Des. Codes Cryptogr. (2016) 80:635–650
DOI 10.1007/s10623-015-0123-1

A Latin square autotopism secret sharing scheme

Rebecca J. Stones[1,2,3] · **Ming Su**[3,4] · **Xiaoguang Liu**[3,4] · **Gang Wang**[3,4] · **Sheng Lin**[5]

Received: 6 March 2015 / Revised: 20 July 2015 / Accepted: 23 July 2015 /
Published online: 4 August 2015
© The Author(s) 2015. This article is published with open access at Springerlink.com

Abstract We present a novel secret sharing scheme where the secret is an autotopism (a symmetry) of a Latin square. Previously proposed secret sharing schemes involving Latin squares have many drawbacks: (a) Latin squares contain n^2 entries, which may be too large, (b) partial information about the secret may be directly revealed, (c) a subsequently discovered subtle "flaw", (d) difficulty in initialization and reconstruction, (e) difficulty in verification, and (f) difficulty in generalizing to a multi-level scheme. We carefully analyze the security of the proposed scheme, and identify how it overcomes all of these problems.

Keywords Autotopism · Latin square · Partial Latin square · Secret sharing scheme

Mathematics Subject Classification 05B15 · 94A62

1 Introduction

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.

Communicated by C. Blundo.

✉ Rebecca J. Stones
rebecca.stones82@gmail.com

1 School of Mathematical Sciences and Faculty of Information Technology, Monash University, Melbourne, Australia

2 Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada

3 College of Computer and Control Engineering, Nankai University, Tianjin, China

4 College of Software, Nankai University, Tianjin, China

5 School of Computer and Communication Engineering, Tianjin University of Technology, Tianjin, China

The concept of secret sharing schemes goes back to Shamir [20] and Blakey [1] (see also [21]). In this work, we present and analyze a secret sharing scheme based on symmetries of Latin squares.

We will consider the case of when the secret message θ can be reconstructed from knowledge of l shares, which are distributed to the l participants, that is, every participant must cooperate to recover the secret L. The primary obstacle in attacking this scheme is the huge number of Latin squares and the huge number of symmetries they might have.

1.1 Latin squares

1.1.1 Introduction

A *Latin square* of order n is an $n \times n$ array $L = (l_{i,j})$ of n symbols such that the symbols in every row and in every column are distinct. We will index the rows and columns of L by the elements of $\mathbb{Z}_n$ and take the symbol set to be $\mathbb{Z}_n$. The *orthogonal array* of L is the set of ordered triples

$$O(L) = \{(i, j, l_{i,j}) : i, j \in \mathbb{Z}_n\}.$$

1.1.2 Symmetries

Let S_n be the symmetric group acting on $\mathbb{Z}_n$. We can act on the set of Latin squares $L = (l_{i,j})$ of order n with $\theta := (\alpha, \beta, \gamma) \in S_n \times S_n \times S_n$ such that the rows of L are permuted according to α, the columns of L are permuted according to β, and the symbols of L are permuted according to γ. In terms of entries, we have

$$\theta\big((i, j, l_{i,j})\big) = \big(\alpha(i), \beta(j), \gamma(l_{i,j})\big) \quad \text{for all } i, j \in \mathbb{Z}_n.$$

The mapping θ is called an *isotopism*, we call L and $\theta(L)$ *isotopic*, and the set of Latin squares isotopic to L is its *isotopism class*.

If $\theta(L) = L$, then θ is said to be an *autotopism* of L. Autotopism are the symmetries we will consider in this paper.

If $(i, j, l_{i,j})$ is an entry in a Latin square that admits an autotopism (α, β, γ), then its *orbit* is the set

$$\left\{\big(\alpha^k(i), \beta^k(j), \gamma^k(l_{i,j})\big) : k \geq 0\right\}$$

of entries of L. An orbit will have size dividing the order of (α, β, γ). Importantly, we can reconstruct the whole orbit from knowledge of a single entry in the orbit and (α, β, γ).

We will primarily focus on isotopisms in which each component decomposes into two disjoint $(n/2)$-cycles, which we will call *suitable*. The purpose of this restriction is to permit a simpler security analysis.

1.1.3 Partial Latin squares and contours

A *partial Latin square* of order n is an $n \times n$ array $P = (p_{i,j})$ of $n + 1$ symbols $\mathbb{Z}_n \cup \{\cdot\}$ such that each symbol in $\mathbb{Z}_n$ occurs at most once in each row and at most once in each column. We say an entry $(i, j, p_{i,j})$ is *defined* if $p_{i,j} \in \mathbb{Z}_n$ and *undefined* otherwise. The definitions of "entry", "orthogonal array", and "isotopism" extend naturally to partial Latin squares by restricting to the defined entries.

Algorithm 1 Generate L from contour C and suitable autotopism θ.

Require: contour C, suitable autotopism θ
  set L to be the empty $n \times n$ matrix
  for each entry e in C **do**
    for $t \in \{0, 1, \ldots, n/2 - 1\}$ **do**
      add entry $\theta^t(e)$ to L
    end for
  end for
  return L

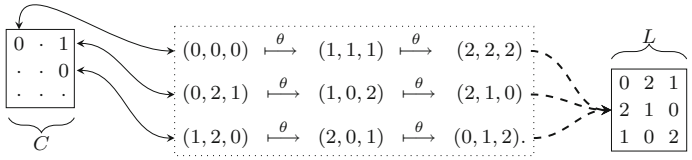

Fig. 1 Illustrating how to recover a Latin square L from an example contour C and an autotopism $\theta = \big((012), (012), (012)\big)$

We can reconstruct a Latin square L from knowledge of an autotopism θ and a partial Latin square with exactly one entry in each orbit in $O(L)$ under $\langle\theta\rangle$ (the group generated by θ); we call this partial Latin square a *contour* C. In this case, we say (C, θ) *generates* L.

Generating the Latin square from (C, θ) Algorithm 1 describes how to generate the Latin square from (C, θ) for an arbitrary suitable autotopism θ. An example is illustrated in Fig. 1.

Algorithm 1 fills in n^2 cells, so takes time $O(n^2)$, but it assumes that (C, θ) generates a Latin square L. Naively, it would take $O(n^3 \log n)$ time to verify that L is indeed a Latin square: for each of the $O(n)$ rows and columns, there are $O(n^2)$ pairs of entries, and we compare their symbols for equality, requiring time $O(\log n)$.

Verifying (C, θ) *generates a Latin square* While it takes time $O(n^3 \log n)$ to verify a matrix is a Latin square, we can verify that (C, θ), for suitable θ, generates a Latin square (without constructing it) in time $O(n^2 \log n)$, as we will now explain.

If $\theta = (\alpha, \beta, \gamma)$ is a suitable isotopism, and α and β can be written in disjoint cycle notation $(a_1 a_2 \cdots a_{n/2})(a'_1 a'_2 \cdots a'_{n/2})$ and $(b_1 b_2 \cdots b_{n/2})(b'_1 b'_2 \cdots b'_{n/2})$, respectively, then we choose the *leading entries* $\ell_1^{\text{row}} = a_1$, $\ell_2^{\text{row}} = a'_1$, $\ell_1^{\text{col}} = b_1$, and $\ell_2^{\text{col}} = b'_1$. Figure 2 illustrates leading entries via an example.

There is flexibility in the choice of leading entry (based on how we write the cycles), but to be concrete, we assume $\ell_1^{\text{row}} = \ell_1^{\text{col}} = 0$ and that ℓ_2^{row} and ℓ_2^{col} are the minimum elements in their respective cycles. The rows indexed by ℓ_1^{row} and ℓ_2^{row} are the *leading rows* and the columns indexed by ℓ_1^{col} and ℓ_2^{col} are the *leading columns*.

To verify that (C, θ) generates a Latin square, it is sufficient to check that there would be no clashes in the leading rows and columns; if there were a clash in a non-leading row or column, the action of $\langle\theta\rangle$ would imply one of the leading rows or columns also contains a clash.

Algorithm 2 describes how we can check the leading rows and columns for clashes without generating the whole Latin square. We can perform this in (worst case) time $O(n^2 \log n)$: for each entry we compute the entry in its orbit in row ℓ_1^{row} or ℓ_2^{row} and column ℓ_1^{col} or ℓ_2^{col}, and check that no clash arises with earlier inspected entries. Thus we check equality (requiring time $O(\log n)$) of $4\binom{n}{2} = O(n^2)$ pairs of entries.

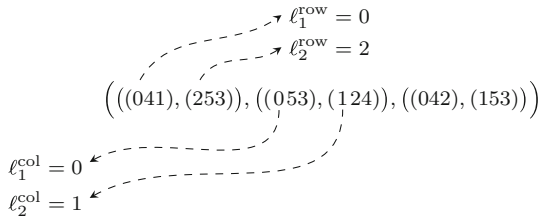

Fig. 2 Highlighting the leading entries ℓ_1^{row}, ℓ_2^{row}, ℓ_1^{col}, and ℓ_2^{col} in an example suitable isotopism

Algorithm 2 Verify that a contour C and suitable autotopism θ generate a Latin square.

Require: contour C with $2n$ entries, suitable autotopism θ
identify leading entries ℓ_1^{row}, ℓ_2^{row}, ℓ_1^{col}, ℓ_2^{col}

set $R_{\text{col}}^{(1)} = R_{\text{col}}^{(2)} = R_{\text{sym}}^{(1)} = R_{\text{sym}}^{(2)} = C_{\text{row}}^{(1)} = C_{\text{row}}^{(2)} = C_{\text{sym}}^{(1)} = C_{\text{sym}}^{(2)} = (\overbrace{0, 0, \ldots, 0}^{n})$.
for each entry e in C **do**
  find $(\ell_q^{\text{row}}, j, k) = \theta^t(e)$ for some $t \in \{0, 1, \ldots, n/2 - 1\}$
  if $R_{\text{col}}^{(q)}[j] = 1$ or $R_{\text{sym}}^{(q)}[k] = 1$ **then**
    return false
  else
    $R_{\text{col}}^{(q)}[j] \leftarrow 1$ and $R_{\text{sym}}^{(q)}[k] \leftarrow 1$
  end if
  find $(i, \ell_q^{\text{col}}, k) = \theta^t(e)$ for some $t \in \{0, 1, \ldots, n/2 - 1\}$
  if $C_{\text{row}}^{(q)}[i] = 1$ or $C_{\text{sym}}^{(q)}[k] = 1$ **then**
    return false
  else
    $C_{\text{row}}^{(q)}[i] \leftarrow 1$ and $C_{\text{sym}}^{(q)}[k] \leftarrow 1$
  end if
end for
return true

1.2 Previous work

1.2.1 Critical set scheme

Cooper et al. [10] proposed a secret sharing scheme based on critical sets of Latin squares. Related secret sharing schemes were proposed in [5,6,13,19] which use other combinatorial objects.

A Latin square $L = (l_{i,j})$ is described as a *completion* of the partial Latin square $P = (p_{i,j})$ if, for all $i, j \in \mathbb{Z}_n$, either $p_{i,j} = l_{i,j}$ or $p_{i,j}$ is undefined. A *critical set* of a Latin square L is a partial Latin square P that has a unique completion L, and any partial Latin square P' with $O(P') \subsetneq O(P)$ admits more than one completion. Figure 3 gives an example of a critical set (sourced from [10]).

Fig. 3 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	·

In [10], the secret is a Latin square L of order n. A critical set S is found, and split into l shares (each of cardinality less than $|S|$) and one share is given to each of the l participants. The key idea is that without full knowledge of S, that is, without the cooperation of every participant, we cannot uniquely recover the Latin square L (since any proper subset of S admits multiple completions). There have been several issues raised with this secret sharing scheme, which we list and expand upon below:

Why a Latin square? There have been many proposed secret sharing schemes using a variety of combinatorial objects as secrets [22]; why would we want a secret Latin square? The number of Latin squares grows rapidly [17,23] and they have some useful redundancy, but so do many other combinatorial objects.

Verification If the participants cooperate and recover a Latin square X, how can they be sure that $X = L$, the secret Latin square? It may be that one or more of the returned shares is erroneous due to e.g. transmission error or deceit.

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 [8] (determining if a partial Latin square admits a completion is NP-complete [9]). A possible workaround for this issue is restricting to certain classes of Latin squares where working with their critical sets is significantly easier [11]. However, this also makes attacking the scheme easier.

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

A subtle "flaw" It was shown in [11] that it is possible that the critical set C could be discovered if a large proportion of the participants collude (see also [16]). Worse still, it is conceivable that if enough participants collude, then they could deceive the remaining participant(s) into believing a false message (such a tactic was raised in [27] regarding Shamir's secret sharing scheme).

Additionally, a proper subset of a critical set is assumed only to admit ≥ 2 completions. It may be possible to find a short list of completions that are consistent with a given collection of shares.

Multi-level scheme A multi-level version of the critical set scheme was also considered in [10]. In it, several critical sets are instead distributed to the participants in such a way that some prescribed subsets of the l participants could jointly access some critical set of L without the cooperation of the remaining participants. However, implementing this generalized scheme would require careful selection of the Latin square L, its critical sets, and how the critical sets are partitioned.

1.2.2 Variations on the critical set scheme

Attempts have been made at modifying the critical set scheme to overcome (some of) these problems.

Chum and Zhang [8] (see also [7]) presented a scheme where cryptographic hash values can be pooled to recover a secret via a so-called Nostradamus attack; they describe the application of this method to a secret 10×10 partial Latin square with a unique completion.

Fitina and Lal [12] instead attempt to overcome some of these problems by using "transformed" critical sets. A related idea of transforming critical sets was mentioned in [8] involving a modification of a technique of Chaudhry et al. [6] to Latin squares: simply make the sum of the shares be equal to the critical set.

1.2.3 An earlier autotopism scheme

A secret sharing scheme involving Latin square autotopisms was briefly mentioned by Ganfornina [15] (see also [14]). However, implementation, complexity, and security of this scheme were not analyzed. Moreover, the method we propose differs in two key aspects: (a) Instead of having a secret Latin square that admits an autotopism, we have a secret autotopism (and we use the Latin square for verification). (b) We enforce particular cycle structures for the autotopism; this allows a concrete theoretical analysis.

2 The proposed secret sharing scheme

2.1 Initialization

We begin in the *initialization* phase, illustrated in Fig. 4. A *dealer* (who will be a certified authority) constructs a random contour C for a random suitable isotopism θ for which (C, θ) generate a Latin square L. The dealer then generates l isotopisms $\sigma_1, \sigma_2, \ldots, \sigma_l$ uniformly at random such that their product is θ; these isotopisms are the shares. The dealer also computes $\xi := \sigma_l \sigma_{l-1} \cdots \sigma_1$ and makes public $C_{\text{public}} = \xi(C)$. Afterwards, the dealer retains no knowledge.

We propose performing this using a four-step protocol. We will consider Latin squares that admit suitable autotopisms which requires n to be even. However, we will further require $n \equiv 2 \pmod 4$ for the proposed construction to work.[1]

Note that at no point do we actually need to construct a whole Latin square, as they will be determined by a contour and an autotopism. However, for the readers' sake, we include the Latin squares in our description.

Step 1: A random contour for a Latin square with the special autotopism Let $\zeta = (\tau, \tau, \tau)$ where $\tau := (0, 1, \ldots, n/2 - 1)(n/2, n/2 + 1, \ldots, n - 1)$. We can randomly generate an $n \times n$ partial Latin square $D = (d_{i,j})$ for which (D, ζ) generates a Latin square that admits the autotopism ζ by randomly choosing $z_i \in \{0, n/2\}$ for each $i \in \{0, 1, \ldots, n/2 - 1\}$, and setting

$$\begin{aligned} d_{n/2-1-i,i} &= d_{n-1-i,n/2+i} = z_i, \text{ and} \\ d_{n-1-i,i} &= d_{n/2-1-i,n/2+i} = n/2 - z_i. \end{aligned}$$

Theorem 21 below ensures that (D, ζ) generates a Latin square, which we call L_{prior}. Note that Theorem 21 implies that the construction only works when $n \equiv 2 \pmod 4$.

This process is illustrated in an example below.

[1] A more complicated construction would work in the $n \equiv 0 \pmod 4$ cases, but, for simplicity, we restrict to $2 \pmod 4$.

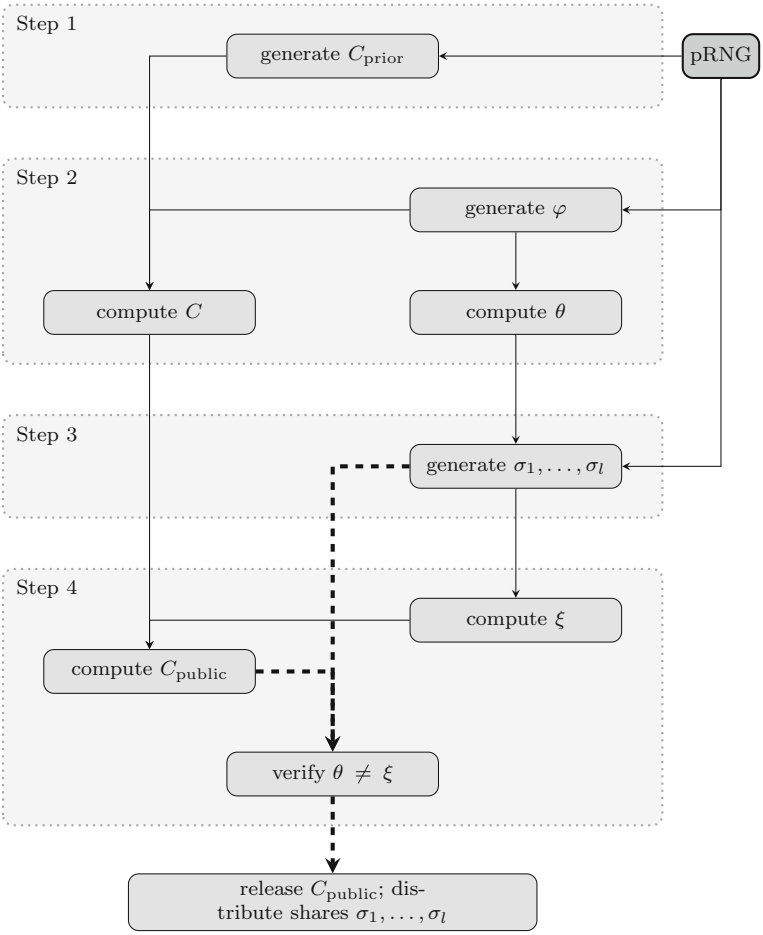

Fig. 4 Flow chart of the proposed secret sharing scheme: initialization phase

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

We name the blocks of D as

$$D = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}.$$

Theorem 21 *Let $D = (d_{i,j})$ be a partial Latin square of order n containing exactly $2n$ defined entries, with every symbol belonging to $\{0, n/2\}$. Then (D, ζ) generates a Latin square if and only if*

- *each block, M_{11}, M_{12}, M_{21} and M_{22}, contains precisely $n/2$ entries and*
- *if $(i, j, d_{i,j})$ and $(i', j', d'_{i,j})$ are two distinct entries within a single block, then $j - j' \not\equiv i - i' \pmod{n/2}$.*

Proof This is a special case of a theorem by [24, pp. 111–112]. □

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. This operation ensures $(C_{\text{prior}}, \zeta)$ generates L_{prior} also. In the earlier example, we might retain

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

where, for example, the entry $(3, 2, 3)$ is replaced by $\zeta^2(3, 2, 3) = (5, 1, 5)$.

In many cases, finding a Latin square that admits a given autotopism is difficult, particularly when n is large. Moreover, often an arbitrary isotopism is not an autotopism of any Latin square of order n [26]. This motivates our choice to use the specific cycle structure.

Step 2: Randomly generate 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 $\theta := \varphi\zeta\varphi^{-1}$. We compute the contour $C := \varphi(C_{\text{prior}})$ of L by applying the isotopism φ to C_{prior}. Moreover, θ is conjugate to ζ, so each component has the same cycle structure (that is, two $(n/2)$-cycles) [3, p. 25].

If we apply the random isotopism

$$\varphi = \big((041352), (124), (1325)\big)$$

to the earlier example, we obtain the Latin square

$$L = \varphi(L_{\text{prior}}) = \left[\begin{array}{cccccc} 0 & 1 & 5 & 2 & 4 & 3 \\ 4 & 2 & 0 & 3 & 1 & 5 \\ 2 & 5 & 1 & 0 & 3 & 4 \\ 3 & 0 & 2 & 4 & 5 & 1 \\ 1 & 4 & 3 & 5 & 0 & 2 \\ 5 & 3 & 4 & 1 & 2 & 0 \end{array}\right]$$

which admits the autotopism

$$\begin{aligned} \theta &= \varphi\zeta\varphi^{-1} \\ &= \big((043)(125), (024)(153), (035)(124)\big). \end{aligned}$$

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.

Step 3: Splitting the autotopism We generate the shares $\sigma_1, \sigma_2, \ldots, \sigma_l$, each of which will be random isotopisms for which $\sigma_1\sigma_2\cdots\sigma_l = \theta$. We can generate σ_1 to σ_{l-1} uniformly at random, then compute $\sigma_l = \sigma_{l-1}^{-1}\cdots\sigma_1^{-1}\theta$.

So, in our example, if $l = 4$, we might generate and compute:

$$\begin{aligned}\sigma_1 &= \big((04)(15), (04531), (051)(243)\big)\\ \sigma_2 &= \big((04)(1352), (025), (013452)\big)\\ \sigma_3 &= \big((01325), (01354), (15)(24)\big)\\ \sigma_4 &= \big((14352), (02531), (052143)\big).\end{aligned}$$

So $\theta = \sigma_1\sigma_2\sigma_3\sigma_4$.

Step 4: Making public a contour We compute $C_{\text{public}} := \xi(C)$ where $\xi := \sigma_l\sigma_{l-1}\cdots\sigma_1$. In our running example, we have the situation

$$\xi = \big((03)(1452), (031)(254), (0243)(15)\big)$$

and so

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

For security, we check if $\theta = \xi$. If this occurs, we restart the initialization from scratch.

Share distribution After the completion of all Steps 1–4, the i-th isotopism σ_i is given to the i-th participant, and we make public $C_{\text{public}} = \xi(C)$.

We're not obligated to make C_{public} public; for increased security, we could instead e.g. split it into l parts and add these parts to the shares.

2.2 Recovery

When all l participants decide to cooperate, we enter the *reconstruction* phase. The l participants securely send the shares $\tilde{\sigma_1}, \tilde{\sigma_2}, \ldots, \tilde{\sigma_l}$ to a *combiner*. The shares may or may not be returned correctly: if share i is correctly sent, we have $\tilde{\sigma_i} = \sigma_i$.

Table 1 Time complexity of the initialization phase

Construct	RN_1	RN_2	Time
C_{prior}	$n/2$	$2n$	$O(n \log n)$
$\varphi, \sigma_1, \sigma_2, \ldots, \sigma_{l-1}$	$3l(n-1)$		$9l(n-1)$
θ			$6n$
σ_l			$6n(l-1)$
ξ			$3n(l-1)$
C_{public}			$6n$
	$O(ln)$	$O(n)$	$O(ln + n \log n)$

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

Afterwards, the dealer retains no knowledge.

It will be assumed that $\theta_{\text{cand}} = \theta$ if $(C, \theta_{\text{cand}})$ generates a Latin square. In Sect. 3.2, we will analyze the chance of this happening without $\theta_{\text{cand}} = \theta$, along with other security matters.

3 Analysis

3.1 Overhead

There are some potential bottlenecks with this scheme: (a) number of memory accesses, (b) number of transfers and transfer size, and (c) random number generation.

Our unit of time complexity will be a memory read/write; we separate random number generation from this calculation. We note that both isotopism multiplication and inversion can be performed in time $3n$. Acting on a contour with an isotopism can be performed by acting entry-by-entry, requiring time $6n$ in total.

3.1.1 Initialization

We describe the computational complexity of the initialization phase in Table 1. Here RN_1 refers to the number of binary random numbers required and RN_2 refers to the number of random numbers in $\{0, 1, \ldots, n/2-1\}$ required.[2] We detail how Table 1 comes about below.

To generate C_{prior}, we need to generate $n/2$ random binary numbers to determine $d_{(n/2-1-i)i}$ for $i \in \{0, 1, \ldots, n/2-1\}$, and $2n$ random numbers in $\{0, 1, \ldots, n/2-1\}$ to determine which entry in each orbit is retained. To calculate the entries in C_{prior}, we will need to add $6n$ pairs of numbers modulo $n/2$. However, addition of these numbers will always result in a number in $\{0, 1, \ldots, n/2-1\}$, so we need only subtract $n/2$ in some cases, individually requiring time $O(\log n)$. This requires $O(n \log n)$ time in total.

[2] In practice, we would make fewer than RN_1 and RN_2 calls to a pseduo-random number generator (pRNG); e.g. $n/2$ random bits could be equally obtained by a single call to a pRNG that generates a random number in $\{0, \ldots, 2^{n/2}-1\}$.

We generate the random isotopisms $\varphi, \sigma_1, \sigma_2, \ldots, \sigma_{l-1}$ uniformly at random using a Fisher–Yates Shuffle, requiring the computation of $3l(n-1)$ random binary numbers and performing $3l(n-1)$ swaps (a swap requiring time 3).

We compute

(a) $\theta = \varphi\zeta\varphi^{-1}$, requiring 1 isotopism inversion and 2 isotopism multiplications,
(b) $\sigma_l = \sigma_{l-1}^{-1} \cdots \sigma_1^{-1}\theta$, requiring $l-1$ isotopism inversions and $l-1$ isotopism multiplications,
(c) $\xi = \sigma_l\sigma_{l-1}\cdots\sigma_1$, requiring $l-1$ isotopism multiplications, and
(d) $C_{\text{public}} = \xi(C)$, requiring $6n$ time.

Thus, we require $O(ln)$ random numbers and $O(ln + n\log n)$ time in the initialization phase. There are $3n$ numbers from $\mathbb{Z}_n$ transferred to each participant (the shares), and $6n$ numbers from $\mathbb{Z}_n$ are made public (C_{public}).

In comparison with [10], in the proposed scheme, the length of each share σ_i is linear in n, whereas the size of the smallest critical sets in Latin squares of order n is conjectured to be $\lfloor n^2/4 \rfloor = \Theta(n^2)$ (see e.g. [4]). The computational complexity of generating critical sets that could be used in a secret sharing scheme is unclear.

3.1.2 Recovery

In the reconstruction phase, the participants transfer $\sigma_1, \sigma_2, \ldots, \sigma_l$ to a certified authority: so $3n$ numbers from $\mathbb{Z}_n$ are transferred. The contour C_{public} is retrieved, comprising $6n$ numbers from $\mathbb{Z}_n$.

The shares are then multiplied together to find θ and ξ, requiring time $6n(l-1)$. We compute ξ^{-1} in time $3n$. We compute $C = \xi^{-1}(C_{\text{public}})$ in time $6n$. We verify that L_{cand} is a Latin square using Algorithm 2, which requires time $O(n^2 \log n)$.

Thus we take time $O(ln + n^2 \log n)$ in the recovery phase.

3.1.3 Remark about practical computation

We remark that the operations required to implement this scheme are simple (mostly random number generation, modular arithmetic, and table lookups). Moreover, the modular additions can be performed as a conditioned regular addition; e.g. typically we want to compute $a \pmod{n/2}$ where $a \in \{0, 1, \ldots, n-1\}$, so, if $a \geq n/2$, then we replace a with $a - n/2$, and no change is required otherwise.

3.2 Security

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 isotopisms σ_i is of no more use in recovering θ or C than is a random suitable isotopism. No partial information about the secret θ is revealed.

Brute-force attack An attacker could attempt to crack the scheme by brute force, e.g., by computing all Latin squares that admit the autotopism ζ, and testing all isotopic Latin squares for correctness.

The number of Latin squares that admit the autotopism ζ was computed in [25] in small cases, given in Table 2. The number of suitable isotopisms is $(2(n-1)!/n)^3$ (see "Number of suitable isotopisms" in Appendix), which we also include in Table 2. These numbers grow very rapidly, so a brute force attack, even for $n = 10$, seems prohibitively expensive.

Table 2 For small $n \equiv 2 \pmod 4$, the number of Latin squares that admit the autotopism ζ, the number of suitable isotopisms, and a lower bound on the size of the isotopism class of any order-n Latin square L

n	nr LS with autotop. ζ	nr suitable isotop.	is(L) lower bound
6	648	6×10^4	2×10^5
10	20,820,000	3×10^{14}	4×10^{14}
14	?	7×10^{26}	1×10^{27}
18	?	6×10^{40}	7×10^{39}

Attack by finding a completion of C_{public} If an attacker managed to find L, they could compute its autotopism group using the method in [18], 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 $1/\text{is}(L)$; a lower bound on is(L) is listed in Table 2 (see "Autotopism group and isotopism class sizes" in Appendix). 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 as we just noted, even full knowledge of the isotopism class is of limited use. This is a benefit to the schemes proposed in [10,15] (if left unmodified) which give away partial information about the secret.

Attack by replacing shares One or several participants might return incorrect shares. Typically this would result in the scheme returning `fail` during the reconstruction phase. If it didn't, it would require that θ_{cand} be a suitable isotopism, and that the generated matrix L_{cand} is a Latin square. If this happens to occur, then the isotopism $\theta_{\text{cand}} \neq \theta$ is returned.

If used as a method of attack, it would be particularly dangerous e.g. in the case of a malicious first or last participant, since, if it succeeds, they could use θ_{cand} to recover θ. For example, if $\tilde{\sigma_1}$ was the only erroneous share returned, then $\theta = \sigma_1 \tilde{\sigma_1}^{-1} \theta_{\text{cand}}$, so participant 1 could recover the real secret, while the remaining participants believe a false secret. We argue that this attack is unlikely to succeed. Specifically, the ability of the combiner to check whether or not θ_{cand} is correct (by checking that L_{cand} is a Latin square) effectively prevents this kind of attack, and is a key reason for the proposed Latin square autotopism scheme (instead of, say, a simpler scheme were participants are assigned random permutations whose product is the secret).

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

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

(see "Probability of suitability" in Appendix for the details).

Obstacle 2 Let p denote the probability of a successful attack assuming Obstacle 1 is overcome, where C is distributed according to the scheme and θ_{cand} is distributed uniformly at random from the set of all suitable isotopisms. We have the lower bound

$$p \geq \frac{n^6 \varphi(n/2)}{8n!^3} \tag{1}$$

Table 3 Estimate that a random contour given by the scheme and a random suitable autotopism generate a Latin square

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

where φ is the Euler phi function, since the right-hand side is the probability of $\theta_{\text{cand}} = \theta^t$ for t coprime to $n/2$. We expect (1) to be not too far away from the correct p-value.[3] Such small probabilities are difficult to estimate empirically, so we give estimates for p only for $n \in \{6, 10\}$, as given in Table 3. The experimental details are given in "Probability $(C, \theta_{\text{cand}})$ generates a Latin square, when θ_{cand} is random" in Appendix.

For sensibly chosen n, this attack will likely require millions of repeated attempts, each of which requiring all participants to return their shares for reconstruction.

Initialization with $\theta = \xi$ There is a chance that $\theta = \xi$, i.e., $\sigma_1\sigma_2 \cdots \sigma_l = \sigma_l\sigma_{l-1} \cdots \sigma_1$, which could conceivably be of use to an attacker. To avoid this, if $\theta = \xi$ occurs we restart from scratch. The probability of this occurring for $l = 2$ is small: e.g. 3×10^{-5} for $n = 6$, and 2×10^{-14} for $n = 10$ (see "Random isotopisms commuting" in Appendix for the details). Experiments suggest this is close to the probability of $\theta = \xi$ for larger l, as we would expect. This ensures that we won't need a large number of restarts in the initialization phase.

3.3 Multi-level scheme

In a *multi-level* secret sharing scheme, we have a family of subsets of the participants $\{P_j\}_{j=1}^k$, say, where if the participants in any P_j pool their shares, they can recover the secret. A common example is a threshold scheme, where the family would consist of all t-subsets of the participants.

Generalizing the proposed secret sharing scheme to a multi-level scheme is straightforward and can be performed "on the fly", i.e., without changing the secret and restarting the entire scheme. If the participants belonging to P_j for some $j \in \{1, 2, \ldots, k-1\}$ decide to extend the scheme to include P_k, then they securely transmit the shares to a certified authority, who reconstructs θ and performs Step 3 again: generating the shares $\sigma_1^{(k)}, \sigma_2^{(k)}, \ldots, \sigma_{|P_j|}^{(k)}$, and distributes them to the participants in P_k accordingly. Afterwards, the certified authority retains no knowledge.

This is a significant advantage over the Latin square schemes of [10,15], which (a) are not naturally extensible to multi-level schemes, (b) give partial information about the secret, and hence more shares imply a larger proportion of the secret is revealed, and (c) would not be able to be performed on the fly (the participants would need to restart from scratch).

One benefit of a multi-level scheme is as a safeguard against lost shares. In a single-level scheme, a lost share would mean the secret is irretrievable, whereas in a multi-level scheme, if a share is lost, some other group of participants can still recover the secret (or a certified authority could regenerate new shares for the group that has lost a share).

[3] We won't have equality in (1) as there are situations where C generates two distinct Latin squares when paired with two distinct autotopisms; see "One contour generating two Latin squares" in Appendix for an example.

4 Concluding remarks

We present a Latin square autotopism-based secret sharing scheme which resolves a multitude of issues arising in previous proposed Latin square secret sharing schemes. These problems are overcome primarily by having a secret autotopism (symmetry), rather than a secret Latin square. Motivated by the significant criticism Latin square secret sharing schemes have received in the past, we perform a careful analysis of the proposed scheme, in terms of both complexity and security.

This work further demonstrates that Latin squares can be useful for designing practical secret sharing schemes, and could provide a practical alternative to established schemes. The proposed scheme also has a substantial benefit over traditional schemes: verification, i.e., the participants can be sure the returned secret is indeed the correct secret.

One future direction this research could take is to look for ways of identifying which shares are incorrect, in the situation that incorrect shares are returned. One method would be to make public cryptographic hash function value of the shares, although this (a) seems to be "overkill" for the problem at hand, and (b) could be used for any secret sharing scheme. Another possibility is encoding random entries of the Latin square L into the shares, however this opens up new possible security vulnerabilities.

Acknowledgments The authors would like to thank Tom McCourt for feedback and assistance tracking down references. Thanks also to Jinjin Sun for proofreading this paper. Lin and Stones were supported by NSFC Grant 61170301. Stones was also partly supported by AARMS, and partially supported by her NSF China Research Fellowship for International Young Scientists (Grant Number 11450110409). Lin thanks the Tianjin Key Lab of Intelligent Computing and Novel Software Technology and Key Laboratory of Computer Vision and System, Ministry of Education for their support. Stones recognizes the use of the online math forum `math.stackexchange.com` to discuss topics arising in this work.

Compliance with ethical standards

Conflicts of interest The authors declare no potential conflicts of interest.

Research involving human and animal rights The research conducted did not involve human participants nor animal testing.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix: Technical comments

Number of suitable isotopisms

The number of permutations in S_n with the cycle structure $(n/2) + (n/2)$ is given by $n!/\left(2!(n/2)^2\right) = 2(n-1)!/n$. Thus the number of suitable isotopisms is $(2(n-1)!/n)^3$.

Autotopism group and isotopism class sizes

By the Orbit-Stabilizer Theorem, we know the size of the isotopism class of the Latin square L is given by $\text{is}(L) = n!^3/\text{aut}(L)$, where $\text{aut}(L)$ is the size of the autotopism group of L. The maximum size of $\text{aut}(L)$ in any order-n Latin square is $n^{2+\lfloor \log_2 n \rfloor}$ [2], which enables us to give a lower bound on $\text{is}(L)$.

Probability of suitability

We know that permutations with the cycle structure $(n/2)+(n/2)$ are even permutations. Of the $n!/2$ even permutations, precisely $2(n-1)!/n$ have the cycle structure $(n/2)+(n/2)$. Thus, if we pick an isotopism uniformly at random from $A_n \times A_n \times A_n$ (where A_n is the alternating group), it has probability

$$\left(\frac{2(n-1)!/n}{n!/2}\right)^3 = 64/n^6$$

of being suitable.

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.

For $n = 6$, we generate 10^9 pairs $(C_{\text{prior}}, \beta)$, 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" [28], we can be 99.995 % confident that $p \leq 7.6/N \approx 2 \times 10^{-11}$.

Random isotopisms commuting

The probability that two random isotopisms with components that are even permutations commute is $(2p_n/n!)^3$, where p_n is the number of partitions of n. This is a special case of a general result in group theory: the probability of two random finite group elements commuting is the number of conjugacy classes divided by the size of the group).

$(C, \theta_1) \mapsto$

5	(4)	(0)	2	(1)	(3)
1	3	5	4	(0)	2
3	2	(4)	0	5	1
2	(1)	3	5	4	(0)
4	0	2	1	3	(5)
0	(5)	(1)	3	2	(4)

$\theta_1 = ((012)(345), (012)(345), (012)(345))$

$(C, \theta_2) \mapsto$

2	(4)	(0)	5	(1)	(3)
4	3	5	1	(0)	2
0	2	(4)	3	5	1
5	(1)	3	2	4	(0)
1	0	2	4	3	(5)
3	(5)	(1)	0	2	(4)

$\theta_2 = ((012)(345), (045)(123), (012)(345))$

Fig. 5 A contour C (formed by the circled entries) for which (C, θ_1) and (C, θ_2) generate two distinct Latin squares

One contour generating two Latin squares

It is possible that a contour C generates a Latin square under the action of two isotopisms: Fig. 5 gives an example. This example swaps a pair of columns which don't intersect the contour; in this way we can generate larger examples.

References

1. Blakey G.R.: Safeguarding cryptographic keys. In: Proceedings of NCC, vol. 48, pp. 313–317 (1979).
2. Browning J., Stones D.S., Wanless I.M.: Bounds on the number of autotopisms and subsquares of a Latin square. Combinatorica **33**, 11–22 (2013).
3. Cameron P.J.: Permutation Groups. Cambridge University Press, Cambridge (1999).
4. Cavenagh N.J.: A superlinear lower bound for the size of a critical set in a Latin square. J. Combin. Des. **15**(4), 269–282 (2007).
5. Chaudhry G., Seberry J.: Secret sharing schemes based on room squares. In: Proceedings of DMTCS '96—Combinatorics, Complexity and Logic, pp. 158–167 (1996).
6. Chaudhry G., Ghodosi H., Seberry J.: Perfect secret sharing schemes from room squares. J. Comb. Math. Comb. Comput. **28**, 55–61 (1998).
7. Chum C.S., Zhang X.: The Latin squares and the secret sharing schemes. Groups Complex. Cryptol. **2**(2), 175–202 (2010).
8. Chum C.S., Zhang X.: Improved Latin square based secret sharing scheme. Comput. Comb. Group Theory Cryptogr. **582**, 51–64 (2012).
9. Colbourn C.J.: The complexity of completing partial Latin squares. Discret. Appl. Math. **8**(1), 25–30 (1984).
10. Cooper J., Donovan D., Seberry J.: Secret sharing schemes arising from Latin squares. Bull. Instrum. Comb. Appl. **12**, 33–43 (1994).
11. Donovan D.M., Lefevre J.G., McCourt T.A., Cavenagh N.J., Khodkar A.: Identifying flaws in the security of critical sets in Latin squares via triangulations. Australas. J. Comb. **52**, 243–268 (2012).
12. Fitina L.F., Lal S.P.: Access schemes based on perfect critical set partitions and transformations. Australas. J. Comb. **34**, 229–237 (2006).
13. Gamble G., Maenhaut B.M., Seberry J., Street A.P.: Further results on strongbox secured secret sharing schemes. Util. Math. **66**, 165–193 (2004).
14. Ganfornina R.M.F.: Decomposition of principal autotopisms into triples of a Latin square. In: Book of Abstracts of the Tenth Meeting on Computer Algebra and Applications, pp. 95–98 (2006).
15. Ganfornina R.M.F.: Latin squares associated to principal autotopisms of long cycles. Application in cryptography. In: Proceedings of Transgressive Computing, pp. 213–230 (2006).
16. Grannell M.J., Griggs T.S., Street A.P.: A flaw in the use of minimal defining sets for secret sharing schemes. Des. Codes Cryptogr. **40**(2), 225–236 (2006).
17. McKay B.D., Wanless I.M.: On the number of Latin squares. Ann. Comb. **9**, 335–344 (2005).
18. McKay B.D., Meynert A., Myrvold W.: Small Latin squares, quasigroups, and loops. J. Comb. Des. **15**, 98–119 (2007).
19. Seberry J., Street A.P.: Strongbox secured secret sharing schemes. Util. Math. **57**, 147–163 (2000).
20. Shamir A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979).
21. Simmons G.J.: An introduction to shared secret and/or shared control schemes and their applications. In: Contemporary Cryptology, the Science of Information Integrity, pp. 441–497. IEEE Press, New York (1991).
22. Stinson D.R.: An explication of secret sharing schemes. Des. Codes Cryptogr. **2**(4), 357–390 (1992).
23. Stones D.S.: The many formulae for the number of Latin rectangles. Electron. J. Comb. **17**(A1) (2010).
24. Stones D.S.: On the number of Latin rectangles. Ph.D. Thesis, Monash University (2010). http://arrow.monash.edu.au/hdl/1959.1/167114.
25. Stones D.S.: The parity of the number of quasigroups. Discret. Math. **310**(21), 3033–3039 (2010).
26. Stones D.S., Vojtěchovský P., Wanless I.M.: Cycle structure of autotopisms of quasigroups and Latin squares. J. Comb. Des. **20**, 227–263 (2012).
27. Tompa M., Woll H.: How to share a secret with cheaters. IBM Res. Rep. RC 11840 (Log #52910) (1986).
28. Wikipedia: Rule of three (statistics). http://en.wikipedia.org/wiki/Rule_of_three_(statistics). Accessed July 2014.