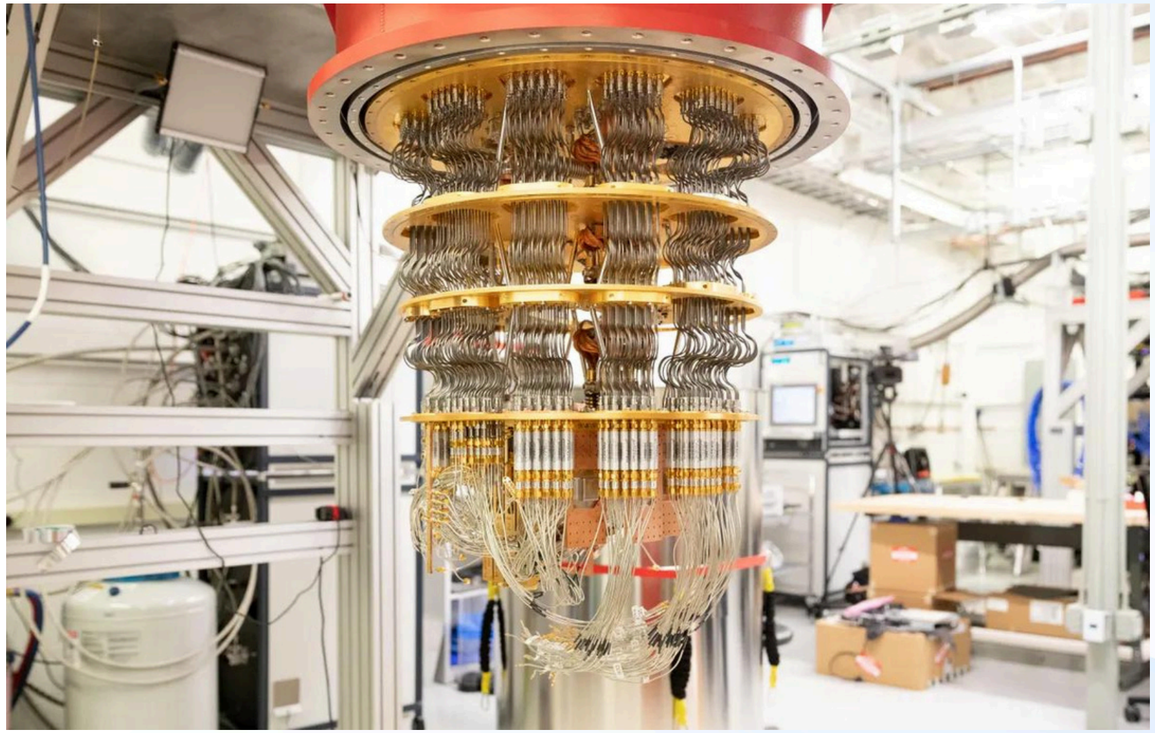
Compact Memory Implementations of the ML-DSA Post-Quantum Digital Signature Algorithm

R. D. Meneses, C. Teixeira, M. A. A. Henriques

16 de Setembro, 2024

Sumário

Criptografia pós-quântica



Computador quântico de 54-qubits da Google

O que é criptografia pós-quântica?

Algoritmos critpográficos resistentes aos ataques de um computador quântico;

NIST 2016 - 2024

Processo de padronização para algoritmos criptográficos pós-quânticos.

- ML-KEM (Kyber)
- **ML-DSA (Dilithium)**
- FN-DSA (Falcon)
- SLH-DSA (SPHINCS+)

Assinaturas digitais

- Buscam garantir a autenticidade e integridade de mensagens e documentos digitais.

Assinaturas digitais

- Buscam garantir a autenticidade e integridade de mensagens e documentos digitais.

1. Geração de chaves (KeyGen);
2. Assinatura (Sign);

Assinaturas digitais

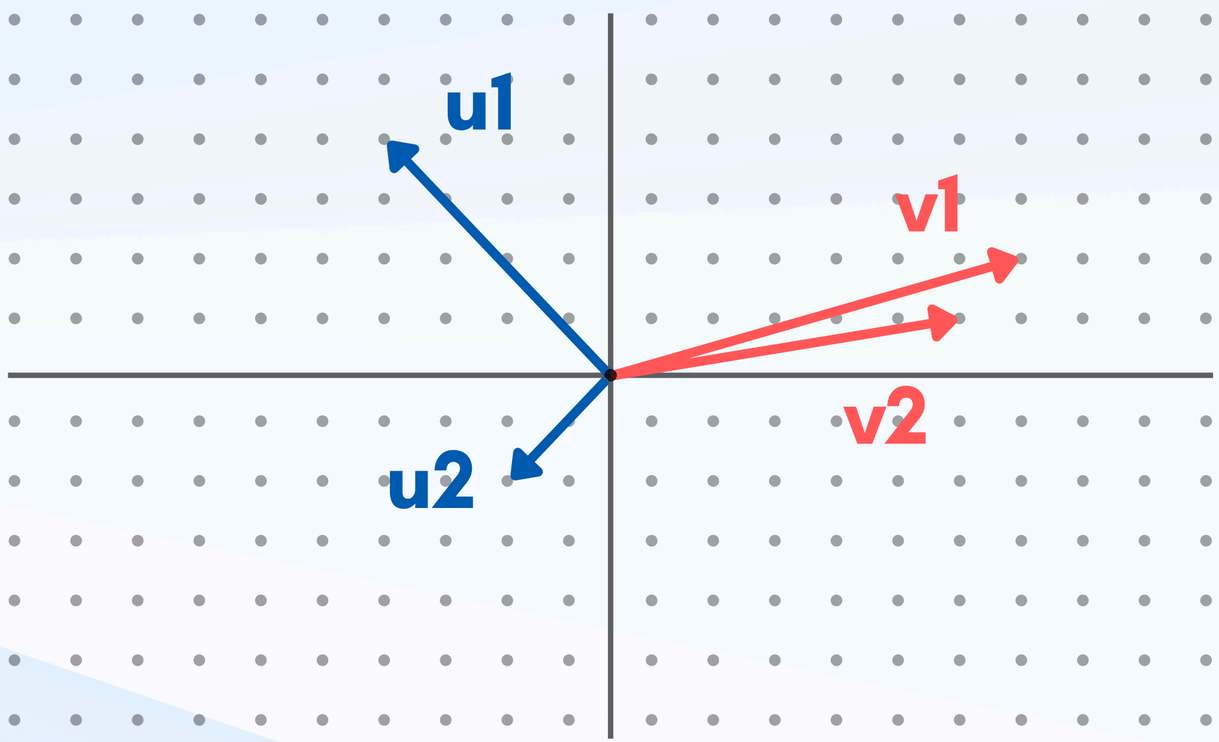
- Buscam garantir a autenticidade e integridade de mensagens e documentos digitais.

1. Geração de chaves (KeyGen);
2. Assinatura (Sign);
3. Verificação (Verify).

Problemas sobre reticulados

O que são reticulados?

- Estruturas algébricas que consistem em um conjunto *S* de pontos em um espaço *n*-dimensional.
- Os pontos são gerados a partir de uma base vetores contida em *S*.



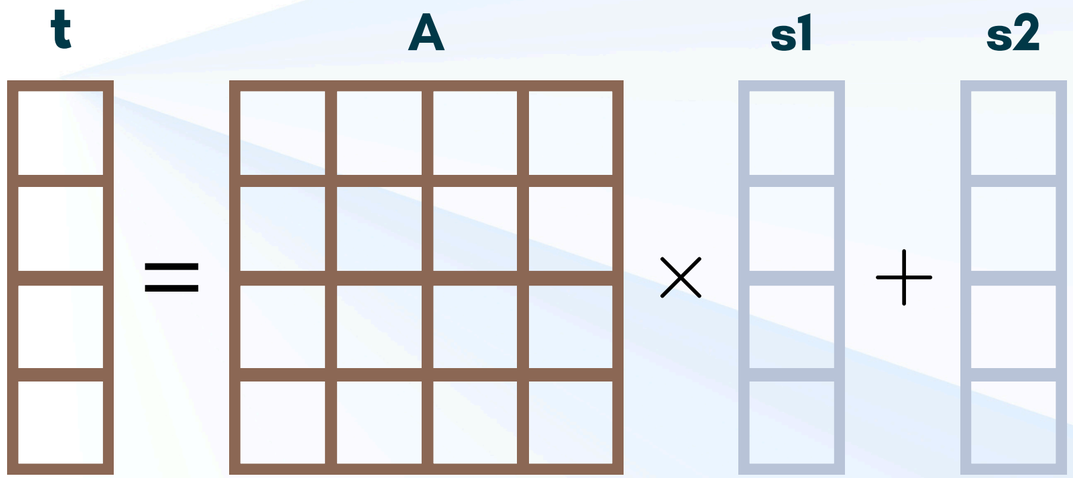
Problemas relevantes:

SVP, CVP, GapCVP, MSIS, **MLWE**...

Funcionamento do ML-DSA

Chave pública = (**A, t**)

Chave privada = (**s1, s2**)

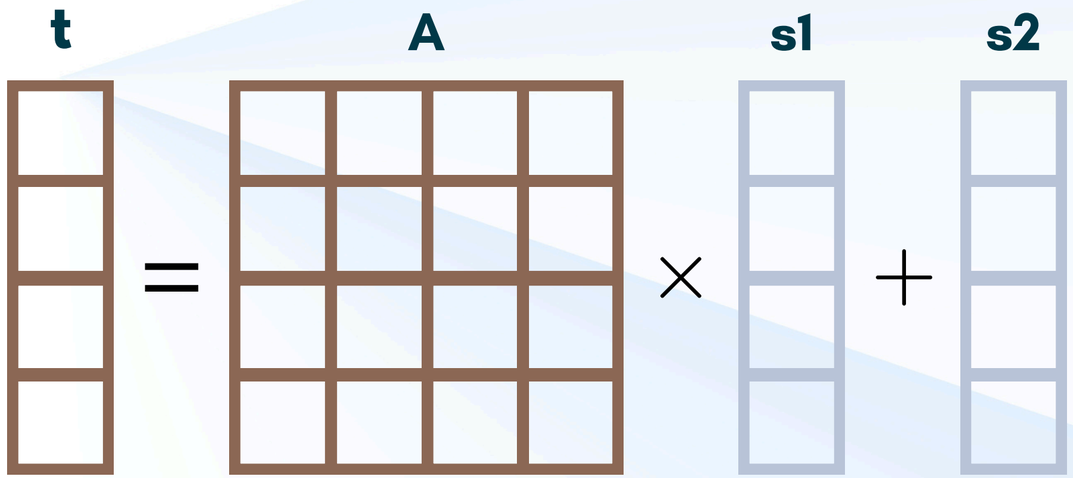


- Matriz de polinômios **A**;
- Vetor de polinômios **t**;
- Vetores secretos de polinômios **s1** e **s2**.

Funcionamento do ML-DSA

Chave pública = (**A, t**)

Chave privada = (**s1, s2**)



- Matriz de polinômios **A**;
- Vetor de polinômios **t**;
- Vetores secretos de polinômios **s1** e **s2**.

Criptografia baseada em reticulados

A dificuldade consiste em, dados **A** e **t**, determinar os vetores **s1** e **s2**.

Cada entrada nos vetores ou matriz é um elemento no anel polinomial $Z_q[X]/(X^{256} - 1)$.

Este problema é chamado **MLWE (Module Learning with Errors)**.

Otimizações de memória no ML-DSA

Problema:

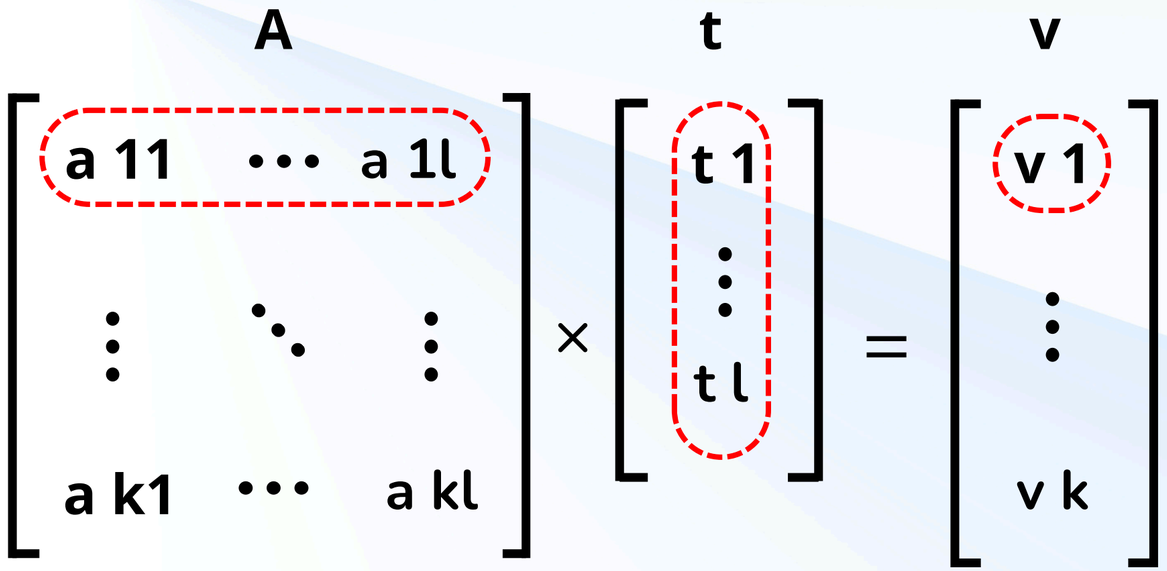
- Armazenamento de matrizes e vetores de polinômios exige muita memória.

Otimizações de memória no ML-DSA

Problema:

- Armazenamento de matrizes e vetores de polinômios exige muita memória.

Método tradicional:

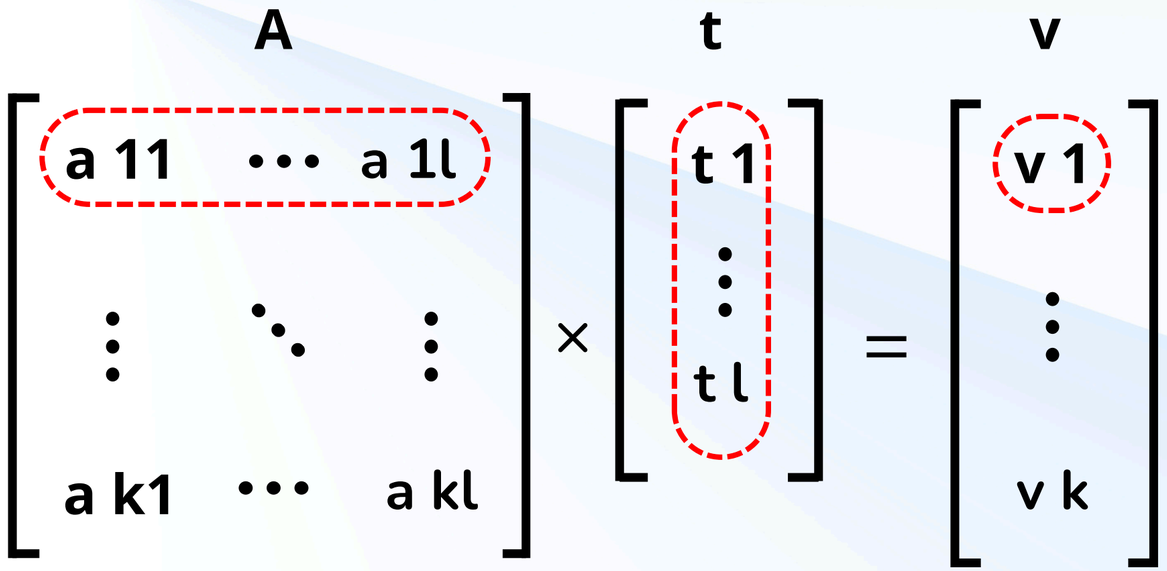


Otimizações de memória no ML-DSA

Problema:

- Armazenamento de matrizes e vetores de polinômios exige muita memória.

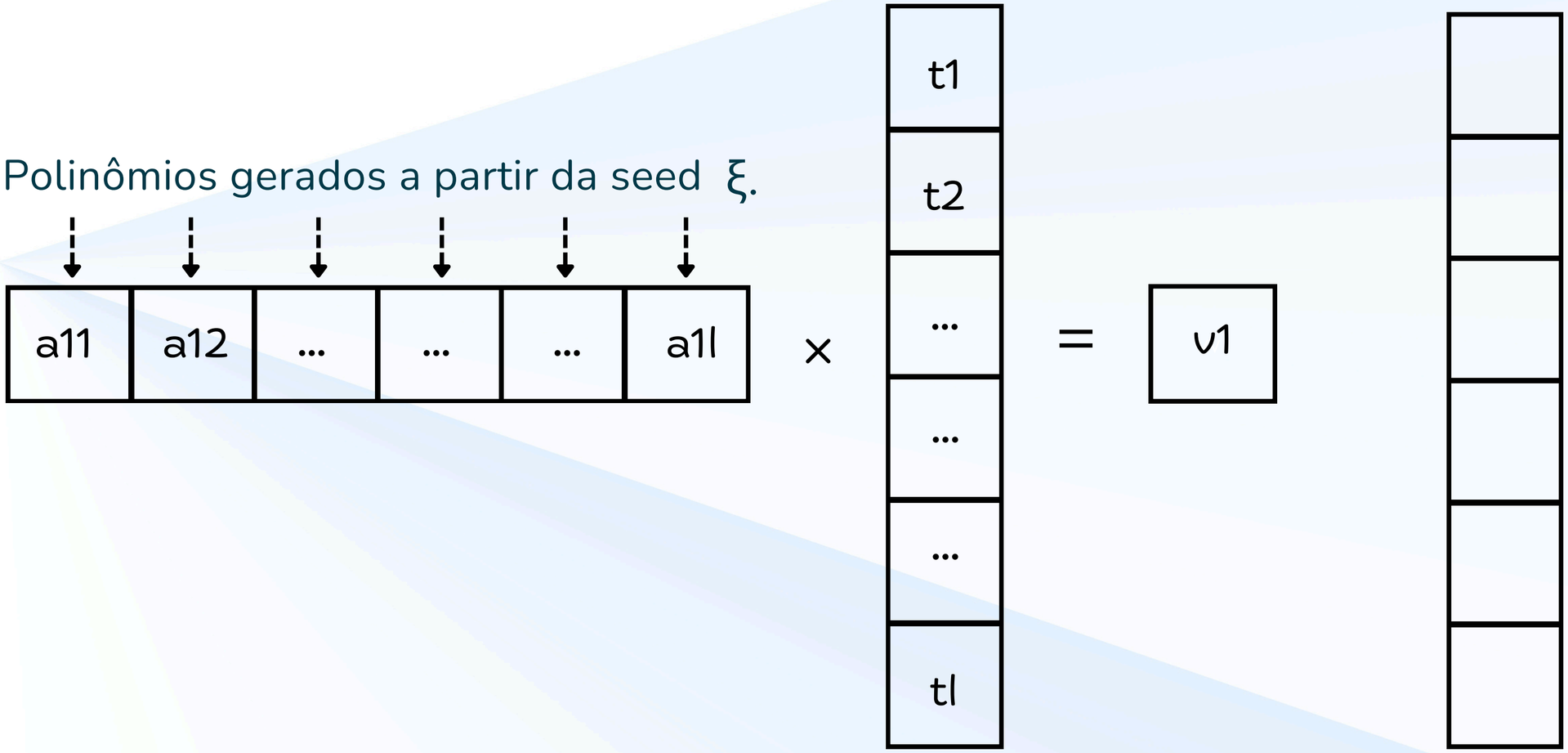
Método tradicional:



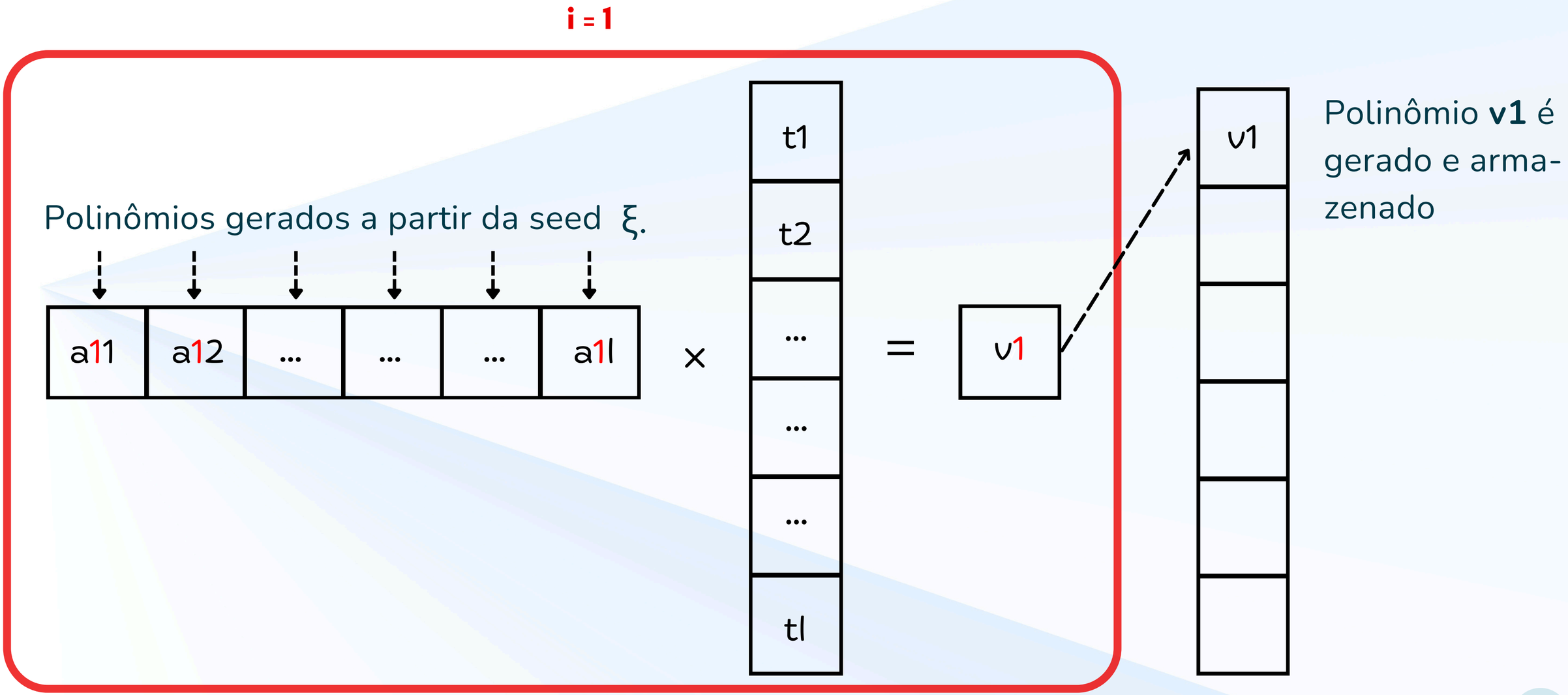
- A matriz pública **A** ocupa {16, 30, 56}-KiB de RAM, tornando inviável a implementação em dispositivos restritos.
- Parâmetros expandidos a partir de uma seed pseudoaleatória ξ.
- **Realizamos a geração dos parâmetros reaproveitando uma mesma região de memória.**

Geração via vetor de polinômios (ML-DSA-v)

Polinômios gerados a partir da seed ξ.



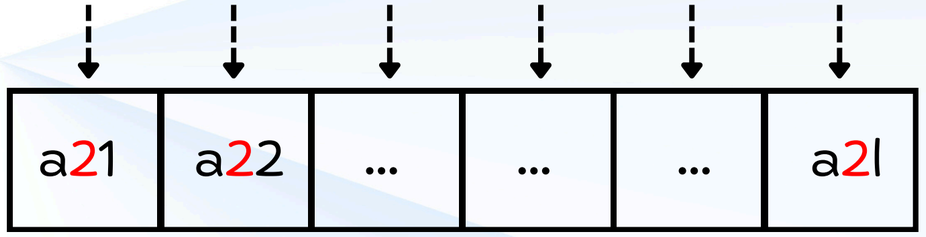
Geração via vetor de polinômios (ML-DSA-v)



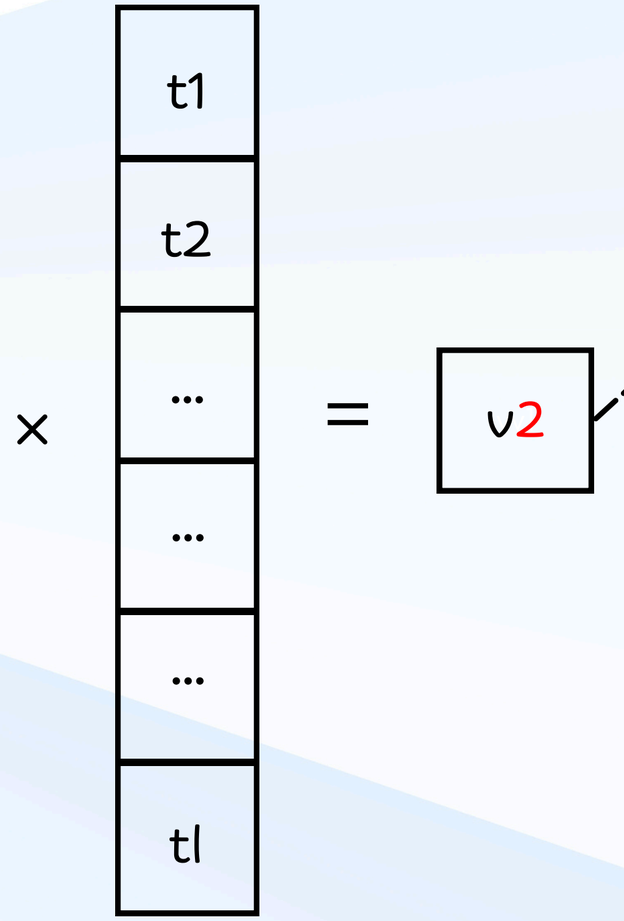
Geração via vetor de polinômios (ML-DSA-v)

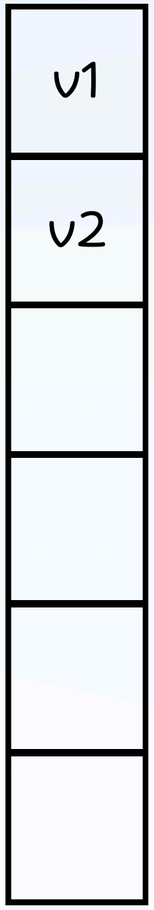
i = 2

Polinômios gerados a partir da seed ξ.



Sobrescrevendo o vetor anterior.



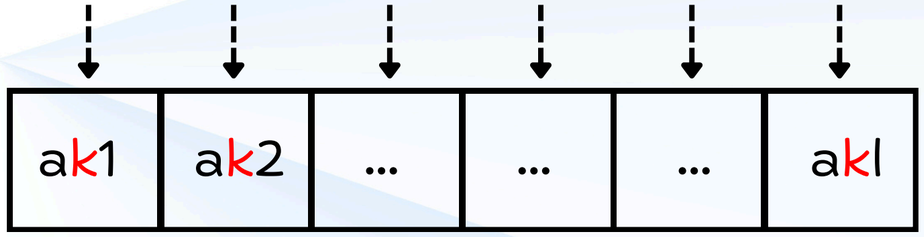


Polinômio **v2** é gerado e armazenado

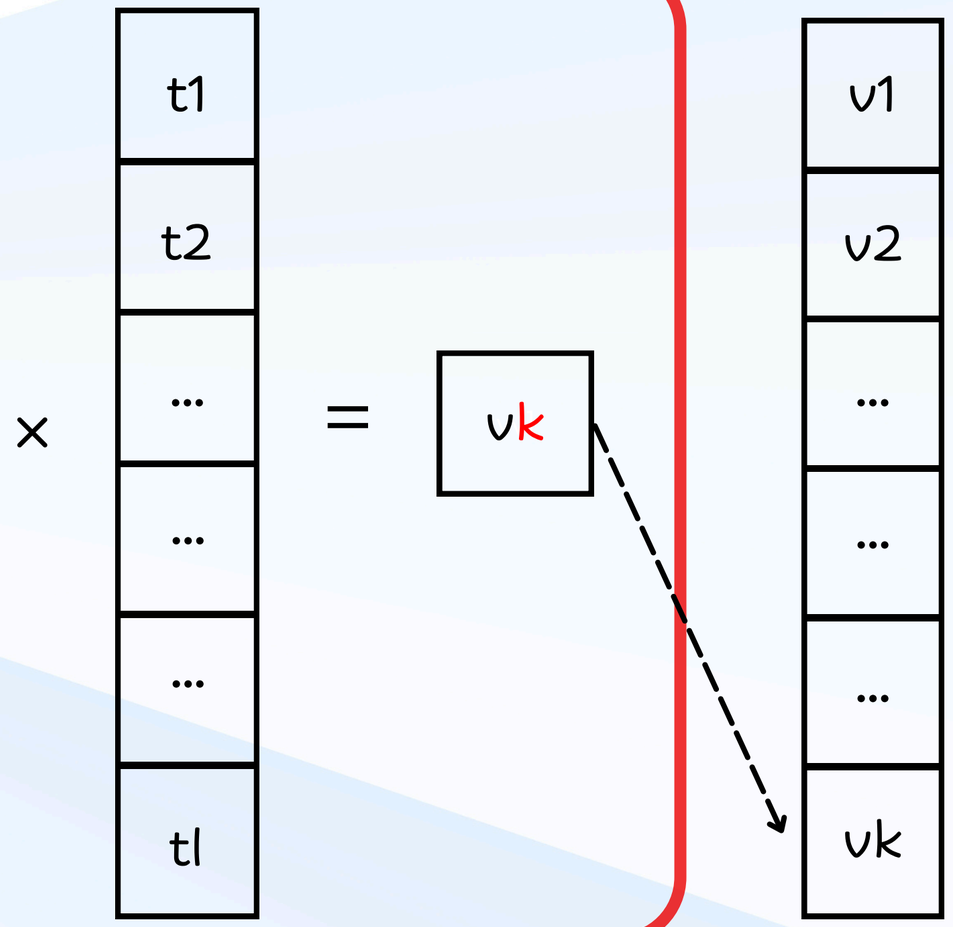
Geração via vetor de polinômios (ML-DSA-v)

i = k

Polinômios gerados a partir da seed ξ.

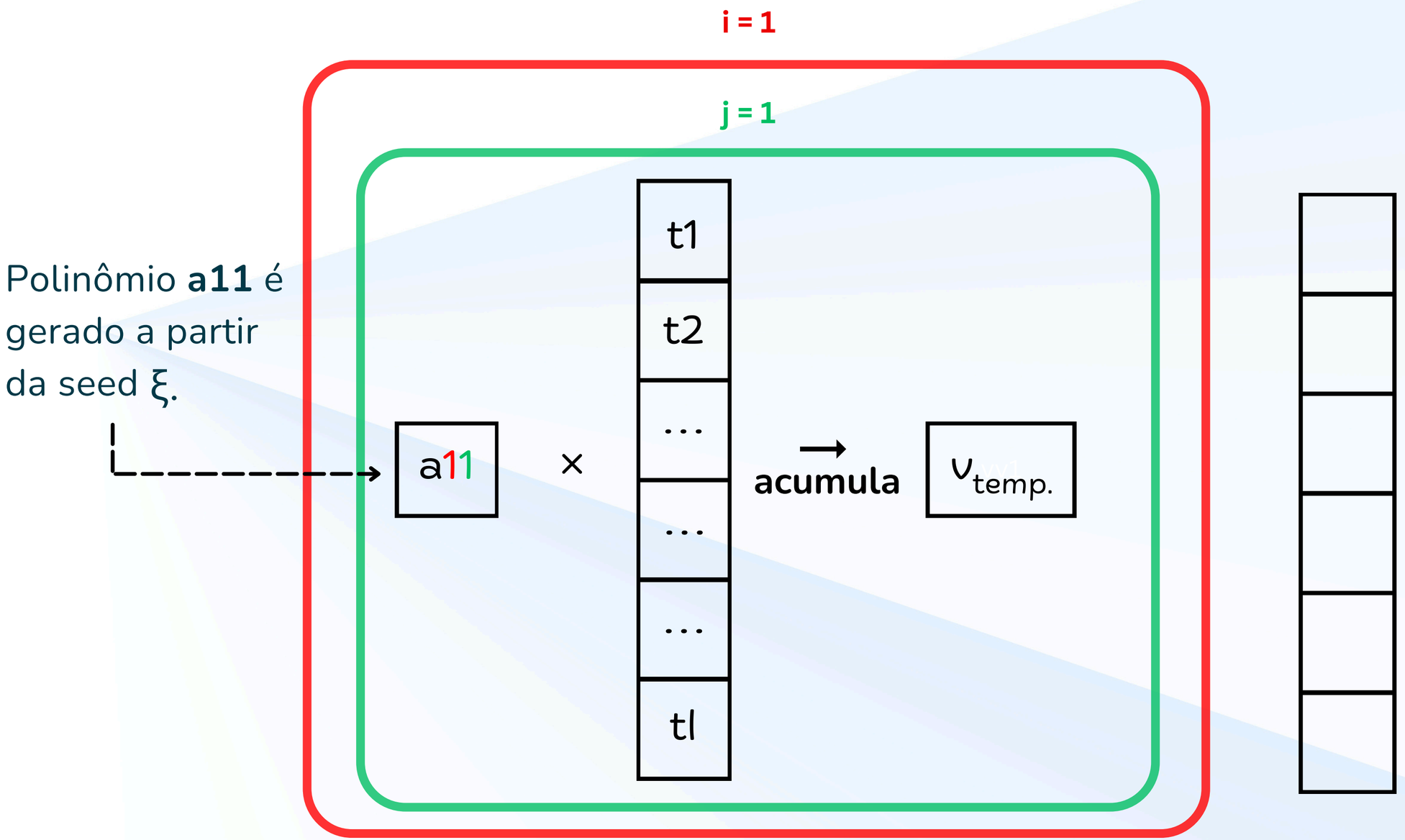


Sobrescrevendo o vetor anterior.

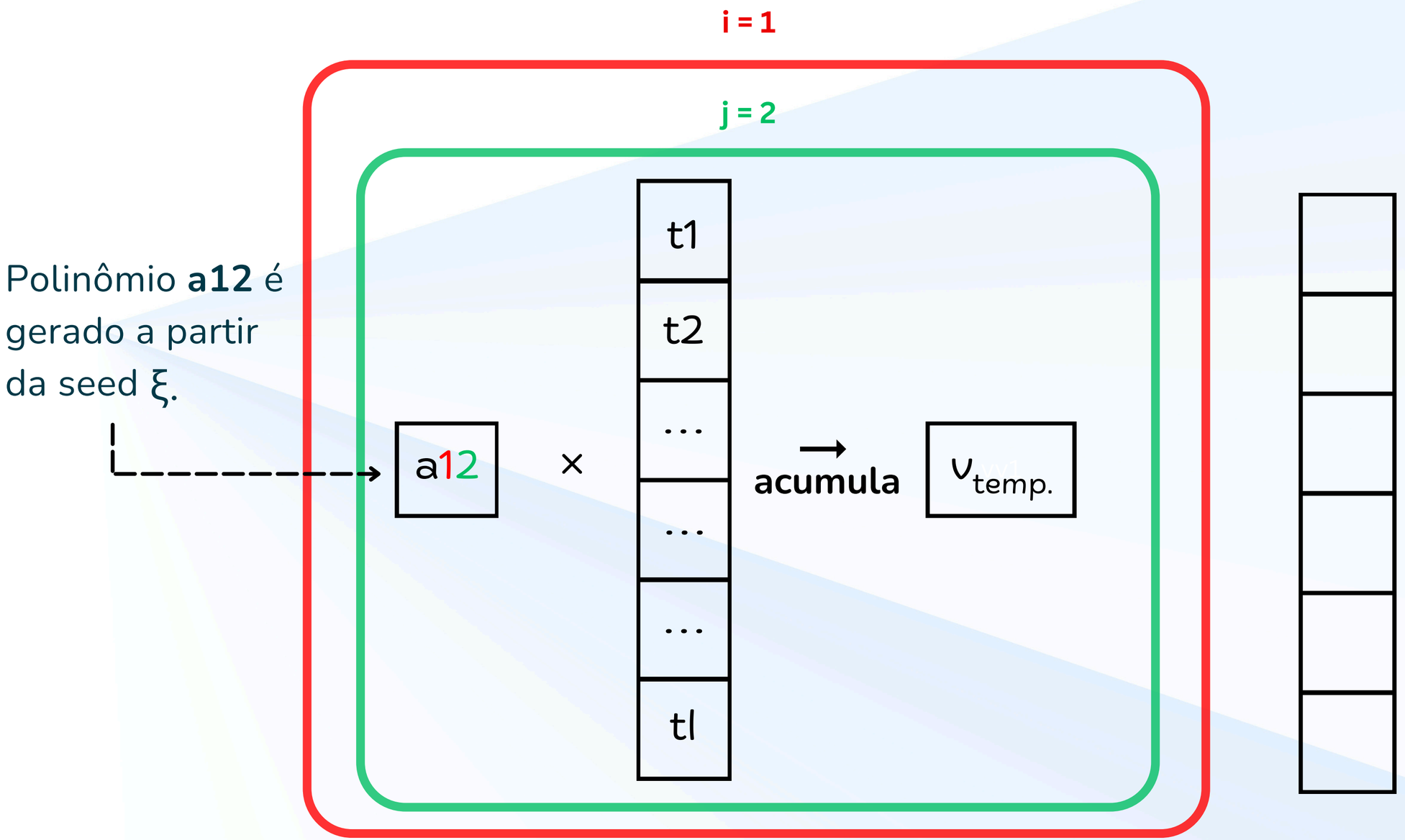


Polinômio **vk** é gerado e armazenado

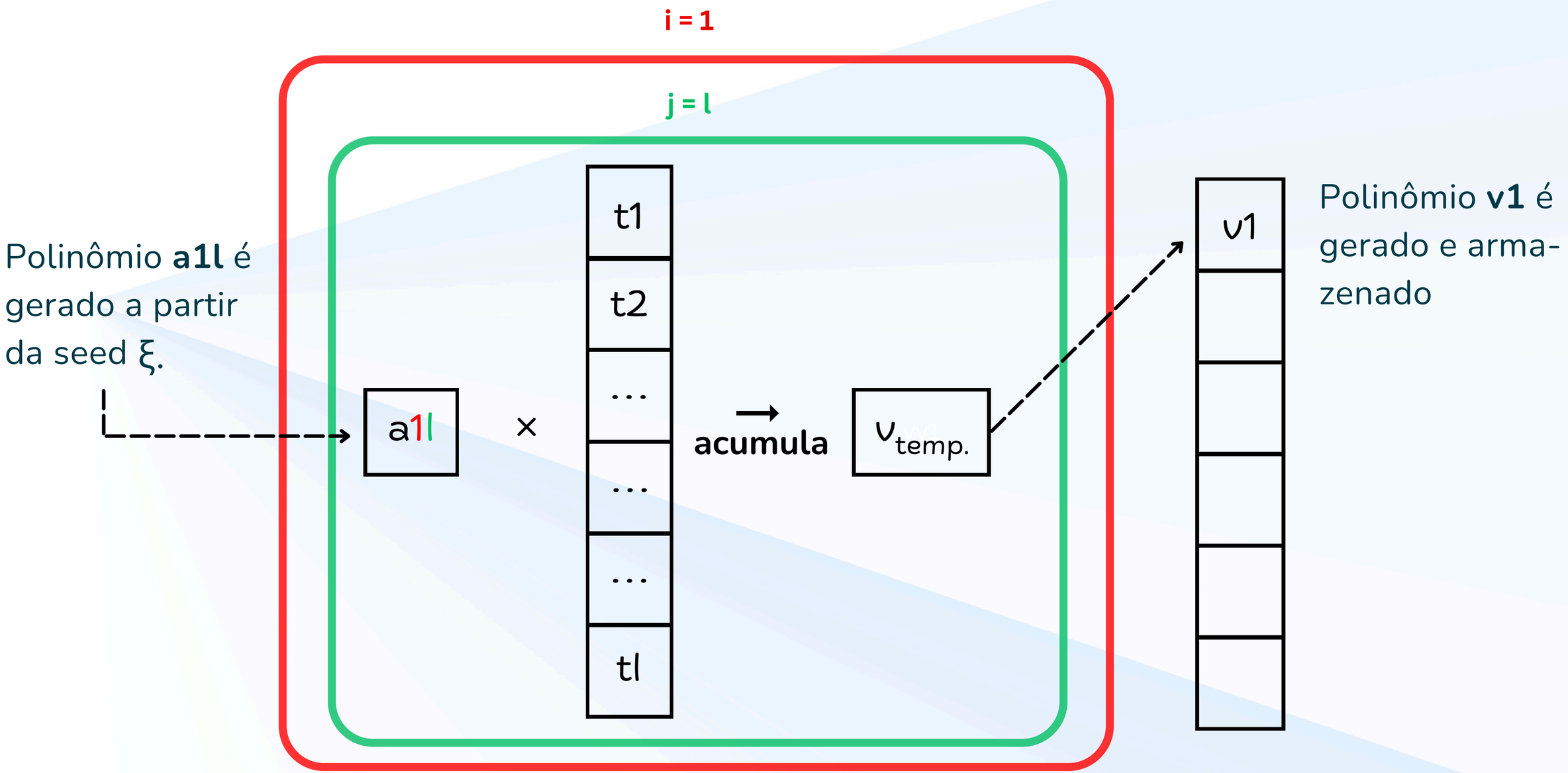
Geração via um único polinômio (ML-DSA-p)



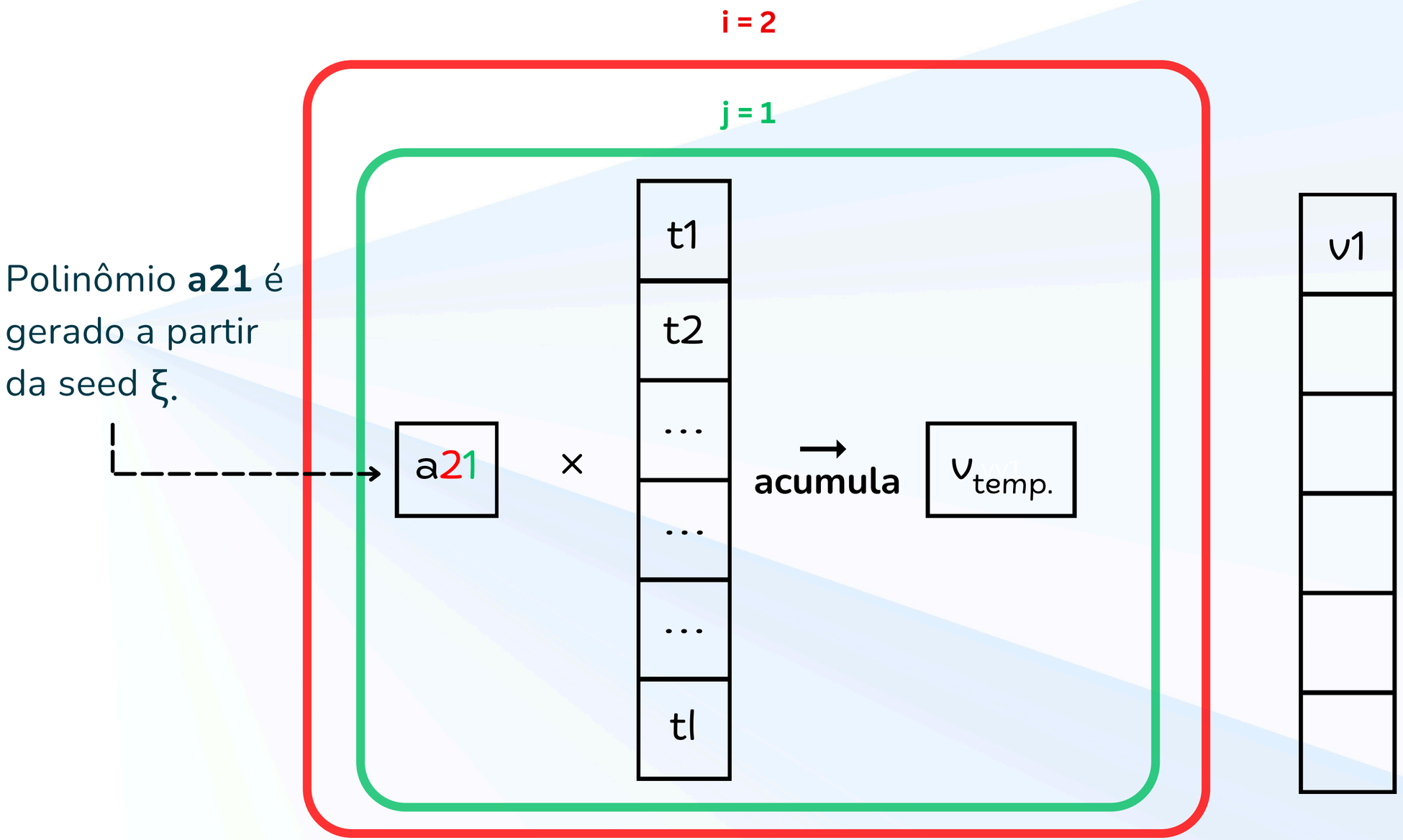
Geração via um único polinômio (ML-DSA-p)



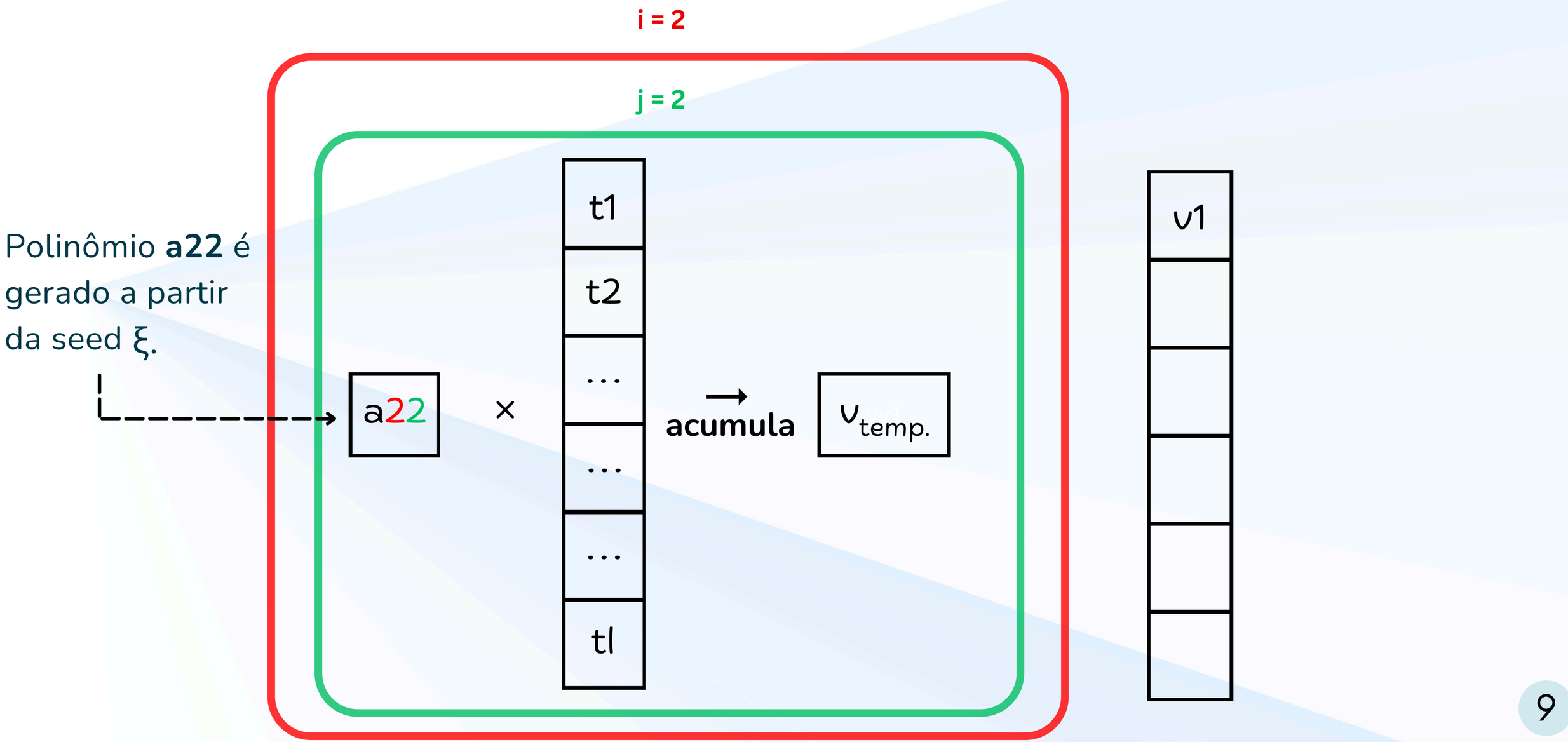
Geração via um único polinômio (ML-DSA-p)



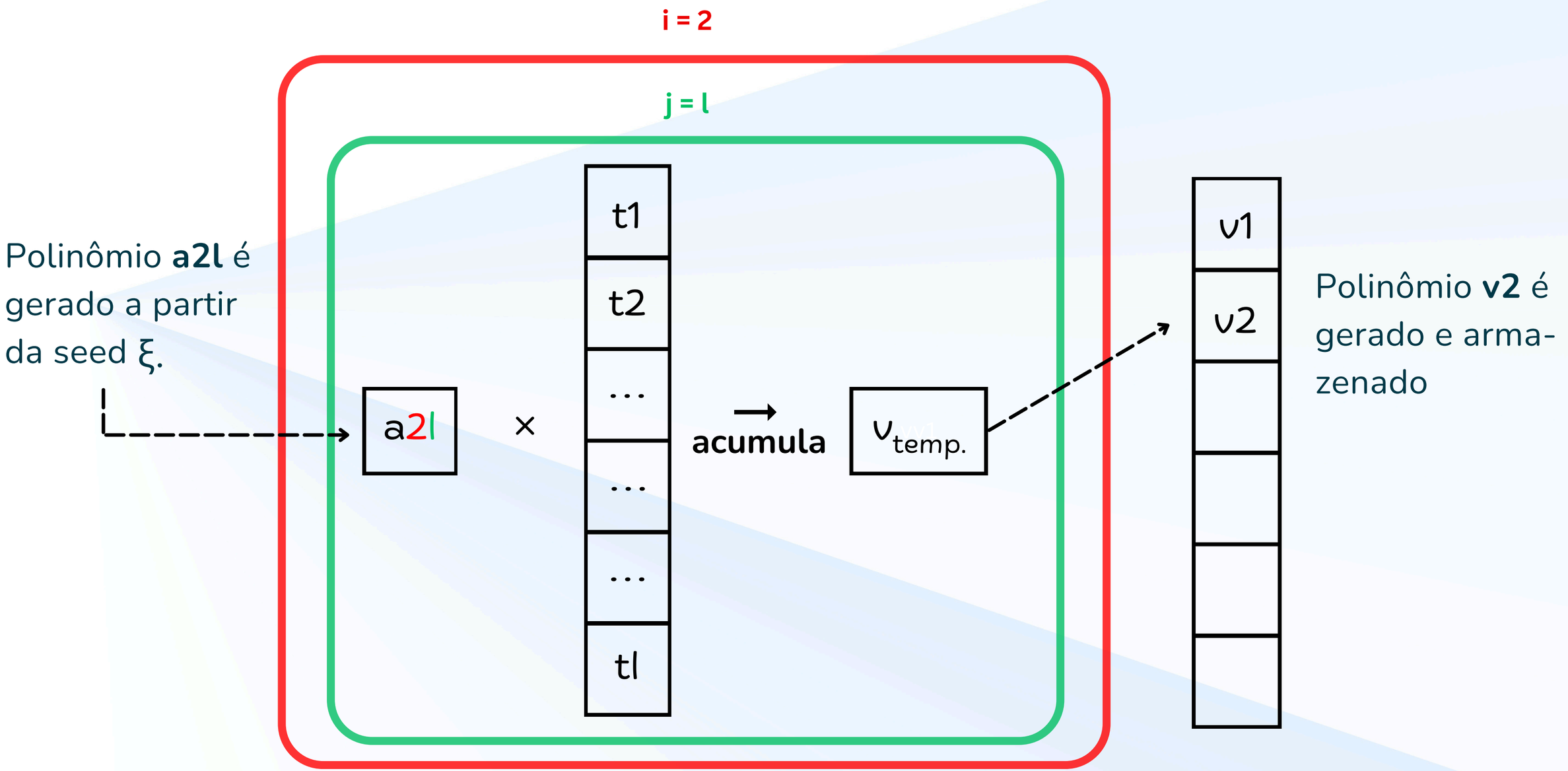
Geração via um único polinômio (ML-DSA-p)



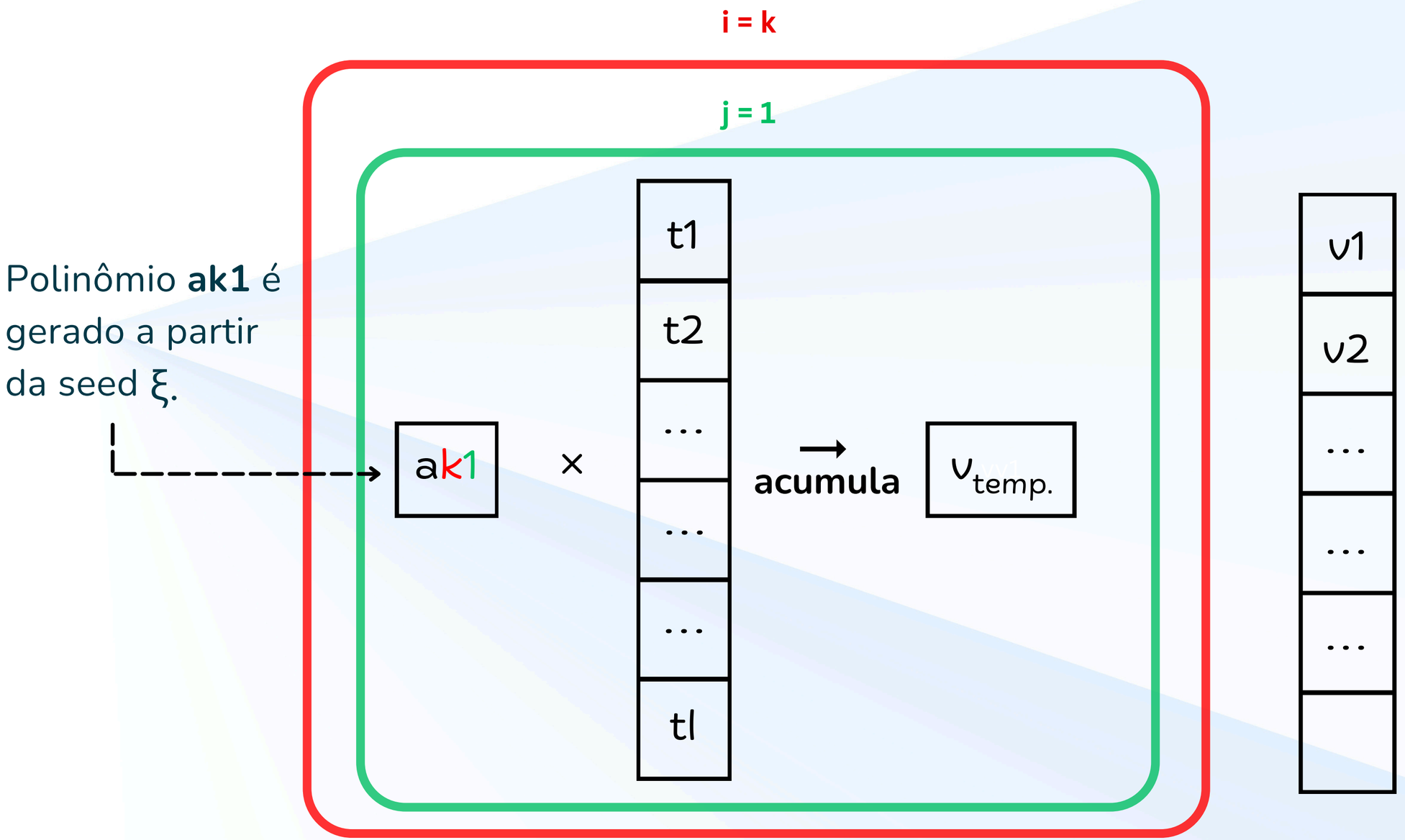
Geração via um único polinômio (ML-DSA-p)



Geração via um único polinômio (ML-DSA-p)

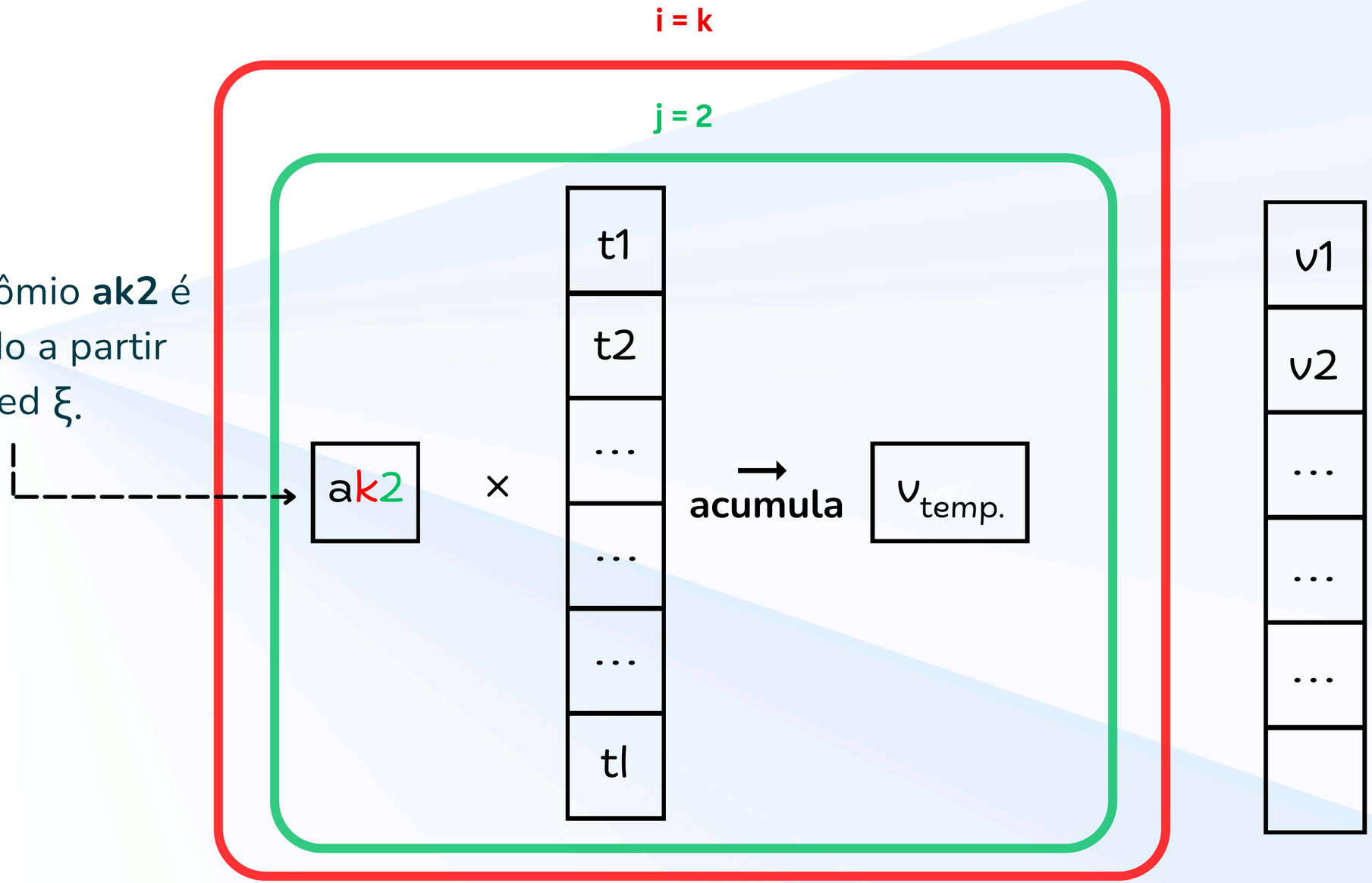


Geração via um único polinômio (ML-DSA-p)

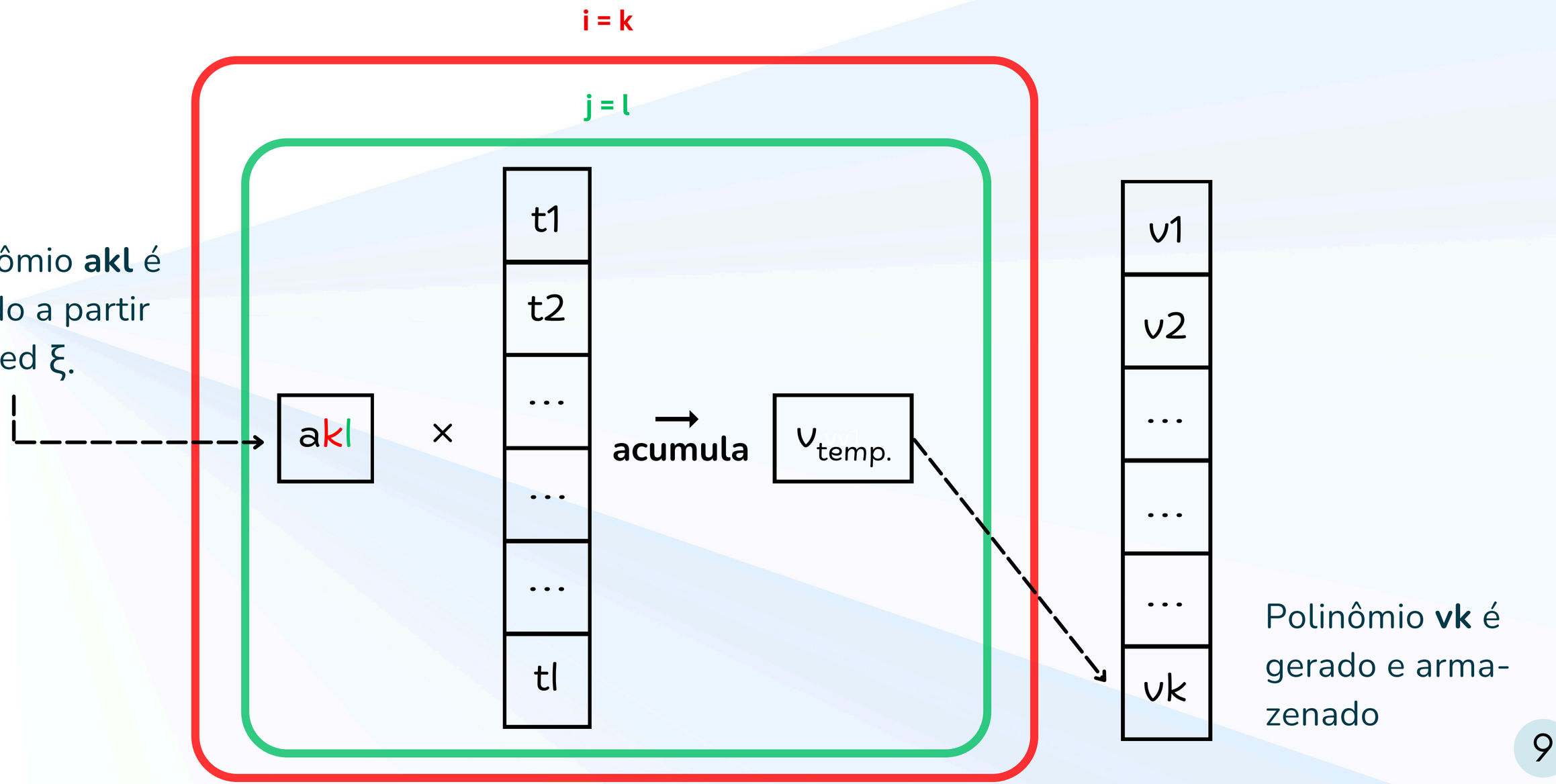


Geração via um único polinômio (ML-DSA-p)

Polinômio **ak2** é gerado a partir da seed ξ.



Geração via um único polinômio (ML-DSA-p)



Método de análise

- Realizamos medidas para o uso de memória e ciclos de CPU para cada um dos níveis de segurança do ML-DSA.

Materiais utilizados

- Utilizamos um laptop com processador Intel Core i7-1185G7, SO Ubuntu 22.04.4 LTS e 16 GiB de memória RAM.

- **PQClean:** implementações de referência em C puro.
- **Valgrind 3.18.1:** utilizado para as medições de uso de memória RAM.

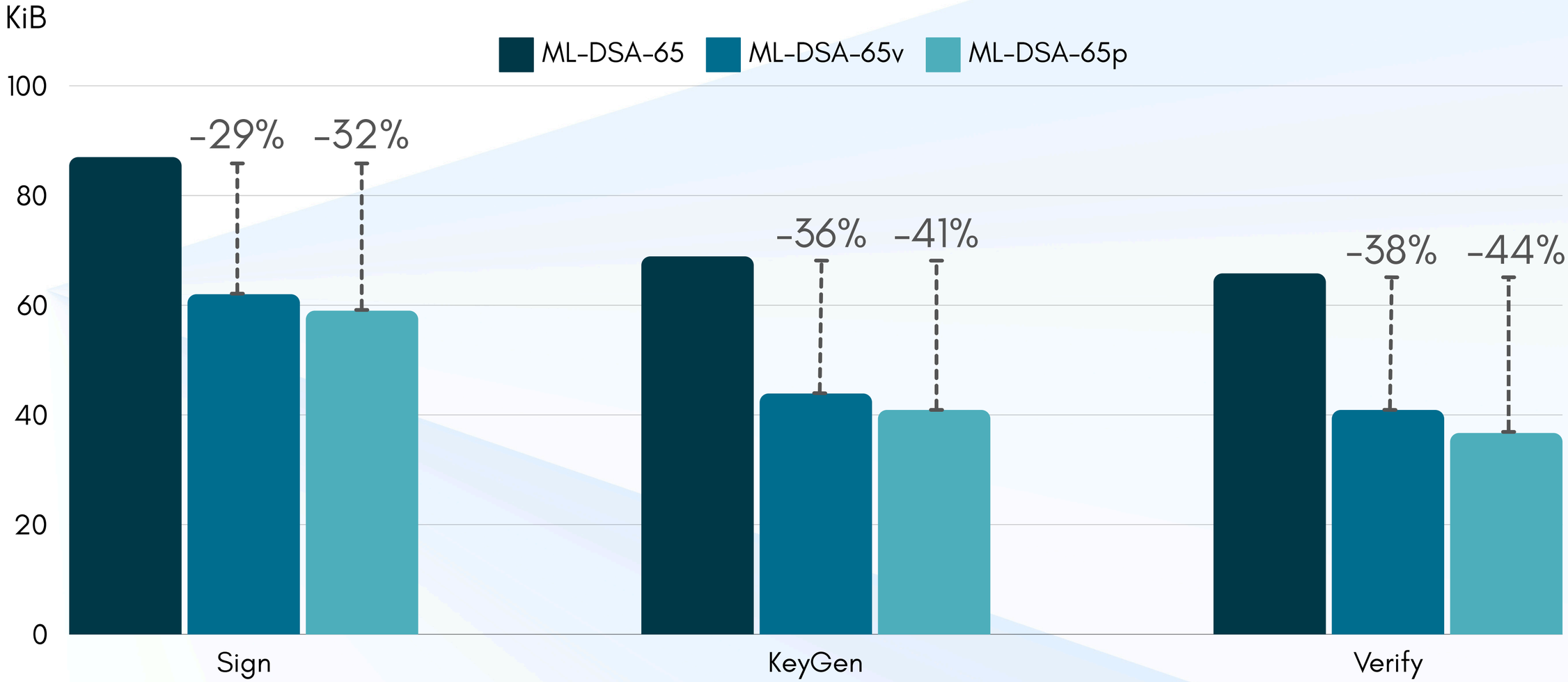
Níveis de segurança:

- ML-DSA-44
- ML-DSA-65
- ML-DSA-87

Procedimento experimental

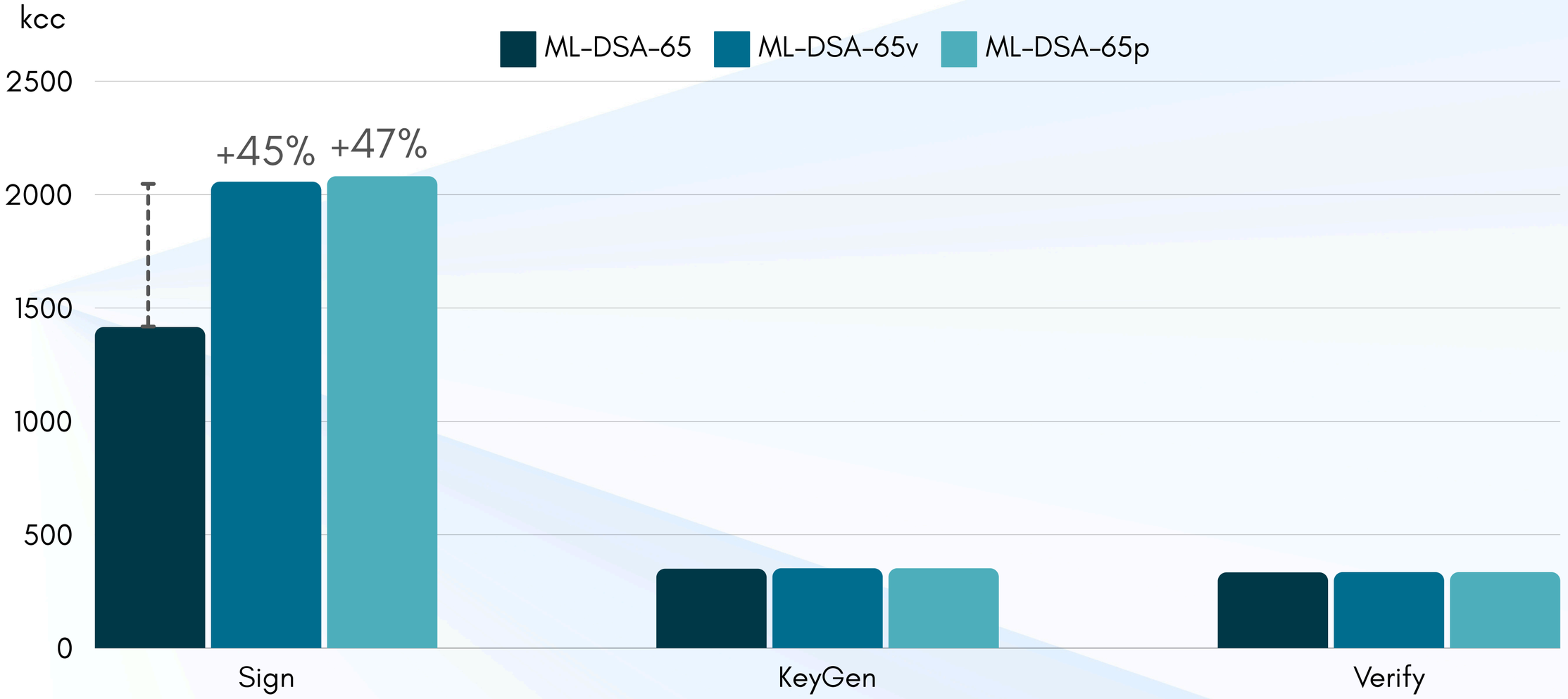
- **Picos de RAM:** análises feitas para uma única execução de KeyGen, Sign e Verify.
- **Ciclos de CPU:** medições feitas para 15000 iterações de KeyGen, Sign e Verify para uma mensagem aleatória de 59 bytes.

Resultados



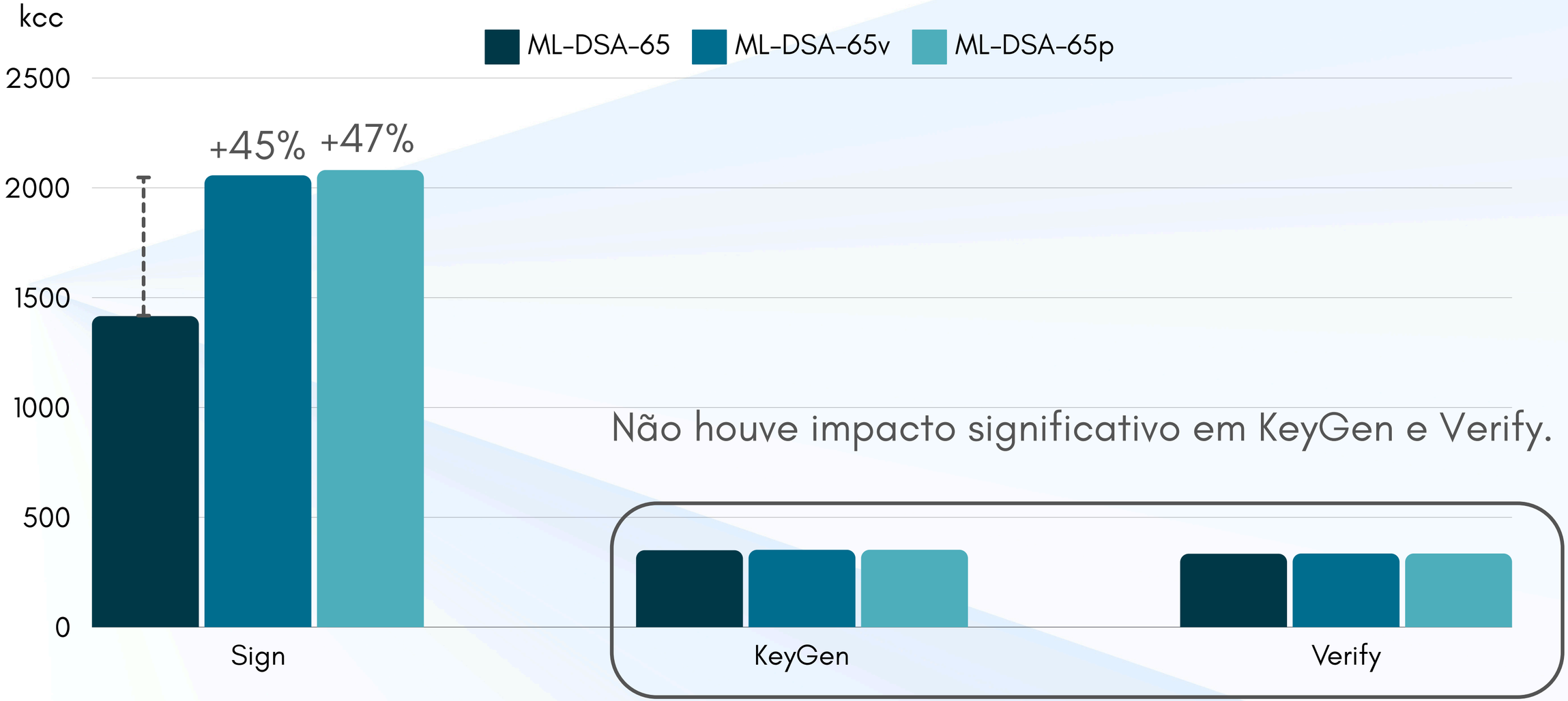
Consumo de memória RAM em ML-DSA-65

Resultados



Média de ciclos de CPU em ML-DSA-65

Resultados



Média de ciclos de CPU em ML-DSA-65

Conclusão e trabalhos futuros

- Redução significativa no uso de memória RAM para todos os níveis de segurança;
- Há um aumento tolerável no número médio de ciclos de processador;
- As otimizações propostas são independentes da arquitetura de processador utilizada.

Conclusão e trabalhos futuros

- Redução significativa no uso de memória RAM para todos os níveis de segurança;
- Há um aumento tolerável no número médio de ciclos de processador;
- As otimizações propostas são independentes da arquitetura de processador utilizada.

- Aplicação das mesmas otimizações para o **ML-KEM (Kyber)**.
- Avaliar as implementações propostas em ambientes computacionalmente restritos.

Obrigado!

r197962@dac.unicamp.br

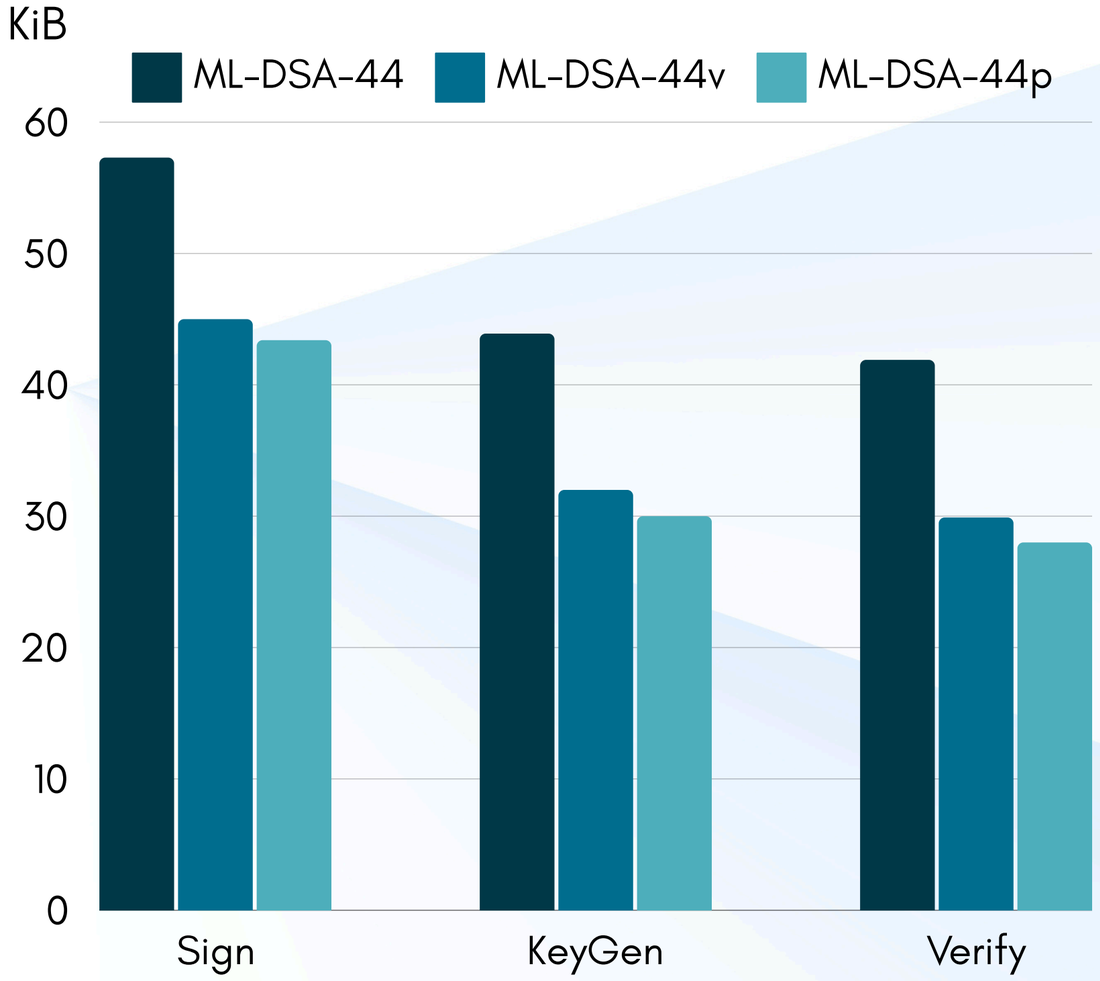
github.com/regras

Compact Memory Implementations of the ML-DSA Post-Quantum Digital Signature Algorithm

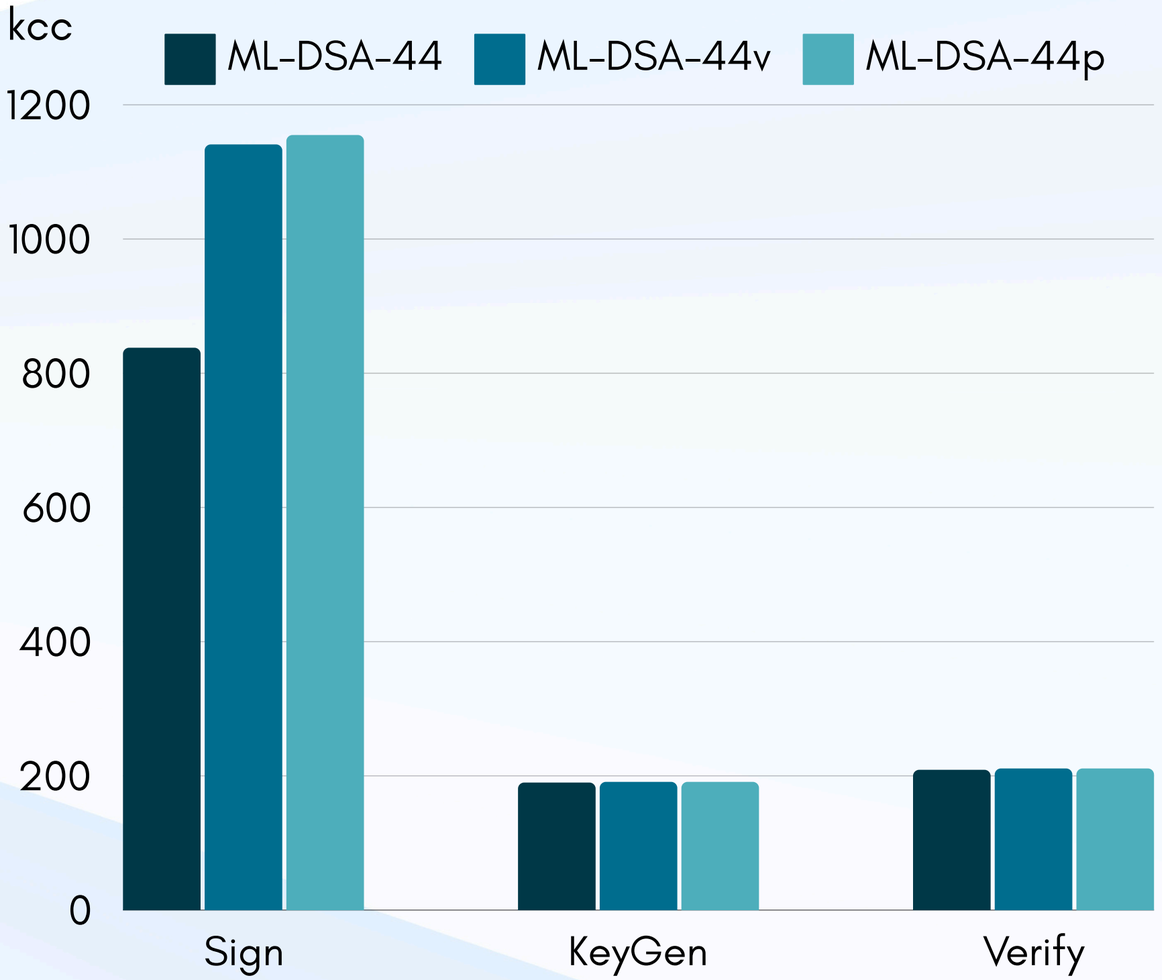
R. D. Meneses, C. Teixeira, M. A. A. Henriques

16 de Setembro, 2024

Apêndice I: resultados para ML-DSA-44

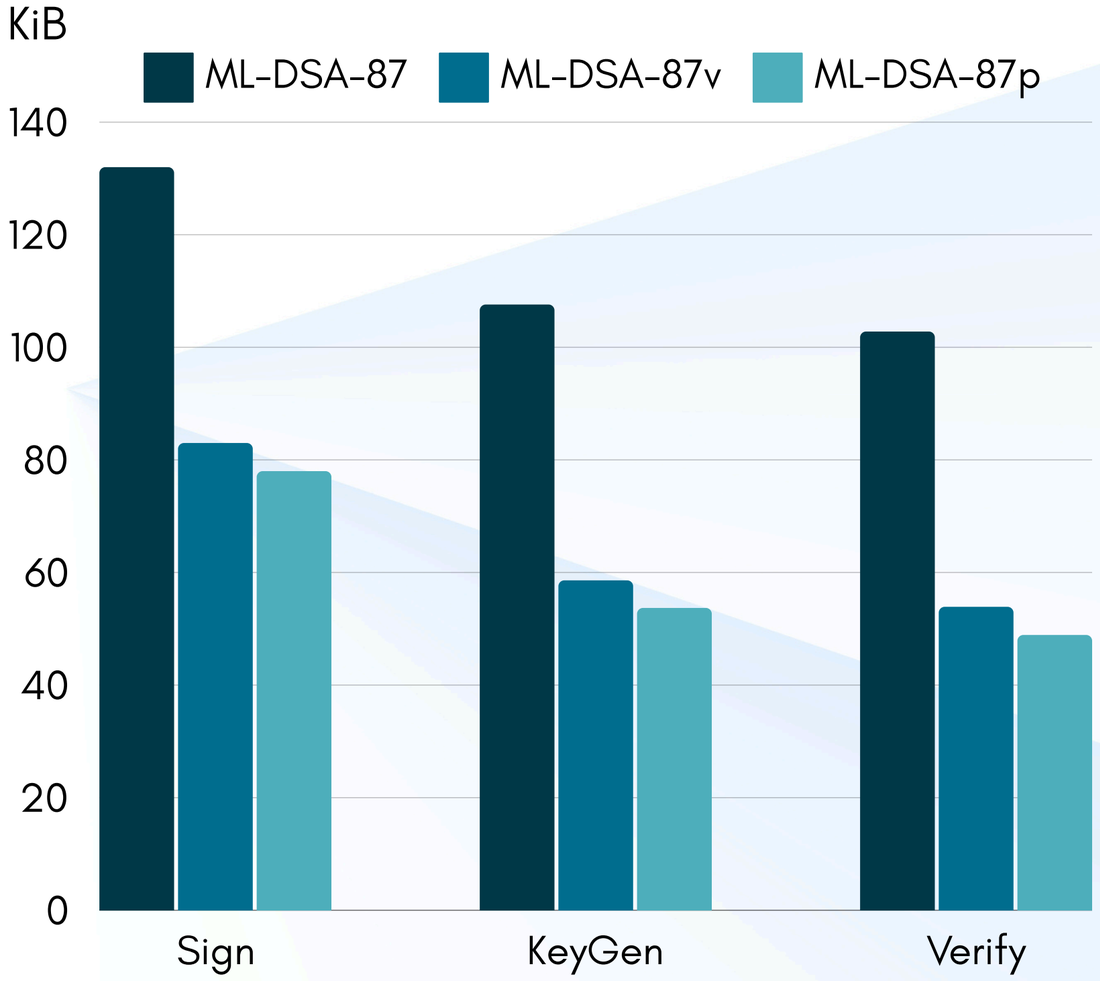


Consumo de memória RAM

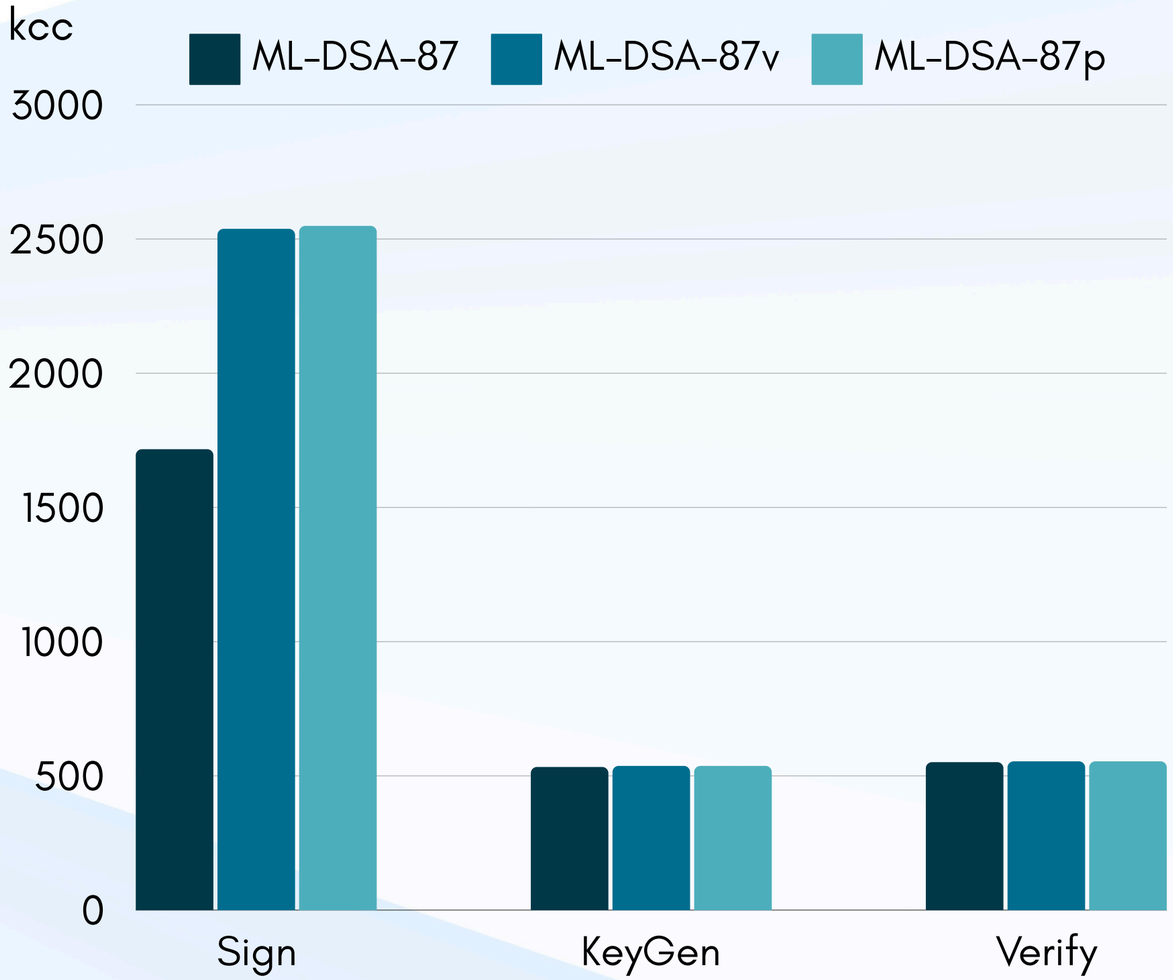


Média de ciclos de CPU

Apêndice II: resultados para ML-DSA-87



Consumo de memória RAM



Média de ciclos de CPU