Aplicações de Secret Sharing para Computação Distribuída

Palavras-chave: Computação distribuída, Criptografia, Códigos polinomiais.

Estudante: Igor Luis Aureliano, IMECC - UNICAMP
Orientador: Marcelo Firer, IMECC - UNICAMP
Coorientador: Rafael G. L. D'Oliveira, SMSS - Clemson University

1 Introdução

Suponha que um indivíduo deseje manipular uma informação privada, conhecida por "Segredo", com a ajuda de um grupo de participantes. Tais participantes são confiáveis, todavia curiosos, o que implica que qualquer grupo específico deles podem tentar deduzir alguma informação sobre o segredo. O usuário deseja realizar tal processo de modo que a partir de qualquer quantidade específica de participantes o segredo seja acessível, mas com uma quantidade menor desse número específico, o segredo seja completamente indeterminado.

Introduzido em 1979, Secret Sharing[1, 2] é um algoritmo criptográfico que permite a divisão de um segredo em compartilhamentos para serem distribuídos de maneira segura entre um conjunto de participantes. A distribuição é feita de uma maneira que somente com um número mínimo de compartilhamentos o segredo pode ser determinado, porém qualquer quantidade menor desse mínimo deixa o segredo completamente indeterminado.

A ideia principal por trás do Secret Sharing é que podemos utilizar polinômios sobre corpos finitos, uma vez que os polinômios proporcionam uma forma simples, eficiente e segura de gerar e combinar os compartilhamentos através da álgebra linear. Além disso, os campos finitos fornecem propriedades algébricas e estatísticas que tornam os esquemas de Secret Sharing perfeitamente seguros, isto é, o esquema não fornece nenhuma informação sobre o segredo mesmo na situação contra adversários com poder computacional e tempo ilimitado.

Devido a tais propriedades, o Secret Sharing possui uma ampla gama de aplicações em problemas da criptografia, como as estudadas nessa iniciação científica, sendo elas o armazenamento seguro e distribuído de dados, recuperação privada de dados e multiplicação segura e distribuída de matrizes. Em inglês, os protocolos de recuperação privada de dados e multiplicação segura e distribuída de matrizes são chamados de, respectivamente, Private Information Retrieval (PIR) e Secure Distributed Matrix Multiplication (SDMM).

2 Medotologia

As metodologias usadas para o desenvolvimento da iniciação científica as descritas abaixo:

- **Revisão bibliográfica:** A realização da revisão bibliográfica abrangente para identificar e compreender os principais conceitos relacionados aos objetivos do estudo. A revisão bibliográfica inclui artigos científicos, livros e dissertações relevantes.

- **Análise teórica:** A realização das análises teóricas das técnicas para avaliar a segurança, eficiência e escalabilidade dos esquemas propostos. A análise inclui o conhecimento de álgebra linear sobre corpos finitos e teoria da informação com foco no conceito de entropia para demonstrar as propriedades dos esquemas e análise da eficiência computacional em relação à otimização dos custos de comunicação.
- **Implementação de algoritmos:** Foram implementados algoritmos para aplicar as técnicas de Secret Sharing nos problemas de armazenamento seguro e distribuído de dados, recuperação privada de dados distribuídos e multiplicação segura e distribuída de matrizes. As implementações foram realizadas em Python e colocadas em três Jupyter Lab para que a aplicação seja apresentada enquanto o código vai sendo criado.
- **Experimentos numéricos:** A realização de experimentos numéricos para, principalmente, evidenciar as propriedades de cada esquema e avaliar o desempenho dos esquemas implementados na literatura de modo a compará-los.
- **Discussão dos resultados:** Foram discutidos os resultados obtidos com as implementações, análises teóricas e experimentos numéricos com o intuito de apontar os pontos fortes e fracos de cada esquema proposto, e discutir os possíveis cenários de aplicação desses esquemas.

3 Resultados

Para realizar as implementações das três aplicações em Python, decidimos criar três Jupyter Lab. De maneira simples, o Jupyter Lab é um caderno de apresentação no qual podemos colocar textos e códigos. A ideia principal é criar uma apresentação para pessoas que não estão familiarizadas com o tema e mostrar os códigos sendo criados à medida que avançamos na explicação. Assim, o leitor pode baixar o Jupyter Lab para executar os códigos na sua máquina. Todos os cadernos foram baseados nas anotações desenvolvidas durante a pesquisa, onde foi registrada toda teoria por trás das três aplicações que, por sua vez, foram baseadas em todos os artigos e livros considerados na revisão bibliográfica. Os cadernos podem ser encontrados aqui: https://github.com/IgorAureliano.

A seguir, introduziremos as principais ideias e relações por trás de cada aplicação, assim como os principais resultados que cada esquema possui.

3.1 Armazenamento Seguro e Distribuído de Dados

Como apresentado na introdução, o Secret Sharing é um algoritmo que permite a partilha de um segredo entre um grupo de participantes de tal forma que apenas subconjuntos com um número específico de participantes podem reconstruir o segredo, enquanto outros menores não podem. Consideremos um exemplo de fixação.

Suponha que uma empresa pretende guardar uma senha de acesso a dados financeiros entre quatro funcionários. A senha só deve ser acessível na presença de pelo menos três funcionários. Assim, cada funcionário recebe uma partilha tal que qualquer grupo de três é capaz de reconstruir o segredo, mas na presença de dois ou menos o segredo é completamente indeterminado. Este problema pode ser resolvido utilizando o esquema conhecido por Threshold Secret Sharing. Assim, seja $\mathbb{F}_q$ um corpo finito de q elementos. O esquema é criado da seguinte maneira:

- **Passo 1:** Seja $S \in \mathbb{F}_q$ o segredo e escolha $R_1, R_2 \in \mathbb{F}_q$ uniformes ao acaso.
- **Passo 2:** Defina o polinômio $f(x) = S + R_1 x + R_2 x^2$ sobre o corpo finito $\mathbb{F}_q$.
- **Passo 3:** Envie para cada funcionário i a avaliação $f(i) = S + R_1 i + R_2 i^2$, $i \in \{1, 2, 3, 4\}$.

O esquema é perfeitamente seguro devido ao fato de que $f(x)$ é uma parábola determinada por três pontos, logo, com até duas avaliações, não existe uma única parábola que passa por tais pontos. Deste modo, como as chaves R_1, R_2 são uniformes ao acaso, temos que S pode ser qualquer valor em $\mathbb{F}_q$ com mesma probabilidade. Ademais, S pode ser acessada pelo fato de que todas as avaliações são linearmente independentes e três ou mais avaliações determinam completamente um polinômio de grau dois.

Outro esquema interessante é conhecido por Ramp Secret Sharing. A ideia por trás dele é que podemos inserir mais segredos para obter as mesmas propriedades de segurança. No exemplo anterior, observe que podemos fazer a definição $f(x) = S_1 + S_2x + R_1x^2 + R_2x^3$. Assim, a diferença é que os segredos são acessíveis somente na presença dos quatro funcionários, uma vez que um polinômio de grau três é completamente determinado por quatro avaliações.

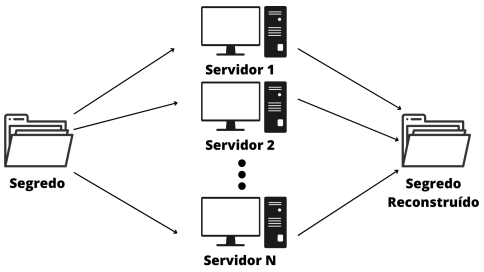

Figura 1: Esquema para armazenamento.

Se considerarmos N e T, respectivamente, como os números de participante e a quantidade máxima deles que se comunicam entre si, as generalizações são análogas para o Secret Sharing definindo um polinômio de grau T. E, para o Ramp Secret Sharing, um polinômio de grau T a N – 1.

3.2 Recuperação Privada de Dados

Uma das aplicações do Secret Sharing é a criação de esquemas de Private Information Retrieval (PIR)[5]. O objetivo do PIR é permitir que um usuário acesse uma mensagem em um banco de dados de M mensagens replicado em N servidores, onde até T deles se comunicam.

A ideia é que podemos interpretar o banco de dados como um vetor e usar os dois esquemas anteriores em sua forma vetorial para obter esquemas assintoticamente ótimos, como o esquema de Ramp Secret Sharing atinge.

O processo se resume ao usuário criar uma avaliação vetorial Q_j do polinômio de tamanho igual ao banco de dados para ser enviada a cada servidor. Assim, cada servidor responde computando o produto interno da avaliação com o banco de dados A_j e envia de volta para o usuário. Baixando todas as respostas dos servidores, o usuário deve ser capaz de determinar a mensagem desejada sem revelar aos servidores qual das mensagens ele tem interesse. A métrica que quantifica a eficiência dos esquemas é a taxa de download, definida pelo número de símbolos recuperados pelo total de símbolos baixados nos processos anteriores.

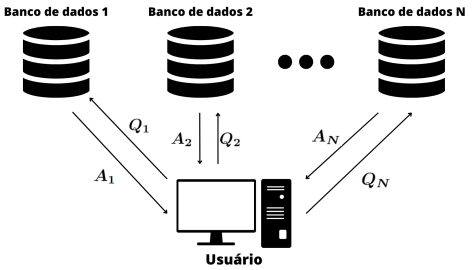

Figura 2: Esquema para recuperação.

Os principais resultados giram em torno das técnicas de Refine e Lift [7, 8, 9] para criar esquemas de PIR que atingem a taxa de download máxima para parâmetros finitos M, N, T, conhecida por Capacidade e determinada em [6]. As técnicas transformam qualquer esquema de Secret Sharing para parâmetros N, T em um esquema ótimo para duas mensagens (M = 2) que por sua vez se transforma em um esquema ótimo para uma quantidade arbitrária M através de um processo combinatório.

3.3 Multiplicação Segura e Distribuída de Matrizes

Outra aplicação do Secret Sharing é para a criação de esquemas de Secure Distributed Matrix Multiplication (SDMM)[10, 11, 12]. O objetivo do SDMM é garantir que o usuário seja capaz de computar

a multiplicação de duas matrizes A e B usando N servidores no qual quaisquer T deles podem se comunicar. O processo deve ser feito de tal forma que não seja revelado nenhum tipo de informação a respeito das entradas de A e B.

Novamente, a ideia gira em torno de usar polinômios sobre corpos finitos para resolver o problema. Neste caso, definimos dois polinômios $f(x)$ e $g(x)$ na forma matricial que codificam, respectivamente, A e B. O usuário envia avaliações $f(a_j)$ e $g(a_j)$ distintas para cada servidor. Ao receber as avaliações, os servidores computam o produto $h(a_j) = f(a_j)g(a_j)$ e enviam de volta para o usuário. O usuário deve ser capaz de determinar AB com todas N avaliações de $h(x)$. Novamente, a taxa de download é a métrica utilizada para avaliar os esquemas e devemos otimizar-lá.

O primeiro resultado que vemos nesse problema é que devemos tomar cuidado ao usar diretamente um esquema de Secret Sharing com A e B sendo os segredos em, respectivamente, $f(x)$ e $g(x)$. Neste caso, as chaves uniformes ao acaso são de mesmo tamanho que A e B, logo, se A e B tiverem dimensões altas, o processo pode ser tão custoso computacionalmente. Assim, uma maneira de resolver o problema anterior é dividir as matrizes A e B em, respectivamente, K e L blocos idênticos de tal modo que o produto pode ser computado como abaixo:

$$\mathrm{A} = \begin{bmatrix} \mathrm{A}_1 \\ \mathrm{A}_2 \\ \vdots \\ \mathrm{A}_\mathrm{K} \end{bmatrix}, \quad \mathrm{B} = \begin{bmatrix} \mathrm{B}_1 & \mathrm{B}_2 & \dots & \mathrm{B}_\mathrm{L} \end{bmatrix} \Longrightarrow \mathrm{AB} = \begin{bmatrix} \mathrm{A}_1\mathrm{B}_1 & \dots & \mathrm{A}_1\mathrm{B}_\mathrm{L} \\ \vdots & \ddots & \vdots \\ \mathrm{A}_\mathrm{K}\mathrm{B}_1 & \dots & \mathrm{A}_\mathrm{K}\mathrm{B}_\mathrm{L} \end{bmatrix}$$

A ideia é definir $f(x)$ e $g(x)$ de modo que todas as multiplicações do tipo $\mathrm{A}_k\mathrm{B}_l$ apareçam como coeficientes de $h(x)$. Todavia, essa divisão apresenta um problema imediato em relação aos graus dos polinômios, pois dependendo de sua definição podemos criar uma combinação de produtos do tipo $\mathrm{A}_k\mathrm{B}_l$ nos coeficientes, logo, não seria possível determinar todos os produtos $\mathrm{A}_k\mathrm{B}_l$ para computar AB. Devido a isso, surgiram vários esquemas de SDMM, como os apresentados em [10, 11] e a família de códigos conhecida por GASP[12, 13] que estudamos com um maior foco durante o projeto.

Os códigos GASP possuem um desempenho superior aos apresentados em [10, 11] e são flexíveis em relação à sua definição para as relações dos paramêtros $\mathrm{K}, \mathrm{L}, \mathrm{T}, \mathrm{N}$. Em conjunto a esses códigos, é apresentada a Tabela de Graus que simplifica o problema de escolha dos graus dos polinômios $f(x)$ e $g(x)$ em um problema combinatório de tabela de soma. O maior resultado é que podemos resolver o problema apenas estudando a tabela de Graus se certas propriedades forem verificadas, pois, por ser uma tabela de soma, podemos ver o que teremos em $h(x)$.

4 Discussão

Na primeira aplicação em armazenamento distribuído de dados, o problema é resolvido fazendo uma aplicação direta do Secret Sharing. Por esse motivo, foi de grande importância começarmos a pesquisa por essa parte enquanto abordávamos os tópicos de Teoria da Probabilidade, Teoria da Informação e Álgebra Linear sobre corpos finitos. Além disso, foi importante para observarmos a semelhança entre o Secret Sharing e os códigos corretores de erros Reed-Solomon[3], como discutido em [4].

Na segunda aplicação, é evidenciada a necessidade das técnicas de Refine e Lift pelo fato de que os esquemas de Secret Sharing e Ramp Secret Sharing na forma vetorial não atingem os valores ótimos para parâmetros finitos, como determinado em [6].

Finalmente, na terceira aplicação, vemos que podemos definir os polinômios $f(x)$ e $g(x)$ através da Tabela de Graus usando as relações de ordem que os parâmetros $\mathrm{K}, \mathrm{L}, \mathrm{T}, \mathrm{N}$ possuem. Todavia, também vemos qual a melhor forma de definir as divisões das matrizes A e B, isto é, os parâmetros K e L. Assim, podemos definir K e L de modo a usar um esquema de [12, 13] que otimiza a taxa de download.

5 Conclusão

Este estudo explorou aplicações de Secret Sharing em computação distribuída, destacando sua versatilidade e importância na segurança da informação. Iniciamos com o armazenamento seguro e distribuído de dados, onde a aplicação direta do esquema Threshold Secret Sharing demonstrou ser uma solução robusta e eficiente. Avançamos para a recuperação privada de dados (PIR), onde as técnicas de Refine e Lift foram importantes para obter esquemas de PIR ótimos para parâmetros finitos. Por fim, na multiplicação segura e distribuída de matrizes (SDMM), os códigos GASP em conjunto com a Tabela de Graus proporcionaram um aumento significativo na eficiência dos esquemas em [10, 11]. Para estudos futuros, sugerimos o problema em aberto de determinar a capacidade dos esquemas de Coded PIR[9] e o problema de multiplicação segura e distribuída de matrizes com pré-computação[14].

Referências

[1] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.

[2] G. R. Blakley, "Safeguarding Cryptographic Keys", Proceedings of the National Computer Conference, 48, 313-317, 1979.

[3] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” Journal of the society for industrial and applied mathematics, vol. 8, no. 2, pp. 300–304, 1960.

[4] R. J. McEliece and D. V. Sarwate, “On sharing secrets and reed-solomon codes,” Communications of the ACM, vol. 24, no. 9, pp. 583–584, 1981.

[5] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, “Private information retrieval,” in IEEE Symposium on Foundations of Computer Science, pp. 41–50, 1995.

[6] H. Sun and S. A. Jafar. "The capacity of private information retrieval."IEEE Transactions on Information Theory 63.7 (2016): 4075-4088.

[7] R. G. L. D’Oliveira and S. El Rouayheb, “One-shot pir: Refinement and lifting,” IEEE Transactions on Information Theory, vol. 66, no. 4, pp. 2443–2455, 2019.

[8] R. G. L. D’Oliveira and S. El Rouayheb, “Lifting Private Information Retrieval from Two to any Number of Messages,” IEEE International Symposium on Information Theory (ISIT), 2018.

[9] R. G. L. D’Oliveira and S. El Rouayheb, "A Guided Walk Through Coded Private Information Retrieval", in IEEE BITS the Information Theory Magazine, 2024.

[10] W.-T. Chang, R. Tandon, “On the Capacity of Secure Distributed Matrix Multiplication”, 2018.

[11] J. Kakar, S. Ebadifar, and A. Sezgin, “Rate-Efficiency and Straggler-Robustness through Partition in Distributed Two-Sided Secure Matrix Computation”, 2018.

[12] R. G. L. D’Oliveira, S. El Rouayheb and D. Karpuk, "GASP Codes for Secure Distributed Matrix Multiplication", in IEEE Transactions on Information Theory, 2020.

[13] R. G. L. D’Oliveira, S. E. Rouayheb, D. Heinlein and D. Karpuk, "Degree Tables for Secure Distributed Matrix Multiplication", in IEEE Journal on Selected Areas in Information Theory, 2021.

[14] R. Cartor, R. G. L. D’Oliveira, S. El Rouayheb, D. Heinlein, D. Karpuk and A. Sprintson, "Secure distributed matrix multiplication with precomputation", 2024.