New generic algorithms for hard knapsacks

Nick Howgrave-Graham[1] Antoine Joux[2]

35 Park St, Arlington, MA 02474
nickhg@gmail.com

DGA and Université de Versailles Saint-Quentin-en-Yvelines
antoine.joux@m4x.org

Eurocrypt 2010, Nice and Monaco, June 1st, 2010

Knapsack problems

- Given positive integers S and $a_1, \ldots, a_n$, consider equation:

$$S = \sum_{i=1}^{n} \epsilon_i a_i,$$

- Decision knapsack problem:

  Does there exist a $\{0, 1\}$ solution?

- Computational knapsack problem: Find it!

Hardness

- Hard in general:
  - The decision knapsack problem is NP-complete
  - The computational knapsack problem is NP-hard

- Easy for a large class:
  - Low density knapsacks
  - Definition of the density:

$$d = \frac{n}{\log_2 \max_i a_i}$$

  - Solved by lattice reduction oracles when $d < 0.94$

Elementary algorithms for generic knapsacks

- Exhaustive search: Time $O(2^n)$ operations
- Birthday algorithm based on:

$$\sum_{i=1}^{n/2} \epsilon_i a_i = S - \sum_{i=n/2+1}^{n} \epsilon_i a_i.$$

Time $O(2^{n/2})$, memory $O(2^{n/2})$.

State of the art: Schroeppel-Shamir algorithm

- To use birthday algorithm, it suffices to enumerate the set

$$\mathcal{S}^{(1)} = \left\{ \sum_{i=1}^{n/2} \epsilon_i a_i \right\}$$

in increasing order.

- Can be done with less memory
- Yields state of the art for generic knapsacks:
Time $O(2^{n/2})$, memory $O(2^{n/4})$.

Description of Schroeppel-Shamir algorithm

- Define the following sets (of size $2^{n/4}$):

$$\mathcal{S}_L^{(1)} = \left\{ \sum_{i=1}^{n/4} \epsilon_i a_i \right\}$$

$$\mathcal{S}_R^{(1)} = \left\{ \sum_{i=n/2+1}^{n/2} \epsilon_i a_i \right\}$$

- Any $\sigma \in \mathcal{S}^{(1)}$ can be written $\sigma = \sigma_L + \sigma_R$

- Moreover:

$$\sigma_L + \sigma_R < \sigma_L + \sigma'_R \quad \text{iff} \quad \sigma_R < \sigma'_R$$

Schroeppel-Shamir continued

- If $\mathcal{S}_R^{(1)}$ is sorted:
  - Finding successor of $\sigma_L + \sigma_R$ with same σ_L is easy
- How to interlace different σ_L values?

- Schamir and Schroeppel idea:
  - Create set of triples $(\sigma_L + \sigma_R^{(0)}, \sigma_L, \sigma_R^{(0)})$
  - Repeat:
    - Extract triple with minimum $\sigma_L + \sigma_R$ from the set
    - Update into successor $(\sigma_L + \sigma'_R, \sigma_L, \sigma'_R)$

  - Require priority queue (heap or balanced tree)

A modular variant of Schroeppel-Shamir

- Let M be a prime near $2^{n/4}$
- For a knapsack solution $\sigma_L^{(1)}$, $\sigma_R^{(1)}$, $\sigma_L^{(2)}$ and $\sigma_R^{(2)}$, we have:

$$\sigma_L^{(1)} + \sigma_R^{(1)} \equiv S - \sigma_L^{(2)} - \sigma_R^{(2)} \pmod{M}.$$

- Let σ_M denotes this "middle value"
- Algorithm becomes:
  - For each possible value of σ_M:
    - For each $\sigma_L^{(1)}$, find all $\sigma_R^{(1)}$ such that:

$$\sigma_L^{(1)} + \sigma_R^{(1)} \equiv \sigma_M \pmod{M}.$$

    - For each $\sigma_L^{(2)}$, find all $\sigma_R^{(2)}$ such that:

$$\sigma_L^{(2)} + \sigma_R^{(2)} \equiv S - \sigma_M \pmod{M}.$$

    - Match the above two lists for exact solution (not only mod M)

Schroeppel-Shamir for unbalanced knapsacks

- Knapsack with extra information:

$$\sum_{i=1}^{n} \epsilon_i = \alpha n$$

- Build sets of $\alpha n/4$ elements in each quarter
- Need "good decomposition" $\Rightarrow$ extra polynomial factor
- Time and memory:

$$\begin{aligned}\binom{n/2}{\alpha n/2} &\approx \left(\frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}}\right)^{n/2} \\ \binom{n/4}{\alpha n/4} &\approx \left(\frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}}\right)^{n/4}.\end{aligned}$$

Schroeppel-Shamir can be improved

- For simplicity, assume exactly $n/2$ elements a_i appear in S
- Consider decompositions:

$$S = \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4,$$

where each σ_j is a sum of exactly $n/8$ values (among n).

- A given solution of the knapsack can be split into:

$$\binom{n/2}{n/8 \; n/8 \; n/8 \; n/8} = \frac{(n/2)!}{(n/8)!^4} \approx 2^n$$

decompositions $\sigma_1 + \sigma_2 + \sigma_3 + \sigma_4$.

Schroeppel-Shamir can be improved (2)

- Fix modulus M, random values R_1, R_2 and R_3
- Search only decompositions with:

$$\sigma_1 \equiv R_1 \pmod{M} \qquad \sigma_2 \equiv R_2 \pmod{M}$$
$$\sigma_3 \equiv R_3 \pmod{M} \qquad \sigma_4 \equiv S - R_1 - R_2 - R_3 \pmod{M}$$

- Since first 3 conditions imply the fourth:
  - Expect one decomposition on average when $M \approx 2^{n/3}$.

Warning: yields a randomized algorithm !

First algorithm

- Given M, R_1, R_2 and R_3
- Solve four unbalanced knapsacks with $\alpha = 1/8$ on n elements modulo M
- Expect:

$$\binom{n}{\alpha n} \cdot M^{-1} \approx \left(\frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}}\right)^n \cdot M^{-1} \approx 2^{0.210\,n}$$

  solutions for each.
  - Costs time $\approx 2^{0.272\,n}$ (and memory $\approx 2^{0.136\,n}$)
- Use Shamir-Schroeppel again to paste the four sets of solutions together:
  - Costs time $\approx 2^{0.420\,n}$ and memory $\approx 2^{0.210\,n}$

Can be improved using smaller value of M (see paper)

Going further

- Instead of cutting in four, cut in two
- Choose M, $R \pmod{M}$ and write $S = \sigma_1 + \sigma_2$
  - With $\sigma_1 \equiv R$ and $\sigma_2 \equiv S - R \pmod{M}$
- Solve two unbalanced knapsacks with $\alpha = 1/4$ on n elements modulo M
  - Assume that subknapsacks can be solved efficiently.

Going further

- Instead of cutting in four, cut in two
- Choose M, $R \pmod{M}$ and write $S = \sigma_1 + \sigma_2$
  - With $\sigma_1 \equiv R$ and $\sigma_2 \equiv S - R \pmod{M}$
- Solve two unbalanced knapsacks with $\alpha = 1/4$ on n elements modulo M
  - Assume that subknapsacks can be solved efficiently.
- Yields $\binom{n/2}{n/4} \approx 2^{n/2}$ decompositions
- Would yield complexity:

$$2^{-n/2} \binom{n}{n/4} \approx 2^{0.311\,n}.$$

Going further

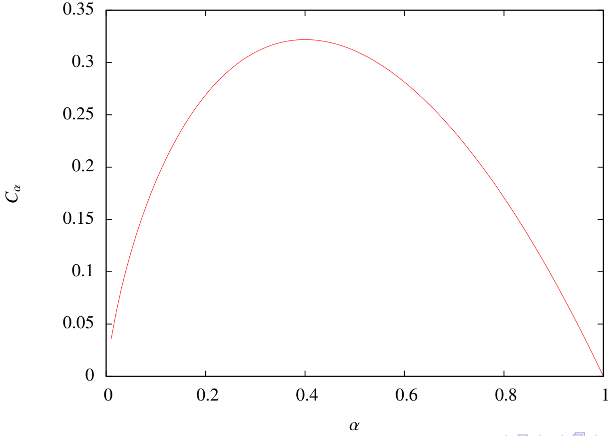
- Instead of cutting in four, cut in two
- Choose M, $R \pmod{M}$ and write $S = \sigma_1 + \sigma_2$
  - With $\sigma_1 \equiv R$ and $\sigma_2 \equiv S - R \pmod{M}$
- Solve two unbalanced knapsacks with $\alpha = 1/4$ on n elements modulo M
  - Assume that subknapsacks can be solved efficiently.
- Yields $\binom{n/2}{n/4} \approx 2^{n/2}$ decompositions
- Would yield complexity:

$$2^{-n/2}\binom{n}{n/4} \approx 2^{0.311\,n}.$$

Does assumption holds ?

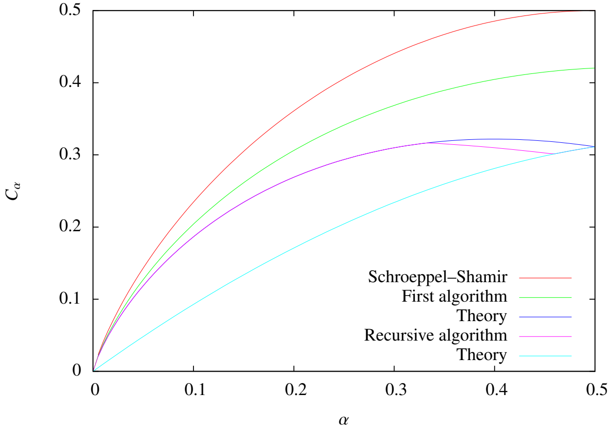
Going further: solving subknapsacks

- Essentially, use idea recursively.
- Problem due to unbalanced knapsacks $\frac{\binom{n}{\alpha\, n/2}}{\binom{\alpha\, n}{\alpha\, n/2}} \approx 2^{C_\alpha\, n}$.



Going further: solving subknapsacks

- Bad news: Clear problem when $\alpha < 1/3$
- Good news: Limited depth of recursion
  - Switch to Schroeppel-Shamir at some point



Proof technique

- Prove that knapsack evaluation mod M are well distributed.
- Basic tool from [Nguyen, Shparlinski, Stern 2001]
  - Use exponential sum techniques
- Need $M \leq 2^n$ and M decreasing during recursion

Proof technique

- Prove that knapsack evaluation mod M are well distributed.
- Basic tool from [Nguyen, Shparlinski, Stern 2001]
  - Use exponential sum techniques
- Need $M \leq 2^n$ and M decreasing during recursion

Algorithm proved for overwhelming fraction of random knapsacks

Is it practical ?

- We chose $n = 96$ and summed 48 elements.
- Schroeppel- Shamir 1:
  - Time 1 500 days, Memory 1.8 Gbytes
- Schroeppel- Shamir 2:
  - Time 4 400 days, Memory 300 Mbytes
- Our best implementation (heuristic):
  - Time 10 hours, Memory 1.7 Gbytes

Is it practical ?

- We chose $n = 96$ and summed 48 elements.
- Schroeppel- Shamir 1:
  - Time 1 500 days, Memory 1.8 Gbytes
- Schroeppel- Shamir 2:
  - Time 4 400 days, Memory 300 Mbytes
- Our best implementation (heuristic):
  - Time 10 hours, Memory 1.7 Gbytes

- For more details:
  - See proceedings or IACR eprint 2010/189.

Generalizations

- Modular knapsacks (already used in the recursion)
- Noisy knapsacks
- Vectorial knapsacks

Conclusion