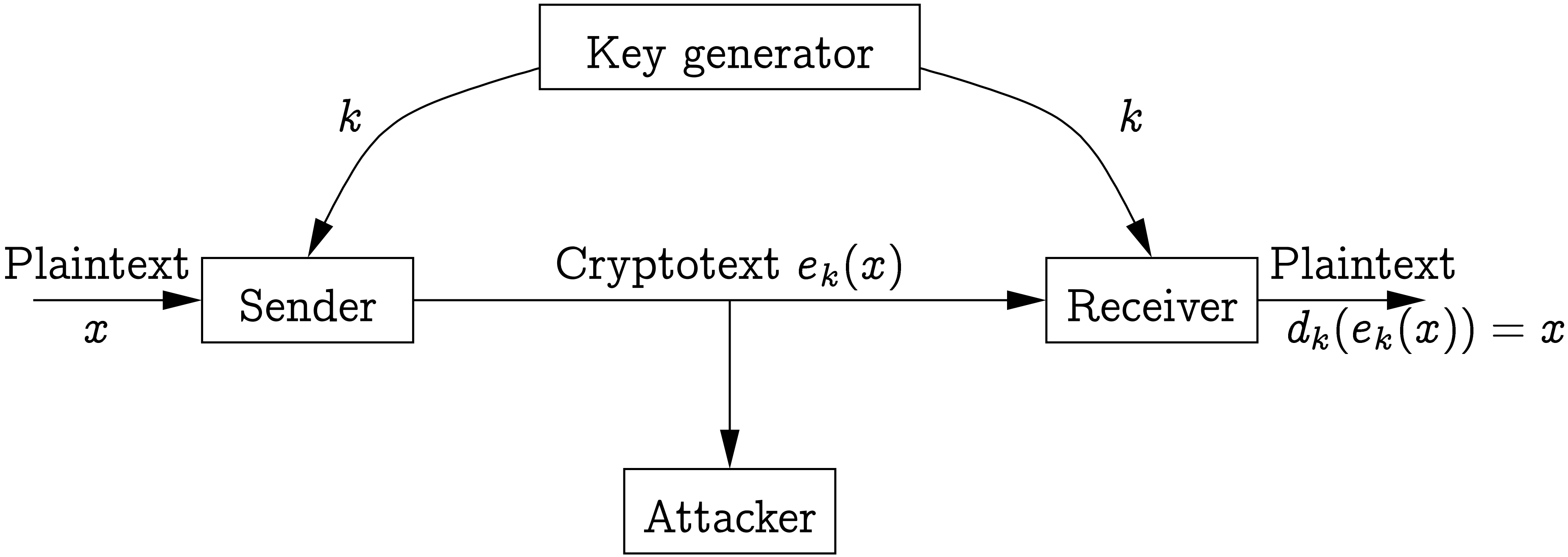
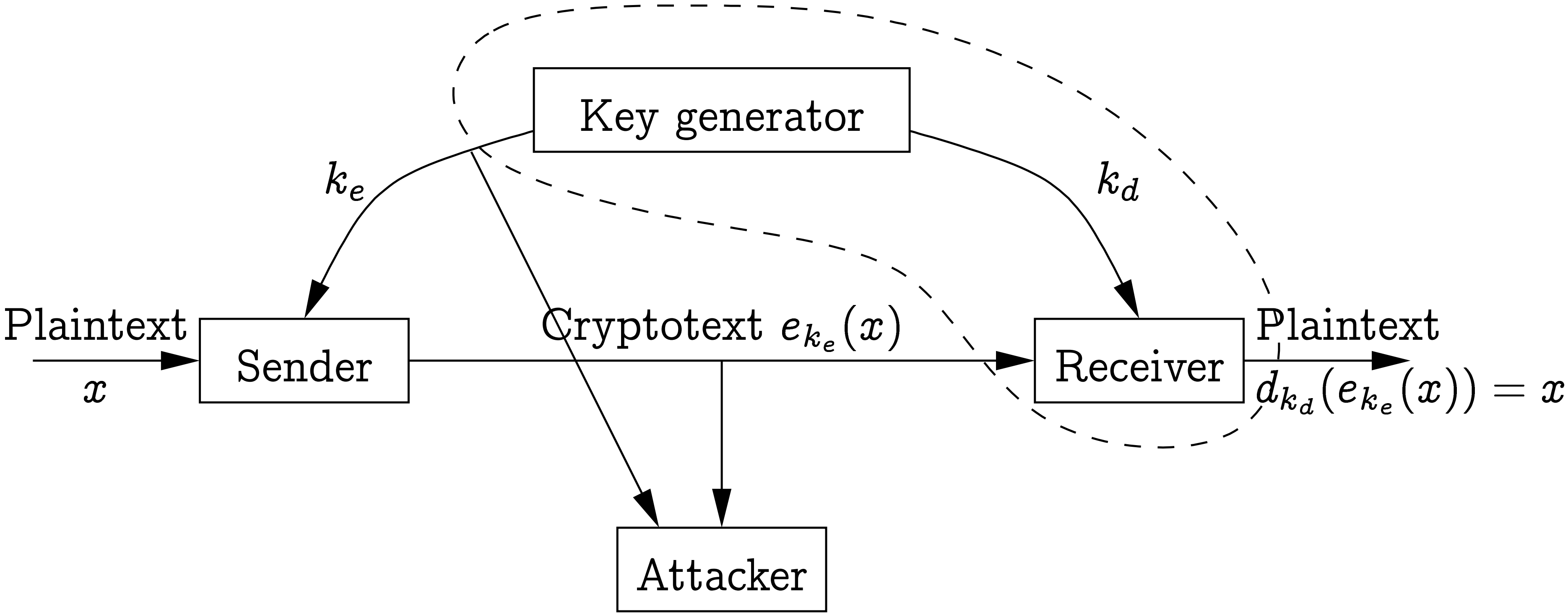
Symmetric encryption:



The rules for encoding and decoding are both given using the same secret k.

Asymmetric encryption:



The rules for encoding and decoding are given by different bit-strings. The bit-string k_e giving the encoding rule is not sensitive.

Finding k_d from k_e should be infeasible.

A function $f : \{0, 1\}^* \to \{0, 1\}^*$ is one-way if

- computing $f(x)$ from x is easy (for almost all x);
- given y, finding an x such that $f(x) = y$, is infeasible on average.

A family of functions $\{f_i\}_{i \in I}$ is one-way if

- computing $f_i(x)$ from x is easy for almost all i and x;
- given y and i, finding an x such that $f_i(x) = y$, is infeasible (averaged over y and i).

The encoding function must be a one-way family (parametrized by the public keys) of functions.

If e is one way, then how does one decode?

A family of functions $\{f_i\}_{i \in I}$ is trapdoor ($tagauksega$) one-way if

- $\{f_i\}_{i \in I}$ is one-way;
- for each i exists i_t, such that given y, i and i_t, it is easy to find an x, such that $f_i(x) = y$.
- Pairs (i, i_t) are easily generated together.

i is the public key. The trapdoor i_t is (a part of) the secret key.

A hard (NP-complete) problem: SUBSET-SUM.

Given: a vector of integers $(a_1, \ldots, a_n)$ and $s \in \mathbb{Z}$.

Determine whether there exist such $x_1, \ldots, x_n$, that $x_i \in \{0, 1\}$ and $\sum_{i=1}^{n} x_i a_i = s$.

Computational version: find those x_i, if they exist.

The vector $(a_1, \ldots, a_n)$ is called the *knapsack*.

Consider the knapsack

$$a = (143, 125, 67, 85, 201, 98, 46, 176, 128, 54, 83) \quad .$$

Then

- $a, 646$ has a solution because $646 = 125 + 201 + 98 + 46 + 176.$
- $a, 589$ has no solutions.
- $a, 833$ has two solutions: $833 = 125 + 67 + 85 + 201 + 98 + 46 + 128 + 83 = 143 + 85 + 201 + 46 + 176 + 128 + 54.$

To solve the instance $(a_1, \ldots, a_n)$, s of SUBSET-SUM:

Generate all possible vectors $(x_1, \ldots, x_n) \in \{0,1\}^n$ and check whether $\sum_{i=1}^{n} x_i a_i = s$.

Time complexity: $O(2^n)$. Space complexity: $O(n)$.

A faster, "meet-in-the-middle" algorithm:

Let $n = 2m$. Define the sets

$$S_1 = \{\sum_{i=1}^{m} x_i a_i \,|\, (x_1, \ldots, x_m) \in \{0,1\}^m\}$$

$$S_2 = \{s - \sum_{i=m+1}^{n} x_i a_i \,|\, (x_{m+1}, \ldots, x_n) \in \{0,1\}^m\}$$

Sort both S_1 and S_2 and check whether some value occurs in both sets.

Time complexity: $O(n2^{n/2})$. Space complexity: $O(2^{n/2})$.

Fastest known algorithm for solving general instances of SUBSET-SUM.

Suppose that $(a_1, \ldots, a_n)$ are such, that all 2^n possible sums are different.

We can define an encoding function

$$e_{(a_1,\ldots,a_n)} : \{0,1\}^n \to \mathbb{Z}$$

$$e_{(a_1,\ldots,a_n)}(x_1 \cdots x_n) = \sum_{i=1}^{n} x_i a_i \quad .$$

The function family e might be one-way...

Where is the trapdoor?

A knapsack $(a_1, \ldots, a_n)$ is superincreasing if $a_i > \sum_{j=1}^{i-1} a_j$ for all $i \in \{1, \ldots, n\}$.

Instances of SUBSET-SUM, where the knapsack is superincreasing, can be easily solved with a greedy algorithm.

In Merkle-Hellman singly-iterated knapsack cryptosystem, the main part of the secret key is a superincreasing knapsack $(b_1, \ldots, b_n)$.

The public key is a transformed version of that knapsack, such that it "looks like a general instance of a knapsack".

Transformation: pick $M \in \mathbb{N}$ such, that $M > \sum_{i=1}^{n} b_i$. Also pick $W \in \mathbb{Z}_M^*$.

Let $a_i = Wb_i \bmod M$. Public key: $(a_1, \ldots, a_n)$.

And the secret key was $((b_1, \ldots, b_n), M, U)$ where $U = W^{-1} \pmod{M}$.

Decoding: when we recieve $s \in \mathbb{Z}$ then compute $s' = s \cdot U \bmod M$. Then solve the SUBSET-SUM instance $((b_1, \ldots, b_n), s')$.

Theorem. If the SUBSET-SUM instance $((a_1, \ldots, a_n), s)$ has a solution then the instance $((b_1, \ldots, b_n), s \cdot U \bmod M)$ also has a unique solution. Moreover, these two solutions are equal.

Example: let $n = 10$ and consider the superincreasing knapsack

$$(1, 2, 5, 9, 20, 39, 81, 159, 318, 643) \quad .$$

Then M must be greater than 1277. Pick $M = 1301$ and $W = 517$. Then $U = 765$.

To construct the public knapsack, multiply the elements of the secret knapsack by $517 \pmod{1301}$, giving

$$(517, 1034, 1284, 750, 1233, 648, 245, 240, 480, 676) \quad .$$

Public key:

$$(517, 1034, 1284, 750, 1233, 648, 245, 240, 480, 676)$$

To encode the bit-string 0110011010 compute

$$0 \cdot 517 + 1 \cdot 1034 + 1 \cdot 1284 + 0 \cdot 750 + 0 \cdot 1233 + 1 \cdot 648 \\ + 1 \cdot 245 + 0 \cdot 240 + 1 \cdot 480 + 0 \cdot 676 = 3691 \quad .$$

The cryptotext is 3691.

Secret key:

$$(1, 2, 5, 9, 20, 39, 81, 159, 318, 643),\ \ 1301,\ \ 765$$

To decode 3691, compute $3691 \cdot 765 \bmod 1301 = 445$. Solve the superincreasing knapsack:

$445 < 643$	$445 - \mathbf{0} \cdot 643 = 445$	$7 < 20$	$7 - \mathbf{0} \cdot 20 = 7$
$445 \geqslant 318$	$445 - \mathbf{1} \cdot 318 = 127$	$7 < 9$	$7 - \mathbf{0} \cdot 9 = 7$
$127 < 159$	$127 - \mathbf{0} \cdot 159 = 127$	$7 \geqslant 5$	$7 - \mathbf{1} \cdot 5 = 2$
$127 \geqslant 81$	$127 - \mathbf{1} \cdot 81 = 46$	$2 \geqslant 2$	$2 - \mathbf{1} \cdot 2 = 0$
$46 \geqslant 39$	$46 - \mathbf{1} \cdot 39 = 7$	$0 < 1$	$0 - \mathbf{0} \cdot 1 = 0$

The plaintext was 0110011010.

The cryptosystem is insecure because $(a_1, \ldots, a_n)$ does not quite "look like a general instance of a knapsack".

We are given $(a_1, \ldots, a_n)$. We want to find a superincreasing $(b_1, \ldots, b_n)$, U and M, such that $b_i = a_i \cdot U \bmod M$ and the previous theorem holds.

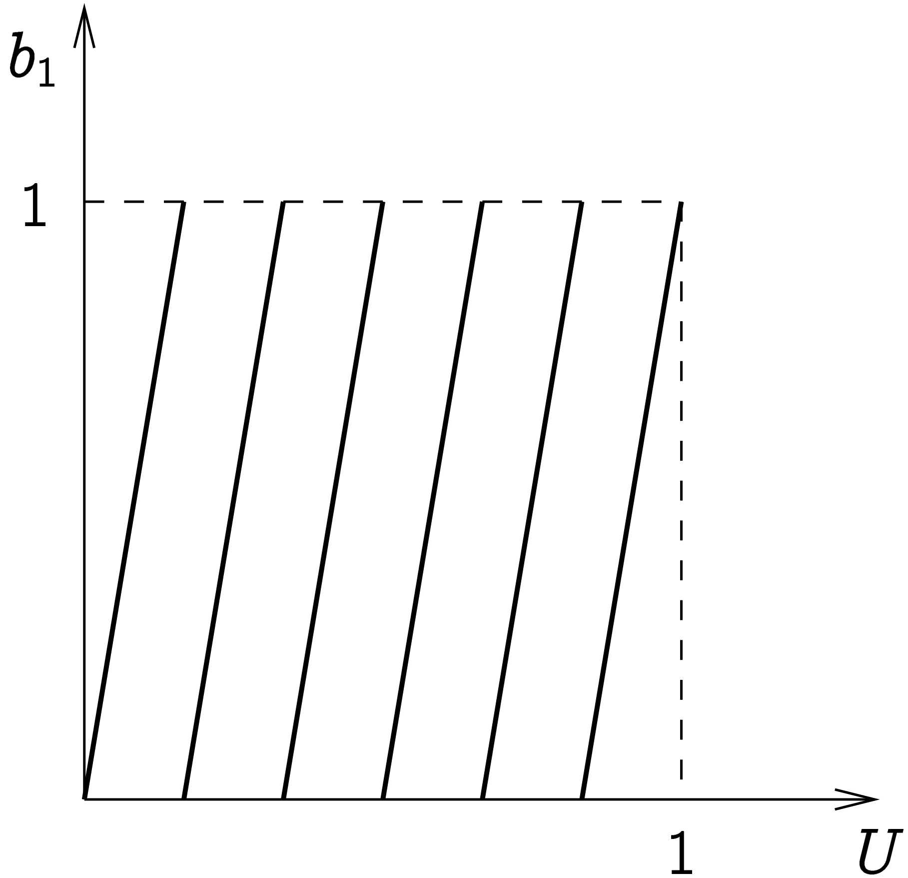
For $x, y \in \mathbb{R}$, $y > 0$ we can define $x \bmod y = x - y \cdot \lfloor x/y \rfloor$.

We also have $(cx) \bmod (cy) = c(x \bmod y)$ for all $c > 0$.

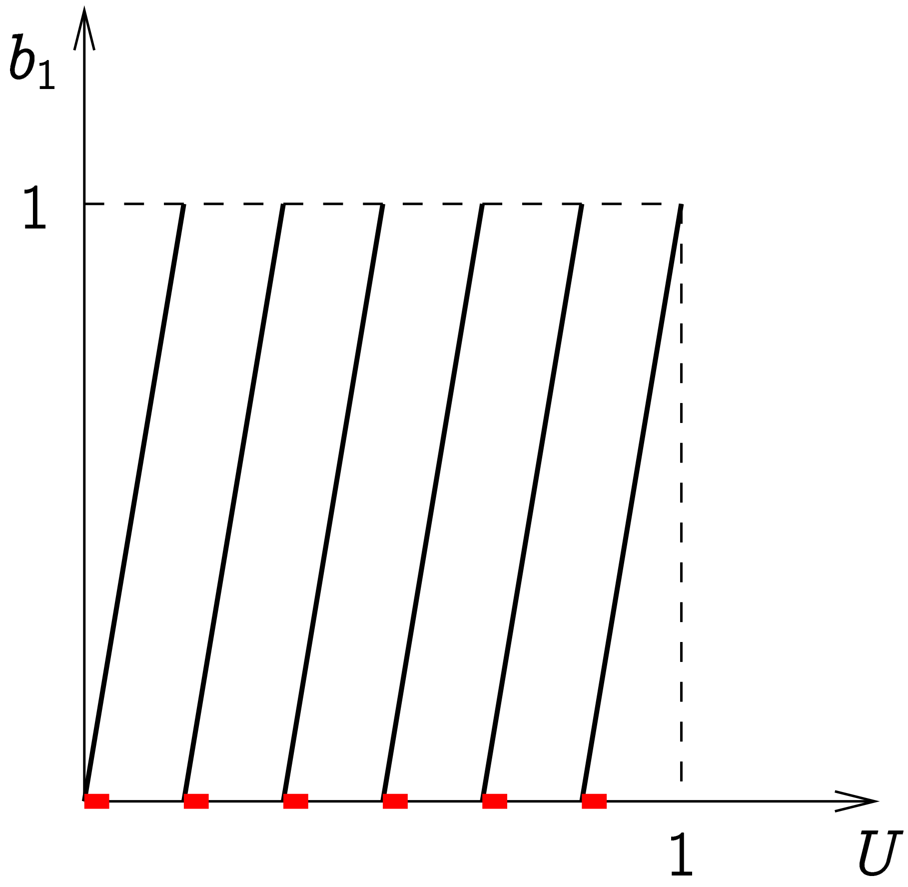
If $(b_1, \ldots, b_n), U, M$ suits us, then $(cb_1, \ldots, cb_n), cU, cM$ suits us as well.

We take $M = 1$. Now our task is to find a suitable $(b_1, \ldots, b_n), U$.

Consider the graph of $b_1 = a_1 \cdot U \bmod 1$. It maps the value of a_1 to the value of b_1, depending on the (unknown) U.



b_1 is the smallest of the knapsack elements (very small compared to $1 = M > \sum_{i=1}^{n} b_i$). Hence U must belong to the marked region.



Also, $b_i = a_i \cdot U \bmod 1$ must be very small if i is small.

The correct U is close to the discontinuation points of both $a_1 \cdot U \bmod 1$ and $a_i \cdot U \bmod 1$.

The discontinuation points of $a_1 \cdot U \bmod 1$ are p/a_1, where $1 \leqslant p \leqslant a_1 - 1$.

The discontinuation points of $a_i \cdot U \bmod 1$ are q/a_i, where $1 \leqslant q \leqslant a_i - 1$.

We are looking for discontinuation points that are close to each other.

$$-\varepsilon < \frac{p}{a_1} - \frac{q}{a_i} < \varepsilon \qquad 1 \leqslant p \leqslant a_1 - 1 \qquad 1 \leqslant q \leqslant a_i - 1$$

$$-\delta < pa_i - qa_1 < \delta \qquad 1 \leqslant p \leqslant a_1 - 1 \qquad 1 \leqslant q \leqslant a_i - 1$$

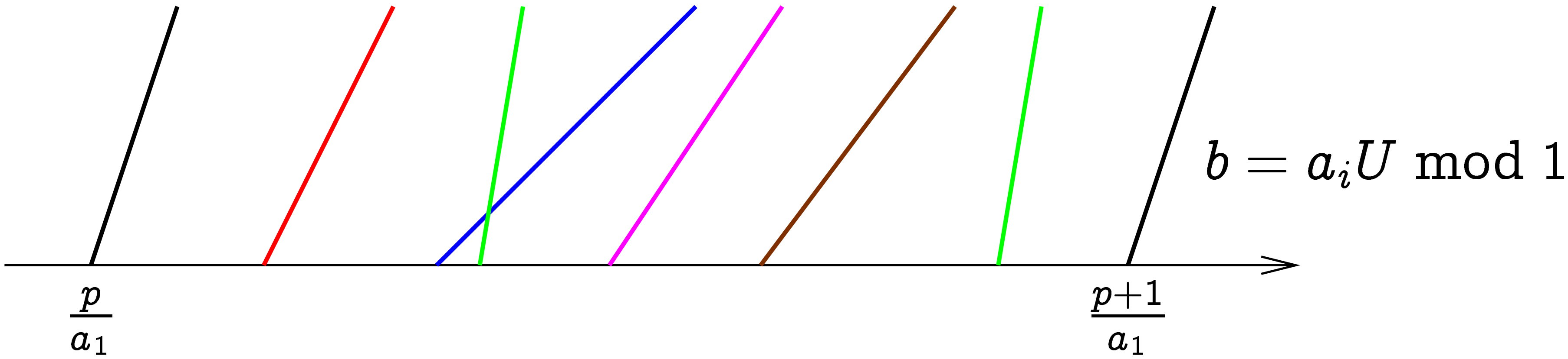
This system of equations gives us candidate p-s. We'll test their suitability.

[Adi Shamir, A poly.-time algo. for breaking the basic MH cryptosystem, Proc. of 32nd Symp. on Foundations of CS, 1982] suggests that $i \in \{2, 3, 4\}$ and $\delta \approx \sqrt{a_1/2}$.

δ may be adjusting depending on the number of candidate p-s.

The system above is solvable in polynomial time (if we treat i as a constant).

Let p be fixed. Consider the interval $[p/a_1, (p+1)/a_1)$.



The discontinuation points of $b_i = a_i U \bmod 1$ partition it to sub-intervals $[x_j, x_{j+1})$ for $j \in \{0, \ldots, m\}$ for some m. Here $x_0 = p/a_1$ and $x_m = (p+1)/a_1$.

In each interval $[x_j, x_{j+1})$ the graph of $b_i = a_i U \bmod 1$ is just a straight line $b_i = a_i U - c_i^j$.

The values x_j and c_i^j are straightforward to find.

The expected number of intervals is $O(n)$.

Consider an interval $[x_j, x_{j+1})$. We are looking for some U in that interval that would make $(b_1, \dots, b_n)$ superincreasing. We have the linear inequalities

$$x_j < U < x_{j+1}$$

$$\sum_{i=1}^{n} (a_i U - c_i^j) < 1$$

$$\forall k \in \{2, \dots, n\} : \sum_{i=1}^{k-1} a_i U - c_i^j < a_k U - c_k^j$$

If these inequalities have a common solution then it is the suitable U.

Example: public key is $(141, 68, 136, 199, 106, 66, 54)$.

We have the following inequalities for p, q_2, q_3, q_4:

$$1 \leqslant p \leqslant 140 \quad 1 \leqslant q_2 \leqslant 67 \quad 1 \leqslant q_3 \leqslant 135 \quad 1 \leqslant q_4 \leqslant 198$$

$$-\delta < 68p - 141q_2 < \delta \quad -\delta < 136p - 141q_3 < \delta$$

$$-\delta < 199p - 141q_4 < \delta$$

Shamir suggests $\delta \approx 8$.

- $-8 < 68p - 141q_2 < 8$ gives

  $p \in \{2, 27, 29, 31, 54, 56, 58, 83, 85, 87, 110, 112, 114, 139\}$

- $-8 < 136p - 141q_3 < 8$ gives

  $p \in \{1, 27, 28, 29, 55, 56, 57, 84, 85, 86, 112, 113, 114, 140\}$

- $-8 < 199p - 141q_4 < 8$ gives

  $p \in \{17, 22, 34, 39, 51, 56, 68, 73, 85, 90, 102, 107, 119, 124\}$

Intersection:

$$p \in \{56, 85\}$$

See [H.W. Lenstra. Integer Programming with a Fixed Number of Variables. Mathematics of Operations Research **8**(4):538–548, 1983] for how these system can actually be solved.

Consider the interval $I = [\frac{56}{141}, \frac{57}{141})$. If $U \in I$ then

- $a_2 U \mod 1$ has no discontinuation points.
- $a_3 U \mod 1$ has no discontinuation points.
- $a_4 U \in \mathbb{Z}$ if $U = 80/199$.
- $a_5 U \mod 1$ has no discontinuation points.
- $a_6 U \mod 1$ has no discontinuation points.
- $a_7 U \mod 1$ has no discontinuation points.

Hence $x_0 = \frac{56}{141}$, $x_1 = \frac{80}{199}$, $x_2 = \frac{57}{141}$.

In $(\frac{56}{141}, \frac{80}{199})$ we have

$$b_1 = 141U - 56 \quad b_2 = 68U - 27 \quad b_3 = 136U - 54$$
$$b_4 = 199U - 79 \quad b_5 = 106U - 42 \quad b_6 = 66U - 26$$
$$b_7 = 54U - 21$$

The inequality $\sum_{i=1}^{n} b_i < 1$ gives $770U - 305 < 1$ or $U < \frac{153}{385}$. The allowed interval for U reduces to $(\frac{56}{141}, \frac{153}{385})$.

Consider the inequalities stating the superincreasing condition.

Interval: $(\frac{56}{141}, \frac{153}{385})$.

Condition: $b_1 < b_2$.

$$141U - 56 < 68U - 27$$

$$U < \frac{29}{73}$$

Ordering: $\frac{56}{141} < \frac{29}{73} < \frac{153}{385}$.

New interval: $(\frac{56}{141}, \frac{29}{73})$.

Interval: $(\frac{56}{141}, \frac{29}{73})$.

Condition: $b_1 + b_2 < b_3$.

$$141U - 56 + 68U - 27 < 136U - 54$$

$$U < \frac{29}{73}$$

New interval: $(\frac{56}{141}, \frac{29}{73})$.

Interval: $(\frac{56}{141}, \frac{29}{73})$.

Condition: $b_1 + b_2 + b_3 < b_4$.

$$141U - 56 + 68U - 27 + 136U - 54 < 199U - 79$$

$$U < \frac{29}{73}$$

New interval: $(\frac{56}{141}, \frac{29}{73})$.

Interval: $(\frac{56}{141}, \frac{29}{73})$.

Condition: $b_1 + b_2 + b_3 + b_4 < b_5$.

$$141U - 56 + 68U - 27 + 136U - 54 + 199U - 79 < 106U - 42$$

$$U < \frac{29}{73}$$

New interval: $(\frac{56}{141}, \frac{29}{73})$.

Interval: $(\frac{56}{141}, \frac{29}{73})$.

Condition: $b_1 + b_2 + b_3 + b_4 + b_5 < b_6$.

$$141U - 56 + 68U - 27 + 136U - 54 + 199U - 79 + 106U - 42 < 66U - 26$$

$$U < \frac{29}{73}$$

New interval: $(\frac{56}{141}, \frac{29}{73})$.

Interval: $(\frac{56}{141}, \frac{29}{73})$.

Condition: $b_1 + b_2 + b_3 + b_4 + b_5 + b_6 < b_7$.

$$141U - 56 + 68U - 27 + 136U - 54 + 199U - 79 + \\ 106U - 42 + 66U - 26 < 54U - 21$$

$$U < \frac{263}{662}$$

Ordering: $\frac{29}{73} < \frac{263}{662}$.

New interval: $(\frac{56}{141}, \frac{29}{73})$.

Any element of this interval is a suitable U.

For example, pick $U = \frac{85}{214}$.

I.e. pick $U = 85$ and $M = 214$.

Computing $b_i = a_i U \bmod M$ gives us the secret knapsack

$$(1, 2, 4, 9, 22, 46, 96) \quad .$$

In the construction of this example I used $U = 114$ and $M = 287$. Their ratio also lies in this interval. They give the knapsack

$$(2, 3, 6, 13, 30, 62, 129) \quad .$$

A variation of the MH knapsack system permutes the elements of the public knapsack $(a_1, \ldots, a_n)$. The permutation is part of the secret key.

We can no longer choose the components of a corresponding to $b_1, \ldots, b_4$, but we can guess them.

- We don't really need four *smallest* b_i-s. Four *small* b_i-s suffices.

When verifying the superincreasing condition, we do not know the ordering of elements $b_1, \ldots, b_n$.

To overcome this, when we partition $[p/a_1, (p+1)/a_1)$ to smaller intervals, we also consider the intersection points of the graphs of some $a_i U \bmod 1$ and $a_j U \bmod 1$.

In all such intervals the ordering of $b_1, \ldots, b_n$ is fixed.

The density of a knapsack $(a_1, \ldots, a_n)$ is

$$R = \frac{n}{\lceil \log \max_i a_i \rceil}$$

The densities of the knapsacks that we have seen:

- $(141, 68, 136, 199, 106, 66, 54)$: $\frac{7}{8}$;
- $(1, 2, 4, 9, 22, 46, 96)$: 1;
- $(2, 3, 6, 13, 30, 62, 129)$: $\frac{7}{8}$.

The knapsacks with densities > 1 usually have multiple decodings of messages.

The public key usually has density less than 1.

When using the parameters suggested by Merkle and Hellman, the public key has the density ≈ 0.5.

Almost all instances of SUBSET-SUM, where the density of the knapsack is less than $0.9408\ldots$, are easily solvable.

Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a basis of the vector space $\mathbb{R}^n$.

The integer lattice determined by this basis is the set of vectors

$$\{m_1\mathbf{b}_1 + \ldots + m_n\mathbf{b}_n \,|\, m_1, \ldots, m_n \in \mathbb{Z}\} \ .$$

Shortest vector problem (SVP): given the basis, determine the shortest non-zero vector (according to the Euclidean norm) of the lattice thus defined.

There exist polynomial-time algorithms for approximating the solution to the SVP.

The LLL-algorithm finds a vector in the lattice that is no more than $2^{(n-1)/2}$ times longer than the shortest vector.

In practice, it often works even better.

The SVP in lattices may be easy on average.

Given a SUBSET-SUM instance $(a_1, \ldots, a_n), s$, consider the integer lattice with the basis

$$\begin{aligned}
\mathbf{b}_1 &= (1, 0, \ldots, 0, Na_1) \\
\mathbf{b}_2 &= (0, 1, \ldots, 0, Na_2) \\
&\ldots\ldots\ldots\ldots \\
\mathbf{b}_n &= (0, 0, \ldots, 1, Na_n) \\
\mathbf{b}_{n+1} &= (\tfrac{1}{2}, \tfrac{1}{2}, \ldots, \tfrac{1}{2}, Ns)
\end{aligned}$$

where $N \in \mathbb{Z}$, $N > \frac{1}{2}\sqrt{n}$.

Let $x_1, \ldots, x_n$ be the solution to the given instance. Then $\left(\sum_{i=1}^{n} x_i \mathbf{b}_i\right) - \mathbf{b}_{n+1} = (x_1 - \frac{1}{2}, \ldots, x_n - \frac{1}{2}, 0)$ is a short vector in that lattice. With high probability, it is a solution to the SVP.

Algorithm for solving SUBSET-SUM instances $(a_1, \ldots, a_n), s$:

1. Construct the basis $\mathbf{b}_1, \ldots, \mathbf{b}_{n+1}$;
2. Solve the SVP for the lattice determined by this basis. Let $\mathbf{e} = (e_1, \ldots, e_{n+1})$ be the result.
3. Check that $e_{n+1} = 0$ and $e_1, \ldots, e_n \in \{\frac{1}{2}, -\frac{1}{2}\}$. If not, then fail.
4. Let $x_i = e_i + \frac{1}{2}$. If $\sum_{i=1}^{n} x_i a_i = s$ then return $(x_1, \ldots, x_n)$.
5. Let $x_i = \frac{1}{2} - e_i$. If $\sum_{i=1}^{n} x_i a_i = s$ then return $(x_1, \ldots, x_n)$.
6. Fail.

When creating the key for the knapsack cryptosystem, we transform a knapsack $(b_1, \ldots, b_n)$ to another one $(a_1, \ldots, a_n)$.

We could iterate this transformation multiple times.

Each time, we must save $U = W^{-1}$ and M in the secret key.

This gives rise to the multiply-iterated knapsack cryptosystem.

In general, multiple iteration makes the elements of the knapsack larger and thus reduces density.