Information Security Lab
Autumn Semester 2023
Module 4, Week 3 – Coppersmith's Method

Kenny Paterson (@kennyog)

Applied Cryptography Group

https://appliedcrypto.ethz.ch/

Overview of today's lecture

- Motivating example
- From polynomials to lattices and back again
- Simple Coppersmith
- Full Coppersmith
- Applications

NB: this lecture is heavily based on Chapter 19 of Steven Galbraith's book, see: https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html

Motivating Example

Motivating Example

- Let ($N = pq, e$) be an RSA public key and suppose an attacker has learned the MSBs of p due to some vulnerability.

- Can the attacker recover the rest of *p?*
  - And hence recover the factorisation of N.
  - Then recover the RSA secret key d by solving $de = 1 \bmod (p-1)(q-1)$.

- This scenario is not artificial!

Motivating Example

- Example: recent analysis of MEGA cloud storage system:
  - In MEGA, a user's RSA private key is encrypted using AES in ECB mode under a per-user master key.
  - It is possible to recover individual 128-plaintext blocks of the RSA private key, but recovery of each block requires a user to engage in many logins.
  - So: target the ECB blocks corresponding to the MSBs of p and set up an instance of our "known MSBs" problem.
  - Full details at https://mega-caveat.github.io/ and https://eprint.iacr.org/2023/329.
  - (Attack actually targets the LSBs for technical reasons.)

Motivating Example

- With MSBs of p known, we can write:

$$p = p^* + x_o$$

where p^* is known and x_o is unknown with $|x_o| < X$ for some bound X that depends on the number of known MSBs.

- e.g. if p has 1024 bits and 768 MSBs of p are known, then we can set $X = 2^{256}$.

- In this situation, if we can somehow recover x_o, then we can recover all of p.

Motivating Example

- Essentially we have to solve a polynomial equation in variable x, of the form:

$$F(x) = 0 \bmod p.$$

  under the guarantee that the solution x_o is relatively small ($|x_o| < X$).

- In our example, $F(x) = p^* + x$ is a particularly simple polynomial.

- But here we don't know the modulus p!

- In other examples F may have higher degree, we may know the modulus but not its factors, or we may have a system in more than one variable.

- We will return to this specific example later.

General Setting of Coppersmith's Method

- Given a polynomial equation in variable x, of the form:

$$F(x) = 0 \bmod M.$$

find all solutions x_o with $|x_o| < X$ for some given bound X.

Two remarks:

- Solving low-degree polynomial equations $F(x) = 0 \bmod p$ for **known prime *p*** can be done efficiently with e.g. Berlekamp or Cantor-Zassenhaus algorithms.

- Solving quadratic polynomial equations $F(x) = 0 \bmod N$ (without limitation on the size of solutions) is equivalent to factoring N.

General Setting of Coppersmith's Method

- Given a polynomial equation in variable x, of the form:

$$F(x) = 0 \bmod M.$$

find all solutions x_0 with $|x_0| < X$ for some given bound X.

<u>Two core ideas behind Coppersmith's method:</u>

1. We can make a transformation of our equation $F(x) = 0 \bmod M$ to produce another equation $F'(x) = 0 \bmod M$ that preserves all of the solutions x_0 but which guarantees that $F'(x)$ will be small if x is.

2. Solutions of $F'(x_0) = 0 \bmod M$ must also be solutions over the integers *if they are sufficiently small*. And we can find solutions of $F'(x) = 0$ over the integers just using numerical methods, e.g. Newton's method.

General Setting of Coppersmith's Method

Example:

- Consider $M = 17.19 = 323$ (a small RSA modulus!)
- Suppose $F(x) = x^2 + 33x + 215$ and we want to find small solutions to $F(x) = 0 \bmod M$.
- Notice that $x_0 = 3$ is a solution to $F(x) = 0 \bmod M$:

$$3^2 + 33.3 + 215 = 9 + 99 + 215 = 323 = 0 \bmod 323.$$

- Yet $x_0 = 3$ is not a solution to "$F(x) = 0$" over the integers!
- We can solve this **small** problem by working mod 17 and mod 19, and combining the results. Does not work if M is a **large** RSA modulus…

General Setting of Coppersmith's Method

Example (ctd):

- $M = 323$, $F(x) = x^2 + 33x + 215$.
- We seek small solutions x_0 to $F(x) = 0 \bmod M$.
- Consider the polynomials $F(x)$, Mx, and "M" (a constant polynomial).
- Suppose x_0 is a solution to $F(x) = 0 \bmod M$.
- Any x_0 is trivially also a solution to $Mx = 0 \bmod M$ and to $M = 0 \bmod M$.
- Hence x_0 is a solution to any *linear combination* of these three polynomials:

$$F(x),\ Mx,\ M.$$

General Setting of Coppersmith's Method

Example (ctd):

- $M = 323$, $F(x) = x^2 + 33x + 215$.
- We seek small solutions x_0 to $F(x) = 0 \bmod M$.
- If x_0 is a solution to $F(x) = 0 \bmod M$ then it is also a solution to:

$$G(x) = 0 \bmod M$$

for

$$G(x) = 9F(x) - Mx - 6M = 9x^2 - 26x + 3.$$

- This G has quite small coefficients, so maybe its values will be small, and we can look for solutions of $G(x) = 0$ over the integers.
- In fact, $G(3) = 9.3^2 - 26.3 + 3 = 81 - 78 + 3 = 0$.

General Setting of Coppersmith's Method

Example (ctd):

- $M = 323$, $F(x) = x^2 + 33x + 215$.
- We seek small solutions x_0 to $F(x) = 0 \bmod M$.
- We chose $G(x) = 9F(x) - Mx - 6M = 9x^2 - 26x + 3$ and found a solution $x_o = 3$ over the integers.

- Where did this choice of G come from?
- How do we guarantee that G will behave well, e.g. have small coefficients or small values when x is small?

- We will translate to a lattice problem where "small polynomials" map to "short vectors" and vice-versa.
- We can use an SVP solver to find a short vector in the lattice.
- We can then map this vector back to a ``small polynomial'' G.
- We find the roots of G over the integers.

From polynomials to vectors and back again

From polynomials to vectors and back again

Notation:

M is the modulus for the equation we wish to solve.

$F(x) := f_o + f_1 x + \ldots + f_d x^d$ is the univariate polynomial of degree d whose solutions we would like to find.

X is a bound on the size of solutions we seek (X is **not** a variable.)

Polynomial to vector translation:

$$G(x) := g_o + g_1 x + \ldots + g_d x^d \rightarrow v_G = (g_o, g_1 X, \ldots, g_d X^d)$$

This translation turns any polynomial G of degree d with integer coefficients into a vector v_G in $\mathbb{R}^{d+1}$.

In particular we can apply it to F to get a vector v_F.

From polynomials to vectors and back again

Polynomial to vector translation:

$$G(x) := g_0 + g_1 x + \ldots + g_d x^d \;\rightarrow\; v_G = (g_0, g_1 X, \ldots, g_d X^d)$$

Note that:

1. Any vector which can be written in the form $(g_0, g_1 X, \ldots, g_d X^d)$ *in which the* g_i *are integers* can be converted back into a polynomial over the integers.
2. Any integer linear combination of vectors of this form will also have this form.
3. Suppose polynomials $F_1, F_2, \ldots$ all have a common solution x_0. Then any integer linear combination G of $F_1, F_2, \ldots$ also has x_0 as a solution.

Moral: we can convert freely backwards and forwards between polynomials and vectors, and solutions x_0 will be nicely preserved under linear combinations.

From polynomials to vectors and back again

Polynomial to vector translation:

$$G(x) := g_0 + g_1 x + \ldots + g_d x^d \;\rightarrow\; v_G = (g_0, g_1 X, \ldots, g_d X^d)$$

Theorem (Howgrave-Graham, 1997).

With notation as above, suppose x_0 is a solution to $F(x) = 0 \bmod M$ with $|x_0| < X$. Suppose $\| v_F \| < M/(d+1)^{1/2}$. Then $F(x_0) = 0$ (over the integers).

Interpretation:
If we can find a polynomial F for which the corresponding vector v_F has small L2 norm, then we can find all small solutions of $F(x) = 0 \bmod M$ by solving $F(x) = 0$ over the integers!

From polynomials to vectors and back again

Theorem (Howgrave-Graham, 1997).

With notation as above, suppose x_o is a solution to $F(x) = 0 \bmod M$ with $|x_o| < X$. Suppose $\| v_F \| < M/(d+1)^{1/2}$. Then $F(x_o) = 0$ (over the integers).

Proof:

From the Cauchy-Schwarz inequality, it is easy to show that, for any n-dimensional vector $x = (x_1, x_2, \ldots, x_n)$ over the real numbers,

$$\sum_{i=1}^{n} x_i \le \left(n \sum_{i=1}^{n} x_i^2\right)^{1/2} = \sqrt{n} \cdot \| x \|$$

(To see this, recall that Cauchy-Schwarz says $|x \cdot y| \le \|x\| \cdot \|y\|$; hence $(\Sigma_i x_i y_i)^2 \le (\Sigma_i x_i^2) \cdot (\Sigma_i y_i^2)$; now take $y = (1,1,\ldots,1)$ and simplify to get the result.)

From polynomials to vectors and back again

Theorem (Howgrave-Graham, 1997).

With notation as above, suppose x_o is a solution to $F(x) = 0 \bmod M$ with $|x_o| < X$. Suppose $\| v_F \| < M/(d+1)^{1/2}$. Then $F(x_o) = 0$ (over the integers).

Proof (ctd):

Now:

$$
\begin{aligned}
|F(x_o)| &= |\textstyle\sum_i f_i x_o^i| \\
&\le \textstyle\sum_i |f_i| |x_o^i| \\
&\le \textstyle\sum_i |f_i| X^i \qquad \text{(using the fact that } |x_o| < X\text{)} \\
&\le \sqrt{(d+1)} \cdot \| v_F \| \quad \text{(by applying the previous result to the vector } v_F\text{)} \\
&< \sqrt{(d+1)} \cdot M / \sqrt{(d+1)} \quad \text{(by assumption on } \|v_F\|\text{)} \\
&= M.
\end{aligned}
$$

Hence $-M < F(x_o) < M$, and we see that $F(x_o) = 0 \bmod M$ implies $F(x_o) = 0$.

Simple Coppersmith

Simple Coppersmith

Notation:

M is the modulus for the equation we wish to solve.

$F(x) := f_0 + f_1 x + \ldots + f_d x^d$ is the univariate polynomial of degree d whose solutions mod M we would like to find.

Let x_0 be such a solution with $|x_0| < X$.

Let's assume F is monic (i.e. $a_d = 1$) – we can convert to this case by multiplying F by $(f_d)^{-1}$ mod M if necessary. (And if the inverse does not exist, we can factor M!)

Consider the set of $d+1$ polynomials $G_i(x) = Mx^i$ for $0 \le i < d$ and $F(x)$.

They all have the solution $x = x_0$ mod M.

Hence any linear combination of them also has the solution $x = x_0$ mod M.

Simple Coppersmith

Consider the d+1 polynomials $G_i(x) = Mx^i$ for o ≤ i < d and $F(x)$.

Convert them all to vectors and consider the lattice L they generate.

Hence L has full-rank basis matrix:

$$B = \begin{vmatrix} M & o & \dots & \dots & o & o \\ o & MX & \dots & \dots & o & o \\ . & & & & . & . \\ . & . & \dots & \dots & . & . \\ . & & & & & \\ o & o & \dots & \dots & MX^{d-1} & o \\ f_o & f_1X & \dots & \dots & f_{d-1}X^{d-1} & X^d \end{vmatrix}$$

Simple Coppersmith

Observations:

- Every **vector** v in lattice L is a linear combination of vectors corresponding to polynomials $G_i(x)$ ($0 \le i < d$) and $F(x)$, and so maps back to a **polynomial** G such that $G(x_0) = 0 \bmod M$.

- We have: $\det(L) = M^d X^{d(d+1)/2}$.

- If we run LLL on B, we are guaranteed to obtain a vector b_1 in the reduced basis for L such that:

$$\| b_1 \| \le 2^{(n-1)/4} \det(L)^{1/n}$$

where $n = d+1$ is the dimension of our lattice.

- This bound evaluates to:

$$\| b_1 \| \le 2^{d/4} M^{d/d+1} X^{d/2}.$$

Simple Coppersmith

Theorem:
With notation as above, let G be the polynomial corresponding to vector b_1 in the LLL-reduced basis for L. Set $X = c(d) \cdot M^{2/d(d+1)}$. Then any solution x_o of $F(x_o) = 0 \bmod M$ such that $|x_o| < X$ satisfies $G(x_o) = 0$ over the integers.

Here $c(d)$ is an explicit constant depending only on d.

Proof
From the previous observations, we know that LLL gives us a vector b_1 in L with:

$$\| b_1 \| \leq 2^{d/4} M^{d/d+1} X^{d/2}.$$

This b_1 maps back to a polynomial G such that $G(x_o) = 0 \bmod M$, and

$$\| v_G \| = \| b_1 \| \leq 2^{d/4} M^{d/d+1} X^{d/2}.$$

Recall that Howgrave-Graham's theorem says that if $\| v_G \| < M/(d+1)^{1/2}$, then $G(x_o) = 0$ (over the integers). Hence it is sufficient that:

$$2^{d/4} M^{d/d+1} X^{d/2} < M/(d+1)^{1/2}.$$

Solving for X gives the result, with $c(d) = 2^{-1/2}(d+1)^{-1/d}$.

Simple Coppersmith

Theorem:
With notation as above, let G be the polynomial corresponding to vector b_1 in the LLL-reduced basis for L. Set $X = c(d) \cdot M^{2/d(d+1)}$. Then any solution x_o of $F(x_o) = 0 \bmod M$ such that $|x_o| < X$ satisfies $G(x_o) = 0$ over the integers.

Here $c(d)$ is an explicit constant depending only on d.

Interpretations
If $d=2$ (we have a quadratic equation defining x_o), it is sufficient that $X \approx M^{1/3}$ to apply the theorem.

If degree $d=3$, we need $X \approx M^{1/6}$. Application: RSA encryption with $e=3$ involves equations of the form $C = \text{encode}(P)^3 \bmod N$. Maybe if P is small enough and the encoding is deterministic, we get a cubic equation to which the method can be applied?

As usual, the actual performance of LLL is often much better than the guaranteed performance, and it may be possible to find solutions beyond these values of X.

Full Coppersmith

Full Coppersmith

- As usual, we have a polynomial $F(x)$ and we want to find solutions x_o satisfying $F(x_o) = \text{mod } M$ such that $|x_o| < X$.

- Simple Coppersmith considered the set of $d+1$ polynomials $G_i(x) = Mx^i$ (for $0 \le i < d$) and $F(x)$.

$$B = \begin{vmatrix} M & o & \dots & \dots & o & o \\ o & MX & \dots & \dots & o & o \\ . & & & & . & . \\ . & . & \dots & \dots & . & . \\ . & & & & & \\ o & o & \dots & \dots & MX^{d-1} & o \\ f_o & f_1X & \dots & \dots & f_{d-1}X^{d-1} & X^d \end{vmatrix}$$

Full Coppersmith

- As usual, we have a polynomial $F(x)$ and we want to find solutions x_o satisfying $F(x_o) = 0 \bmod M$ such that $|x_o| < X$.
- In simple Coppersmith, we considered the set of $d+1$ polynomials $G_i(x) = Mx^i$ for $0 \le i < d$ and $F(x)$.
- Idea: add more polynomials, increasing the lattice dimension without increasing the determinant too much.
- Then, we can hope that the sufficient condition needed for applying Howgrave-Graham's theorem:

$$\| b_1 \| \le 2^{(n-1)/4} \det(L)^{1/n} < M/(d+1)^{1/2}$$

will be satisfied with a larger n and for a larger bound X.

- More polynomials come from shifting F and taking powers of F:

$$F(x_o) = 0 \bmod M \Rightarrow x^j \cdot F(x_o) = 0 \bmod M \text{ for any } j$$
$$F(x_o) = 0 \bmod M \Rightarrow F(x_o)^i = 0 \bmod M^i \text{ for any } i.$$

Full Coppersmith

Theorem (Coppersmith)
Let F be a monic polynomial of degree d.

Suppose $0 < \varepsilon < \min\{0.18, 1/d\}$ and let $X = \frac{1}{2} M^{1/d - \varepsilon}$.
Then any solution x_0 satisfying $F(x_0) = 0 \bmod M$ with $|x_0| < X$ can be found in time polynomial in d, $1/\varepsilon$, and $\log(M)$.

Proof
Construct a lattice L of dimension $d \cdot h$, where $h > 1$ is a parameter of the attack, from the polynomials:

$$G_{i,j} = M^{h-1-j} \cdot F(x)^j \cdot x^i$$

with $0 \le i < d$, $0 \le j < h$.

These all satisfy $G_{i,j}(x_0) = 0 \bmod M^{h-1}$.

Full Coppersmith

Proof (ctd).

The proof constructs a lattice L of dimension $d \cdot h$, where $h>1$ is a parameter of the attack from polynomials:

$$G_{i,j} = M^{h-1-j} \cdot F(x)^j \cdot x^i$$

with $0 \le i < d$, $0 \le j < h$.

These all satisfy $G_{i,j}(x_0) = 0 \bmod M^{h-1}$.

L can be represented by a lower triangular matrix with diagonal entries:

$$M^{h-1-j} \cdot X^{jd+i}, \quad 0 \le i < d, 0 \le j < h$$

Hence $\det(L) = M^{(h-1)hd/2} \cdot X^{(dh-1)dh/2}$ and running LLL gives an LLL-reduced basis with first vector b_1 satisfying:

$$\| b_1 \| \le 2^{(dh-1)/4} \cdot M^{(h-1)/2} \cdot X^{(dh-1)/2}.$$

Full Coppersmith

Proof (ctd).

We have a basis for L with first vector b_1 such that:

$$\| b_1 \| \leq 2^{(dh-1)/4} \cdot M^{(h-1)/2} \cdot X^{(dh-1)/2}.$$

Now we apply Howgrave-Graham's theorem, for which it is sufficient that:

$$2^{(dh-1)/4} \cdot M^{(h-1)/2} \cdot X^{(dh-1)/2} < M^{h-1}/(dh)^{1/2}$$

(Here we have M^{h-1} on the RHS because our modulus became M^{h-1} instead of M, and our polynomial degree is now $dh-1$ instead of d.)

Further analysis shows taking $h = 1/d\varepsilon$ gives the desired result.
For full details, see Galbraith, Theorem 19.1.9.
Running time follows from analysis of parameter sizes and guaranteed performance of LLL.

Full Coppersmith

Remarks

- It is possible to eliminate the "$-\varepsilon$" in the exponent and achieve $X = M^{1/d}$, e.g. through exhaustive search over topmost bits of x_0.

- Full Coppersmith method shows we can approach $X = M^{1/d}$, significantly better than simple Coppersmith with $X \approx M^{2/d(d+1)}$.

- However, we cannot hope to significantly exceed this bound in general.
- To see why, consider the polynomial:

$$F(x) = x^2 + px \bmod M$$

with $d=2$, $M = p^2$. If we set X *to* $M^{1/2+\varepsilon}$, this equation has many solutions, so we cannot hope to enumerate them all. (To see this, consider $x_0 = kp$ for $|k| < M^{\varepsilon}$.)

- See the exercises for further challenges in using Coppersmith's method.

Applications

Back to the Motivating Example

- Let ($N = pq, e$) be an RSA public key and suppose an attacker has learned the MSBs of p due to some vulnerability.

- Can the attacker recover the rest of *p?*
  - And hence recover the factorisation of N.
  - Then recover the RSA secret key d by solving $de = 1 \bmod (p-1)(q-1)$.

Back to the Motivating Example

- With MSBs of p known, we can write:

$$p = p^* + x_o$$

where p^* is known and x_o is unknown with $|x_o| < X$ for some bound X that depends on the number of known MSBs.

- We have to solve a polynomial equation in variable x, of the form:

$$F(x) = 0 \bmod p.$$

under the guarantee that the solution x_o is relatively small ($|x_o| < X$).

- In our example, $F(x) = p^* + x$.

Recovering p from its MSBs

Theorem

Suppose $N = pq$ and $p < q < 2p$ (so p and q are about the same size).
Let $0 < \varepsilon < 1/4$ and suppose p^* is such that $|p - p^*| \leq (2\sqrt{2})^{-1} \cdot N^{1/4 - \varepsilon}$. Then, given N and p^*, one can factor N in time polynomial in $\log(N)$ and $1/\varepsilon$.

Informally

If we know half the MSBs of p, we can recover the rest of p.

Recovering p from its MSBs

Theorem

Suppose $N = pq$ and $p < q < 2p$ (so p and q are about the same size).
Let $0 < \varepsilon < 1/4$ and suppose p^* is such that $|p - p^*| \leq (2\sqrt{2})^{-1} \cdot N^{1/4 - \varepsilon}$. Then, given N and p^*, one can factor N in time polynomial in $\log(N)$ and $1/\varepsilon$.

Proof

Write $F(x) = x + p^*$, and let x_0 satisfy $F(x_0) = 0 \bmod p$ and:

$$|x_0| < X = (2\sqrt{2})^{-1} \cdot N^{1/4 - \varepsilon}.$$

Set $k = 2h$ for some integer h to be determined.
Consider the $k+1$ polynomials obtained from shifts and powers:

$$N^h,\ N^{h-1}F(x),\ N^{h-2}F(x)^2,\ldots,\ NF(x)^{h-1},\ F(x)^h,\ x\,F(x)^h,\ldots,\ x^{k-h}\,F(x)^h.$$

For all polynomials G in this set, we have $G(x_0) = 0 \bmod p^h$.

Recovering p from its MSBs

Proof (ctd):
So our M is the **unknown** value p^h. We also have maximum degree k.
We build the lattice L in the usual way; we have $\det(L) = N^{h(h+1)/2} X^{k(k+1)/2}$.

To apply Howgrave-Graham's theorem, we need:

$$2^{k/4} \cdot N^{h(h+1)/(2(k+1))} \cdot X^{k/2} < p^h/(k+1)^{1/2}.$$

Since $p > (N/2)^{1/2}$, it is sufficient that

$$(k+1)^{1/2} \cdot 2^{k/4} \cdot N^{h(h+1)/(2(k+1))} \cdot X^{k/2} < (N/2)^{h/2}.$$

This eliminates the unknown p from our analysis, replacing it with a bound in terms of N.
After some more tedious analysis, we get the desired result on taking $h \geq \max\{4, 1/(4\varepsilon)\}$.

Recovering p from its MSBs: An example

Example (Galbraith,Example 19.4.3)

Let $N = 16{,}803{,}551$, $p^* = 2830$, and suppose we are guaranteed that we can take $X = 10$ in Coppersmith's method.

Write $F(x) = x + p^*$ and consider the *reduced* set of polynomials:

$$N,\ F(x),\ xF(x),\ x^2F(x).$$

which should all have the same solution x_o mod p with $|x_o| < 10$.

We build lattice L with basis matrix:

$$B = \begin{vmatrix} N & o & o & o \\ p^* & X & o & o \\ o & p^*X & X^2 & o \\ o & o & p^*X^2 & X^3 \end{vmatrix}$$

Recovering p from its MSBs: An example

Example (Galbraith,Example 19.4.3)
Lattice L with basis matrix B:

$$B = \begin{vmatrix} N & o & o & o \\ p^* & X & o & o \\ o & p^*X & X^2 & o \\ o & o & p^*X^2 & X^3 \end{vmatrix}$$

Running LLL on this basis matrix gives $b_1 = (105,-1200,800,1000)$ which we convert back to the polynomial $G(x) = 105 - 120x + 8x^2 + x^3$.

We then find the root $G(7) = 0$ over the integers.

Hence we recover $p = p^* + 7 = 2837$ and finally $q = N/p = 5923$.

Further Applications

- Coppersmith's method has many other applications in cryptanalysis.
- You will explore some of these in the lab this week.
- See also Galbraith, Chapter 19 for examples.
- It's not always so clear exactly which combinations of polynomials (shifts, powers, etc.) to include when building your lattices. You will need to experiment. (See also this week's exercise class.)
- There are also multivariate versions of Coppersmith's method. These are more advanced and harder to analyse rigorously; you will not need them in the lab this week.