteq \{1,2\}$

m_1 m_2

derive “pre-decryption” key

Decrypt

pre-decryption key should be short

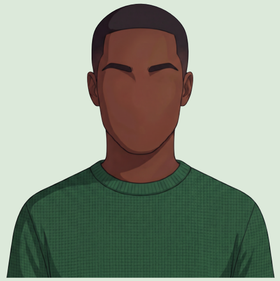
m_3

S

sbk

Batch *Threshold* Encryption

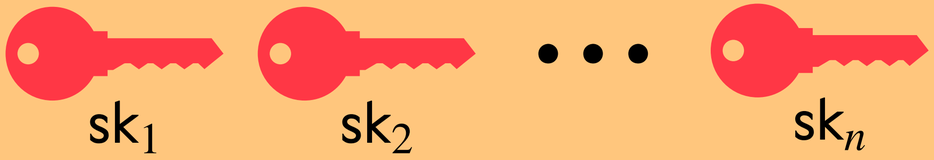
introduced by [CGPP24]



pk

sk

can be split between multiple parties with



sk_1 sk_2 ••• sk_n

and threshold t, s.t. at least t parties required to decrypt subset.

Decryption is non-interactive!

m_1

m_3

remains seman
secure

Batch *Threshold* Encryption

Batch *Threshold* Encryption

Highly active area of research

Loads of work on this in various settings!

See e.g. [SAS24, ABD+25, AFP25, BCF+25, BFO+25, CGP+25, FPT+25, GWW+25, ADG+26, Pol26]

Achieve either **a relaxed notion of the primitive or suffer from a large CRS**. (We will revisit this toward the end…)

Batch *Threshold* Encryption

Highly active area of research

Loads of work on this in various settings!

See e.g. [SAS24, ABD+25, AFP25, BCF+25, BFO+25, CGP+25, FPT+25, GWW+25, ADG+26, Pol26]

Achieve either **a relaxed notion of the primitive or suffer from a large CRS**. (We will revisit this toward the end…)

Why do we care?

Batch encryption has a **strong application in blockchains.**

It **allows transactions to stay hidden (encrypted)** until the block is confirmed.

Helps **prevent MEV attacks** (e.g. front-running)

The Core Technique [Jutla-N-Roy26]

The Core Technique [Jutla-N-Roy26]

Use *simple* rational functions as first-class primitives

The Core Technique

[Jutla-N-Roy26]

Use *simple* rational functions as first-class primitives

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

finite field of prime order p

The Core Technique

[Jutla-N-Roy26]

Use *simple* rational functions as first-class primitives

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

finite field of prime order p

We will **encode keys and ciphertexts using fractions**, instantiated in a bilinear pairing group $\mathscr{G} = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\right)$

The Core Technique

[Jutla-N-Roy26]

Use *simple* rational functions as first-class primitives

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

finite field of prime order p

We will **encode keys and ciphertexts using fractions**, instantiated in a bilinear pairing group $\mathscr{G} = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\right)$

The Core Technique [Jutla-N-Roy26]

Use *simple* rational functions as first-class primitives

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

finite field of prime order p

We will **encode keys and ciphertexts using fractions**, instantiated in a bilinear pairing group $\mathcal{G} = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\right)$

These fractions have two elementary yet powerful properties:

(1) Decomposition

(2) Linear Independence

The Core Technique [Jutla-N-Roy26]

Use *simple* rational functions as first-class primitives

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

finite field of prime order p

We will **encode keys and ciphertexts using fractions**, instantiated in a bilinear pairing group $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$

These fractions have two eleme rful properties:

we'll focus on this one for this talk

(1) Decomposition

(2) Linear Independence

Property 1: Decomposition [Jutla-N-Roy26]

Property 1: Decomposition [Jutla-N-Roy26]

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

Property 1: Decomposition [Jutla-N-Roy26]

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

$$\text{e.g. } \frac{1}{X+3}\cdot\frac{1}{X+5}=\frac{1}{2}\left(\frac{1}{X+3}-\frac{1}{X+5}\right);$$

Property 1: Decomposition

[Jutla-N-Roy26]

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

$$\text{e.g. } \frac{1}{X+3}\cdot\frac{1}{X+5}=\frac{1}{2}\left(\frac{1}{X+3}-\frac{1}{X+5}\right);$$

$$\text{in general: } \frac{1}{X+i}\cdot\frac{1}{X+j}=\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$$

Property 1: Decomposition

[Jutla-N-Roy26]

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

$$\text{e.g. } \frac{1}{X+3}\cdot\frac{1}{X+5}=\frac{1}{2}\left(\frac{1}{X+3}-\frac{1}{X+5}\right);$$

$$\text{in general: } \frac{1}{X+i}\cdot\frac{1}{X+j}=\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$$

Property 1: Decomposition [Jutla-N-Roy26]

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

$$\text{e.g. } \frac{1}{X+3}\cdot\frac{1}{X+5}=\frac{1}{2}\left(\frac{1}{X+3}-\frac{1}{X+5}\right);$$

$$\text{in general: } \frac{1}{X+i}\cdot\frac{1}{X+j}=\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$$

Product with itself cannot be decomposed

Property 1: Decomposition [Jutla-N-Roy26]

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

$$\text{e.g. } \frac{1}{X+3}\cdot\frac{1}{X+5}=\frac{1}{2}\left(\frac{1}{X+3}-\frac{1}{X+5}\right);$$

$$\text{in general: } \frac{1}{X+i}\cdot\frac{1}{X+j}=\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$$

Product with itself cannot be decomposed

$$\text{e.g. } \frac{1}{X+3}\cdot\frac{1}{X+3}=\frac{1}{(X+3)^2}$$

Property 1: Decomposition

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{X+i} \cdot \frac{1}{X+i}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{X+j} \cdot \frac{1}{X+j}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{X+k} \cdot \frac{1}{X+k}$

Matrix of pairwise products

Property 1: Decomposition

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Products on the *diagonal* do not decompose. Remain *quadratic*.

Property 1: Decomposition

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$
$-j$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{(X+j)^2}$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$
$-k$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$	$\frac{1}{(X+k)^2}$

Off-diagonal/cross-terms *linearize*!

Our Construction: Roadmap

Our Construction: Roadmap

Index-dependent
batch encryption

Ciphertext is tied to a **fixed position** in the batch **known at encryption time**

Enc(pk, **index** $i \in [\ell]$, **msg** m)

Our Construction: Roadmap

Index-dependent
batch encryption

Ciphertext is tied to a **fixed position** in the batch **known at encryption time**

Enc(pk, **index** $i \in [\ell]$, **msg** m)

+

Broadcast Encryption

Short ciphertext can be decrypted by only authorized secret keys

Dec(**authorized** $\mathsf{sk}_j, \mathsf{ct}) \to m$

Dec(**unauthorized** $\mathsf{sk}_j, \mathsf{ct}) \to \perp$

from [JNR26]:
built using
partial fractions

Our Construction: Roadmap

Index-dependent
batch encryption

Ciphertext is tied to a **fixed position** in the batch **known at encryption time**

Enc(pk, **index** $i \in [\ell]$, **msg** m)

+

Broadcast Encryption

Short ciphertext can be decrypted by only authorized secret keys

Dec(**authorized** $\mathsf{sk}_j, \mathsf{ct}) \to m$

Dec(**unauthorized** $\mathsf{sk}_j, \mathsf{ct}) \to \perp$

→

Batch Encryption

from [JNR26]:
built using
partial fractions

Our Construction: Roadmap

Index-dependent
batch encryption

Ciphertext is tied to a **fixed position** in the batch **known at encryption time**

Enc(pk, **index** $i \in [\ell]$, **msg** m)

+

Broadcast Encryption

Short ciphertext can be decrypted by only authorized secret keys

Dec(**authorized** $\mathsf{sk}_j, \mathsf{ct}) \to m$

Dec(**unauthorized** $\mathsf{sk}_j, \mathsf{ct}) \to \perp$

from [JNR26]:
built using
partial fractions

→

Batch Encryption

Thresholdize using LSS

↓

Batch Threshold Encryption

Our Construction: Roadmap

Index-dependent
batch encryption

Ciphertext is tied to a **fixed position** in the batch **known at encryption time**

Enc(pk, **index** $i \in [\ell]$, **msg** m)

+

Broadcast Encryption

Short ciphertext can be decrypted by only authorized secret keys

Dec(**authorized** $\mathsf{sk}_j, \mathsf{ct}) \to m$

Dec(**unauthorized** $\mathsf{sk}_j, \mathsf{ct}) \to \bot$

→

Batch Encryption

Thresholdize using LSS

↓

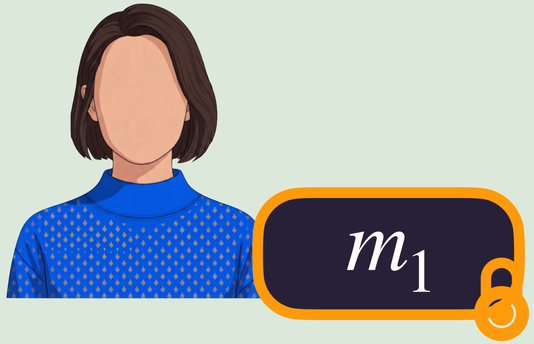
Batch Threshold Encryption

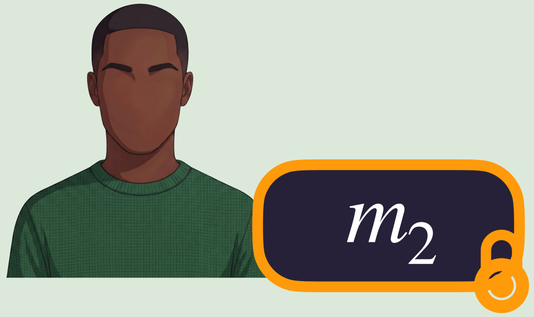
today's focus:
we will construct this using the *partial fractions* technique we saw earlier

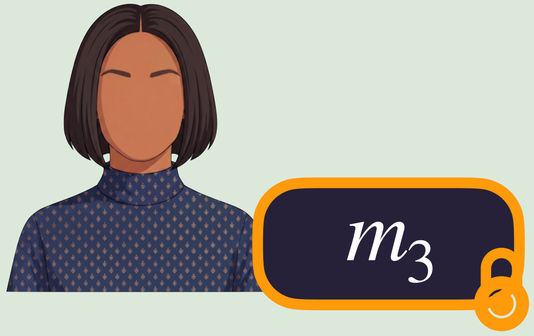
om [JNR26]:
uilt using
ial fractions

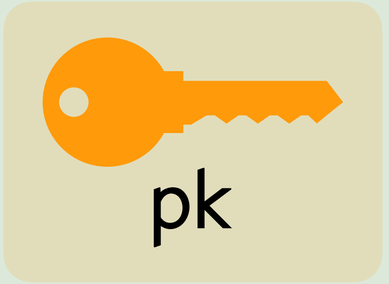
Index Dependence

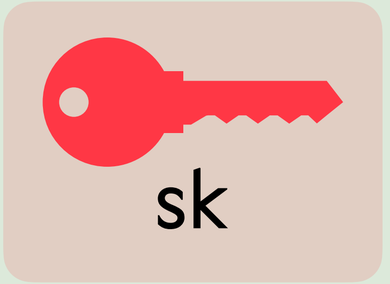
ℓ: max batch size = 3





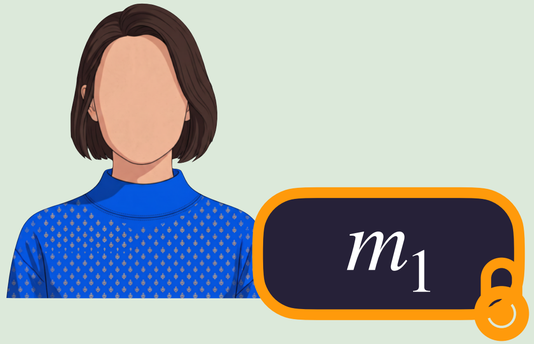


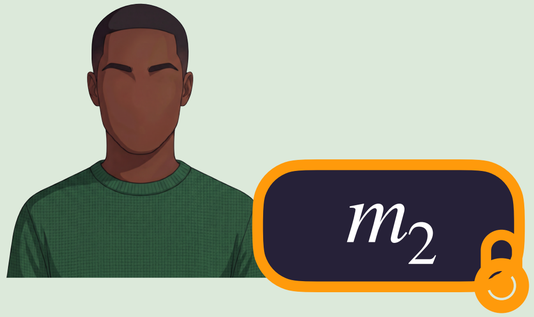


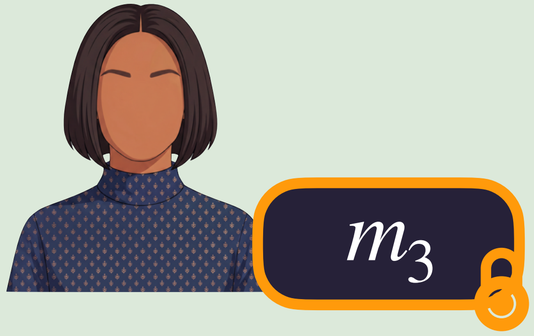


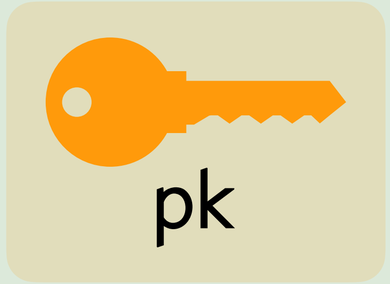
Index Dependence

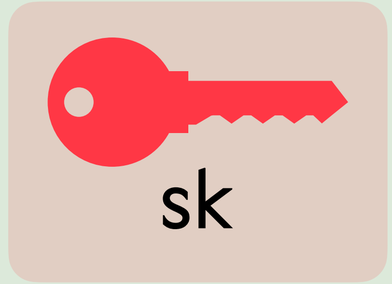
ℓ: max batch size = 3

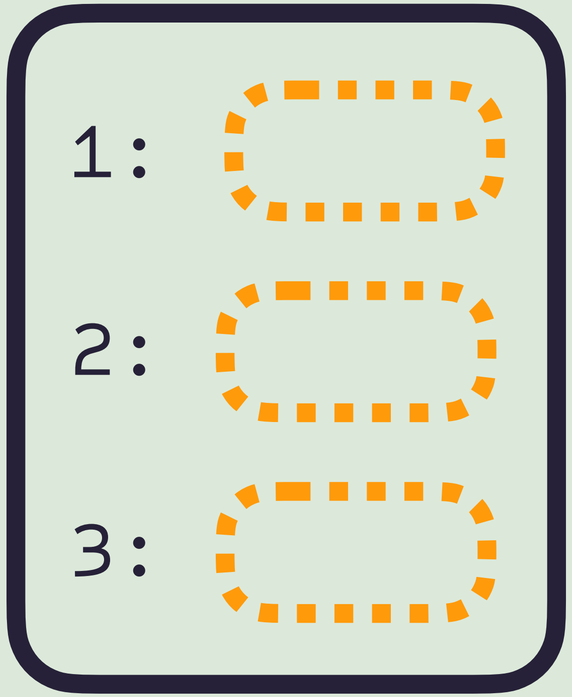






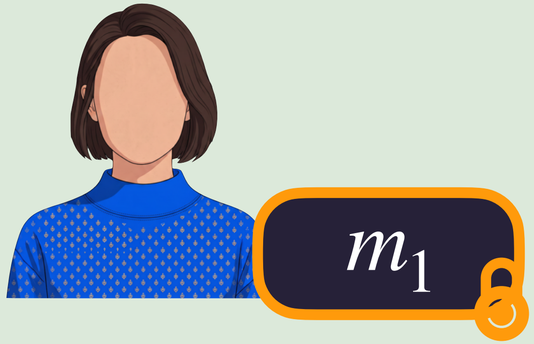




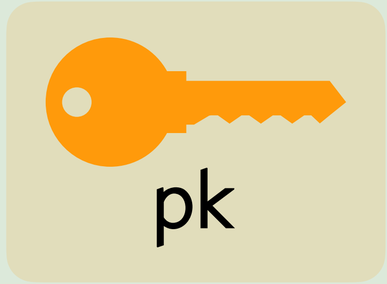


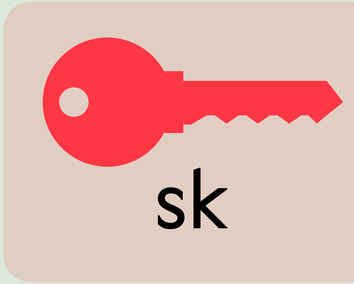
Index Dependence

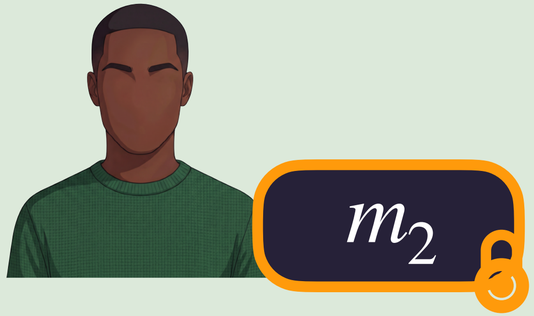
ℓ: max batch size = 3

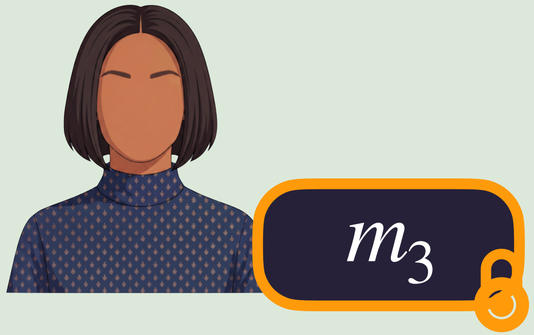


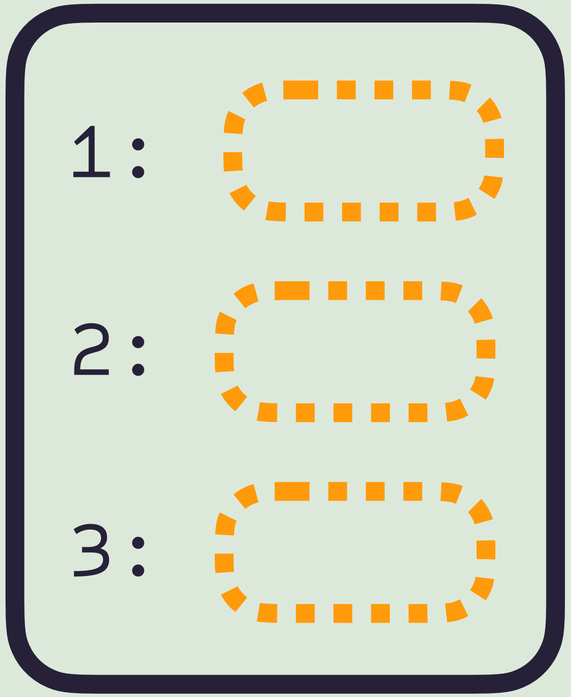
, 1





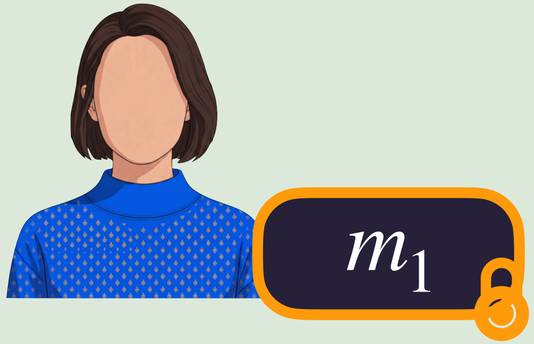




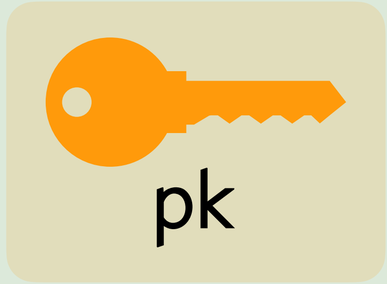


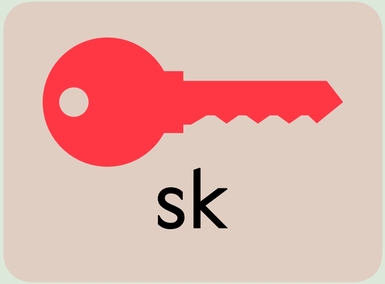
Index Dependence

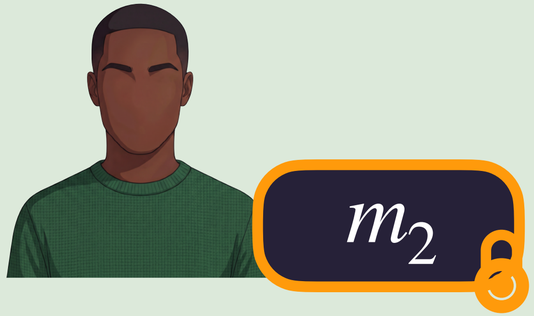
ℓ: max batch size = 3



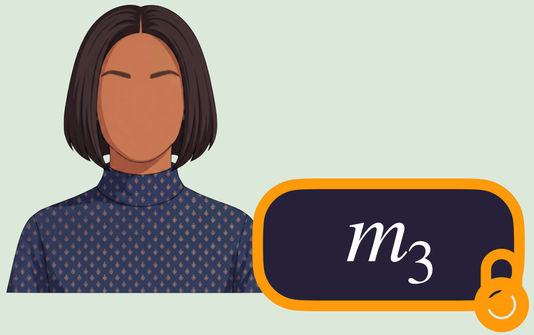
, 1

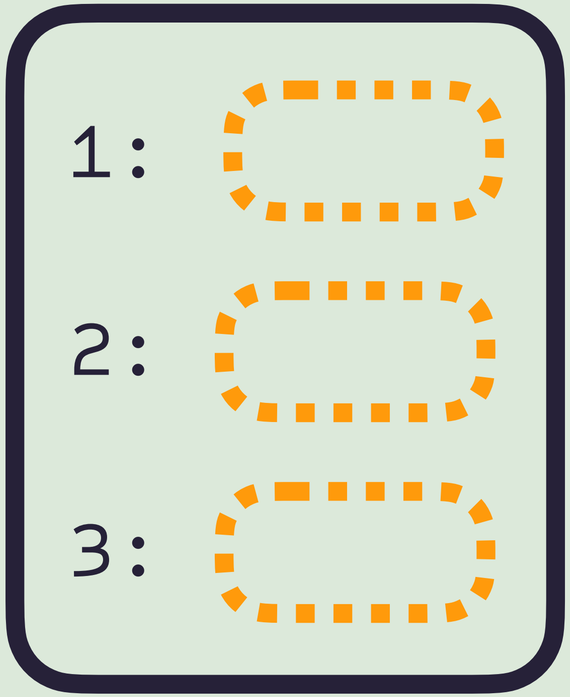






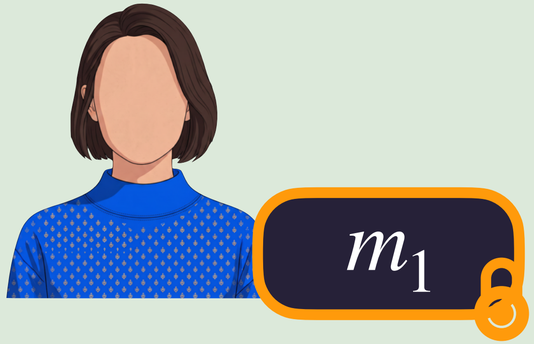
, 2



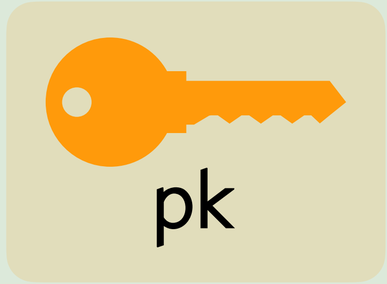


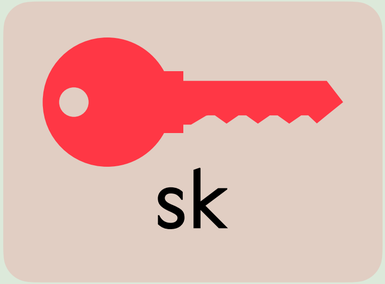
Index Dependence

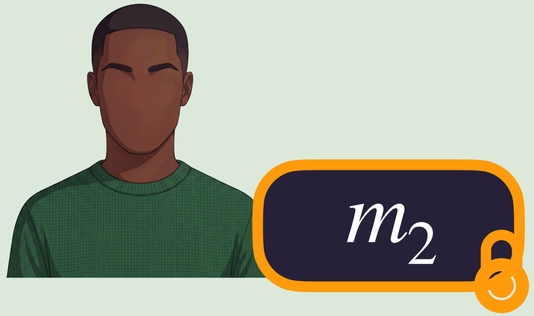
ℓ: max batch size = 3



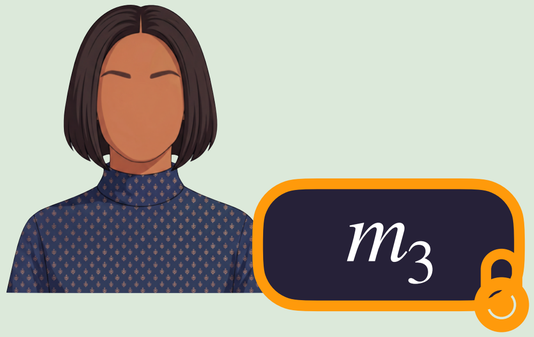
, 1



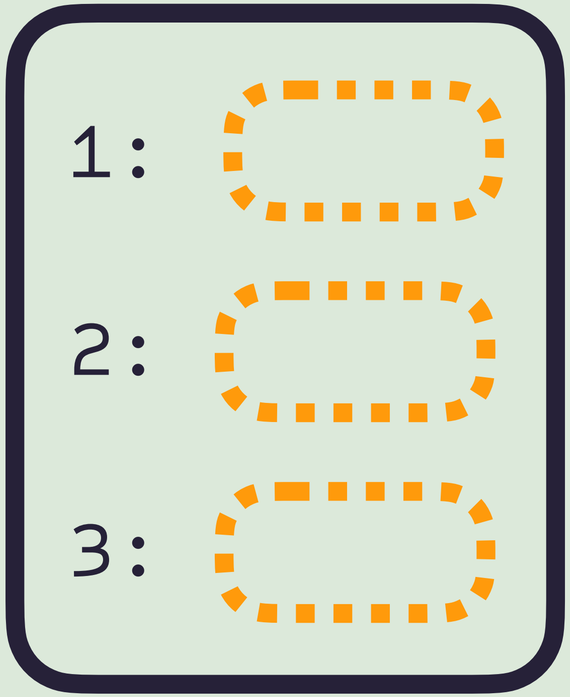




, 2

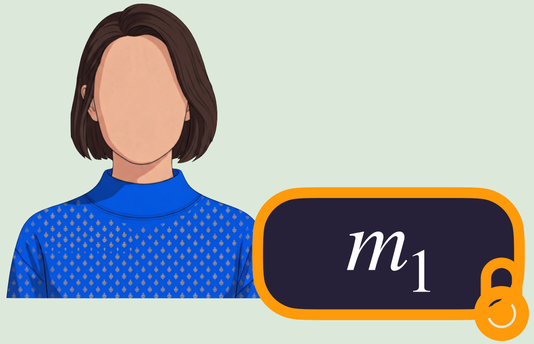


, 3



Index Dependence

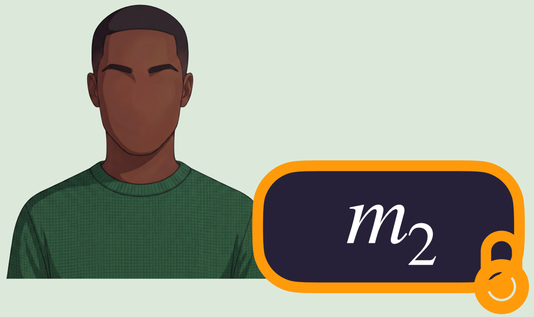
ℓ: max batch size = 3



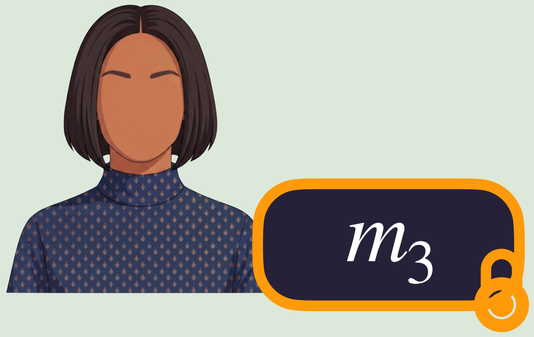
m_1 , 1

pk

sk



m_2 , 2



m_3 , 3

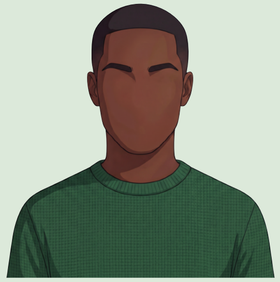
1:

2:

3:

Index Dependence

ℓ: max batch size = 3



pk

sk

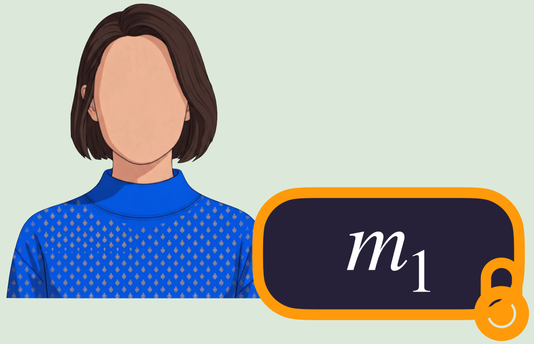
1: m_1

2: m_2

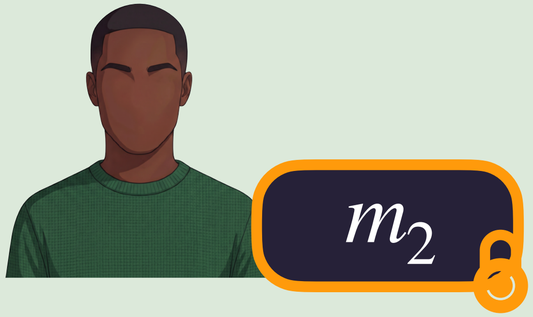
3: m_3

Index Dependence

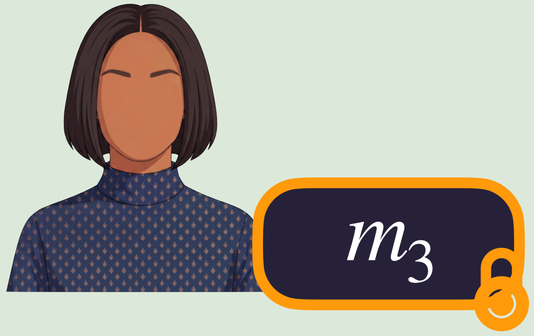
ℓ: max batch size = 3



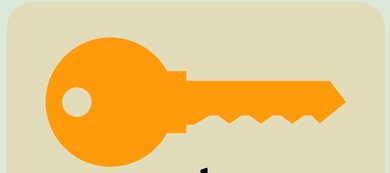
, 1



, 2

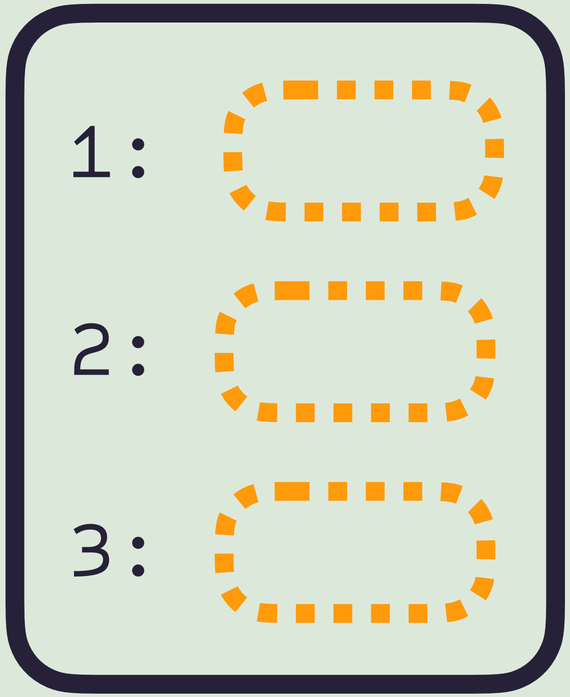


, 1



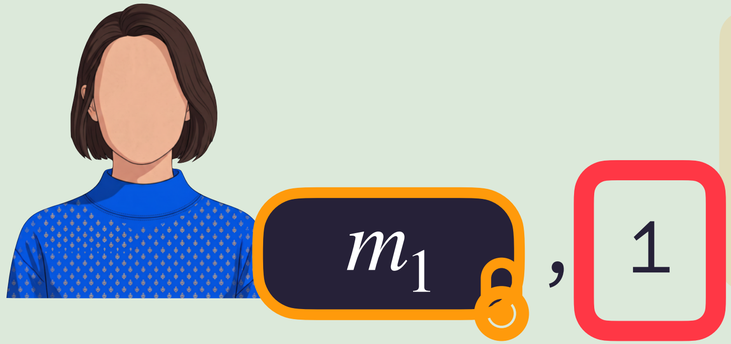
pk

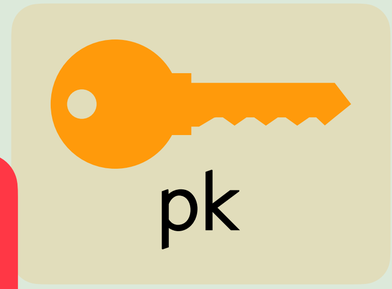
sk

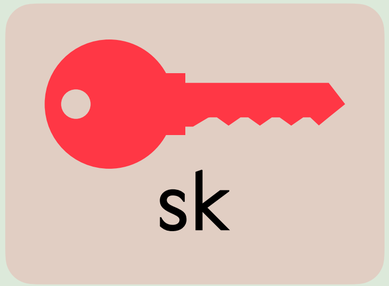


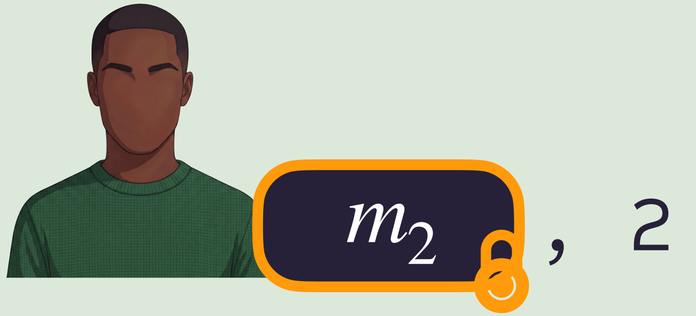
Index Dependence

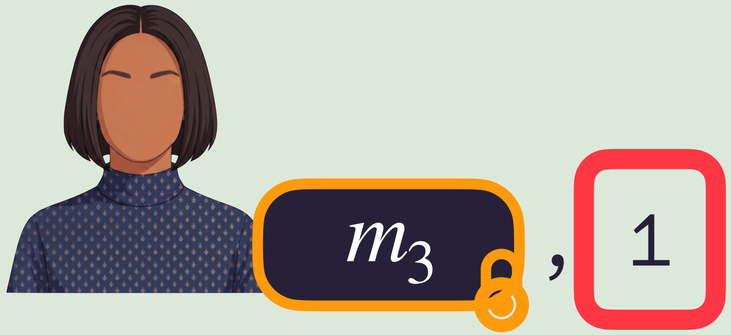
ℓ: max batch size = 3

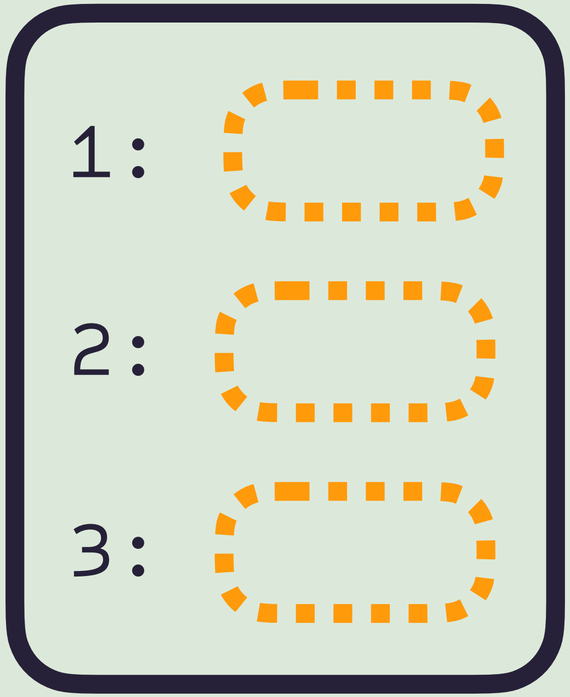












Index Dependence

ℓ: max batch size = 3

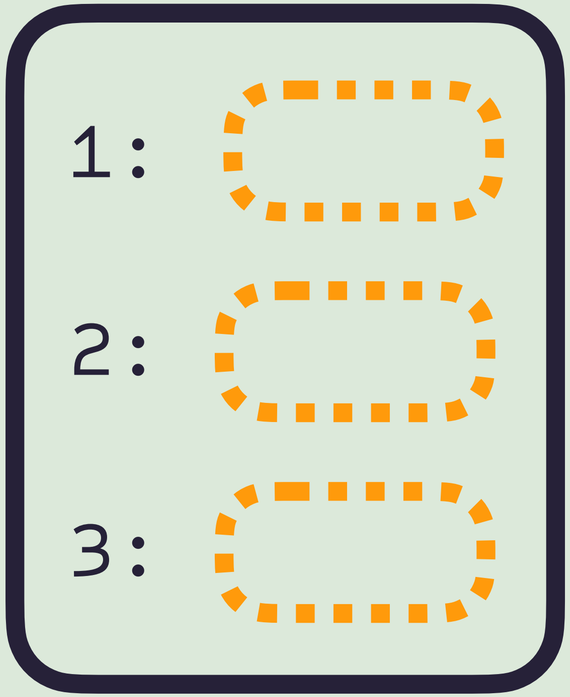
m_1 , 1

pk

sk

m_2 , 2

m_3 , 1



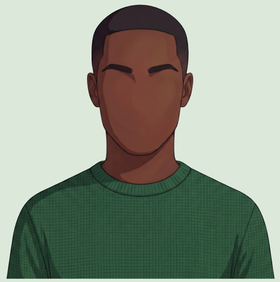
Index Dependence

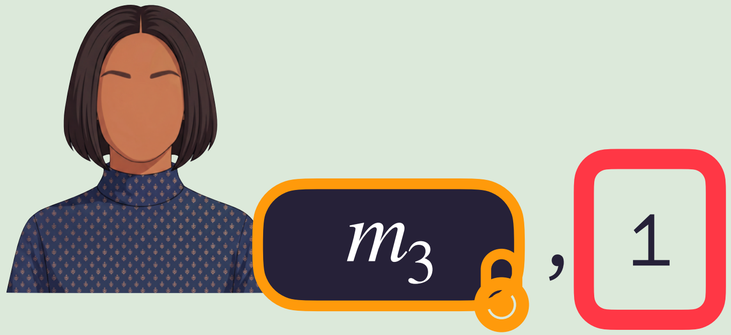
ℓ: max batch size = 3

m_1 , 1

pk

sk





m_3 , 1

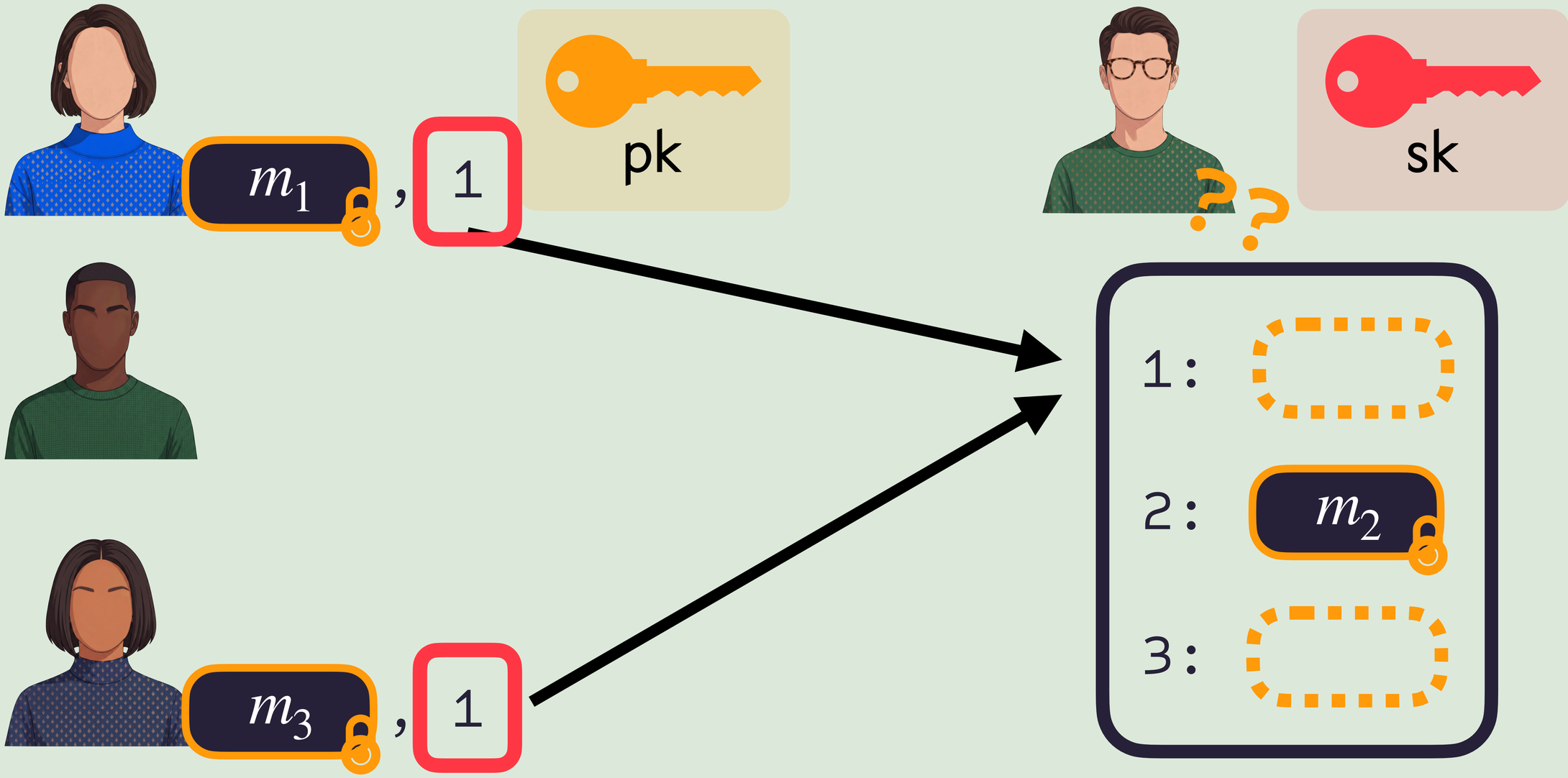
1:

2: m_2

3:

Index Dependence

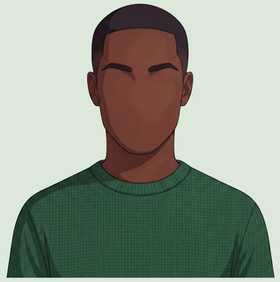
ℓ: max batch size = 3



Index Dependence

ℓ: max batch size = 3

m_1 , 1



m_3 , 1

pk

sk

only one of them can be included…

1:

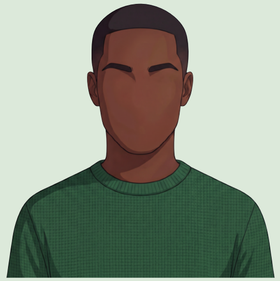
2: m_2

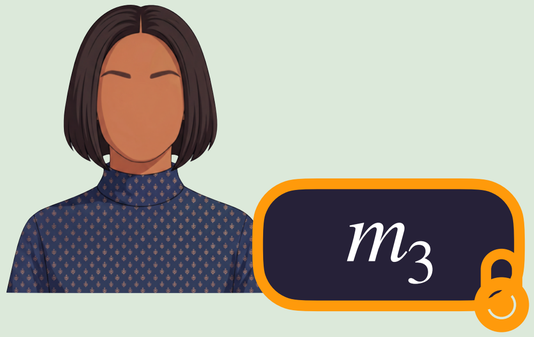
3:

Index Dependence

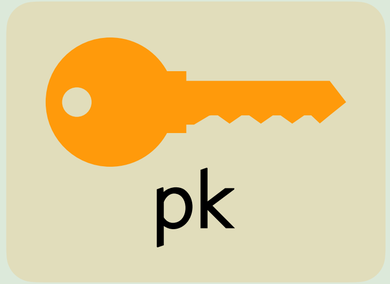
ℓ: max batch size = 3

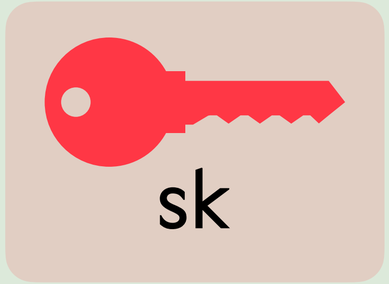
m_1 , 1

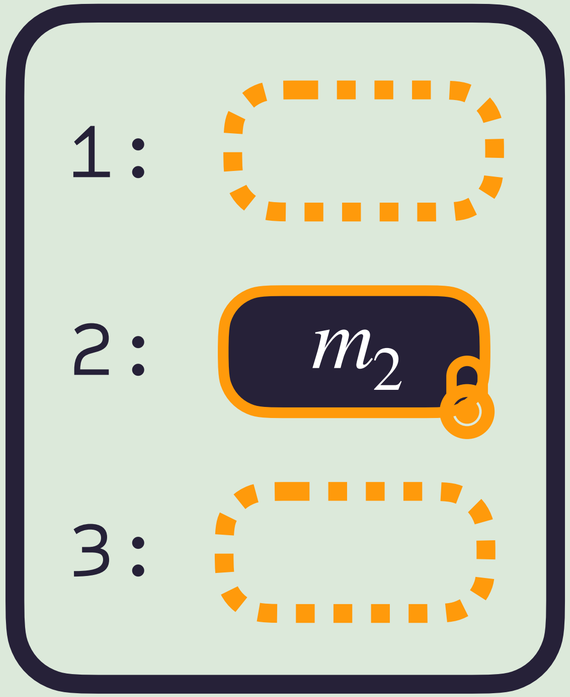




m_3 , 1

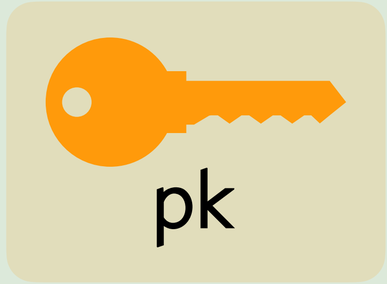


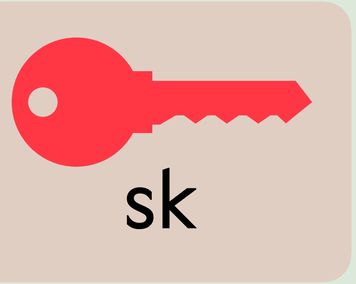


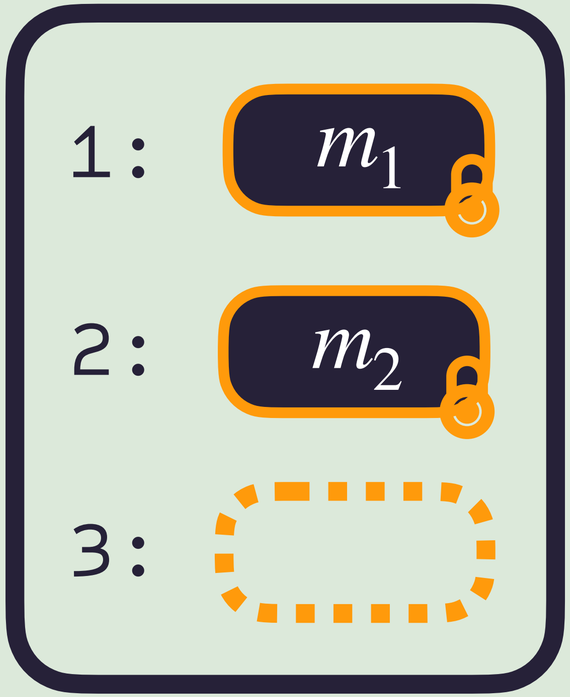


Index Dependence

ℓ: max batch size = 3

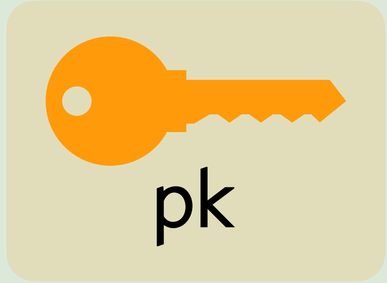


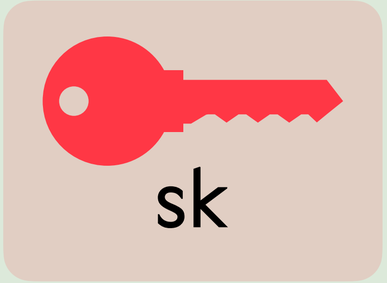




Index Dependence

ℓ: max batch size = 3





for the rest of this talk, we'll look at this weaker notion.

See the paper on how we achieve batch encryption without index dependence

Intuition

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:
$\frac{1}{X+j}$ $\forall j \in [\ell]$;

and *quadratic* fractions in $\mathbb{G}_T$:
$\frac{1}{(X+j)^2}$ $\forall j \in [\ell]$.

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction

on the diagonal times a random value:

$$m + \frac{r_i}{(X+i)^2}$$

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction

on the diagonal times a random value:

$$m + \frac{r_i}{(X+i)^2}$$

Pre-decrypting batch

The pre-decryptor computes the pre-decryption key:

$$\mathsf{sbk} = \sum_{j \in [\ell]} \frac{r_j}{X+j}$$

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction

on the diagonal times a random value:

$$m + \frac{r_i}{(X+i)^2}$$

Pre-decrypting batch

The pre-decry
computes the
decryption ke

$$\mathsf{sbk} = \sum_{j \in [\ell]} \frac{r_j}{X+j}$$

r_j **is used to tie ciphertext to index** j

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{X+i} \cdot \frac{1}{X+j}$	$\frac{1}{X+i} \cdot \frac{1}{X+k}$
$-j$	$\frac{1}{X+j} \cdot \frac{1}{X+i}$	$\frac{1}{(X+j)^2}$	$\frac{1}{X+j} \cdot \frac{1}{X+k}$
$-k$	$\frac{1}{X+k} \cdot \frac{1}{X+i}$	$\frac{1}{X+k} \cdot \frac{1}{X+j}$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction

on the diagonal times a random value:

$$m + \frac{r_i}{(X+i)^2}$$

Pre-decrypting batch

The pre-decry
computes the
decryption ke

$$\mathsf{sbk} = \sum_{j \in [\ell]} \frac{r_j}{X+j}$$

r_j **is used to tie ciphertext to index** j

Decryption for index $i \in [\ell]$

Compute $\mathsf{sbk} \cdot \left(\frac{1}{X+i}\right)$

$$= \frac{r_i}{(X+i)^2} + \sum_{j \neq i} \frac{r_i}{(X+j)} \cdot \frac{1}{(X+i)}$$

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$
$-j$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{(X+j)^2}$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$
$-k$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction *on the diagonal multiplied by a random value:*

$$m + \frac{r_i}{(X+i)^2}$$

Pre-decrypting batch

The pre-decryptor computes the pre-decryption key:

$$\mathsf{sbk} = \sum_{j \in [\ell]} \frac{r_j}{X+j}$$

Decryption for index $i \in [\ell]$

Compute $\mathsf{sbk} \cdot \left(\frac{1}{X+i}\right)$

$$= \frac{r_i}{(X+i)^2} + \sum_{j \neq i} \frac{r_i}{(X+j)} \cdot \frac{1}{(X+i)}$$

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$
$-j$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{(X+j)^2}$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$
$-k$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction *on the diagonal multiplied by a random value:*

$$m + \frac{r_i}{(X+i)^2}$$

Pre-decrypting batch

The pre-decryptor computes the pre-decryption key:

$$\mathsf{sbk} = \sum_{j \in [\ell]} \frac{r_j}{X+j}$$

Decryption for index $i \in [\ell]$

Compute $\mathsf{sbk} \cdot \left(\frac{1}{X+i}\right)$

$$= \frac{r_i}{(X+i)^2} + \sum_{j \neq i} \frac{r_i}{(X+j)} \cdot \frac{1}{(X+i)}$$

off-diagonal products linearize and can be subtracted out

Intuition

All parties hold *linear* fractions in $\mathbb{G}_2$:

$$\frac{1}{X+j} \quad \forall j \in [\ell];$$

and *quadratic* fractions in $\mathbb{G}_T$:

$$\frac{1}{(X+j)^2} \quad \forall j \in [\ell].$$

Poles	$-i$	$-j$	$-k$
$-i$	$\frac{1}{(X+i)^2}$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$
$-j$	$\frac{1}{j-i}\left(\frac{1}{X+i}-\frac{1}{X+j}\right)$	$\frac{1}{(X+j)^2}$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$
$-k$	$\frac{1}{k-i}\left(\frac{1}{X+i}-\frac{1}{X+k}\right)$	$\frac{1}{k-j}\left(\frac{1}{X+k}-\frac{1}{X+j}\right)$	$\frac{1}{(X+k)^2}$

Ciphertext for index $i \in [\ell]$

One-time pad* message using *quadratic fraction *on the diagonal multiplied by a random value:*

$$m + \frac{r_i}{(X+i)^2}$$

Pre-decrypting batch

The pre-decryptor computes the pre-decryption key:

$$\mathsf{sbk} = \sum_{j \in [\ell]} \frac{r_j}{X+j}$$

Decryption for index $i \in [\ell]$

Compute $\mathsf{sbk} \cdot \left(\frac{1}{X+i}\right)$

$$= \frac{r_i}{(X+i)^2} + \sum_{j \neq i} \frac{r_j}{(X+j)} \cdot \frac{1}{(X+i)}$$

and then you're just left with the one-time pad!

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e\right)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2,$ and $[z]_T = z \cdot e(g_1, g_2)$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

Setup(1^λ, batch size 1^ℓ)

Sample $x \xleftarrow{\$} \mathbb{Z}_p$, and output

$\mathsf{sk} := x$, and $\mathsf{pk} :=$

(1) linear fractions: $\left[\frac{1}{x+1}\right]_2, \ldots, \left[\frac{1}{x+\ell}\right]_2$, and

(2) quadratic fractions: $\left[\frac{1}{(x+1)^2}\right]_T, \ldots, \left[\frac{1}{(x+\ell)^2}\right]_T$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

Setup(1^λ, batch size 1^ℓ)

Sample $x \xleftarrow{\$} \mathbb{Z}_p$, and output

$\mathsf{sk} := x$, and $\mathsf{pk} :=$

(1) linear fractions: $\left[\frac{1}{x+1}\right]_2, \ldots, \left[\frac{1}{x+\ell}\right]_2$, and

(2) quadratic fractions: $\left[\frac{1}{(x+1)^2}\right]_T, \ldots, \left[\frac{1}{(x+\ell)^2}\right]_T$

Enc(pk, index $i \in [\ell]$, msg m)

Sample $r \xleftarrow{\$} \mathbb{Z}_p$, and output

ciphertext with two components

(1) $[r]_1$ (2) $r\left[\frac{1}{(x+i)^2}\right]_T + m$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

Setup(1^λ, batch size 1^ℓ)

Sample $x \xleftarrow{\$} \mathbb{Z}_p$, and output

$\mathsf{sk} := x$, and $\mathsf{pk} :=$

(1) linear fractions: $\left[\frac{1}{x+1}\right]_2, \ldots, \left[\frac{1}{x+\ell}\right]_2$, and

(2) quadratic fractions: $\left[\frac{1}{(x+1)^2}\right]_T, \ldots, \left[\frac{1}{(x+\ell)^2}\right]_T$

Enc(pk, index $i \in [\ell]$, msg m)

Sample $r \xleftarrow{\$} \mathbb{Z}_p$, and output

ciphertext with two components

(1) $[r]_1$ (2) $r\left[\frac{1}{(x+i)^2}\right]_T + m$

(3) NIZK proof of randomness

$\Pi.\mathsf{Prove}(x = (\mathsf{ct}[1], \mathsf{ct}[2]), w = r)$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

Setup(1^λ, batch size 1^ℓ)

Sample $x \xleftarrow{\$} \mathbb{Z}_p$, and output

$\mathsf{sk} := x$, and $\mathsf{pk} :=$

(1) linear fractions: $\left[\frac{1}{x+1}\right]_2, \ldots, \left[\frac{1}{x+\ell}\right]_2$, and

(2) quadratic fractions: $\left[\frac{1}{(x+1)^2}\right]_T, \ldots, \left[\frac{1}{(x+\ell)^2}\right]_T$

Enc(pk, index $i \in [\ell]$, msg m)

Sample $r \xleftarrow{\$} \mathbb{Z}_p$, and output

ciphertext with two components

(1) $[r]_1$ (2) $r\left[\frac{1}{(x+i)^2}\right] + m$

(3) NIZK proof of rando

$\Pi.\mathsf{Prove}(x = (\mathsf{ct}[1], \mathsf{ct}[2]), w = r)$

needed for CCA-2 security

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

$$\mathsf{PreDec}(\mathsf{sk}, \{\mathsf{ct}_i\}_i)$$

Check NIZK proofs, and if valid, **output pre-decryption key as sum of partial fractions:**

$$\mathsf{sbk} := \sum_{i \in [\ell]} \frac{1}{x+i} \cdot \mathsf{ct}_i[1] = \sum_{i \in [\ell]} \left[\frac{r_i}{x+i} \right]_1$$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2,$ and $[z]_T = z \cdot e(g_1, g_2)$

$\mathsf{PreDec}(\mathsf{sk}, \{\mathsf{ct}_i\}_i)$

Check NIZK proofs, and if valid, **output pre-decryption key as sum of partial fractions:**

$$\mathsf{sbk} := \sum_{i \in [\ell]} \frac{1}{x+i} \cdot \mathsf{ct}_i[1] = \sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1$$

$\mathsf{Dec}(\mathsf{pk}, \mathsf{sbk}, \{\mathsf{ct}_i\}_i)$

To recover m_j, compute:

$$e\left(\mathsf{sbk}, \left[\frac{1}{x+j}\right]_2\right) = e\left(\sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1, \left[\frac{1}{x+j}\right]_2\right)$$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

$\mathsf{PreDec}(\mathsf{sk}, \{\mathsf{ct}_i\}_i)$

Check NIZK proofs, and if valid, **output pre-decryption key as sum of partial fractions:**

$$\mathsf{sbk} := \sum_{i \in [\ell]} \frac{1}{x+i} \cdot \mathsf{ct}_i[1] = \sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1$$

$\mathsf{Dec}(\mathsf{pk}, \mathsf{sbk}, \{\mathsf{ct}_i\}_i)$

To recover m_j, compute:

$$e\left(\mathsf{sbk}, \left[\frac{1}{x+j}\right]_2\right) = e\left(\sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1, \left[\frac{1}{x+j}\right]_2\right)$$

$$= r_j \left[\frac{1}{(x+j)^2}\right]_T + r_j \sum_{i \neq j} \left[\frac{1}{(x+i)} \cdot \frac{1}{(x+j)}\right]_T$$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

$\mathsf{PreDec}(\mathsf{sk}, \{\mathsf{ct}_i\}_i)$

Check NIZK proofs, and if valid, **output pre-decryption key as sum of partial fractions:**

$$\mathsf{sbk} := \sum_{i \in [\ell]} \frac{1}{x+i} \cdot \mathsf{ct}_i[1] = \sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1$$

$\mathsf{Dec}(\mathsf{pk}, \mathsf{sbk}, \{\mathsf{ct}_i\}_i)$

To recover m_j, compute:

$$e\left(\mathsf{sbk}, \left[\frac{1}{x+j}\right]_2\right) = e\left(\sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1, \left[\frac{1}{x+j}\right]_2\right)$$

$$= r_j \left[\frac{1}{(x+j)^2}\right]_T + r_j \sum_{i \neq j} \left[\frac{1}{(x+i)} \cdot \frac{1}{(x+j)}\right]_T$$

$$= r_j \left[\frac{1}{(x+j)^2}\right]_T + \sum_{i \neq j} \frac{1}{j-i} \cdot e\left([r_j]_1, \left[\frac{1}{(x+i)}\right]_2 - \left[\frac{1}{(x+j)}\right]_2\right)$$

The Index-Dependent Construction

ℓ: **max block size**

Recall (asymmetric) bilinear group: $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$;

additive notation: $[x]_1 = x \cdot g_1, [y]_2 = y \cdot g_2$, and $[z]_T = z \cdot e(g_1, g_2)$

$\mathsf{PreDec}(\mathsf{sk}, \{\mathsf{ct}_i\}_i)$

Check NIZK proofs, and if valid, **output pre-decryption key as sum of partial fractions:**

$$\mathsf{sbk} := \sum_{i \in [\ell]} \frac{1}{x+i} \cdot \mathsf{ct}_i[1] = \sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1$$

$\mathsf{Dec}(\mathsf{pk}, \mathsf{sbk}, \{\mathsf{ct}_i\}_i)$

To recover m_j, compute:

$$e\left(\mathsf{sbk}, \left[\frac{1}{x+j}\right]_2\right) = e\left(\sum_{i \in [\ell]} \left[\frac{r_i}{x+i}\right]_1, \left[\frac{1}{x+j}\right]_2\right)$$

$$= r_j \left[\frac{1}{(x+j)^2}\right]_T + r_j \sum_{i \neq j} \left[\frac{1}{(x+i)} \cdot \frac{1}{(x+j)}\right]_T$$

$$= r_j \left[\frac{1}{(x+j)^2}\right]_T + \sum_{i \neq j} \frac{1}{j-i} \cdot e\left([r_j]_1, \left[\frac{1}{(x+i)}\right]_2 - \left[\frac{1}{(x+j)}\right]_2\right)$$

subtract out linear terms

Prior Work Limitations

Prior Work Limitations

Ciphertext is tied to an index

Users must **encrypt to an index/position** in the block/batch. Thus, they **need to coordinate** and know their **position ahead of time**.

The ciphertext is *only* for the fixed index.

“Index-dependent” Batch Encryption

Prior Work Limitations

Ciphertext is tied to an index

Users must **encrypt to an index/position** in the block/batch. Thus, they **need to coordinate** and know their **position ahead of time**.

The ciphertext is *only* for the fixed index.

“Index-dependent” Batch Encryption

Ciphertext is tied to an epoch

Ciphertext is no longer tied to a position but is **only valid for the current batch** identified by an epoch.

If it’s not included, it **needs to be re-encrypted**.

“Epoch-based”

Prior Work Limitations

Ciphertext is tied to an index

Users must **encrypt to an index/position** in the block/batch. Thus, they **need to coordinate** and know their **position ahead of time**.

The ciphertext is *only* for the fixed index.

"Index-dependent" Batch Encryption

Ciphertext is tied to an epoch

Ciphertext is no longer tied to a position but is **only valid for the current batch** identified by an epoch.

If it's not included, it **needs to be re-encrypted**.

"Epoch-based"

Large CRS

Index-independent and *epoch-less* schemes suffer from a large exponential-size CRS

Our Work

Our Work

Best of all worlds!

We achieve batch encryption that is

1. **Independent of index:** no coordination required
2. **Epochless:** pending messages don’t need to be re-encrypted
3. **Much shorter practical CRS:** linear in batch size
4. **Shortest ciphertext:** only 2 group elements!

Our Work

(Almost) Best of all worlds!

We achieve batch encryption that is

1. **Independent of index:** no coordination required
2. **Epochless:** pending messages don’t need to be re-encrypted
3. **Much shorter practical CRS:** linear in batch size
4. **Shortest ciphertext:** only 2 group elements!

Secret key size is $\mathcal{O}(\ell)$:
linear in the maximum size of the batch

Our Work

(Almost) Best of all worlds!

We achieve batch encryption that is

1. **Independent of index:** no coordination required
2. **Epochless:** pending messages don’t need to be re-encrypted
3. **Much shorter practical CRS:** linear in batch size
4. **Shortest ciphertext:** only 2 group elements!

Secret key size is $\mathcal{O}(\ell)$:

linear in the maximum size of the batch

Shortly after our eprint was out, two subsequent works

[Pol26, ADG+26]

achieved the same using a different technique

Implementation

Implementation

Prototypical implementation (not very optimized) using arkworks:
BLS12-381, single core Apple M4 Pro, 24 GB memory

Implementation

Prototypical implementation (not very optimized) using arkworks:
BLS12-381, single core Apple M4 Pro, 24 GB memory

BE: Index-independent batch encryption with **3 group elements**

Implementation

Prototypical implementation (not very optimized) using arkworks:
BLS12-381, single core Apple M4 Pro, 24 GB memory

BE: Index-independent batch encryption with **3 group elements**

$\mathsf{BE}_{\mathrm{short}}$: Index-independent batch encryption with **2 group elements**

Implementation

Prototypical implementation (not very optimized) using arkworks:
BLS12-381, single core Apple M4 Pro, 24 GB memory

BE: Index-independent batch encryption with **3 group elements**

$\mathsf{BE}_{\mathrm{short}}$: Index-independent batch encryption with **2 group elements**

Batch size ℓ	Enc		PreDec		Dec	
	BE	$\mathsf{BE}_{\mathrm{short}}$	BE	$\mathsf{BE}_{\mathrm{short}}$	BE	$\mathsf{BE}_{\mathrm{short}}$
4	599 µs	437 µs	1.53 ms	0.82 ms	19.2 ms	15.3 ms
16	599 µs	437 µs	2.83 ms	1.93 ms	101 ms	76.3 ms
64	599 µs	437 µs	8.12 ms	6.93 ms	501 ms	364 ms
128	599 µs	437 µs	14.5 ms	13.2 ms	1.09 s	787 ms
256	599 µs	437 µs	27.9 ms	25.6 ms	2.38 s	1.69 s
512	599 µs	437 µs	54.3 ms	50.1 ms	5.12 s	3.61 s

Takeaways

Partial Fraction Techniques

Allows you to **linearize cross-terms:** $\frac{1}{X+i} \cdot \frac{1}{X+j} = \frac{1}{j-i}\left(\frac{1}{X+i} - \frac{1}{X+j}\right),$

and **mask values with higher-order poles:** $\frac{1}{(X+k)^2}$

introduces another interesting property using Cauchy matrices

Part 1
Partial Fraction Techniques for Cryptography (2025/2081)

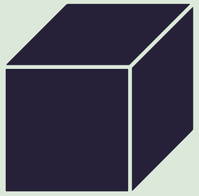
this paper

Part 2
Efficient Batch Threshold Encryption Using Partial Fraction Techniques (2026/674)

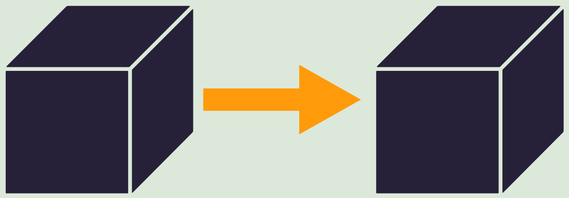
Appendix

Application: *Encrypted Mempools* for Blockchain

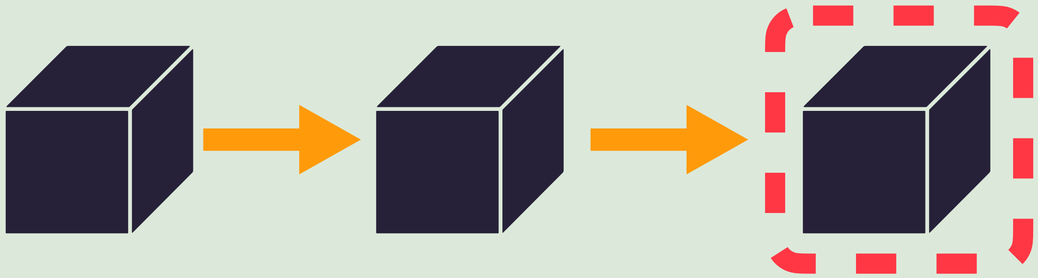
Application: *Encrypted Mempools* for Blockchain



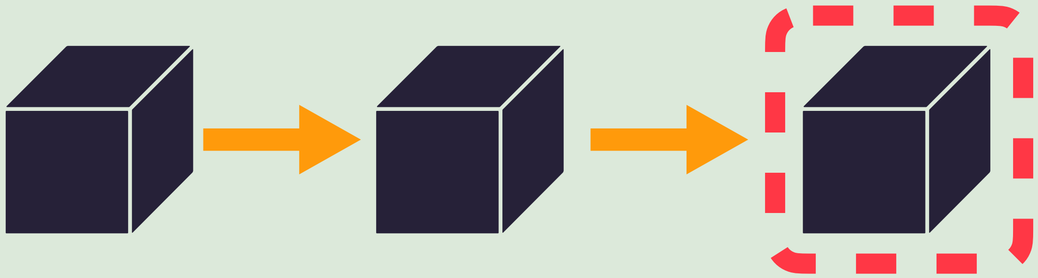
Application: *Encrypted Mempools* for Blockchain



Application: *Encrypted Mempools* for Blockchain



Application: *Encrypted Mempools* for Blockchain



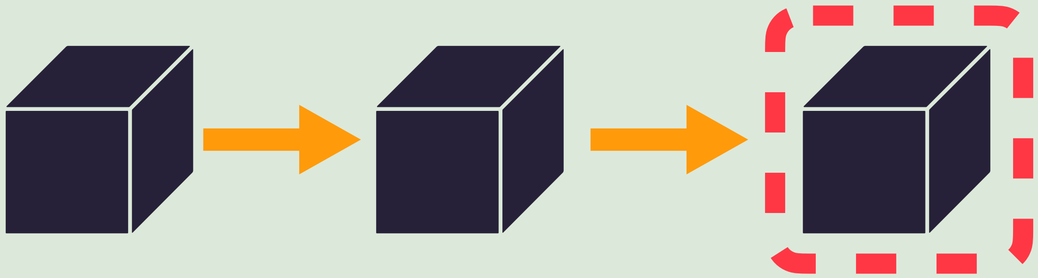
users with transactions

tx_1

tx_2

tx_3 tx_4

Application: *Encrypted Mempools* for Blockchain



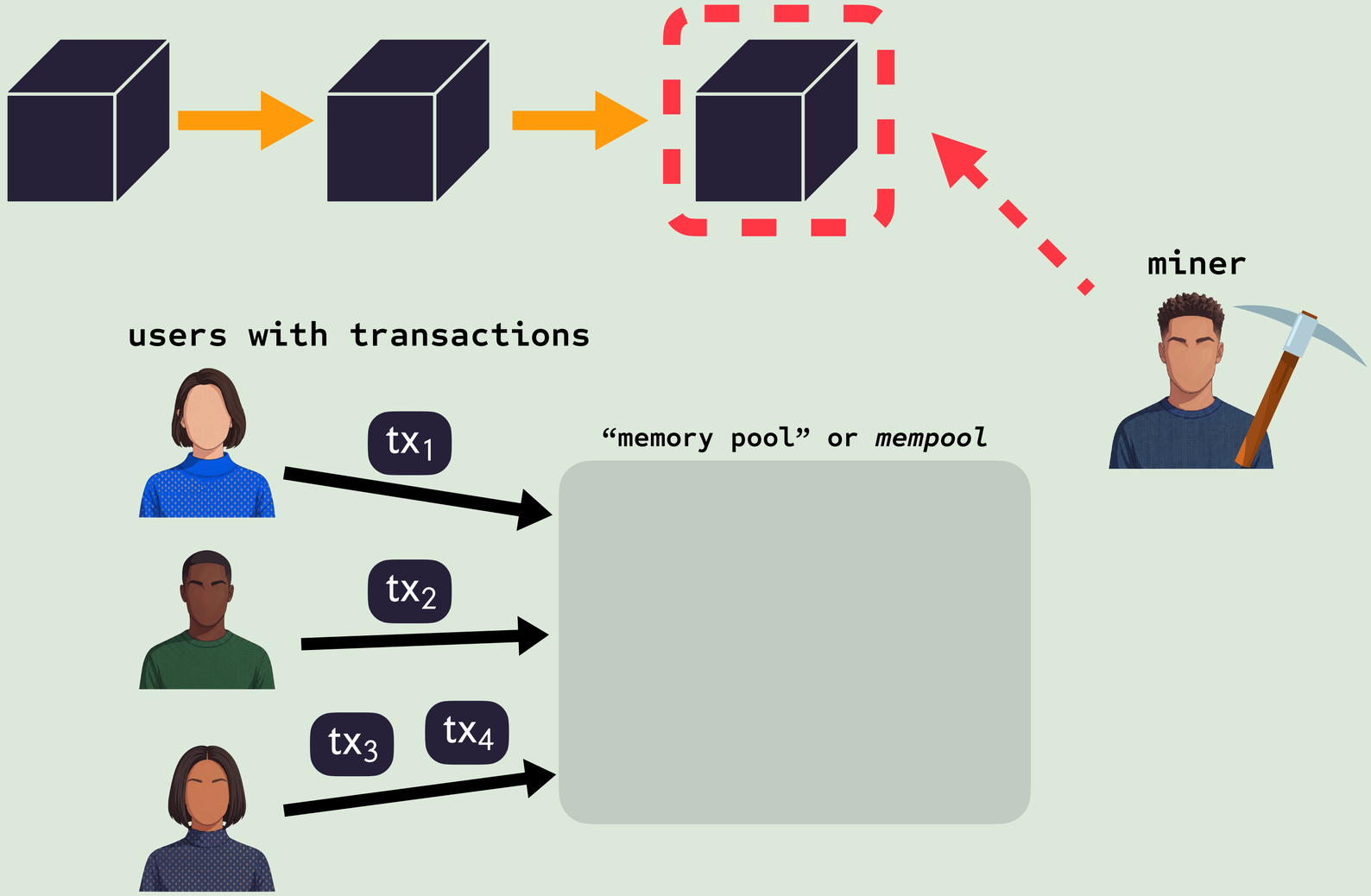
users with transactions

tx_1

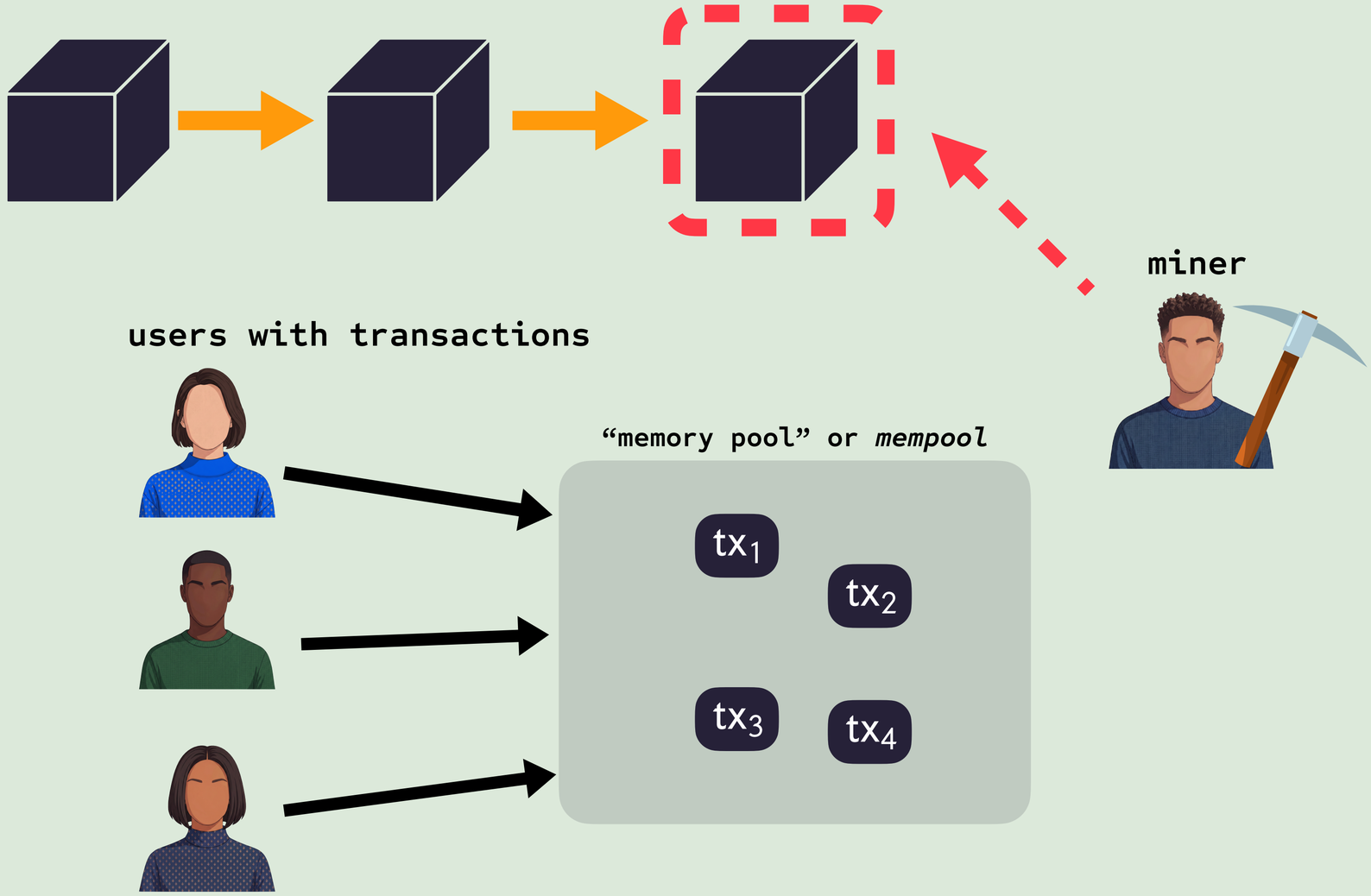
tx_2

tx_3 tx_4

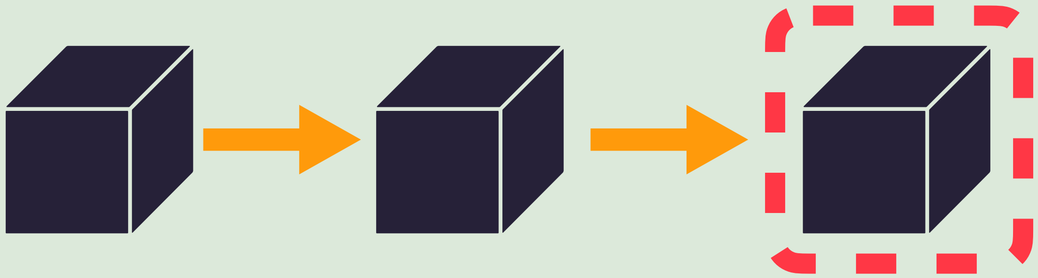
Application: *Encrypted Mempools* for Blockchain



Application: *Encrypted Mempools* for Blockchain

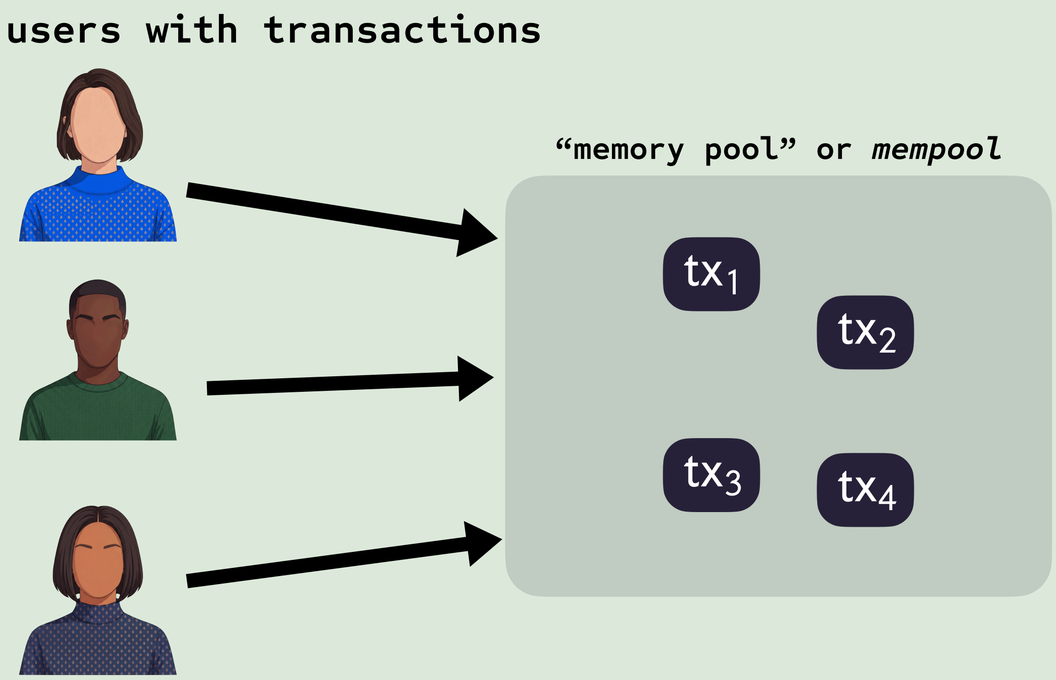


Application: *Encrypted Mempools* for Blockchain



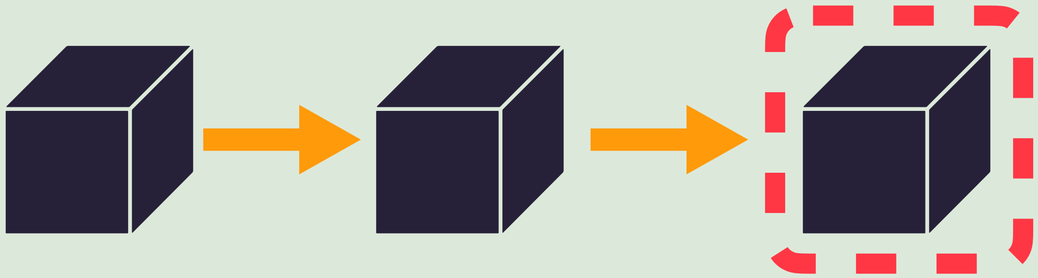
Transactions on block are executed sequentially.

Order matters!



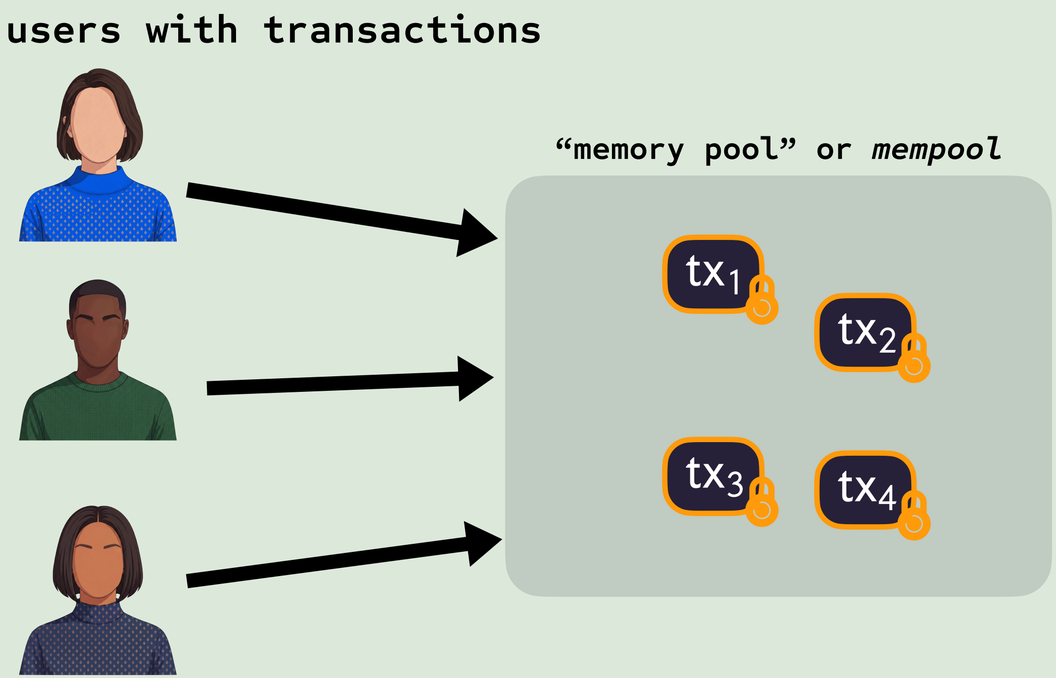
Block proposer can order/ pick in a way that’s beneficial to them (MEV attacks e.g. front-running)

Application: *Encrypted Mempools* for Blockchain



Transactions on block are executed sequentially.

Order matters!

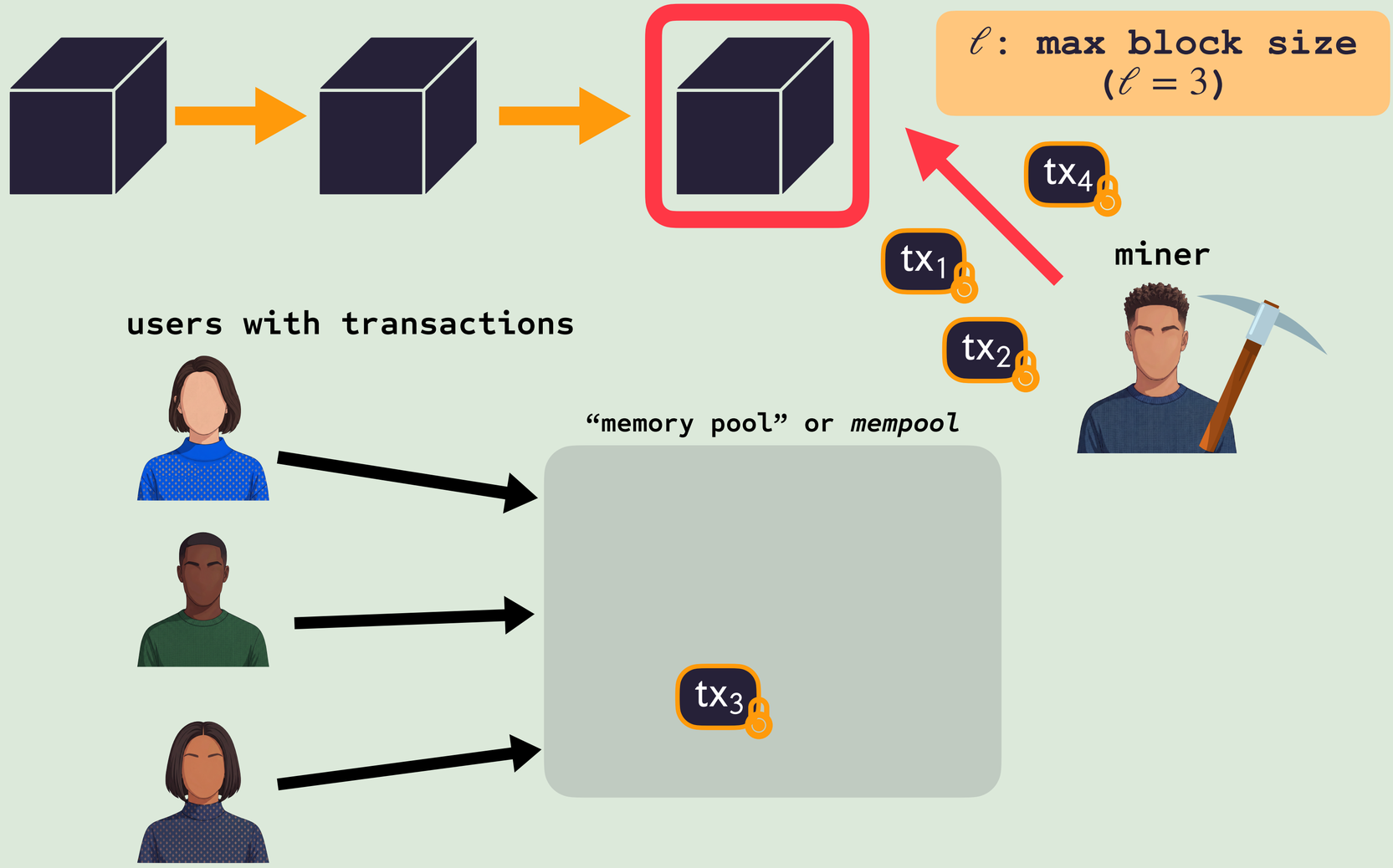


Block proposer can order/ pick in a way that’s beneficial to them (MEV attacks e.g. front-running)

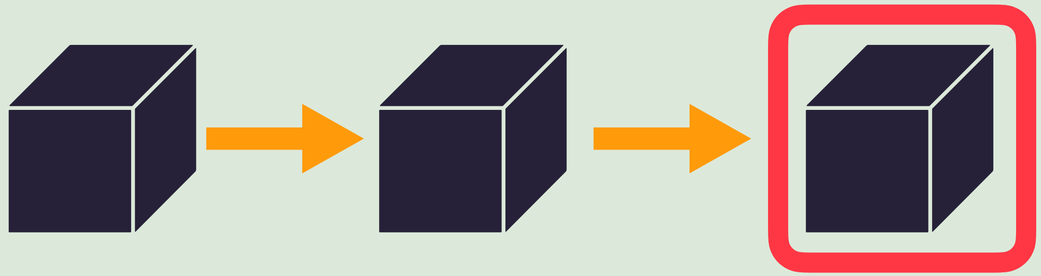
Can we hide them?

Yes, we could encrypt…

Application: *Encrypted Mempools* for Blockchain



Application: *Encrypted Mempools* for Blockchain



ℓ: **max block size**
($\ell = 3$)

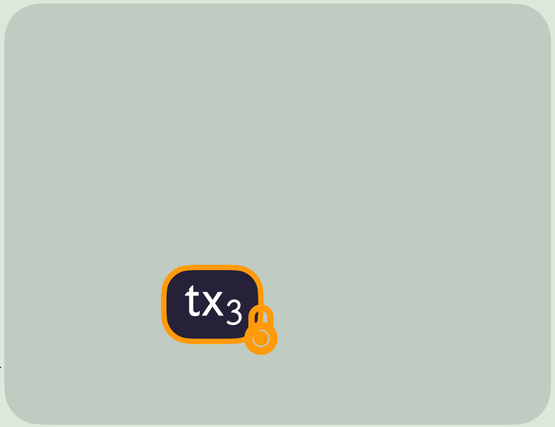
miner

Block is confirmed.

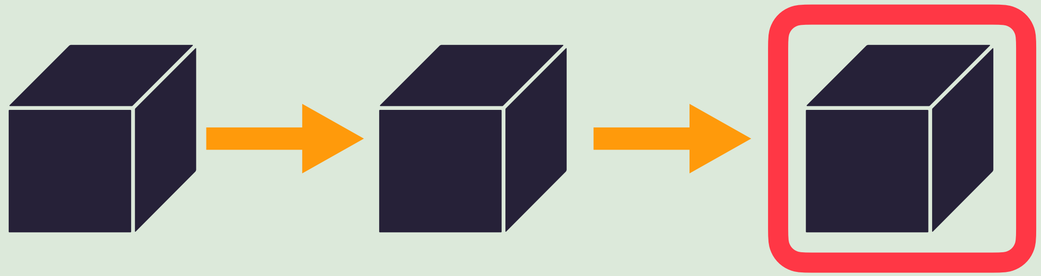
How to decrypt?

users with transactions

“memory pool” or *mempool*



Application: *Encrypted Mempools* for Blockchain



ℓ: **max block size**
($\ell = 3$)

miner

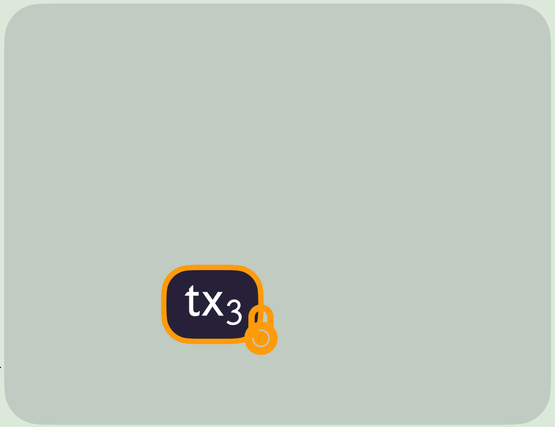
Block is confirmed.

How to decrypt?

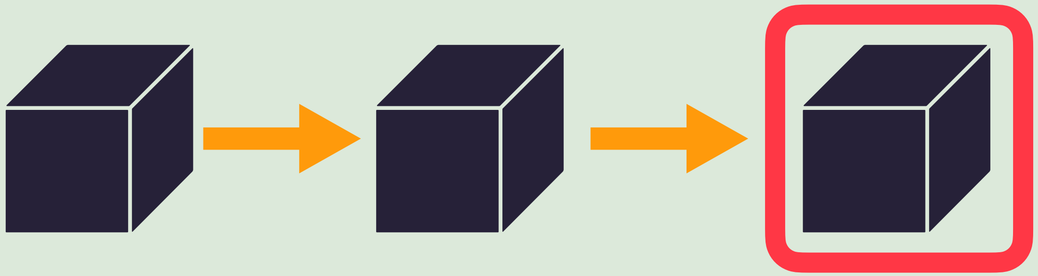
What about transactions that were **not included**?

users with transactions

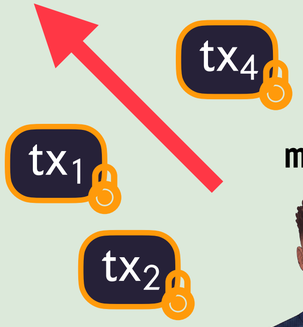
“memory pool” or *mempool*



Application: *Encrypted Mempools* for Blockchain



ℓ: **max block size** ($\ell = 3$)



miner

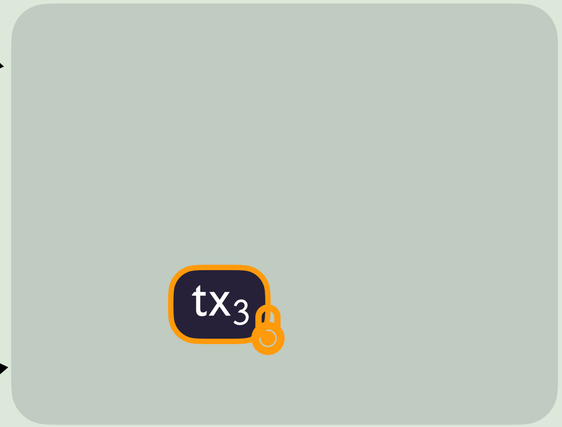
Block is confirmed.

How to decrypt?

What about transactions that were **not included**?

users with transactions

“memory pool” or *mempool*



Batch Encryption enables **efficient decryption of *only* included transactions**: tx_1, tx_2, tx_4

while keeping **excluded transactions** (tx_3) **secret** (semantically secure)