Planos para a implementação do decodificador de Leech

Ramon Ribeiro

Instituto de Computação - UNICAMP

18/03/2018

Introdução

Vou implementar o decodificador de Vardy e Berry "Maximum likelihood decoding algorithm".
A principal preocupação sera em aderir aos requerimentos da criptografia. O tempo de implementação fica em segundo plano, mas continua sendo importante para praticidade do programa.

Tópicos

- Definições
- Decodificador
- Requerimentos
- Suposições
- Exatidão
- Performance

Definições

Codificar

Transformar dados em outro formato publicamente conhecido. Para manter a usabilidade do dado.

Encriptar

Transformar dados em outro formato tal que somente indivíduos especifico possam reverter esse processo. Para manter a confidencialidade do dado.

Definições

Decriptar

Reverter dados transformados em uma plataforma diferente.
Somente possível com a chave secreta correspondente.

Decodificar

Reverter dados transformados em uma plataforma diferente.
Qualquer individuo tem essa capacidade.

Maximum likelihood decoder

O decodificador usa uma construção do reticulado de Leech baseado no campo de Galois $\mathbb{F}_4 = \{0, 1, w, \overline{w}\}$, conhecido como *hexacode*, H_6.

Escolha do decodificador

O decodificador BDD possui um tempo muito mais efetivo que o nosso, no entanto, como a nossa prioridade é conseguir uma codificação mais eficaz, escolhemos o MLD.

Requerimentos

Em um sistema criptográfico deve-se ter cuidado para não deixar vazar informações que podem ser exploradas.
Para isso, proteger-se de ataques de tempo e cache são essenciais.
Os seguintes itens são os cuidados importantes para o nosso objetivo:

- Algorítimo de tempo constante
- Ramificação dependente de dados
- Acessos a memoria dependente de dados

Algorítimo de tempo constante

O algorítimo deve terminar em um numero de passos constante, independente de sua entrada.

Ramificação

CPUs modernas possuem arquiteturas pipeline. Então, varias instruções sequenciais são executadas simultaneamente.
Isso quer dizer que o o sistema deve prever qual ramificação das instruções simultâneas deve ser escolhida.
Se houver algum atraso, pode-se prever algum tipo de informação, caso os atrasos dependerem da entrada.

Acessos a memoria

Acessos a memoria são lentos.
Então, devemos implementar caminhos de acessos que são iguais para todas as entradas.

Suposições

Assumimos que as instruções de adição, subtração, multiplicação e bit-shift são de tempo constante. Isso não é verdade para todas arquiteturas computacionais.

Exatidão

Nessa secção, veremos os testes que verificam a exatidão do código.

Error-correction radius

Nesse teste, geramos pontos aleatórios do reticulado de Leech, para cada ponto adicionamos um erro aleatório dento do raio de erro do reticulado $(\lambda_1(\Lambda_{24})/2)$.
O ponto criado é decodificado e verificado contra o ponto inicial. Assim, verificamos também a distancia entre o ponto inicial mais o erro e o vetor mais próximo que a decodificação retorna.

Consistência

Nesse teste, geramos pontos contidos no espaco do reticulado.
Para cada ponto, usa-se dois decodificadores diferentes e verifica-se se eles divergem em resultados.
Para os pontos que divergem, checa-se a distancia entre os pontos do reticulado encontrados e o inicial.

Segurança por tempo

Nesse teste, geramos pontos contidos no espaco do reticulado.
Para cada ponto, usa-se o decodificador N vezes.
Depois, calculamos uma media de todos os tempos e as variaças delas. Idealmente, todas as decodificações devem terminar em tempos iguais.

Performance em codificação

Examinaremos a performance de codificar e decodificar as informações em comparação com os métodos já conhecidos para o reticulado inteiro. Para isso compararemos seu tempo médio de uso em cada processo.

Performance em encriptação

Além disso, analisaremos o funcionamento do reticulado de Leech em interação com métodos de encriptação LWE, devido a sua dimensão e características únicas.