A Polynomial-Time Algorithm for Solving the Hidden Subset Sum Problem

Jean-Sébastien Coron and **Agnese Gini**

University of Luxembourg

CRYPTO2020

Hidden Subset Sum Problem

~~Hidden~~ Subset Sum Problem

$$h = \alpha_1 x_1 + \cdots + \alpha_n x_n \pmod{Q}$$

with $x_1, \ldots, x_n \in \{0, 1\}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}/Q\mathbb{Z}^n$.

Given Q, h and $\alpha_1, \ldots, \alpha_n$, recover $x_1, \ldots, x_n$.

Hidden Subset Sum Problem

$$
\begin{aligned}
h_1 &= \alpha_1 x_{1,1} + \cdots + \alpha_n x_{n,1} && \pmod{Q} \\
&\vdots \\
h_m &= \alpha_1 x_{1,m} + \cdots + \alpha_n x_{n,m} && \pmod{Q}
\end{aligned}
$$

with $x_{i,j} \in \{0,1\}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}/Q\mathbb{Z}^n$.

Given Q and $h_1, \ldots, h_m$, recover $\alpha_1, \ldots, \alpha_n$ and $x_{i,j}$ for $i \in [n]$ and $j \in [m]$.

The weights α_i's are hidden!!

Hidden Subset Sum Problem

$$
\begin{aligned}
h_1 &= \alpha_1 x_{1,1} + \cdots + \alpha_n x_{n,1} && \pmod{Q} \\
&\vdots \\
h_m &= \alpha_1 x_{1,m} + \cdots + \alpha_n x_{n,m} && \pmod{Q}
\end{aligned}
$$

with $x_{i,j} \in \{0,1\}$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}/Q\mathbb{Z}^n$.

Given Q and $h_1, \ldots, h_m$, recover $\alpha_1, \ldots, \alpha_n$ and $x_{i,j}$ for $i \in [n]$ and $j \in [m]$.

$$
\begin{bmatrix} h_1 & \cdots & \cdots & h_m \end{bmatrix} = \begin{bmatrix} \alpha_1 & \cdots & \alpha_n \end{bmatrix} \begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,m} \\ & \vdots & \vdots & \\ x_{n,1} & \cdots & \cdots & x_{n,m} \end{bmatrix} \pmod{Q}
$$

Hidden Subset Sum Problem

Let Q be an integer, and let $\alpha_1, \ldots, \alpha_n$ be random integers in $\mathbb{Z}/Q\mathbb{Z}$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{Z}^m$ be random vectors with components in $\{0, 1\}$. Let $\mathbf{h} \in \mathbb{Z}^m$ satisfying:

$$\mathbf{h} = \alpha_1 \mathbf{x}_1 + \cdots + \alpha_n \mathbf{x}_n \pmod{Q}$$

Given Q and $\mathbf{h}$, recover the integers α_i's and the vectors $\mathbf{x}_i$'s.

$$\mathrm{h} = \boldsymbol{\alpha} \; \mathbf{X} \pmod{Q}$$

Timeline

1998 Boyko, Peinado and Venkatesan presented a fast generator of random pairs $(x, g^x \pmod{p})$ introducing the HSSP.

1999 Nguyen and Stern described a lattice based algorithm for solving the HSSP.

2020 Our main contributions:

- detailed analysis of the Nguyen-Stern algorithm,
- variant working in polynomial-time.

Overview

- The Nguyen-Stern attack.
- Our polynomial-time attack.
- The affine hidden subset sum.
- Final remarks and open questions.

The Nguyen-Stern Attack

$$\boxed{\mathbf{h} = \alpha_1 \mathbf{x}_1 + \cdots + \alpha_n \mathbf{x}_n \pmod{Q}}$$

The idea:

- If a vector $\mathbf{u}$ is orthogonal to $\mathbf{h}$ modulo Q:

$$\langle \mathbf{u}, \mathbf{h} \rangle \equiv \alpha_1 \langle \mathbf{u}, \mathbf{x}_1 \rangle + \cdots + \alpha_n \langle \mathbf{u}, \mathbf{x}_n \rangle \equiv 0 \pmod{Q}$$

  $\Rightarrow \mathbf{p_u} = (\langle \mathbf{u}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{u}, \mathbf{x}_n \rangle)$ is orthogonal to $\boldsymbol{\alpha}$ modulo Q.

- If $\|\mathbf{p_u}\| < \lambda_1(\Lambda_Q^{\perp}(\boldsymbol{\alpha}))$, we must have $\mathbf{p_u} = 0$, and therefore the vector $\mathbf{u}$ is orthogonal in $\mathbb{Z}$ to all vectors $\mathbf{x}_i$.

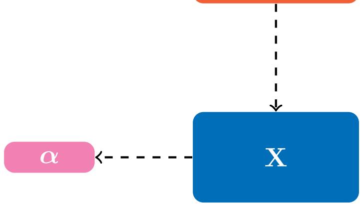
$$\mathbf{h} = \alpha \, \mathbf{X} \pmod{Q}$$

The algorithm:

Step 1

$$\mathbf{h} = \gamma \, \mathbf{C} \pmod{Q}$$

Step 2



$$\mathbf{h} = \boldsymbol{\alpha} \, \mathbf{X} \pmod{Q}$$

The algorithm:

Step 1 From the samples $\mathbf{h}$ and Q, determine the lattice $\bar{\mathcal{L}}_{\mathbf{x}} = (\mathcal{L}_{\mathbf{x}}^{\perp})^{\perp}$, where $\mathcal{L}_{\mathbf{x}}$ is the lattice generated by the $\mathbf{x}_i$'s.

$$\mathbf{h} \dashrightarrow \mathbf{C}$$

Step 2 From $\bar{\mathcal{L}}_{\mathbf{x}} \supseteq \mathcal{L}_{\mathbf{x}}$, recover the hidden vectors $\mathbf{x}_i$'s. From $\mathbf{h}$, the $\mathbf{x}_i$'s and Q, recover the weights α_i's.

$$\mathbf{C} \dashrightarrow \mathbf{X} \dashrightarrow \boldsymbol{\alpha}$$

Our analysis:

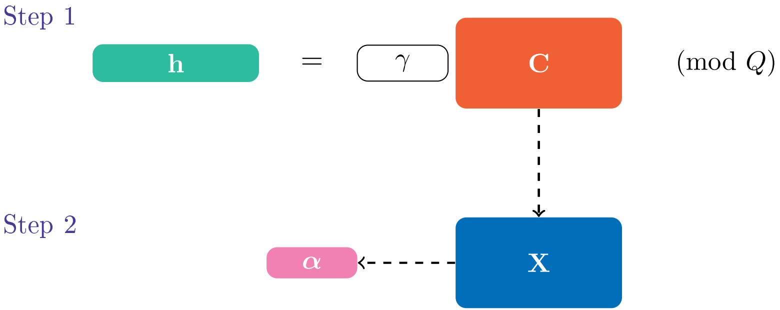
Step 1:

- With probability at least $1/2$ over the choice of $\boldsymbol{\alpha}$, the algorithm recovers a basis of $\bar{\mathcal{L}}_{\mathbf{x}}$ in polynomial time, assuming that Q is a prime integer of bitsize at least $2mn \log m$.
- For $m = 2n$, if the density is $d = n/\log Q = \mathcal{O}(1/(n \log n))$ we recover $\bar{\mathcal{L}}_{\mathbf{x}}$ in polynomial time.
- Heuristically, $d = \mathcal{O}(1/n)$ is sufficient.

Step 2:

- The $\mathbf{x}_i$'s are short vectors of $\bar{\mathcal{L}}_{\mathbf{x}}$.
- Using BKZ the asymptotic complexity is $2^{\Omega(n/\log n)}$.

Our polynomial-time algorithm



- We require $m \approx (n^2+n)/2$ instead of $m = 2n$.
- Improved step 1: fast generation of orthogonal vectors.
- New step 2: recover binary vectors.

New step 2: multivariate approach

Ingredients:

- $\mathcal{L}_{\mathbf{x}}$ is a sublattice of $\bar{\mathcal{L}}_{\mathbf{x}}$: there exists $\mathbf{W} \in \mathbb{Z}^{n \times n} \cap \mathrm{GL}(\mathbb{Q}, n)$

$$\mathbf{X} = \mathbf{W} \; \mathbf{C}$$

- Being binary is an algebraic condition:

$$y \in \{0, 1\} \iff y^2 - y = 0$$

Mixing...

$$\mathbf{x}_i = \mathbf{w}_i \begin{bmatrix} \tilde{\mathbf{c}}_1 & \cdots & \tilde{\mathbf{c}}_m \end{bmatrix}$$

For each $i = 1, \ldots, n$ and $j = 1, \ldots, m$ we have

- $x_{i,j} \in \{0, 1\}$
- $\mathbf{w}_i \cdot \tilde{\mathbf{c}}_j = x_{i,j}$

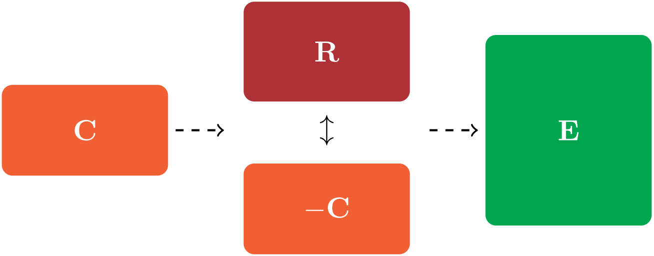
$$\implies \quad (\mathbf{w}_i \cdot \tilde{\mathbf{c}}_j)^2 - \mathbf{w}_i \cdot \tilde{\mathbf{c}}_j = 0$$

The rows of $\mathbf{W}$ are solutions of multivariate quadratic polynomial system

$$\begin{cases} \mathbf{w} \cdot \tilde{\mathbf{c}}_1 \tilde{\mathbf{c}}_1^\intercal \cdot \mathbf{w}^\intercal - \mathbf{w} \cdot \tilde{\mathbf{c}}_1 = 0 \\ \qquad\qquad \vdots \\ \mathbf{w} \cdot \tilde{\mathbf{c}}_m \tilde{\mathbf{c}}_m^\intercal \cdot \mathbf{w}^\intercal - \mathbf{w} \cdot \tilde{\mathbf{c}}_m = 0 \end{cases}$$

- For $m \approx (n^2 + n)/2$ we expect to solve this system and recover the $\mathbf{x}_i$'s by $\mathcal{O}(n^6)$ bit operations and $\mathcal{O}(n^4)$ space complexity, via linear algebra.

The coefficient of $w_i w_k$ in the jth equation is $(2 - \delta_{i,k})\mathbf{C}_{ij}\mathbf{C}_{kj}$.



- $\mathbf{E}$ is the matrix of the coefficients of the system.
- The rows of $\mathbf{W}$ are eigenvectors of certain submatrices of a basis of $\ker \mathbf{E}$.

Lemma

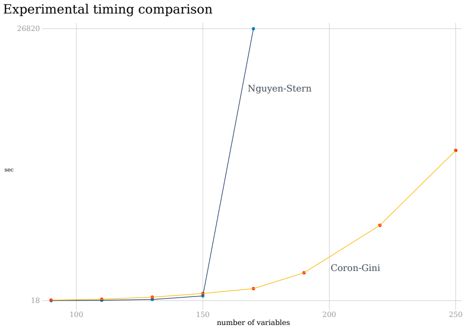
If $\mathbf{R}$ has rank $\frac{n^2+n}{2}$, then the vectors $\mathbf{x}_i$'s can be recovered in $\mathcal{O}(n^6)$ arithmetic operations.

Reducing the matrix relation $\mathbf{X} = \mathbf{WC} \mod p$ we can obtain a system defined over $\mathbb{F}_p$.

$\Rightarrow$ For $m \approx (n^2+n)/2$ we expect to solve this system and recover the $\mathbf{x}_i$'s by $\mathcal{O}(n^6)$ bit operations and $\mathcal{O}(n^4)$ space complexity, via linear algebra.

Comparison

	NS99	CG20
m	$2n$	$(n^2+n)/2$
$\log Q$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
time	$2^{\Omega(n/\log n)}$	$\mathcal{O}(n^9)$



Affine Hidden Subset Sum Problem

$$\mathbf{h} + s\mathbf{e} = \alpha_1\mathbf{x}_1 + \cdots + \alpha_n\mathbf{x}_n \pmod{Q}$$

Given Q, $\mathbf{h}$ and $\mathbf{e}$, recover the α_i's and s and the vectors $\mathbf{x}_i$'s.

Step 1 From $(\mathbf{h}, \mathbf{e})$, determine the lattice $\bar{\mathcal{L}}_{\mathbf{x}}$, where $\mathcal{L}_{\mathbf{x}}$ is the lattice generated by the $\mathbf{x}_i$'s.

Step 2 From $\bar{\mathcal{L}}_{\mathbf{x}} \supseteq \mathcal{L}_{\mathbf{x}}$, recover the hidden vectors $\mathbf{x}_i$'s. From $\mathbf{h}$, the $\mathbf{x}_i$'s and Q, recover the weights α_i's and s.

Conclusions

- We argue that the asymptotic complexity of the full Nguyen-Stern algorithm is $2^{\Omega(n/\log n)}$.
- We propose a new second step to recover short (binary) vectors using $m \simeq n^2/2$ samples, via a multivariate technique.
- Asymptotically the heuristic complexity of our full algorithm is $\mathcal{O}(n^9)$.

Can we further reduce m? and $\log Q$?

In addition, we show how to slightly reduce the number of samples m required for our attack, with two different methods; in both cases the attack remains heuristically polynomial time under the condition $m = n^2/2 - \mathcal{O}(n \log n)$.

Thank you for your attention!

Full paper at
https://ia.cr/2020/461

Code at
https://pastebin.com/ZFk1qjfP