DOI：10.13718/j.cnki.xdzk.2025.11.017
张渝，李佳朔，李乾燃，等. 基于集合分配的高效安全门限秘密共享方案 [J]. 西南大学学报(自然科学版)，2025，47(11)：221-233.

基于集合分配的高效安全门限秘密共享方案

张渝[1]，　李佳朔[1]，　林已杰[2]，　李乾燃[3]

1. 西南大学 计算机与信息科学学院、软件学院，重庆 400715；
2. 西南大学 信息化建设办公室，重庆 400715；3. 重庆信安网络安全等级测评有限公司，重庆 401120

摘要：在信息安全领域，集中式密钥管理模式由于存在单点失效的风险而面临严峻挑战。虽然传统的门限方案通过分布式密钥管理有效解决了这一问题，但其依赖多项式插值的实现方式带来了高昂的计算开销，难以适应低资源场景的应用需求。针对这一核心问题，提出了一种基于集合分配(Allocation Scheme based on Combination，ASC)的安全门限方案(Secure Threshold Scheme based on ASC，STSA)。该方案采用离散集合划分替代多项式插值算法，通过 ASC 元素分配机制与哈希函数的协同作用构建秘密信息，实现算法复杂度的结构性优化。理论分析与实验验证表明，STSA 在满足(t，n)门限安全性的同时，将秘密恢复阶段的运算简化为集合交操作，实现线性计算复杂度。

关　键　词：隐私保护；集合分配方法；安全门限方案；隐私保护
中图分类号：TP311　　**文献标识码**：A
文 章 编 号：1673-9868(2025)11-0221-13　　开放科学(资源服务)标识码(OSID)：

A Secure (t, n) Threshold Scheme Based on Set Partition

ZHANG Yu[1]，　LI Jiashuo[1]，　LIN Yijie[2]，　LI Qianran[3]

1. *College of Computer and Information Science, School of Software, Southwest University, Chongqing 400715, China*;
2. *Information Construction Office, Southwest University, Chongqing 400715, China*;
3. *Chongqing Xinan Network Security Classified Testing and Evaluation Co. Ltd, Chongqing 401120, China*

Abstract: In the field of information security, the centralized key management model faces serious challenges due to the risk of single-point failure. Although the traditional (t, n) threshold scheme effectively solved this problem through distributed key management, but its implementation relying on polynomial interpolation brought the high computational overhead, which made it difficult to adapt to the application requirements of low-resource scenarios. To address this core problem, this paper proposes a Secure

收稿日期：2025-02-24
基金项目：国家自然科学基金(重点)项目(62032019)；国家重点研发计划项目(2023YFB3209000)。
作者简介：张渝，博士，教授，主要从事智能控制、机器学习、信息安全等的研究。
通信作者：林已杰，高级工程师。

Threshold Scheme based on ASC (STSA). The scheme breaks through the traditional cryptographic framework, adopts discrete set division instead of polynomial interpolation algorithm, and constructs the secret information through the synergy of ASC element allocation mechanism and hash function to realize the structural optimization of algorithm complexity. Theoretical analysis and experimental validation show that STSA simplified the operations in the secret recovery phase to set intersection operations and realized linear computational complexity while satisfying the (t, n) threshold security. This research provides a lightweight security solution for low-resource scenarios such as IoT terminals and edge computing. At the meantime, it opens the source algorithm implementation code, which has practical value for building an efficient secure computing ecosystem.

Key words: privacy protection; set allocation method; secure threshold scheme; privacy protection

密钥管理作为网络系统安全的基石，其可靠性直接影响整个系统的安全架构。当前主流的对称加密与非对称加密技术均采用单一实体集中保管密钥的模式，这种架构不仅存在密钥泄露的风险，还会因为管理节点被攻破而引发系统性安全崩塌。为应对这一挑战，门限密码学应运而生[1-2]。该技术通过将密钥分割为多个份额并分发给不同参与者，要求超过预设门限值的参与者协作方可恢复密钥，从根本上杜绝了单点失效风险。研究表明，门限方案在区块链[3]、物联网[4]、隐私计算等领域展现出独特优势，其安全性提升效果已在分布式存储、多方安全计算等场景中得到验证。

1979年，文献[1]提出了一种基于多项式插值的门限方案，该方案基于以下定理：给定二维平面的n个点(x_1, y_1)，…，(x_n, y_n)，这些点的x坐标都不相同，即$x_i \neq x_j$。如果$i \neq j$，$1 \leqslant i, j \leqslant n$，那么有且只有一个$t-1$阶多项式$q(x)$满足$q(x_i)=y_i$，$1 \leqslant i \leqslant n$[5]。对于秘密信息$K$，随机选取$t-1$阶多项式$q(x)=a_0+a_1x+a_2x^2+$，…$+a_{t-1}x^{t-1}$，使得$a_0=K$，然后计算$D_1=q(1)$，$D_2=q(2)$，…，$D_n=q(n)$。只要知道$D_1$，…，$D_n$中的$t$个，即可通过插值法找出多项式$q(x)$的系数，进而计算出$q(0)$，得到$K=q(0)$。如果只知道$D_1$，…，$D_n$中的$t-1$个或更少，则不能求出多项式$q(x)$的系数，因此不能计算出正确的秘密信息$K$。为突破传统门限方案的性能瓶颈，文献[6]提出基于向量空间的创新性门限方案。该方案通过线性代数框架重构秘密管理：首先构建d维向量空间，将原始秘密信息编码为特定向量；随后为每个参与者生成与该向量空间正交的子空间投影向量作为秘密份额，其数学本质是通过线性变换实现秘密信息的分割。当t个及以上参与者协作时，其持有的投影向量可通过线性组合重构原始向量空间，从而解码出初始秘密信息。相较于传统多项式插值方法，该方案依托向量空间的正交性特征，将秘密信息的恢复过程转化为高效的矩阵运算，其计算复杂度降低至$O(t^2)$量级。这种方案的优势在于其数学结构清晰，计算效率高，在分布式安全计算中具有重要应用。

1983年，文献[7]提出了一种基于中国剩余定理的方案(CRT)。该方案首先选择一个大整数N作为模数，并将其分解为互质的数n_1，n_2，…，n_k，满足$N=n_1 \times n_2 \times \cdots \times n_k$。对于秘密信息$SE$，计算$SE_i \equiv SE \bmod n_i$，并将$SE_i$分发给第$i$个参与者。当至少$t$个参与者提供各自的$SE_i$时，利用$CRT$可以唯一确定$SE \bmod N$，从而恢复秘密信息。该方案具有高安全性和计算效率，广泛应用于分布式系统中的安全计算和密钥管理。

在基础理论体系不断完备的同时，门限方案的设计范式逐步向多元化演进。从数学工具维度看，研究者相继提出了基于矩阵乘法的分布式验证架构[8]、依托多维向量投影的轻量化方案[9]，以及采用矢量空间线性组合的高效方法[10]。在新型计算模型方面，细胞自动机方案[11]展现了离散动力学系统的独特优势，而多级秘密共享机制[12-13]通过多项式插值扩展实现了细粒度权限控制。值得关注的是，单调跨度程序[14]与有

理正态曲线[15]等代数几何工具的引入，进一步丰富了方案的理论边界。2019 年后，随着量子计算技术的快速发展，传统的公钥密码体系面临着被量子算法破解的风险。因此，抗量子密码学(Post-Quantum Cryptography，PQC)成为密码学研究的核心攻关方向之一。在这一背景下，基于格密码(Lattice-based Cryptography)的方案[16]因其在理论上的安全性和实际应用中的潜力，受到了广泛关注。近 5 年的研究更呈现出跨学科融合态势：2021 年，文献[17]将 Shamir 阈值机制与区块链智能合约结合，通过 STCChain 方案实现密钥分片的防篡改存证；2023 年，文献[18]构建的线性哈希阈值协议，首次在离散对数与 RSA 双重假设下达成两轮通信安全认证；2024 年，文献[19]基于类群同态加密的创新设计解决了动态门限调整中的密钥更新难题，文献[20]提出的分层量子密钥共享方案通过引入理性参与者博弈模型并使用量子相移运算来分发和重构密钥，使方案能够抵御一系列典型的外部攻击，并能够检测参与者的伪造和串通攻击。

尽管已有研究取得显著进展，但现有方案在资源受限场景的适用性仍存在关键瓶颈，例如 Shamir 方案虽然具有信息论的安全特性，但是其 $O(t^2)$的插值复杂度难以满足实时性要求；Blakley 方案的高维矩阵运算产生了 $O(t^3)$的计算开销，制约其在边缘设备的部署；中国剩余定理方案在动态门限调整时面临模数重构难题。更本质地，现有方法普遍依赖连续数学空间操作，这种范式性特征导致算法复杂度与安全强度呈正相关，尚未出现能同时满足轻量化与强安全需求的新型理论框架。基于此，本文提出基于集合分配的安全门限方案 STSA(Secure Threshold Scheme based on ASC)，主要贡献包括：

1) 提出一种简单、易理解、易实现的基于集合分配的安全(t，n)门限方案。STSA 使用集合分配替代复杂数学结构，通过离散集合的逻辑划分替代多项式、超平面或向量空间，避免高次插值、矩阵运算等复杂操作。利用集合交集运算恢复秘密，复杂度可降至 $O(t)$。

2) 在数学上证明了提出的 STSA 门限方案的正确性和安全性，同时通过程序验证了该方案的正确性和可行性。

3) 提出的 STSA 方案为该领域引入了新的思路，是相关研究的补充和拓展。STSA 采用轻量化设计减少模数运算与浮点计算需求，适用于物联网终端、边缘设备等资源受限场景。集合分配机制可无缝集成至多方计算协议与零知识证明框架，为隐私计算生态提供基础工具。

4) 使用 Python 实现了 STSA，同时也复现了其他一些门限方案。实现的源代码都进行了公开(https://gitee.com/Tinff/stsa)，供其他学者对我们的工作进行验证，并希望激起更多的研究和创新。

1　基于集合分配的(t，n)门限方案

(t，n)门限方案是一种在 n 个参与者之间分配密钥的方法，任何 t(门限值，$t>1$)个或多于 t 个的参与者都可以合作重构密钥，如果少于 t 个则不能进行重构。

定义 1　参与者(Participant)

参与者是指参与门限方案的个人或其他实体(例如计算机、通信节点等)。

定义 2　秘密信息(Secret Information，SI)

(t，n)门限方案的秘密信息是需要私密传输的数据。可以通过一些方法将原始数据或在原始数据基础上计算出的派生信息分割为 n 个部分，分配给 n 个参与者。系统的密钥是一种秘密信息。

定义 3　分配密钥(Key Assigned，KA)

指(t，n)门限方案将密钥分配给各个参与者，任何不少于 t 个的参与者，将其持有的分配密钥 KA 组合在一起，可以恢复出秘密信息 SI。

1.1　设计思路

为了实现(t，n) 门限方案，可以选定一些元素 e_i 构成一个集合全集 $U=\{e_i\}$，$i=1$、2…，使集合

中的元素两两不同。集合的全体元素共同用于生成秘密信息。将集合中的元素 e_i 分配给 n 个参与者(p_1，p_2，…，p_n)，分得的集合分别为 S_1、S_2、…、S_n。n 个参与者组成的集合为 $S=\bigcup_j S_j$，$1\leqslant j\leqslant n$，每个元素 e_i 可以多次分配给不同的参与者。只需保证分配后 t 个或更多的参与者将各自持有的集合合并在一起，去除重复的元素后，能够恢复出全集 U。

因此，可以考虑将集合 U 中的元素 e_i 只分配给其中的 $t-1$ 个参与者，这样就保证在少于 t 个参与者的情况下，不能恢复出全集 U。而 t 个或更多参与者合作，能够恢复出全集 U。

1.1.1 ASC(Allocation Scheme based on Combination) 方法

步骤 1 随机选取两两不同的 $N=C_n^{t-1}$ 个元素组成集合 $U=\{e_i\}$，$1\leqslant i\leqslant N$。

步骤 2 记录 $N=C_n^{t-1}$ 种组合的排列 $C=[C_i]$，

$$C_i=\begin{bmatrix}c_{i,1}\\c_{i,2}\\\cdots\\c_{i,n}\end{bmatrix}$$

$$c_{i,j}=\begin{cases}1 & \text{该组合中 } p_j \text{ 被选出}\\0 & \text{没有被选出}\end{cases}$$

$$1\leqslant i\leqslant N,\ 1\leqslant j\leqslant n$$

步骤 3 将元素 e_i 分配给 p_j，当且仅当有序集合 C_i 中 $c_{i,j}=0$，即 p_j 分配到的元素组成的集合为 S_j。

$$S_j=\bigcup_{i=1}^{N}\{e_j\times(1-c_{i,j})\}-\{0\}$$

$$1\leqslant i\leqslant N,\ 1\leqslant j\leqslant n$$

示例：构造基于集合划分的(2，3)门限方案

步骤 1 选取两两不同的 $N=C_3^1=3$ 个元素组成全集 $U=\{7,11,23\}$。

步骤 2 记录 $N=C_n^{t-1}=3$ 种组合的有序排列：

$$C_1=\begin{bmatrix}1\\0\\0\end{bmatrix}\qquad C_2=\begin{bmatrix}0\\1\\0\end{bmatrix}\qquad C_3=\begin{bmatrix}0\\0\\1\end{bmatrix}$$

步骤 3 将元素 e_i 分配给 p_i，当且仅当有序集合 C_i 中 $c_i=0$，$1\leqslant i\leqslant 3$。

得到表 1 所示的分配方案。

表 1 分配方案

	$e_1=7$	$e_2=11$	$e_3=23$
p_1	×	√	√
p_2	√	×	√
p_3	√	√	×

分配的结果为：

p_1 分配得到的元素集合为 $S_1=\{11,23\}$

p_2 分配得到的元素集合为 $S_2=\{7,23\}$

p_3 分配得到的元素集合为 $S_3=\{7,11\}$

如果少于两个参与者则不能恢复出集合 U。

当不少于 2 时给出自己的集合，

$$S_1 \cup S_2 = S_1 \cup S_3 = S_2 \cup S_3 = S_1 \cup S_2 \cup S_3 = \{7, 11, 23\} = U$$

1.1.2　正确性证明

定理 1　ASC 方法能成功恢复全集 U。

证明　按照分配方法，集合 U 中的任意元素 e_i 分配给了 n 个参与者中的 $n-t+1$ 个参与者，$1 \leqslant i \leqslant N$。当不少于 t 个参与者给出所持有的集合时，至少其中有一个参与者具有元素 e_i。从而在它们组合的集合 U' 中会存在该元素。

因为以上元素 e_i 具有任意性，因此在集合 U' 中包含所有的元素 e_i，$1 \leqslant i \leqslant N$。故 $U' = U$。

定理 2　ASC 构造的门限方案是所用元素最少的方案。

证明　用反证法证明。

一方面，如果将全集 U 中的元素 e_k 整体去掉，得到集合 $U' = U - \{e_k\}$。

假设使用 ASC 方法分配元素后，去掉集合 U 中的一个元素 e_k，仍能满足(t, n)门限方案。

选取其中的 t 个参与者，其中只有一个参与者具有元素 e_k。不失一般性，设该参与者为 p_v，$1 \leqslant v \leqslant n$。

因为 t 个参与者所持有的集合，可以组合为集合 U，所以除去 p_v 持有的集合，其他 $t-1$ 个参与者中，必定可以组合出除了元素 e_k 之外的其他元素，即这 $t-1$ 个参与者能够组合出 $U' = U - \{e_k\}$。此时其门限值不再是 t，这与假设矛盾。

另一方面，如果从 ASC 方法产生的参与者原始集合 S 中将某一个元素 e_i 去掉，那么 S 中将存在 $n-t$ 个元素 e_i，有 t 个参与者没有分配到元素 e_i。如果刚好选中了没有分配到元素 e_i 的这 t 个参与者，则会因为缺少元素 e_i，从而无法计算出秘密信息。

综合以上两方面可见 ASC 方法是所用元素最少的。

例如，用 ASC 方法构造的(5，6)门限方案如表 2 所示，其中“√”表示分配，“×”表示未分配。

表 2　(5，6)门限方案

	p_1	p_2	p_3	p_4	p_5	p_6
e_1	×	×	×	×	√	√
e_2	×	×	×	√	×	√
e_3	×	×	×	√	√	×
e_4	×	×	√	×	×	√
e_5	×	×	√	×	√	×
e_6	×	×	√	√	×	×
e_7	×	√	×	×	×	√
e_8	×	√	×	×	√	×
e_9	×	√	×	√	×	×
e_{10}	×	√	√	×	×	×
e_{11}	√	×	×	×	×	√
e_{12}	√	×	×	×	√	×
e_{13}	√	×	×	√	×	×
e_{14}	√	×	√	×	×	×
e_{15}	√	√	×	×	×	×

若去除元素 e_2，$U'=U-\{e_2\}=\{e_1, e_3, \cdots, e_{15}\}$。若选择 p_1、p_2、p_3、p_5，则 $S_1 \cup S_2 \cup S_3 \cup S_5=\{e_1, e_3, \cdots, e_{15}\}=U'$。因此只要4个参与者给出所持的集合即可恢复出密钥集合 U'。这时不满足门限值为5的条件，故不能去除元素 e_2。同理也不能去除其他的元素。

推论 构造(t, n)门限方案至少需要 C_n^{t-1} 个元素。

证明 ASC方法将最少的元素个数分配给各个参与者。由定理1知，不能再缩减元素。因此，构造(t, n)门限方案至少需要 C_n^{t-1} 个元素。

1.2 ASC-2方法

ASC方法在已知 t 和 n 的情况下，选取 C_n^{t-1} 个元素来构造分配方案。有时为了增强安全性，可以使用 $d \times C_n^{t-1}$ 个元素来构建加密方案。d 为安全系数，$d \in I^+$，d 越大所需元素越多，整体安全性越好。

步骤1 使用ASC方法生成前1、…、C_n^{t-1} 个元素的分配方案。

步骤2 对于第 $C_n^{t-1}+1$、…、$2C_n^{t-1}$ 个元素，可以将这些元素编号。对于编号为 u 的元素 e_u，按照元素 $e_{u \bmod (N+1)}$ 的分配方案进行分配。

步骤3 对于第 $2C_n^{t-1}+1$、…、$3C_n^{t-1}$ 个元素，也采用步骤2操作进行分配，直到所有的元素分配完毕。

例如，$d=3$ 即安全系数为3时，用ASC-2方法构造的(5, 6)门限方案如表3所示，其中“√”表示分配，“×”表示未分配。

表3 $d=3$, (5, 6)门限方案

	p_1	p_2	p_3	p_4	p_5	p_6
e_1	×	×	×	×	√	√
e_2	×	×	×	√	×	√
e_3	×	×	×	√	√	×
e_4	×	×	√	×	×	√
e_5	×	×	√	×	√	×
e_6	×	×	√	√	×	×
e_7	×	√	×	×	×	√
e_8	×	√	×	×	√	×
e_9	×	√	×	√	×	×
e_{10}	×	√	√	×	×	×
e_{11}	√	×	×	×	×	√
e_{12}	√	×	×	×	√	×
e_{13}	√	×	×	√	×	×
e_{14}	√	×	√	×	×	×
e_{15}	√	√	×	×	×	×
e_{16}	×	×	×	×	√	√
e_{17}	×	×	×	√	×	√
e_{18}	×	×	×	√	√	×
e_{19}	×	×	√	×	×	√
e_{20}	×	×	√	×	√	×
e_{21}	×	×	√	√	×	×
e_{22}	×	√	×	×	×	√

续表 3

	p_1	p_2	p_3	p_4	p_5	p_6
e_{23}	×	√	×	×	√	×
e_{24}	×	√	×	√	×	×
e_{25}	×	√	√	×	×	×
e_{26}	√	×	×	×	×	√
e_{27}	√	×	×	×	√	×
e_{28}	√	×	×	√	×	×
e_{29}	√	×	√	×	×	×
e_{30}	√	√	×	×	×	×
e_{31}	×	×	×	×	√	√
e_{32}	×	×	×	√	×	√
e_{33}	×	×	×	√	√	×
e_{34}	×	×	√	×	×	√
e_{35}	×	×	√	×	√	×
e_{36}	×	×	√	√	×	×
e_{37}	×	√	×	×	×	√
e_{38}	×	√	×	×	√	×
e_{39}	×	√	×	√	×	×
e_{40}	×	√	√	×	×	×
e_{41}	√	×	×	×	×	√
e_{42}	√	×	×	×	√	×
e_{43}	√	×	×	√	×	×
e_{44}	√	×	√	×	×	×
e_{45}	√	√	×	×	×	×

ASC-2 的正确性证明如下。

定理 3　ASC-2 方法能成功恢复全集 U。

证明　对于第 1、…、C_n^{t-1} 个元素，采用 ASC 方法进行分配，因此能恢复出包含这些元素的集合 U_1。对于第 $C_n^{t-1}+1$、…、$2C_n^{t-1}$ 个元素，对于编号为 u 的元素 e_u，按照元素 $e_{u \bmod (N+1)}$ 的分配方案进行分配 $C_n^{t-1}+1 \leqslant u \leqslant 2C_n^{t-1}$。因此，这些元素将一一对应到 1、…、$C_n^{t-1}$ 各个元素的分配方法。也相当于采用 *ASC* 方法进行分配，因此能恢复出包含这些元素的集合 U_2。

同理，可以恢复出 U_3、…、U_d，进而可以得到全集 $U=\bigcup_{i=1}^{d} U_i$。

2　STSA 安全门限方案

2.1　STSA 方案

2.1.1　集合划分和计算秘密信息 P_{key}

步骤 1　输入门限值 t，参与者数量 n 和安全系数 d。

步骤 2　选取两两不同的 $N(N=d*C_n^{t-1}, d\in I^+)$个大质数组成全集 $U=\{e_i\}$。通过元素相乘来生成秘密信息，$P_{\text{key}}=hash(\prod\{e|e\in U\})$。$hash()$为哈希函数(如 SHA－256[21]等)。利用秘密信息 P_{key} 对信息

进行加密。

步骤3 按照ASC-2方法生成参与者原始集合$S=\{S_j\}$，$1\leqslant j\leqslant n$。

步骤4 将参与者的原始集合S_j分配给参与者p_j。

2.1.2 参与者合作计算秘密信息的恢复过程RecoverProcess

步骤1 对于参与者p_j，利用自己的参与者原始集合S_j，将元素相乘得到生成信息I_j，$I_j=\prod_i\{s\mid s\in S_j\}$，$1\leqslant j\leqslant n$。

步骤2 在计算秘密信息时，对于愿意参与的参与者，检查是否有参与者公开了生成信息，若没有，则公开自己的生成信息；若已有其他参与者公开了生成信息，则获取该信息。如果获取到多个生成信息，则选择数值最大的一个，记为I'。

步骤3 对自己的参与者原始集合中每个元素做除法，生成信息I'，如果余数不为零，则把这个元素值乘以生成信息，得到新的生成信息。如果余数为零，则忽略重新选取I'。对每一个元素都进行这样的处理，得到处理后的生成信息I''。若$I''\neq I'$则公开I''。

步骤4 所有参与者都经过步骤1到步骤3的处理后，可以得到一个最大的生成信息$I_{\max}$。如果有不少于t个参与者参与了以上过程，则$I_{\max}$将是唯一的，可用于生成秘密信息$P_{\text{key}}=hash(I_{\max})$。

2.2 恢复过程RecoverProcess的正确性证明

定理4 RecoverProcess能正确地计算出P_{key}。

证明 使用ASC-2方法生成各个参与者的原始集合S_j，$1\leqslant j\leqslant n$，当不少于t个参与者将自身S_j组合在一起时，可以得到全集U。

对于有t个参与者的秘密信息恢复，分别记为p_1、p_2、…、p_t，$1\leqslant t\leqslant n$。它们一起进行RecoverProcess操作。

第1个参与者p_1计算出生成信息I_1，并公开给下一个参与者：

$$I_1=\prod\{e\mid e\in S_1\}$$

第2个参与者p_2收到生成信息I_1，计算出生成信息I_2，并公开给下一个参与者：

$$I_2=I_1\times\prod\{e\mid e\in S_2 \text{ and } (I_1 \bmod s\neq 0)\}=$$

$$\prod\{e\mid e\in S_1\}\prod\{e\mid e\in S_2 \text{ and } (I_1 \bmod s\neq 0)\}=$$

$$\prod\{e\mid e\in (S_1\cup S_2)\}$$

第3个参与者p_3收到生成信息I_2，计算I_3，并公开给下一个参与者：

$$I_3=I_2\times\prod\{e\mid e\in S_3 \text{ and } (I_2 \bmod s\neq 0)\}=$$

$$\prod\{e\mid e\in (S_1\cup S_2\cup S_3)\}$$

重复以上过程，最后得到：

$$I_t=I_{t-1}\times\prod\{e\mid e\in S_t \text{ and } (I_{t-1} \bmod s\neq 0)\}=$$

$$\prod\{e\mid e\in (S_1\cup S_2\cup\cdots\cup S_3)\}=\prod\{e\mid e\in U\}$$

可见$P_{\text{key}}=hash(I_t)$。

有多于t个参与者参与秘密信息的恢复时，计算出的秘密信息也将是P_{key}。

因此RecoverProcess能正确地计算出P_{key}。

2.3 STSA的安全性证明

在恢复秘密信息的过程中，参与者会给出一些信息，这些信息如果设计不当，可能会泄露自己的参与

者原始集合。进而导致身份的泄露。此处，作者证明 STSA 的安全性，保证不会导致泄露这些信息。

因为选择的元素个数 $N=d*C_n^{t-1}$，$d\in I^+$。可以保证分配给各个参与者的原始集合，两两之间至少有两个元素是不相同的。

每个参与者公开出去的是元素的乘积信息。如果想要通过该信息恢复参与者的原始集合，则要进行大数分解。因为选择的是大质数，要分解至少两个大质数的乘积，是一个困难问题[22−23]。RSA 算法也是基于分解大素数的困难性进行设计的[24]。在当前计算能力条件下，STSA 具有安全性。所以其他参与者都不能将公开的生成信息还原为参与者的原始集合 S。

最后，得到的秘密信息是一个经过哈希运算后的秘密信息 P_{key}，P_{key} 也不能反向计算出全集 U。

因此，从 STSA 协议公开的生成信息不能恢复参与者原始集合，也不能找出参与者的身份。

3　STSA 安全方案的程序实现与验证

本节通过程序验证第 1 节提出的 ASC 方法和第 2 节提出的 STSA 安全门限方案的正确性和可行性。程序主要包括秘密信息分配、恢复过程以及安全性分析，确保在不少于 t 个参与者的情况下能够正确恢复秘密信息，并在少于 t 个参与者的情况下无法恢复。

3.1　程序实现

本节使用 Python 实现 STSA 安全门限方案，针对这样一个应用场景：在控制系统中存在多个控制器，对于一些关键操作，需要得到不低于门限值个数的控制器允许，才能执行某些关键操作，即使用 STSA 安全(t, n)门限方案，当有不少于 t 个控制器允许执行时，系统才执行关键操作。同时保证控制器隐私安全。不能暴露有哪些控制器参与，也不能暴露参与控制器所持有的分配密钥。

3.1.1　分配秘密信息

设控制器个数为 n，门限值为 t，安全系数为 d。为了安全性可以把质数选得更大。更大的质数需要更多的时间来完成计算。元素分配并计算秘密信息的伪代码如下。

```
step 1:    input(t, n, d)   #输入参数门限值 t，参与者数量 n，安全系数 d
step2:     N=CalCombiNum(n, t−1)   #计算组合数量
step3:     PrimeNumbers=GenPrimeNumbers(d * N)   #选择 d * N 个大质数
step4:     pAssignedElements=ASC-2(PrimeNumbers)   #将 d * N 个大质数按照 ASC-2 方法分配给 n 个参与者
step5:     Pkey=1
step6:     for i in range(0, len(PrimeNumbers), 1):
step7:       Pkey *=PrimeNumbers[i]
step8:     Pkey=hashlib.sha256(str(Pkey).encode('utf8').hexdigest()   #step5-8 计算加密秘密信息，并用于对信息的加密
```

3.1.2　恢复秘密信息

在具有至少 t 个参与者进行秘密信息恢复时，恢复秘密信息的伪代码如下。

```
step1:     defRecoverSI(listParticipants):   #定义恢复秘密信息的函数，listParticipants 变量是参与者的列表
step2:
        I=1
step3:       for i in range(0, len(pAssignedElements[0]), 1):
```

step4: I *=$pAssignedElements[listParticipants[0]][i]$ #计算出第1个参与者的生成信息，并存储到I中

step5: for j in $range(1, len(listParticipants), 1)$: #参与者

step6: for i in $range(0, len(pAssignedElements[0]), 1)$: #参与者原始集合

step7: $tmp=pAssignedElements[listParticipants[j]][i]$

step8: if I % tmp! =0:

step9: I *=tmp

step10: $I=hashlib.sha256(str(I).encode('utf8').herdigest()$ #step5-10其他的参与者根据接收到的生成信息进行操作，最后计算出来总的生成信息，并更新I

step11: output(I) #输出恢复后的秘密信息I

step12: return($I==P_{key}$) #返回恢复是否成功的结果

3.2 正确性验证

正确性验证分为两个部分，第一部分验证任何不少于t个参与者分配密钥均能够恢复出秘密信息，第二部分验证任何少于t个参与者分配密钥均不能恢复出秘密信息。其伪代码如下。

step1: $listPall=list(range(0, n)$ #列出所有的参与者

step2: $listPnotLessThanT=[]$

step3: for j in $range(0, n-t+1, 1)$:

step4: for $listTmp$ in $itertools.combinations(list1, t+j)$:

step5: $listPnotLessThanT.append(list(listTmp))$ #step2-5找出所有不少于t个参与者的组合，保存在$listParticipants$中

step6:

$forlistP$ in $listPnotLessThanT$:

step7: $rst=RecoverSI(listP)$

step8: $assert(rst)$ #step 6-8验证不少于t个参与者时，恢复秘密信息。失败则触断言

step9: $listPLessThanT=[]$

step10: for j in $range(1, t, 1)$:

step11: for $listTmp$ in $itertools.combinations(list1, t-j)$ #step9-12找出所有少于t个参与者的组合，保存在$ElistPLessThanT$中

step12: $listPLessThanT.append(list(listTmp))$

step13: for $listP$ in $listPLessThanT$:

step14: $rst=RecoverSI(listP)$

step15: $assert$(not rst)

程序运行可验证如下结论：

1) 任何不少于t个参与者分配密钥能够恢复出秘密信息；

2) 任何少于t个参与者分配密钥不能恢复出秘密信息。

因此STSA方案能够实现(t, n)门限方案。

完整的代码见https://gitee.com/Tinff/stsa。

3.3 安全性分析

在$t=5$, $n=9$, $d=8$和$random.seed(8)$条件下，所有元素设置为质数，并通过$random.randint(2\times$

10^9，1×10^{12}）随机选取。通过全集计算出的生成信息为：28905983057226665170470287 17…61065926636993556844746678793115449024977244883551138951842817746685488537191377510834212818884119308256496410760911995576833488794631978375692880577019145877383614849344 3(共有 7030 十进制位)，这是一个非常大的整数。SHA256 计算后的加密秘密信息和恢复秘密信息都为：3d974c5fba1201295d1aac36cdf5c938a78a910e4e23a467cb05f2e10736c618。

在安全系数大于 1($d>1$)的条件下，参与者给出的生成信息为多个质数的乘积。质数的大小是可以选择的。如果选择大质数，其他参与者就很难分解出该乘积。因此其他参与者很难恢复出参与者的原始集合。如果要求的安全性更高可以设置更大的安全系数 d 或选择更大的质数来形成全集 U。

可见能够实现(t，n)门限方案的秘密信息计算和恢复。经过程序验证，并能够保证安全性。

3.4　仿真实验

计算机配置为英特尔 Core i7-7700 @ 3.60 GHz 四核、内存 8 GB (DDR42400 MHz)。在 $t=5$，$n=9$ 时，不同质数取值范围和设置不同安全系数($d=2$，4，6，8)的运行性能如图 1 所示。在安全系数 $d=2$，质数取值范围设置为(2×10^8，1×10^{10})时，不同门限值($t=4$，5，6，7，8，9)和参与者数量($n=9$，10，11，12，13，14)下的运行性能如图 2 所示。

图 1 展示了 STSA 方案在 $t=5$，$n=9$ 参数配置下的性能特征，重点分析安全系数 $d=2$、4、6、8 与质数取值范围对运算时间的双重影响。实验结果表明，当质数取值范围为(2×10^{10}，1×10^{12})，安全系数 d 从 2 递增至 8 时，运算时间呈现近似指数增长的趋势，这是因为本方法中元素数量随 d 值增大而增加，导致集合划分、秘密计算及恢复等环节的计算复杂度显著提升。同时，质数范围的扩展也会加剧运算耗时，因为大质数在模乘、取余等运算中需要更多计算资源。实验结论表明：提升安全系数 d 或采用更大的质数范围虽然可以增强安全性，但是会导致运算效率的下降，实际部署需根据应用场景在安全性与性能之间寻求平衡。

图 2 呈现了 STSA 方案在安全系数 $d=2$、质数范围(2×10^8，1×10^{10})的条件下，运算时间随门限值 t 和参与方数量 n 的动态关系。实验表明：运算时长与参与方数量 n 呈显著正相关，当 n 由 9 增至 14 时，ASC-2 方法的运算时间增长较快，同时秘密恢复阶段需处理更多参与方的信息交互，导致计算量急剧升高。相较而言，门限值 t 从 4 提升至 9 时运算时长增幅较缓，主要因为 t 值增加仅线性扩展了恢复阶段的验证步骤。该结论揭示了在此参数框架下，系统效率对参与方规模更为敏感。实际部署于多控制器系统时，须通过 n 与 t 的协同配置实现安全门限与运算效能的动态平衡。

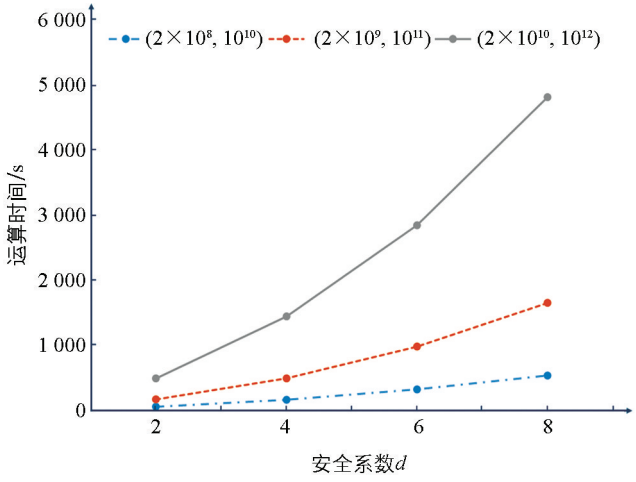

图 1　STSA 性能分析

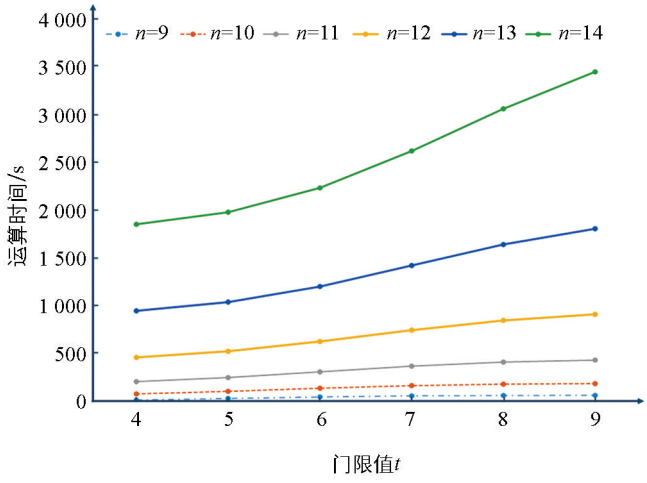

图 2　STSA 性能分析

4 结论

针对当前一些(t, n)门限方案的设计和实现复杂性的问题，本文提出一种简单、易理解、易实现的门限方案。首先使用ASC集合分配方法分配$N=C_n^{t-1}$个元素给n个参与者，进而设计ASC-2方法将安全系数为d的全集元素分配给各个参与者，在数学上证明了ASC和ASC-2方法的正确性。在ASC-2基础上提出了STSA安全(t, n)门限方案，也在数学上证明了STSA方案的正确性和安全性。相较于传统门限方案，STSA避免了高次插值和矩阵运算等复杂操作，降低了计算资源消耗，具有较强的理论意义。此外，STSA方案还为物联网终端、边缘设备等资源受限的场景提供了新的思路和实践路径，拓展了门限方案在隐私保护和多方计算中的应用潜力。

针对多控制器执行关键操作的应用场景，使用Python实现了STSA(t, n)门限方案。实现的STSA安全(t, n)门限方案，可以实现只有不少于t个控制器允许执行时，系统才执行关键操作；少于t个时则不能执行关键操作。同时该方案可保证控制器隐私安全，未暴露参与控制器及所持有的分配密钥。通过程序验证的方式，确认了STSA方案的正确性和安全性，同时也实现了其他学者提出的一些门限方案。

开展了多个STSA方案的性能实验并进行了分析。STSA运算时间随安全系数d，质数选取范围的增大而增加；随参与者数量n增大而急速增加；随门限值t的增大而缓慢增加。本文主要提出并验证了STSA的可行性。接下来的工作包括：增强在抗量子攻击方面的能力；对算法进行进一步优化，提升时间效率；在分配密钥复用方面探讨STSA方案与其他密码学协议的结合，特别是零知识证明和同态加密等技术，以增强其在更广泛场景下的应用能力。

参考文献：

[1] SHAMIR A. How to Share a Secret [J]. Communications of the ACM, 1979, 22(11): 612-613.

[2] BLAKLEY G R. Safeguarding Cryptographic Keys [C] //1979 International Workshop on Managing Requirements Knowledge (MARK). New York: IEEE, 2019: 313-318.

[3] 涂彬彬，陈宇. 门限密码系统综述 [J]. 密码学报，2020，7(1)：1-14.

[4] 赵宗渠，郭小杰，殷明辉，等. 物联网环境下基于身份匿签密的认证方法研究 [J]. 重庆邮电大学学报(自然科学版)，2023，35(2)：343-351.

[5] 赵武超，许文超. 插值多项式的存在性和唯一性 [J]. 洛阳师范学院学报，2008，27(2)：17-18.

[6] BRICKELL E F. Some Ideal Secret Sharing Schemes [C] //Advances in Cryptology-EUROCRYPT '89. Berlin, Heidelberg: Springer, 1990: 468-475.

[7] ERSOY O, PEDERSEN T B, ANARIM E. Homomorphic Extensions of CRT-Based Secret Sharing [J]. Discrete Applied Mathematics, 2020, 285: 317-329.

[8] KARNIN E, GREENE J, HELLMAN M. On Secret Sharing Systems [J]. IEEE Transactions on Information Theory, 1983, 29(1): 35-41.

[9] BRICKELL E F, DAVENPORT D M. On the Classification of Ideal Secret Sharing Schemes [J]. Journal of Cryptology, 1991, 4(2): 123-134.

[10] CHIEN H Y, JAN J K, TSENG Y M. A Practical (t, n) Multi-Secret Sharing Scheme [J]. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, 2000, 83(12): 2762 - 2765.

[11] MARTÍN DEL REY A, MATEUS J P, SÁNCHEZ G R. A Secret Sharing Scheme Based on Cellular Automata [J]. Applied Mathematics and Computation, 2005, 170(2): 1356-1364.

[12] 蔡兆政，瞿云云，包小敏. 一个可公开验证的多重秘密共享门限方案 [J]. 西南大学学报(自然科学版)，2021，43(7)：105-110.

[13] 张剑，林昌露，黄可可，等. 基于多项式插值的多等级秘密共享方案 [J]. 密码学报，2022，9(4)：743-754.

[14] HSU C F，CHENG Q，TANG X M，et al. An Ideal Multi-Secret Sharing Scheme Based on MSP [J]. Information Sciences，2011，181(7)：1403-1409.

[15] CAPUTO S，KORCHMÁROS G，SONNINO A. Multilevel Secret Sharing Schemes Arising from the Normal Rational Curve [J]. Discrete Applied Mathematics，2020，284：158-165.

[16] BENDLIN R，DAMGÅRD I. Threshold Decryption and Zero-Knowledge Proofs for Lattice-Based Cryptosystems [C] //Theory of Cryptography. Berlin，Heidelberg：Springer，2010：201-218.

[17] YU K P，TAN L，YANG C X，et al. A Blockchain-Based Shamir's Threshold Cryptography Scheme for Data Protection in Industrial Internet of Things Settings [J]. IEEE Internet of Things Journal，2022，9(11)：8154-8167.

[18] TESSARO S，ZHU C Z. Thresholdand Multi-Signature Schemes from Linear Hash Functions [M] //Advances in Cryptology-EUROCRYPT 2023. Cham：Springer Nature，2023：628-658.

[19] BRAUN L，CASTAGNOS G，DAMGÅRD I，et al. An Improved Threshold Homomorphic Cryptosystem Based on Class Groups [M] //Security and Cryptography for Networks. Cham：Springer Nature，2024：24-46.

[20] LI F L，LIU Z，LIU L，et al. A Rational Hierarchical (T，n)-Threshold Quantum Secret Sharing Scheme [J]. Quantum Information Processing，2024，23(2)：60.

[21] 邓鸿，林金朝，庞宇，等. 基于 FPGA 的 BAN 认证算法硬件实现 [J]. 重庆邮电大学学报(自然科学版)，2019，31(6)：799-804.

[22] LI X Y，MOU H J，LU D J. An Improved Ciphertext Retrieval Scheme Based on Fully Homomorphic Encryption [J]. Wuhan University Journal of Natural Sciences，2019，24(3)：218-222.

[23] 刘莺迎. 大整数分解算法的设计与实现 [J]. 科学技术创新，2020(36)：109-110.

[24] 杜俊. RSA 算法中大素数快速生成和运算方法实现 [J]. 中阿科技论坛(中英文)，2024(9)：108-112.

责任编辑　张枸