A Latin square autotopism secret sharing scheme

Talk by Rebecca J. Stones

Co-authors: Ming Su, Xiaoguang Liu, Gang Wang,
(Nankai University)
and Sheng Lin
(Tianjin University of Technology).

September 12, 2014

Secret sharing schemes

Secret sharing schemes describe how to distribute pieces of information, called *shares*, among *participants* so that:

Secret sharing schemes

Secret sharing schemes describe how to distribute pieces of information, called *shares*, among *participants* so that:

> if the participants cooperate, their collective shares can be used to recover a secret message, and

Secret sharing schemes

Secret sharing schemes describe how to distribute pieces of information, called *shares*, among *participants* so that:

- if the participants cooperate, their collective shares can be used to recover a secret message, and
- if too few participants cooperate, then the secret cannot be recovered.

A toy example...

$$\overbrace{\begin{bmatrix} 1&0&1&0&0&1&1&0&1&0\\ 1&1&1&1&1&1&1&0&0&0\\ 0&0&1&0&0&1&0&1&1&0\\ 1&0&1&1&1&0&1&1&1&0\\ 0&1&1&1&0&0&0&0&1&1\\ 0&0&1&1&0&0&0&1&1&0\\ 1&1&0&1&0&1&0&1&0&1\\ 0&1&1&1&0&0&0&0&1&0\\ 1&1&1&0&1&1&0&1&0&0\\ 1&0&1&1&0&1&0&0&0&1 \end{bmatrix}}^{\text{share 1}} \quad \overbrace{\begin{bmatrix} 1&1&1&0&1&0&1&0&1&0\\ 0&1&0&1&1&1&0&0&0&0\\ 0&1&1&1&0&1&0&0&1&0\\ 1&0&0&1&0&0&1&1&0&0\\ 1&1&1&0&0&0&0&0&1&0\\ 1&0&1&1&0&0&0&0&0&0\\ 1&0&0&1&0&1&0&1&1&0\\ 0&1&0&1&0&1&0&0&1&1\\ 1&1&1&1&1&0&1&1&0&0\\ 1&0&1&1&1&1&1&1&0&1 \end{bmatrix}}^{\text{share 2}}$$

A toy example...

$$
\overbrace{\begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}}^{\text{share 1}}
+
\overbrace{\begin{bmatrix}
1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{bmatrix}}^{\text{share 2}}
=
\overbrace{\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{bmatrix}}^{\text{addition modulo 2 reveals secret}}
$$

A toy example...

$$\underbrace{\begin{bmatrix} 1&0&1&0&0&1&1&0&1&0\\ 1&1&1&1&1&1&1&0&0&0\\ 0&0&1&0&0&1&0&1&1&0\\ 1&0&1&1&1&0&1&1&1&0\\ 0&1&1&1&0&0&0&0&1&1\\ 0&0&1&1&0&0&0&1&1&0\\ 1&1&0&1&0&1&0&1&0&1\\ 0&1&1&1&0&0&0&0&1&0\\ 1&1&1&0&1&1&0&1&0&0\\ 1&0&1&1&0&1&0&0&0&1 \end{bmatrix}}_{\text{share 1}} + \underbrace{\begin{bmatrix} 1&1&1&0&1&0&1&0&1&0\\ 0&1&0&1&1&1&0&0&0&0\\ 0&1&1&1&0&1&0&0&1&0\\ 1&0&0&1&0&0&1&1&0&0\\ 1&1&1&0&0&0&0&0&1&0\\ 1&0&1&1&0&0&0&0&0&0\\ 1&0&0&1&0&1&0&1&1&0\\ 0&1&0&1&0&1&0&0&1&1\\ 1&1&1&1&1&0&1&1&0&0\\ 1&0&1&1&1&1&1&1&0&1 \end{bmatrix}}_{\text{share 2}} = \underbrace{\begin{bmatrix} 0&1&0&0&1&1&0&0&0&0\\ 1&0&1&0&0&0&1&0&0&0\\ 0&1&0&1&0&0&0&1&0&0\\ 0&0&1&0&1&0&0&0&1&0\\ 1&0&0&1&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&1&1&0\\ 0&1&0&0&0&0&0&0&1&1\\ 0&0&1&0&0&1&0&0&0&1\\ 0&0&0&1&0&1&1&0&0&0\\ 0&0&0&0&1&0&1&1&0&0 \end{bmatrix}}_{\text{addition modulo 2 reveals secret}}$$

We can't find the secret without both shares.

A toy example...

$$\underbrace{\begin{bmatrix}1&0&1&0&0&1&1&0&1&0\\1&1&1&1&1&1&1&0&0&0\\0&0&1&0&0&1&0&1&1&0\\1&0&1&1&1&0&1&1&1&0\\0&1&1&1&0&0&0&0&1&1\\0&0&1&1&0&0&0&1&1&0\\1&1&0&1&0&1&0&1&0&1\\0&1&1&1&0&0&0&0&1&0\\1&1&1&0&1&1&0&1&0&0\\1&0&1&1&0&1&0&0&0&1\end{bmatrix}}_{\text{share 1}} + \underbrace{\begin{bmatrix}1&1&1&0&1&0&1&0&1&0\\0&1&0&1&1&1&0&0&0&0\\0&1&1&1&0&1&0&0&1&0\\1&0&0&1&0&0&1&1&0&0\\1&1&1&0&0&0&0&0&1&0\\1&0&1&1&0&0&0&0&0&0\\1&0&0&1&0&1&0&1&1&0\\0&1&0&1&0&1&0&0&1&1\\1&1&1&1&1&0&1&1&0&0\\1&0&1&1&1&1&1&1&0&1\end{bmatrix}}_{\text{share 2}} = \underbrace{\begin{bmatrix}0&1&0&0&1&1&0&0&0&0\\1&0&1&0&0&0&1&0&0&0\\0&1&0&1&0&0&0&1&0&0\\0&0&1&0&1&0&0&0&1&0\\1&0&0&1&0&0&0&0&0&1\\1&0&0&0&0&0&0&1&1&0\\0&1&0&0&0&0&0&0&1&1\\0&0&1&0&0&1&0&0&0&1\\0&0&0&1&0&1&1&0&0&0\\0&0&0&0&1&0&1&1&0&0\end{bmatrix}}_{\text{addition modulo 2 reveals secret}}$$

We can't find the secret without both shares.

We can choose share 1 uniformly at random.

A toy example...

$$\overbrace{\begin{bmatrix}1&0&1&0&0&1&1&0&1&0\\1&1&1&1&1&1&1&0&0&0\\0&0&1&0&0&1&0&1&1&0\\1&0&1&1&1&0&1&1&1&0\\0&1&1&1&0&0&0&0&1&1\\0&0&1&1&0&0&0&1&1&0\\1&1&0&1&0&1&0&1&0&1\\0&1&1&1&0&0&0&0&1&0\\1&1&1&0&1&1&0&1&0&0\\1&0&1&1&0&1&0&0&0&1\end{bmatrix}}^{\text{share 1}} + \overbrace{\begin{bmatrix}1&1&1&0&1&0&1&0&1&0\\0&1&0&1&1&1&0&0&0&0\\0&1&1&1&0&1&0&0&1&0\\1&0&0&1&0&0&1&1&0&0\\1&1&1&0&0&0&0&0&1&0\\1&0&1&1&0&0&0&0&0&0\\1&0&0&1&0&1&0&1&1&0\\0&1&0&1&0&1&0&0&1&1\\1&1&1&1&1&0&1&1&0&0\\1&0&1&1&1&1&1&1&0&1\end{bmatrix}}^{\text{share 2}} = \overbrace{\begin{bmatrix}0&1&0&0&1&1&0&0&0&0\\1&0&1&0&0&0&1&0&0&0\\0&1&0&1&0&0&0&1&0&0\\0&0&1&0&1&0&0&0&1&0\\1&0&0&1&0&0&0&0&0&1\\1&0&0&0&0&0&0&1&1&0\\0&1&0&0&0&0&0&0&1&1\\0&0&1&0&0&1&0&0&0&1\\0&0&0&1&0&1&1&0&0&0\\0&0&0&0&1&0&1&1&0&0\end{bmatrix}}^{\text{addition modulo 2 reveals secret}}$$

We can't find the secret without both shares.

We can choose share 1 uniformly at random. And choose share 2 to so that "share 1 + share 2" reveals the secret.

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme.
(*How to share a secret* (1979), Comm. ACM.)

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme.
(*How to share a secret* (1979), Comm. ACM.)

We have

ℓ participants,

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme.
(*How to share a secret* (1979), Comm. ACM.)

We have

ℓ participants,

a secret number c, and

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme.
(*How to share a secret* (1979), Comm. ACM.)

We have

- ℓ participants,
- a secret number c, and
- we want any t of the participants to be able to recover the secret.

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme. (*How to share a secret* (1979), Comm. ACM.)

We have

ℓ participants,

a secret number c, and

we want any t of the participants to be able to recover the secret.

We generate a polynomial f of degree $t-1$ with constant term c and the other coefficients are chosen at random.

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme. (*How to share a secret* (1979), Comm. ACM.)

We have

- ℓ participants,
- a secret number c, and
- we want any t of the participants to be able to recover the secret.

We generate a polynomial f of degree $t - 1$ with constant term c and the other coefficients are chosen at random. The shares are distinct points $(x, f(x))$ (except for when $x = 0$).

Shamir's Secret Sharing Scheme

Adi Shamir (of RSA fame) developed a secret sharing scheme.
(*How to share a secret* (1979), Comm. ACM.)

We have

- ℓ participants,
- a secret number c, and
- we want any t of the participants to be able to recover the secret.

We generate a polynomial f of degree $t-1$ with constant term c and the other coefficients are chosen at random. The shares are distinct points $(x, f(x))$ (except for when $x = 0$).

Given any t points, we can use Lagrange Interpolation to recover f, and find the secret $f(0)$.

Shamir's secret sharing scheme is in widespread use and has withstood the test of time.

Shamir's secret sharing scheme is in widespread use and has withstood the test of time. This relegates most subsequently studied secret sharing schemes to be primarily of academic interest

Shamir's secret sharing scheme is in widespread use and has withstood the test of time. This relegates most subsequently studied secret sharing schemes to be primarily of academic interest (including the one I'm presenting, but it could be thought of as an alternative).

Shamir's secret sharing scheme is in widespread use and has withstood the test of time. This relegates most subsequently studied secret sharing schemes to be primarily of academic interest (including the one I'm presenting, but it could be thought of as an alternative).

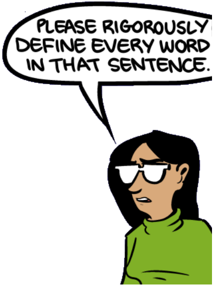
Blakely developed a different secret sharing scheme where the shares are hyperplanes and the secret is their unique intersection point (via linear algebra). (*Safeguarding cryptographic keys* (1979).)

Shamir's secret sharing scheme is in widespread use and has withstood the test of time. This relegates most subsequently studied secret sharing schemes to be primarily of academic interest (including the one I'm presenting, but it could be thought of as an alternative).

Blakely developed a different secret sharing scheme where the shares are hyperplanes and the secret is their unique intersection point (via linear algebra). (*Safeguarding cryptographic keys* (1979).)

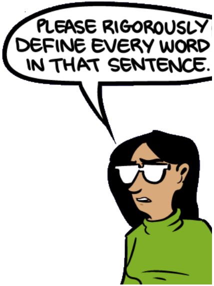
> *Secret sharing was invented independently by Adi Shamir and George Blakley in 1979. — Wikipedia.*

Latin squares (intro)



(Image source: SMBC)

Latin squares (intro)

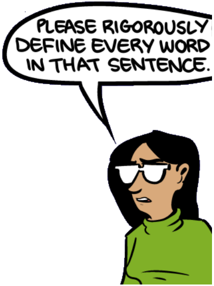


(Image source: SMBC)

A *Latin square* of order $n = 3$:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}.$$

Latin squares (intro)



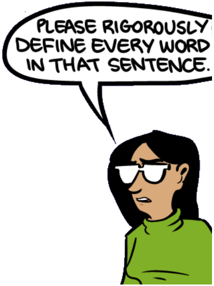
(Image source: SMBC)

A *Latin square* of order $n = 3$:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}.$$

It contains *entries* e.g. $(0, 0, 0)$, $(1, 2, 0)$, $(2, 0, 2)$.

Latin squares (intro)



(Image source: SMBC)

A *Latin square* of order $n = 3$:

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}.$$

It contains *entries* e.g. $(0, 0, 0)$, $(1, 2, 0)$, $(2, 0, 2)$. It has *autotopisms* (or *symmetries*) e.g.

$$\big(\overbrace{(0, 1, 2)}^{\text{row perm}}, \overbrace{(0, 1, 2)}^{\text{col perm}}, \overbrace{(0, 2, 1)}^{\text{sym perm}} \big).$$

Reconstruction from partial Latin squares

A Latin square of order 4 and a critical set:

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	·	·
·	·	·	2
·	3	·	·
·	·	1	·

Reconstruction from partial Latin squares

A Latin square of order 4 and a critical set:

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	·	·
·	·	·	2
·	3	·	·
·	·	1	·

A critical set

Reconstruction from partial Latin squares

A Latin square of order 4 and a critical set:

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	·	·
·	·	·	2
·	3	·	·
·	·	1	·

A critical set (a) completes to a *unique* Latin square

Reconstruction from partial Latin squares

A Latin square of order 4 and a critical set:

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	·	·
·	·	·	2
·	3	·	·
·	·	1	·

A critical set (a) completes to a *unique* Latin square and (b) any proper subset of these entries completes to ≥ 2 Latin squares.

Reconstruction from partial Latin squares

A Latin square of order 4 and a critical set:

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	·	·
·	·	·	2
·	3	·	·
·	·	1	·

A critical set (a) completes to a *unique* Latin square and (b) any proper subset of these entries completes to ≥ 2 Latin squares.

Cooper, Donovan, and Seberry (1994) proposed having a secret Latin square, and splitting critical sets among the participants.

Reconstruction from partial Latin squares

A Latin square of order 4 and a critical set:

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	·	·
·	·	·	2
·	3	·	·
·	·	1	·

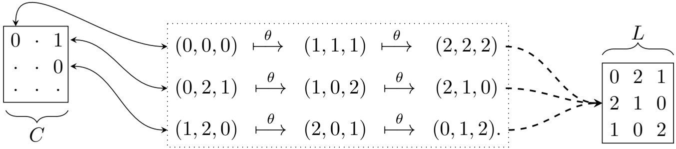
A critical set (a) completes to a *unique* Latin square and (b) any proper subset of these entries completes to ≥ 2 Latin squares.

Cooper, Donovan, and Seberry (1994) proposed having a secret Latin square, and splitting critical sets among the participants.

This scheme has been (harshly) criticized in the literature as impractical. (More about this later...)

Reconstruction from contours

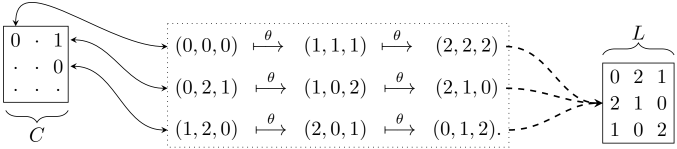
We can reconstruct a Latin square L from knowledge of a *contour* C and an autotopism θ.



Here $\theta = \big((0, 1, 2), (0, 1, 2), (0, 1, 2)\big)$.

Reconstruction from contours

We can reconstruct a Latin square L from knowledge of a *contour* C and an autotopism θ.

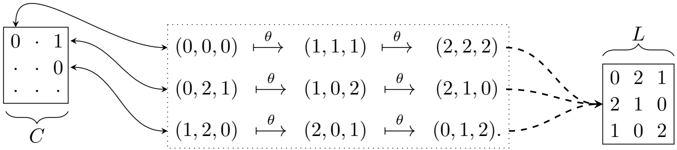


Here $\theta = \big((0,1,2),(0,1,2),(0,1,2)\big)$.

Ganfornina (2006) proposed having a secret Latin square, and splitting contours among participants.

Reconstruction from contours

We can reconstruct a Latin square L from knowledge of a *contour* C and an autotopism θ.



Here $\theta = \big((0,1,2),(0,1,2),(0,1,2)\big)$.

Ganfornina (2006) proposed having a secret Latin square, and splitting contours among participants. This was not carefully analyzed in his work (it felt more like he was proposing a potential application).

Criticisms

Why a Latin square? There have been many proposed secret sharing schemes using a variety of combinatorial objects as secrets; why would we want a secret Latin square?

Criticisms

Why a Latin square? There have been many proposed secret sharing schemes using a variety of combinatorial objects as secrets; why would we want a secret Latin square? Latin squares also have $O(n^2)$ entries, which might be "too much" for some applications (in terms of time and/or space).

Criticisms

Why a Latin square? There have been many proposed secret sharing schemes using a variety of combinatorial objects as secrets; why would we want a secret Latin square? Latin squares also have $O(n^2)$ entries, which might be "too much" for some applications (in terms of time and/or space).

Verification If the participants cooperate and recover a Latin square X, how can they be sure that $X = L$, the secret Latin square?

Criticisms

Why a Latin square? There have been many proposed secret sharing schemes using a variety of combinatorial objects as secrets; why would we want a secret Latin square? Latin squares also have $O(n^2)$ entries, which might be "too much" for some applications (in terms of time and/or space).

Verification If the participants cooperate and recover a Latin square X, how can they be sure that $X = L$, the secret Latin square?

Initialization and reconstruction complexity Typically, it is difficult to find a critical set C, and given a critical set C, it is difficult to find the completion of C (determining if a partial Latin square admits a completion is NP-complete; Colbourn 1984).

More criticisms

Partial information The shares reveal partial information about the secret Latin square to the participants.

More criticisms

Partial information The shares reveal partial information about the secret Latin square to the participants.

A subtle “flaw” It was shown in Donovan et al. (2012) that some partial critical sets embed in only one critical set (so the secret can be determined without knowledge of the full critical set).

More criticisms

Partial information The shares reveal partial information about the secret Latin square to the participants.

A subtle "flaw" It was shown in Donovan et al. (2012) that some partial critical sets embed in only one critical set (so the secret can be determined without knowledge of the full critical set).

Multi-level scheme It is impractical to extend these schemes to multi-level schemes (where certain subsets of the participants can combine to find the secret).

The proposed scheme

The method we propose differs in two key aspects:

- Instead of having a secret Latin square that admits an autotopism, we have a secret autotopism (and we use the Latin square for verification).

The proposed scheme

The method we propose differs in two key aspects:

- Instead of having a secret Latin square that admits an autotopism, we have a secret autotopism (and we use the Latin square for verification).
- We enforce particular cycle structures for the autotopism; this allows a concrete theoretical analysis.

The proposed scheme

The method we propose differs in two key aspects:

- Instead of having a secret Latin square that admits an autotopism, we have a secret autotopism (and we use the Latin square for verification).
- We enforce particular cycle structures for the autotopism; this allows a concrete theoretical analysis.

We call an isotopism $\theta = (\alpha, \beta, \gamma)$ *suitable* if α, β, and γ all decompose into 2 disjoint $(n/2)$-cycles.

Generating the "prior" contour

We generate a random contour for the autotopism $\zeta = (\tau, \tau, \tau)$ where $\tau := (0, 1, \ldots, n/2 - 1)(n/2, n/2 + 1, \ldots, n - 1)$ by sticking 0's and $n/2$'s along the diagonals indicated below:

$$D = \left[\begin{array}{ccc|ccc} \cdot & \cdot & 0 & \cdot & \cdot & 3 \\ \cdot & 3 & \cdot & \cdot & 0 & \cdot \\ 0 & \cdot & \cdot & 3 & \cdot & \cdot \\ \hline \cdot & \cdot & 3 & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & 3 & \cdot \\ 3 & \cdot & \cdot & 0 & \cdot & \cdot \end{array}\right] \xrightarrow{\text{contour}} L_{\text{prior}} = \left[\begin{array}{ccc|ccc} 5 & 1 & 0 & 2 & 4 & 3 \\ 1 & 3 & 2 & 4 & 0 & 5 \\ 0 & 2 & 4 & 3 & 5 & 1 \\ \hline 2 & 4 & 3 & 5 & 1 & 0 \\ 4 & 0 & 5 & 1 & 3 & 2 \\ 3 & 5 & 1 & 0 & 2 & 4 \end{array}\right]$$

Generating the "prior" contour

We generate a random contour for the autotopism $\zeta = (\tau, \tau, \tau)$ where $\tau := (0, 1, \ldots, n/2-1)(n/2, n/2+1, \ldots, n-1)$ by sticking 0's and $n/2$'s along the diagonals indicated below:

$$D = \begin{array}{|ccc|ccc|} \hline \cdot & \cdot & 0 & \cdot & \cdot & 3 \\ \cdot & 3 & \cdot & \cdot & 0 & \cdot \\ 0 & \cdot & \cdot & 3 & \cdot & \cdot \\ \hline \cdot & \cdot & 3 & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & 3 & \cdot \\ 3 & \cdot & \cdot & 0 & \cdot & \cdot \\ \hline \end{array} \xrightarrow{\text{contour}} L_{\text{prior}} = \begin{array}{|ccc|ccc|} \hline 5 & 1 & 0 & 2 & 4 & 3 \\ 1 & 3 & 2 & 4 & 0 & 5 \\ 0 & 2 & 4 & 3 & 5 & 1 \\ \hline 2 & 4 & 3 & 5 & 1 & 0 \\ 4 & 0 & 5 & 1 & 3 & 2 \\ 3 & 5 & 1 & 0 & 2 & 4 \\ \hline \end{array}$$

(for this to work we need, and hence assume $n \equiv 0 \pmod 4$).

Generating the "prior" contour

We generate a random contour for the autotopism $\zeta = (\tau, \tau, \tau)$ where $\tau := (0, 1, \ldots, n/2 - 1)(n/2, n/2 + 1, \ldots, n - 1)$ by sticking 0's and $n/2$'s along the diagonals indicated below:

$$D = \left[\begin{array}{ccc|ccc} \cdot & \cdot & 0 & \cdot & \cdot & 3 \\ \cdot & 3 & \cdot & \cdot & 0 & \cdot \\ 0 & \cdot & \cdot & 3 & \cdot & \cdot \\ \hline \cdot & \cdot & 3 & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & 3 & \cdot \\ 3 & \cdot & \cdot & 0 & \cdot & \cdot \end{array}\right] \xrightarrow{\text{contour}} L_{\text{prior}} = \left[\begin{array}{ccc|ccc} 5 & 1 & 0 & 2 & 4 & 3 \\ 1 & 3 & 2 & 4 & 0 & 5 \\ 0 & 2 & 4 & 3 & 5 & 1 \\ \hline 2 & 4 & 3 & 5 & 1 & 0 \\ 4 & 0 & 5 & 1 & 3 & 2 \\ 3 & 5 & 1 & 0 & 2 & 4 \end{array}\right]$$

(for this to work we need, and hence assume $n \equiv 0 \pmod 4$).

Instead of the original contour for D, we retain a random contour C_{prior} by replacing each entry $(i, j, d_{i,j})$ in the contour with $\zeta^t(i, j, d_{i,j})$ for $t \in \{0, 1, \ldots, n/2 - 1\}$ randomly chosen for each entry.

$$C_{\text{prior}} = \left[\begin{array}{ccc|ccc} 5 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & 4 & 0 & \cdot \\ 0 & \cdot & \cdot & 3 & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 5 & \cdot & \cdot & 2 \\ \cdot & 5 & \cdot & \cdot & 2 & 4 \end{array}\right]$$

Randomizing the contour and autotopism

We randomly generate an isotopism φ.

Randomizing the contour and autotopism

We randomly generate an isotopism φ.

If L_{prior} is a Latin square that admits the autotopism ζ, then $L := \varphi(L_{\text{prior}})$ admits the autotopism $\theta := \varphi\zeta\varphi^{-1}$.

Randomizing the contour and autotopism

We randomly generate an isotopism φ.

If L_{prior} is a Latin square that admits the autotopism ζ, then $L := \varphi(L_{\text{prior}})$ admits the autotopism $\theta := \varphi\zeta\varphi^{-1}$. Note: θ is a suitable autotopism.

Randomizing the contour and autotopism

We randomly generate an isotopism φ.

If L_{prior} is a Latin square that admits the autotopism ζ, then $L := \varphi(L_{\text{prior}})$ admits the autotopism $\theta := \varphi\zeta\varphi^{-1}$. Note: θ is a suitable autotopism.

If we apply the random isotopism

$$\varphi = \big((0,4,1,3,5,2),(1,2,4),(1,3,2,5)\big)$$

to the earlier example, we obtain the Latin square

$L = \varphi(L_{\text{prior}}) =$

0	1	5	2	4	3
4	2	0	3	1	5
2	5	1	0	3	4
3	0	2	4	5	1
1	4	3	5	0	2
5	3	4	1	2	0

which admits the autotopism

$$\begin{aligned}\theta &= \varphi\zeta\varphi^{-1} \\ &= \big((0,4,3)(1,2,5),(0,2,4)(1,5,3),(0,3,5)(1,2,4)\big).\end{aligned}$$

Randomizing the contour (cont.)

Further, it is generated by the contour

$$C = \varphi(C_{\text{prior}}) = \begin{array}{|cccccc|} \hline 0 & \cdot & \cdot & 2 & \cdot & \cdot \\ \cdot & \cdot & 0 & \cdot & 1 & 5 \\ \cdot & 5 & 1 & \cdot & \cdot & 4 \\ 3 & 0 & \cdot & 4 & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline \end{array}$$

and the autotopism θ.

Splitting the autotopism

If we have e.g. 4 participants, we split the autotopism θ into 3 random isotopisms $\sigma_1, \sigma_2, \sigma_3$, and we choose σ_4 such that $\theta = \sigma_1\sigma_2\sigma_3\sigma_4$.

Splitting the autotopism

If we have e.g. 4 participants, we split the autotopism θ into 3 random isotopisms $\sigma_1, \sigma_2, \sigma_3$, and we choose σ_4 such that $\theta = \sigma_1\sigma_2\sigma_3\sigma_4$. E.g., we might end up with:

$$\begin{aligned}
\sigma_1 &= \big((0,4)(1,5),(0,4,5,3,1),(0,5,1)(2,4,3)\big)\\
\sigma_2 &= \big((0,4)(1,3,5,2),(0,2,5),(0,1,3,4,5,2)\big)\\
\sigma_3 &= \big((0,1,3,2,5),(0,1,3,5,4),(1,5)(2,4)\big)\\
\sigma_4 &= \big((1,4,3,5,2),(0,2,5,3,1),(0,5,2,1,4,3)\big).
\end{aligned}$$

Splitting the autotopism

If we have e.g. 4 participants, we split the autotopism θ into 3 random isotopisms $\sigma_1, \sigma_2, \sigma_3$, and we choose σ_4 such that $\theta = \sigma_1\sigma_2\sigma_3\sigma_4$. E.g., we might end up with:

$$\begin{aligned}
\sigma_1 &= \big((0,4)(1,5),(0,4,5,3,1),(0,5,1)(2,4,3)\big)\\
\sigma_2 &= \big((0,4)(1,3,5,2),(0,2,5),(0,1,3,4,5,2)\big)\\
\sigma_3 &= \big((0,1,3,2,5),(0,1,3,5,4),(1,5)(2,4)\big)\\
\sigma_4 &= \big((1,4,3,5,2),(0,2,5,3,1),(0,5,2,1,4,3)\big).
\end{aligned}$$

These are our shares and we distribute one to each participant.

Public contour

We compute $C_{\mathsf{public}} := \xi(C)$ where $\xi := \sigma_\ell \sigma_{\ell-1} \cdots \sigma_1$.

Public contour

We compute $C_{\text{public}} := \xi(C)$ where $\xi := \sigma_\ell \sigma_{\ell-1} \cdots \sigma_1$. In our running example, we have the situation

$$\xi = \big((0,3)(1,4,5,2), (0,3,1)(2,5,4), (0,2,4,3)(1,5)\big)$$

and so

$$C_{\text{public}} = \xi(C) = \begin{array}{|cccccc|} \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & \cdot & 2 & \cdot & \cdot \\ 1 & \cdot & 4 & \cdot & 3 & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & 3 & 2 & 5 & \cdot & \cdot \\ 2 & \cdot & \cdot & \cdot & 4 & 1 \\ \hline \end{array}$$

which we make public.

Public contour

We compute $C_{\text{public}} := \xi(C)$ where $\xi := \sigma_\ell \sigma_{\ell-1} \cdots \sigma_1$. In our running example, we have the situation

$$\xi = \big((0,3)(1,4,5,2),(0,3,1)(2,5,4),(0,2,4,3)(1,5)\big)$$

and so

$$C_{\text{public}} = \xi(C) = \begin{array}{|cccccc|}\hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & \cdot & 2 & \cdot & \cdot \\ 1 & \cdot & 4 & \cdot & 3 & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & 3 & 2 & 5 & \cdot & \cdot \\ 2 & \cdot & \cdot & \cdot & 4 & 1 \\ \hline \end{array}$$

which we make public. When the shares are returned to reveal the secret, we use this to verify that the shares combine correctly.

Review

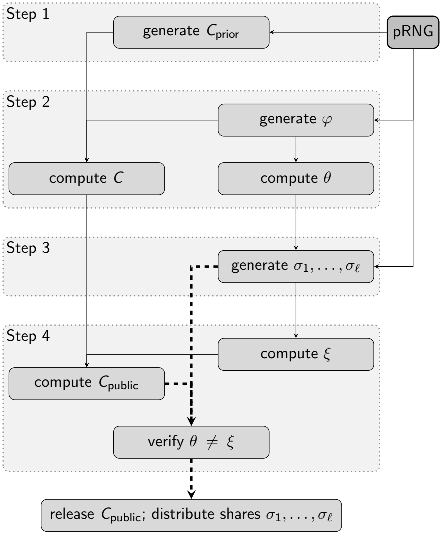


Figure : Flow chart of the proposed secret sharing scheme: initialization phase. (We also check $\theta = \xi$, restarting if this happens.)

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_\ell$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma}_i = \sigma_i$).

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\mathsf{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\mathsf{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\mathsf{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`. Otherwise we verify that L_{cand}, determined from the contour $C = \xi^{-1}(C_{\mathsf{public}}) = \sigma_1^{-1}\sigma_2^{-1}\cdots\sigma_\ell^{-1}(C_{\mathsf{public}})$ and θ_{cand}, is a Latin square.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\mathsf{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`. Otherwise we verify that L_{cand}, determined from the contour $C = \xi^{-1}(C_{\mathsf{public}}) = \sigma_1^{-1}\sigma_2^{-1}\cdots\sigma_\ell^{-1}(C_{\mathsf{public}})$ and θ_{cand}, is a Latin square.
3. If L_{cand} is not a Latin square, then we return `fail`.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\mathsf{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`. Otherwise we verify that L_{cand}, determined from the contour $C = \xi^{-1}(C_{\mathsf{public}}) = \sigma_1^{-1}\sigma_2^{-1}\cdots\sigma_\ell^{-1}(C_{\mathsf{public}})$ and θ_{cand}, is a Latin square.
3. If L_{cand} is not a Latin square, then we return `fail`. Otherwise θ_{cand} is revealed to the participants.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\text{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`. Otherwise we verify that L_{cand}, determined from the contour $C = \xi^{-1}(C_{\text{public}}) = \sigma_1^{-1}\sigma_2^{-1}\cdots\sigma_\ell^{-1}(C_{\text{public}})$ and θ_{cand}, is a Latin square.
3. If L_{cand} is not a Latin square, then we return `fail`. Otherwise θ_{cand} is revealed to the participants.

Security The security of this scheme depends on the small chance of θ_{cand} being returned when $\theta_{\text{cand}} \neq \theta$.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\text{cand}} := \tilde{\sigma_1}\tilde{\sigma_2}\cdots\tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`. Otherwise we verify that L_{cand}, determined from the contour $C = \xi^{-1}(C_{\text{public}}) = \sigma_1^{-1}\sigma_2^{-1}\cdots\sigma_\ell^{-1}(C_{\text{public}})$ and θ_{cand}, is a Latin square.
3. If L_{cand} is not a Latin square, then we return `fail`. Otherwise θ_{cand} is revealed to the participants.

Security The security of this scheme depends on the small chance of θ_{cand} being returned when $\theta_{\text{cand}} \neq \theta$.

Efficiency We don't need to generate the Latin square L for verification.

Recovery

When all participants decide to cooperate, the participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_\ell}$ to a *combiner* (possibly incorrectly—if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$).

1. The combiner computes $\theta_{\text{cand}} := \tilde{\sigma_1}\tilde{\sigma_2} \cdots \tilde{\sigma_\ell}$.
2. If θ_{cand} is not suitable, then we return `fail`. Otherwise we verify that L_{cand}, determined from the contour $C = \xi^{-1}(C_{\text{public}}) = \sigma_1^{-1}\sigma_2^{-1} \cdots \sigma_\ell^{-1}(C_{\text{public}})$ and θ_{cand}, is a Latin square.
3. If L_{cand} is not a Latin square, then we return `fail`. Otherwise θ_{cand} is revealed to the participants.

Security The security of this scheme depends on the small chance of θ_{cand} being returned when $\theta_{\text{cand}} \neq \theta$.

Efficiency We don't need to generate the Latin square L for verification. It suffices, and is more efficient to check the two "leading" rows and columns for clashes.

Security analysis

Collusion Each σ_i is a random isotopism (distributed uniformly at random from $S_n \times S_n \times S_n$); knowledge of fewer than all ℓ shares σ_i is of no more use in recovering θ or C than is a random suitable isotopism.

Security analysis

Collusion Each σ_i is a random isotopism (distributed uniformly at random from $S_n \times S_n \times S_n$); knowledge of fewer than all ℓ shares σ_i is of no more use in recovering θ or C than is a random suitable isotopism.

Brute-force attack Search spaces are too large:

n	nr LS with autotop. ζ	nr suitable isotop.	$\mathrm{is}(L)$ lower bound
6	648	6×10^4	2×10^5
10	20820000	3×10^{14}	4×10^{14}
14	?	7×10^{26}	1×10^{27}
18	?	6×10^{40}	7×10^{39}

Security analysis (cont.)

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group, and find the secret θ.

Security analysis (cont.)

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group, and find the secret θ. So we need to ensure C_{public} cannot be used to find L.

Security analysis (cont.)

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group, and find the secret θ. So we need to ensure C_{public} cannot be used to find L.

Assuming an attacker managed to find a completion of C_{public}, this would at most give the attacker knowledge of the isotopism class containing L.

Security analysis (cont.)

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group, and find the secret θ. So we need to ensure C_{public} cannot be used to find L.

Assuming an attacker managed to find a completion of C_{public}, this would at most give the attacker knowledge of the isotopism class containing L. If the attacker attempted to randomly guess L from knowledge of M, their probability of being correct is $1/\mathrm{is}(L)$. This probability is prohibitively small, even for $n = 10$.

Security analysis (cont.)

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group, and find the secret θ. So we need to ensure C_{public} cannot be used to find L.

Assuming an attacker managed to find a completion of C_{public}, this would at most give the attacker knowledge of the isotopism class containing L. If the attacker attempted to randomly guess L from knowledge of M, their probability of being correct is $1/\mathrm{is}(L)$. This probability is prohibitively small, even for $n = 10$.

Partial information about L Since the isotopisms σ_i are random, they provide no information about L.

Security analysis (cont.)

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group, and find the secret θ. So we need to ensure C_{public} cannot be used to find L.

Assuming an attacker managed to find a completion of C_{public}, this would at most give the attacker knowledge of the isotopism class containing L. If the attacker attempted to randomly guess L from knowledge of M, their probability of being correct is $1/\mathrm{is}(L)$. This probability is prohibitively small, even for $n = 10$.

Partial information about L Since the isotopisms σ_i are random, they provide no information about L. The public contour C_{public} might give some information about the isotopism class that L belongs to (such as the existence of subsquares), but even full knowledge of the isotopism class is of limited use.

Security analysis (cont.)

Attack by replacing shares How likely is it that an isotopism $\theta_{\mathsf{cand}} \neq \theta$ is returned?

Security analysis (cont.)

Attack by replacing shares How likely is it that an isotopism $\theta_{\mathsf{cand}} \neq \theta$ is returned?

Obstacle 1: If participant i returns the share $\tilde{\sigma}_i$ chosen uniformly at random from those whose components are even permutations, we have

$$\Pr[\theta_{\mathsf{cand}} \text{ suitable} \mid \tilde{\sigma}_i \text{ returned}] = \frac{64}{n^6}.$$

Security analysis (cont.)

Attack by replacing shares How likely is it that an isotopism $\theta_{\mathsf{cand}} \neq \theta$ is returned?

Obstacle 1: If participant i returns the share $\tilde{\sigma}_i$ chosen uniformly at random from those whose components are even permutations, we have

$$\Pr[\theta_{\mathsf{cand}} \text{ suitable} \mid \tilde{\sigma}_i \text{ returned}] = \frac{64}{n^6}.$$

Obstacle 2: Let p denote the probability of $\theta_{\mathsf{cand}} \neq \theta$ returned assuming Obstacle 1 is overcome.

Security analysis (cont.)

Attack by replacing shares How likely is it that an isotopism $\theta_{\mathsf{cand}} \neq \theta$ is returned?

Obstacle 1: If participant i returns the share $\tilde{\sigma}_i$ chosen uniformly at random from those whose components are even permutations, we have

$$\Pr[\theta_{\mathsf{cand}} \text{ suitable} \mid \tilde{\sigma}_i \text{ returned}] = \frac{64}{n^6}.$$

Obstacle 2: Let p denote the probability of $\theta_{\mathsf{cand}} \neq \theta$ returned assuming Obstacle 1 is overcome. This is tested experimentally:

n	experimentally $p \leq$		theoretically $p \geq$
6	4.5×10^{-5}	(99.995% confidence)	3.13×10^{-5}
10	2×10^{-11}	(99.995% confidence)	1.04×10^{-14}

Concluding remarks

1. The ability to verify the secret is correct is an advantage overy Shamir's scheme.

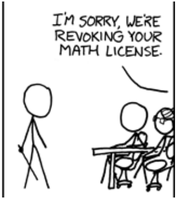
Concluding remarks

1. The ability to verify the secret is correct is an advantage overy Shamir's scheme.
2. We can easily extend to a multi-level scheme on-the-fly.

Concluding remarks

1. The ability to verify the secret is correct is an advantage overy Shamir's scheme.
2. We can easily extend to a multi-level scheme on-the-fly.
3. We can eliminate working with Latin squares altogether (they're "behind the scenes"); this saves on space and time complexity.

Thank you!



(Image source: xkcd)

Probability $(C, \theta_{\text{cand}})$ generates a Latin square, when θ_{cand} is random We have

$$\begin{aligned} p &:= \Pr[(C, \theta_{\text{cand}}) \text{ generates a Latin square}] \\ &= \Pr[(\varphi^{-1}(C), \varphi^{-1}\theta_{\text{cand}}\varphi) \text{ generates a Latin square}] \\ &= \Pr[(C_{\text{prior}}, \varphi^{-1}\theta_{\text{cand}}\varphi) \text{ generates a Latin square}] \\ &= \Pr[(C_{\text{prior}}, \theta_{\text{cand}}) \text{ generates a Latin square}] \end{aligned}$$

since θ_{cand} and $\varphi^{-1}\theta_{\text{cand}}\varphi$ are equal in distribution. This was used to simplify method used in the simulations.

Probability $(C, \theta_{\mathsf{cand}})$ generates a Latin square, when θ_{cand} is random We have

$$
\begin{aligned}
p &:= \Pr[(C, \theta_{\mathsf{cand}}) \text{ generates a Latin square}] \\
&= \Pr[(\varphi^{-1}(C), \varphi^{-1}\theta_{\mathsf{cand}}\varphi) \text{ generates a Latin square}] \\
&= \Pr[(C_{\mathsf{prior}}, \varphi^{-1}\theta_{\mathsf{cand}}\varphi) \text{ generates a Latin square}] \\
&= \Pr[(C_{\mathsf{prior}}, \theta_{\mathsf{cand}}) \text{ generates a Latin square}]
\end{aligned}
$$

since θ_{cand} and $\varphi^{-1}\theta_{\mathsf{cand}}\varphi$ are equal in distribution. This was used to simplify method used in the simulations.

For $n = 6$, we generate 10^9 pairs $(C_{\mathsf{prior}}, \beta)$, for random suitable autotopism β, and find 43409 generate a Latin square. The upper bound on the Wald confidence interval is 4.5×10^{-5} with 99.995% confidence. For $n = 10$, we made $N := 3.6 \times 10^{11}$ samples, and no Latin square was generated this way. Using a modified "rule of three", we can be 99.995% confident that $p \leq 7.6/N \approx 2 \times 10^{-11}$.