New Algorithms and Analyses for Sum-Preserving Encryption

Sarah Miracle and Scott Yilek

University of St. Thomas

Sum-Preserving Encryption

<u>Sum-Preserving Encryption schemes:</u>

Encryption schemes in which ciphertexts and plaintexts are both integer vectors with the same sum.

plaintext	ciphertext
[0, 10, 20, 5, 50]	[15, 7, 8, 1, 54]
[2, 10]	[5, 7]
[1, 3, 7, 2, 8, 20, 9]	[10, 3, 5, 8, 1, 2, 21
[5, 3, 1]	[2, 4, 3]

plaintext → ??? → ciphertext

Vector components are typically bounded between 0 and d.

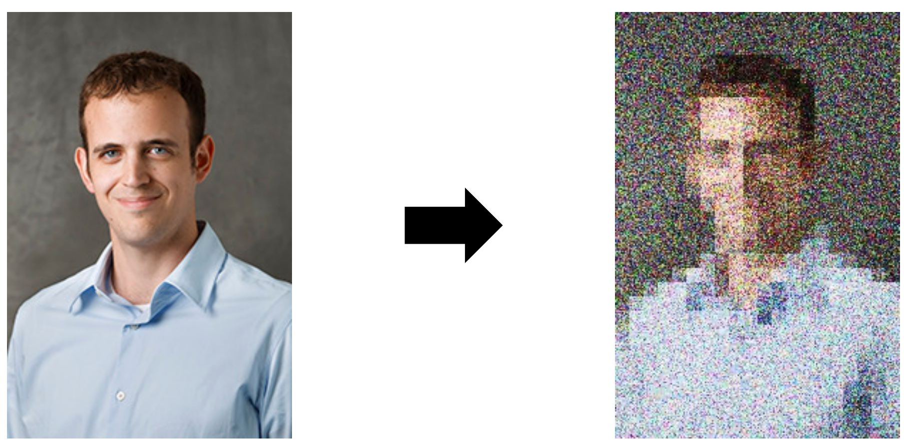
Introduced by Tajik et al. [NDSS 2019]

Sum-Preserving Encryption

Application: Thumbnail-preserving encryption

Image encryption where the thumbnail of an encrypted image matches the thumbnail of the unencrypted image.

- Divide the image into *b* x *b* blocks of pixels
- Apply sum-preserving encryption to each block with component bound 255.

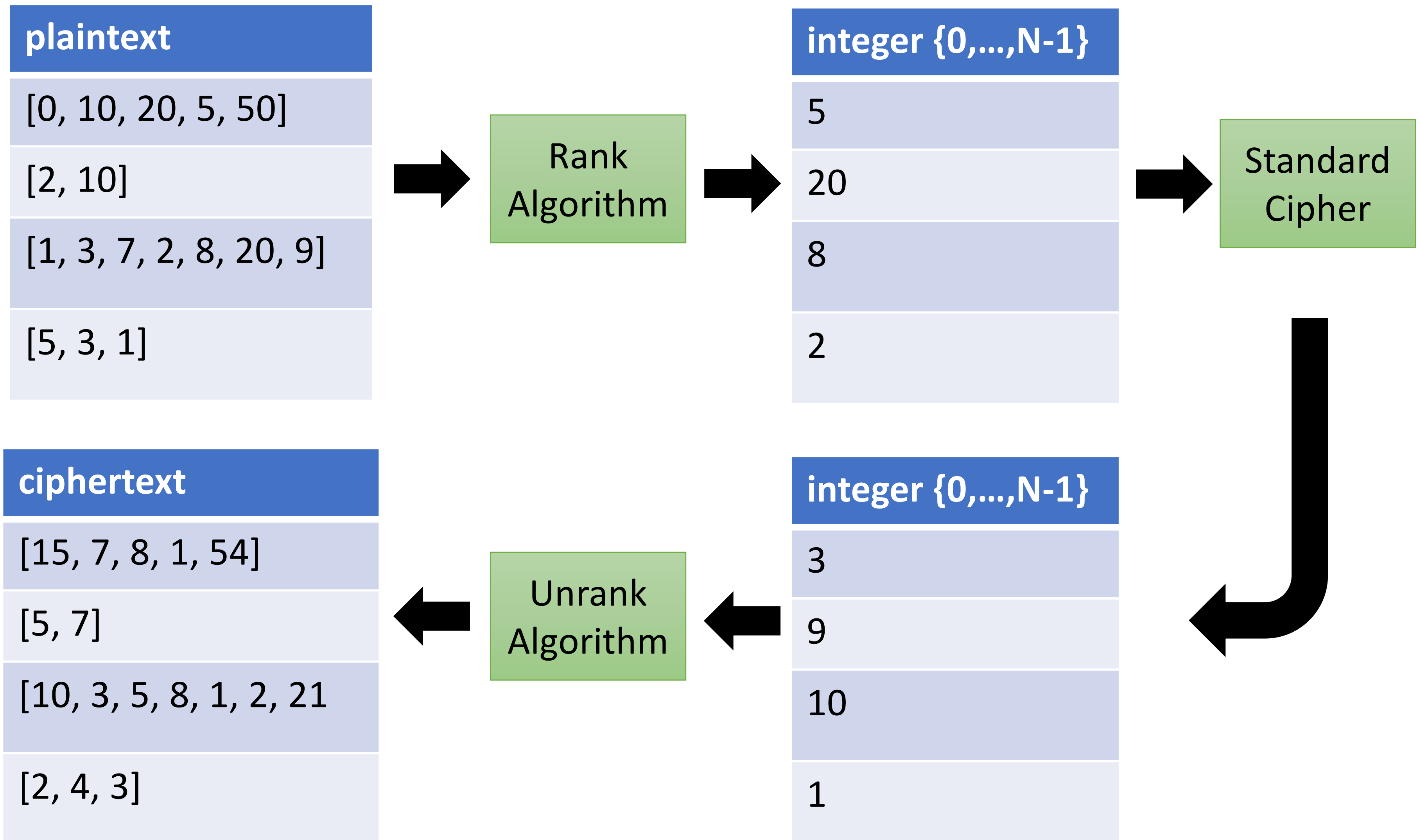


Background

A special type of **format-preserving encryption**.

- Originally studied by Brightwell and Smith [‘97]
- Formally defined and analyzed by Bellare, Ristenpart, Rogaway, and Stegers [‘09]
  (use a rank-encipher-unrank construction)
- Widely studied and even standardized
- Tajik, Gunasekaran, Dutta, Ellis, Bobba, Rosulek, Wright, Feng, focus on problem of creating a thumbnail encryption scheme [‘19]

Rank-Encipher-Unrank



Rank-Encipher-Unrank

Idea [Tajik et al.]:

[5, 3, 1] ➡ 11111011101

- Represent as a string of 1's and 0's
- The number of 1's is the same as the sum and 0's act as separators
- This is a regular language and thus we can use known techniques for ranking DFAs [see Bellare et al. '09]

However, high time complexity and impractical!

Rank-Encipher-Unrank

Idea [Tajik et al.]:

[5, 3, 1] ➡ 11111011101

- Represent as a string of 1's and 0's
- The number of 1's is the same as the sum and 0's act as separators
- This is a regular language and thus we can use known techniques for ranking DFAs [see Bellare et al. '09]

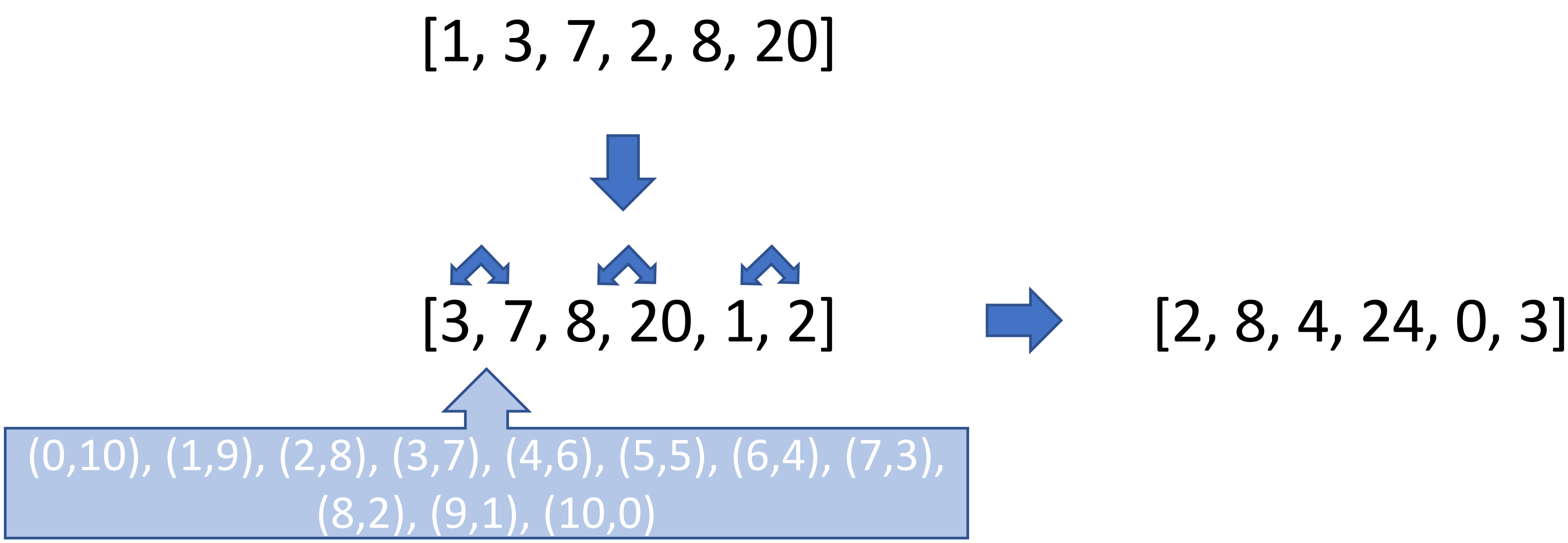
Ranking vectors of length 2 is simple & efficient!

Tajik et al. Construction

Sum-Preserving Shuffle Markov chain:

Repeat:

1. Choose a random shuffling on all points uniformly at random.
2. Pair adjacent points to create a perfect matching.
3. Independently for each matched pair select a pair of values u.a.r. from all valid choices that preserve the sum.



Our Results

1. First proof bounding the mixing time of the Tajik et al. algorithm
2. Give practical rank and unrank algorithms for sum-preserving encryption
3. Create prototype implementations with performance comparisons

Tajik et al. Construction

<u>Sum-Preserving Shuffle Markov chain:</u>

Repeat:

1. Choose a random shuffling on all points uniformly at random.
2. Pair adjacent points to create a perfect matching.
3. Independently for each matched pair select a pair of values u.a.r. from all valid choices that preserve the sum.

How many times do you need to repeat?

Tajik et al. Construction

<u>Sum-Preserving Shuffle Markov chain:</u>

Repeat:

1. Choose a random shuffling on all points uniformly at random.
2. Pair adjacent points to create a perfect matching.
3. Independently for each matched pair select a pair of values u.a.r. from all valid choices that preserve the sum.

- Tajik et al. give heuristic arguments for what secure round choices might be but no proof.
- Test performance with 1000, 3000, and 5000 rounds.

Mixing Time

Definition: The total variation distance is

$$\| P^t, \pi \| = \max_{x \in \Omega} \tfrac{1}{2} \sum_{y \in \Omega} |P^t(x,y) - \pi(y)|.$$

Definition: Given ε, the mixing time is

$$\tau(\varepsilon) = \min \{t: \|P^{t'},\pi\| < \varepsilon, \quad \forall t' \geq t\}.$$

Mixing Time Bound

<u>Sum-Preserving Shuffle Markov chain:</u>

Repeat:

1. Choose a random shuffling on all points uniformly at random.
2. Pair adjacent points to create a perfect matching.
3. Independently for each matched pair select a pair of values u.a.r. from all valid choices that preserve the sum.

Let *n* be the vector length, *d* the component bound, and *S* the fixed sum. We show the mixing time satisfies

$$\tau(\epsilon) \leq n \ln(\min(dn, 2S)\epsilon^{-1})$$

Proof Idea

Let *n* be the vector length, *d* the component bound, and *S* the fixed sum. We show the mixing time satisfies

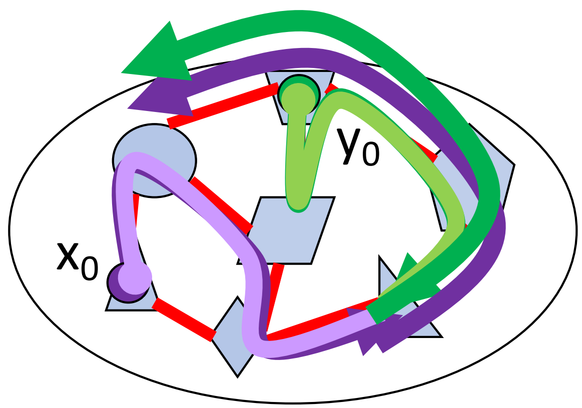
$$\tau(\epsilon) \leq n \ln(\min(dn, 2S)\epsilon^{-1})$$

Use path coupling due to Dyer, Greenhill ['98]

Coupling

Simulate 2 processes:

- Start at any x_0 and y_0
- Couple moves, but each simulates the MC
- Once they agree, they move in sync ($x_t = y_t \rightarrow x_{t+1} = y_{t+1}$)



Expected Coupling Time > Mixing time

Prove chains getting closer in expectation in each step

Manhattan distance

Path Coupling

- **Coupling**: Show for all states x,y,
  $E[\, \Delta\, (dist(x,y))\,] < 0$.

- **Path coupling**: Show for all u,v s.t. u,v differ by 2 points, that $E[\, \Delta\, (dist(u,v))\,] < 0$.

Consider a shortest path:

$x = z_0,\ z_1,\ z_2, \ldots, z_r = y$, z_i, z_{i+1} differ by 2 points.

$$E[\, \Delta\, (dist(x,y))\,] \le \Sigma_i\, E[\, \Delta\, (dist(z_i, z_{i+1})) \le 0.$$

Path Coupling

Consider 2 configuration that differ on exactly 2 points:

$$x = [1, 2, 3, 8, 10, 7, 4, 1, 0]$$
$$y = [1, 2, 5, 8, 10, 5, 4, 1, 0]$$

- If these 2 points are paired together the distance **decreases** to 0.
- Otherwise we show the distances **stays the same**.

Rank-Encipher-Unrank

Our Approach:

- Rank vectors directly using lexicographical order.
- Build on algorithms for unranking developed by Stein ['20] for use in random sampling.
- Uses a dynamic programming approach and pre-computes a table C_d where $C_d(n,S)$ stores the number of vectors of length n with sum S and component bound d (we instead store the cumulative sum)

Configuration	Rank	Configuration	Rank
(0,3,3)	0	(2,3,1)	5
(1,2,3)	1	(3,0,3)	6
(1,3,2)	2	(3,1,2)	7
(2,1,3)	3	(3,2,1)	8
(2,2,2)	4	(3,3,0)	9

Component bound = 3

Recursive Block Order

To order x and y, divide each in half and compute the sum of each half.

$$
\begin{aligned}
S_{x_L} &< S_{y_L} && \implies x <_B y \\
S_{x_L} &> S_{y_L} && \implies x >_B y \\
S_{x_L} &= S_{y_L} \text{ and } x_L <_B y_L && \implies x <_B y \\
S_{x_L} &= S_{y_L} \text{ and } x_L >_B y_L && \implies x >_B y \\
S_{x_L} &= S_{y_L} \text{ and } x_L = y_L \text{ and } x_R <_B x_L && \implies x <_B y \\
S_{x_L} &= S_{y_L} \text{ and } x_L = y_L \text{ and } x_R >_B x_L && \implies x >_B y
\end{aligned}
$$

Recursive Block Order

$$
\begin{aligned}
&S_{x_L} < S_{y_L} && \implies x <_B y \\
&S_{x_L} > S_{y_L} && \implies x >_B y \\
&S_{x_L} = S_{y_L} \text{ and } x_L <_B y_L && \implies x <_B y \\
&S_{x_L} = S_{y_L} \text{ and } x_L >_B y_L && \implies x >_B y \\
&S_{x_L} = S_{y_L} \text{ and } x_L = y_L \text{ and } x_R <_B x_L && \implies x <_B y \\
&S_{x_L} = S_{y_L} \text{ and } x_L = y_L \text{ and } x_R >_B x_L && \implies x >_B y
\end{aligned}
$$

Component bound = 3

Configuration	Rank	Configuration	Rank
(0,2,3,3)	0	(2,1,2,3)	7
(1,1,3,3)	1	(2,1,3,2)	8
(2,0,3,3)	2	(3,0,2,3)	9
(0,3,2,3)	3	(3,0,3,2)	10
(0,3,3,2)	4	(1,3,1,3)	11
(1,2,2,3)	5	(1,3,2,2)	12
(1,2,3,2)	6	(1,3,3,1)	13

Recursive Block Order – Rank

$$S_{x_L} < S_{y_L} \implies x <_B y$$
$$S_{x_L} > S_{y_L} \implies x >_B y$$
$$S_{x_L} = S_{y_L} \text{ and } x_L <_B y_L \implies x <_B y$$
$$S_{x_L} = S_{y_L} \text{ and } x_L >_B y_L \implies x >_B y$$
$$S_{x_L} = S_{y_L} \text{ and } x_L = y_L \text{ and } x_R <_B x_L \implies x <_B y$$
$$S_{x_L} = S_{y_L} \text{ and } x_L = y_L \text{ and } x_R >_B x_L \implies x >_B y$$

$$\text{rank}_n(x) = \sum_{s=0}^{S_{x_L}-1} C(n/2, s) \cdot C(n/2, S - s)$$
$$+ \text{rank}_{n/2}(x_L) \cdot C(n/2, S_{x_R})$$
$$+ \text{rank}_{n/2}(x_R)$$

We only need 2logn rows!

Filling the C table

- Dynamic Programming

$$C_d(n,S) = \begin{cases} 1 & \text{if } n=1, S \le d \\ 0 & \text{if } n=1, S > d \\ 1 & \text{if } n=0 \\ C_d(n{-}1,S) + C_d(n,S{-}1) & \text{if } n>0, S \le d \\ C_d(n{-}1,S) + C_d(n,S{-}1) - C_d(n{-}1,S{-}d{-}1) & \text{otherwise} \end{cases}$$

Start with 0

Start with number > 0

Overcount by those that start with d

- Generating Functions

$$C_d(n,S) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} \binom{n+S-k(d+1)-1}{n-1}$$

Performance Tests and Results

Our implementation of Tajik et al. algorithm

Application	50 rounds	500 rounds	1000 rounds
10x10 image block ($n = 100, d = 255$)	0.06s	0.39s	0.77s
16x16 image block ($n = 256, d = 255$)	0.11s	0.98s	1.98s
32x32 image block ($n = 1024, d = 255$)	0.41s	3.91s	7.81s
Exam scores ($n = 300, d = 100$)	0.12s	1.14s	2.26s
Salaries ($n = 30, d = 100000$)	0.02s	0.13s	0.25s
Ratings ($n = 5000, d = 4$)	1.84s	18.5s	37.8s

Application	Lexicographic			Recursive Block		
	Table size	Table time	Enc. time	Table size	Table time	Enc. time
10x10	162 MB	1.81s	0.009s	9 MB	0.87s	0.06s
16x16	1885 MB	13.1s	0.013	32 MB	5.8s	.18s
32x32	fail	-	-	271 MB	316s	3.50s
Exams	988MB	7.17s	0.011s	15 MB	3.81s	.096s
Salaries	4756 MB	79s	0.34s	672 MB	40.5s	4.2s
Ratings	fail	-	-	25 MB	271 s	0.40 s

Summary

- First proof bounding the mixing time of the Tajik et al. algorithm
- Give practical rank and unrank algorithms for sum-preserving encryption
- Create prototype implementations with performance comparisons

Thank you!