Computing over $\mathbb{Z}, \mathbb{Q}, K[X]$

Clément PERNET

M2-MIA Calcul Exact

Outline

Outline

Exact computations and arithmetic

Domain of Computation

- $\mathbb{Z}, \mathbb{Q}$ $\Rightarrow$variable size
- $\mathbb{Z}_p, \mathsf{GF}(p^k)$ $\Rightarrow$fixed size but specific arithmetic
- $K[X]$ for $K = \mathbb{Z}_p, \mathbb{Z}....$

Key idea: change of representation

Deal with size of arithmetic: reduce to $\mathbb{Z}_p$

- Chinese Remainder Algorithm: $\mathbb{Z} \to \mathbb{Z}_m \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}$

$$x < m_1 \times \cdots \times m_k \in \mathbb{Z} \Leftrightarrow (x_1 \in \mathbb{Z}_{m_1}, \ldots, x_k \in \mathbb{Z}_{m_k})$$

- p-adic Lifting: $\mathbb{Z} \to \mathbb{Z}_{p^k} \to \mathbb{Z}_p$

$$x = x_0 + x_1 p + \ldots x_k p^k < p^{k+1} \in \mathbb{Z}_{p^k} \Leftrightarrow (x_1 \in \mathbb{Z}_p, \ldots, x_k \in \mathbb{Z}_p)$$

- Rational reconstruction: $\mathbb{Q} \to \mathbb{Z}_k \to \mathbb{Z}_p$

$$x = \frac{n}{d} = x_0 + x_1 p + \ldots x_k p^k [p^{k+1}] \Leftrightarrow (x_1 \in \mathbb{Z}_p, \ldots, x_k \in \mathbb{Z}_p)$$

Outline

Chinese remainder algorithm

If $m_1, \ldots, m_k$ pariwise relatively prime:

$$\mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \equiv \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$$

Computation of $y = f(x)$ for $f \in \mathbb{Z}[X], x \in \mathbb{Z}^m$

begin
- Compute an upper bound β on $|f(x)|$;
- Pick $m_1, \ldots m_k$, pairwise prime, s.t. $m_1 \ldots m_k > \beta$;
- **for** $i = 1 \ldots k$ **do**
  - Compute $y_i = f(x \mod m_i) \mod m_i$
- Compute $y = \text{CRT}(y_1, \ldots, y_k)$

$$\text{CRT}: \quad \begin{array}{rcl} \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} & \to & \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \\ (x_1, \ldots, x_k) & \mapsto & \sum_{i=1}^k x_i \Pi_i Y_i \mod \Pi \end{array}$$

$$\text{where} \begin{cases} \Pi & = & \prod_{i=1}^k m_i \\ \Pi_i & = & \Pi/m_i \\ Y_i & = & \Pi_i^{-1} \mod m_i \end{cases}$$

Chinese remainder algorithm

If $m_1, \ldots, m_k$ pariwise relatively prime:

$$\mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \equiv \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$$

Computation of $y = f(x)$ for $f \in \mathbb{Z}[X], x \in \mathbb{Z}^m$

begin

Compute an upper bound β on $|f(x)|$;

Pick $m_1, \ldots m_k$, pairwise prime, s.t. $m_1 \ldots m_k > \beta$;

for $i = 1 \ldots k$ **do**

Compute $y_i = f(x \mod m_i) \mod m_i$; /* Evaluation */

Compute $y = \mathrm{CRT}(y_1, \ldots, y_k)$; /* Interpolation */

$$\begin{array}{rccl} \mathrm{CRT}: & \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} & \rightarrow & \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \\ & (x_1, \ldots, x_k) & \mapsto & \sum_{i=1}^{k} x_i \Pi_i Y_i \mod \Pi \end{array}$$

$$\text{where} \begin{cases} \Pi & = & \prod_{i=1}^{k} m_i \\ \Pi_i & = & \Pi / m_i \\ Y_i & = & \Pi_i^{-1} \mod m_i \end{cases}$$

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Polynomials	
Evaluation: $P \mod X - a$ Evaluate P in a	
Interpolation: $P = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Polynomials	**Integers**
Evaluation: $P \mod X - a$ Evaluate P in a	$N \mod m$ "Evaluate" N in m
Interpolation: $P = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	$N = \sum_{i=1}^{k} y_i \prod_{j \neq i} m_j (\prod_{j \neq i} m_j)^{-1[m_i]}$

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Polynomials	**Integers**
Evaluation: $P \mod X - a$ Evaluate P in a	$N \mod m$ "Evaluate" N in m
Interpolation: $P = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	$N = \sum_{i=1}^{k} y_i \prod_{j \neq i} m_j (\prod_{j \neq i} m_j)^{-1[m_i]}$

Analogy: complexities over $\mathbb{Z} \leftrightarrow$ over $K[X]$

- size of coefficients
- $\mathcal{O}\left(\log \|\text{result}\| \times T_{\text{algebr.}}\right)$
- degree of polynomials
- $\mathcal{O}\left(\deg(\text{result}) \times T_{\text{algebr.}}\right)$

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Polynomials	**Integers**
Evaluation: $P \mod X - a$ Evaluate P in a	$N \mod m$ "Evaluate" N in m
Interpolation: $P = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	$N = \sum_{i=1}^{k} y_i \prod_{j \neq i} m_j (\prod_{j \neq i} m_j)^{-1[m_i]}$

Analogy: complexities over $\mathbb{Z} \leftrightarrow$ over $K[X]$

- size of coefficients
- $\mathcal{O}\left(\log \|\text{result}\| \times T_{\text{algebr.}}\right)$
- $\det(n, \|A\|) = \tilde{\mathcal{O}}\left(n \log |A\| \times n^{\omega}\right)$

- degree of polynomials
- $\mathcal{O}\left(\deg(\text{result}) \times T_{\text{algebr.}}\right)$
- $\det(n, d) = \tilde{\mathcal{O}}\left(nd \times n^{\omega}\right)$

Early termination

Classic Chinese remaindering — Deterministic

- bound β on the result
- Choice of the m_i: such that $m_1 \dots m_k > \beta$

Early termination

Classic Chinese remaindering — Deterministic

- bound β on the result
- Choice of the m_i: such that $m_1 \dots m_k > \beta$

Early termination — Probabilistic Monte Carlo

- For each new modulo m_i:
  - reconstruct $y_i = f(x) \mod m_1 \times \cdots \times m_i$
  - If $y_i == y_{i-1}$ $\Rightarrow$terminated

Early termination

Classic Chinese remaindering — Deterministic

- bound β on the result
- Choice of the m_i: such that $m_1 \dots m_k > \beta$

Early termination — Probabilistic Monte Carlo

- For each new modulo m_i:
  - reconstruct $y_i = f(x) \mod m_1 \times \cdots \times m_i$
  - If $y_i == y_{i-1}$ $\Rightarrow$terminated

Advantage:

- Adaptive number of moduli depending on the output value
- Interesting when
  - pessimistic bound: sparse/structured matrices, ...
  - no bound available

Outline

Informally

(Black-board)

Problem Statement

Definition (Rational Reconstruction Problem)

Let $A \in K[X]$ *of degree* n*, and* $B \in K[X]$ *of degree* $< n$*.*
For a fixed $k \in \{1 \ldots n\}$*, find* $(R, V) \in K[X]$ *satisfying:*

$$gcd(V, A) = 1, \deg(R) < k, \deg(V) \leq n - k \text{ and } R = VB \mod A.$$

$A = X^n$: Padé Approximation

$A = \prod_{i=1}^{n}(X - u_i)$: Cauchy Interpolation

Rational Reconstruction Problem

Existence of a solution: $k + n - k + 1 = n + 1$ unknowns, for n equations

Uniqueness: A divides $R_1 - V_1 B$ and $R_2 - V_2 B$. Thus A divides

$$R_1 V_2 - R_2 V_1 = (R_1 - V_1 B)V_2 - (R_2 - V_2 B)V_1)$$

of degree $< n$.

Algorithm

$$R = VB \mod A \Leftrightarrow VB + UA = R$$

begin

$R_0 \leftarrow A; U_0 \leftarrow 1; V_0 \leftarrow 0;$

$R_0 \leftarrow B; U_0 \leftarrow 0; V_0 \leftarrow 1;$

$i \leftarrow 1;$

while $\deg R_i \geq k$ **do**

$(Q_i, R_{i+1}) \leftarrow \mathsf{QuoRem}(R_{i-1}, R_i);$

$U_{i+1} \leftarrow U_{i-1} - Q_i U_i;$

$V_{i+1} \leftarrow V_{i-1} - Q_i V_i;$

$i \leftarrow i + 1;$

if $gcd(A, V_i) = 1$ **then**

return (R_i, V_i)

else

return 0

Outline

Dense interpolation with errors

Problem : CRT with errors

Given (y_i, m_i) for $i = 1 \ldots n$,
Find Y such that $Y = y_i \mod m_i$ on at least $n - e$ values.

CRT codes [Mandelbaum]

$\Rightarrow$Based on Rational reconstruction

Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

$x \in \mathbb{Z}$ $\longleftrightarrow$

x_1	x_2	$\dots$	x_k

where $m_1 \times \cdots \times m_k > x$ and $x_i = x \mod m_i \, \forall i$

Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

$x \in \mathbb{Z}$ $\longleftrightarrow$

x_1	x_2	$\ldots$	x_k	x_{k+1}	$\ldots$	x_n

where $m_1 \times \cdots \times m_n > x$ and $x_i = x \mod m_i \, \forall i$

Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

$x \in \mathbb{Z} \longleftrightarrow$

x_1	x_2	$\dots$	x_k	x_{k+1}	$\dots$	x_n

where $m_1 \times \cdots \times m_n > x$ and $x_i = x \mod m_i \, \forall i$

Definition

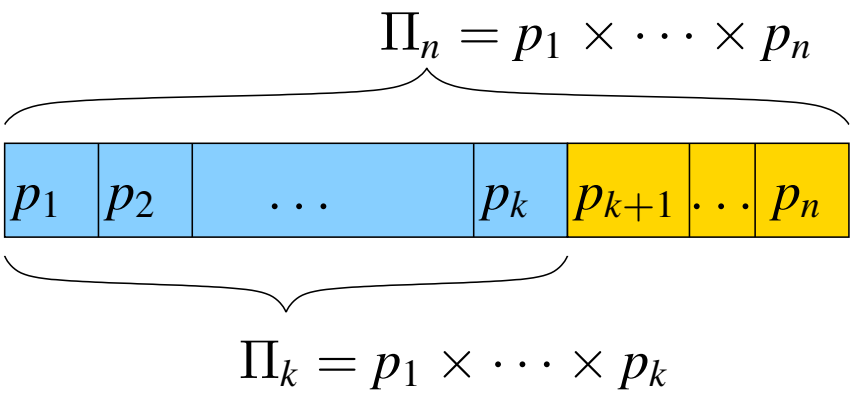
(n,k)-*code:* $C =$

$$\left\{ (x_1, \dots, x_n) \in \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_n} \textbf{ s.t. } \exists x, \left\{ \begin{array}{lcl} x & < & m_1 \dots m_k \\ x_i & = & x \mod m_i \, \forall i \end{array} \right. \right\}$$

Principle

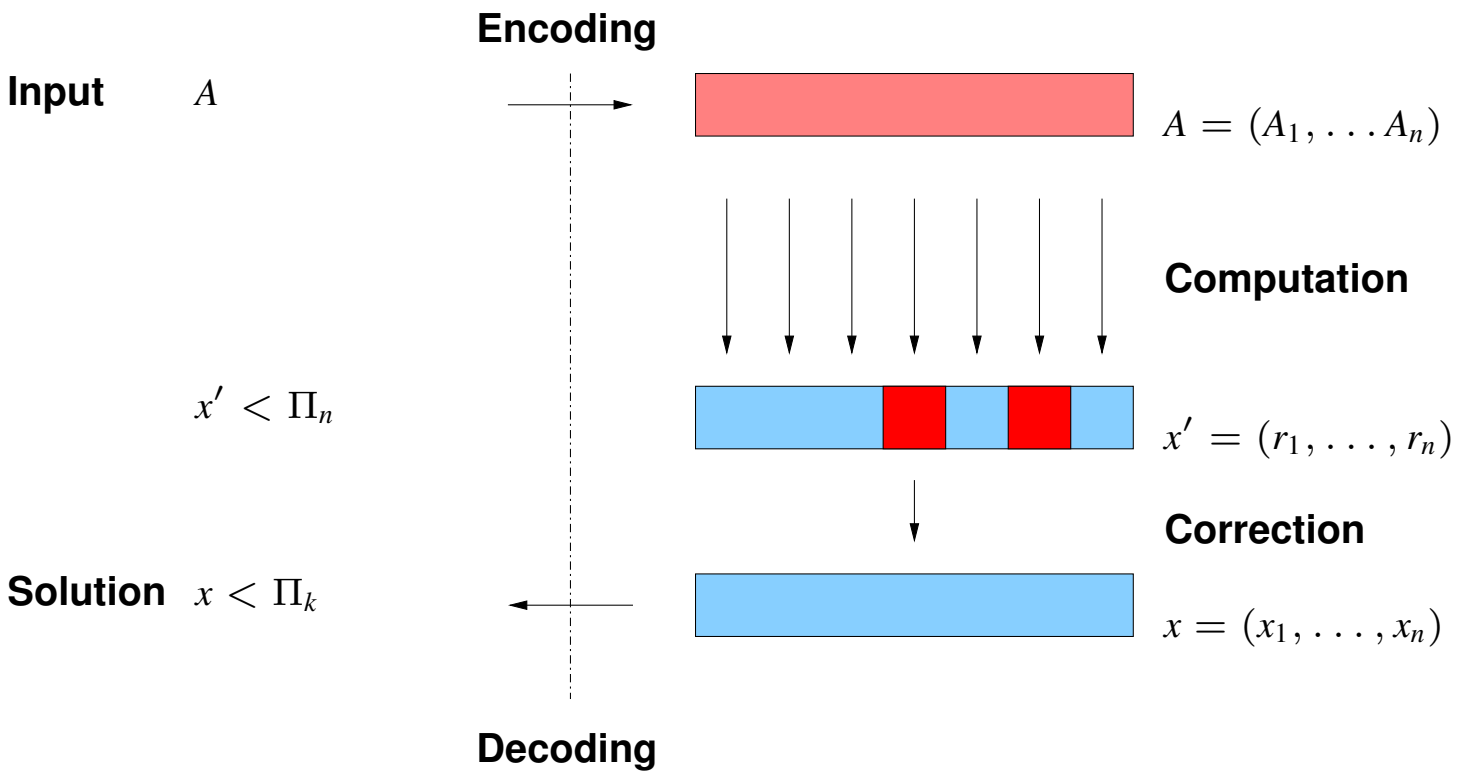
Property

$$X \in C \textit{ iff } X < \Pi_k.$$



Redundancy : $r = n - k$

ABFT with Chinese remainder algorithm



Properties of the code

Error model:

- Error: $E = X' - X$
- Error support: $I = \{i \in 1 \ldots n, E \neq 0 \mod m_i\}$
- Impact of the error: $\Pi_F = \prod_{i \in I} m_i$

Properties of the code

Error model:

- Error: $E = X' - X$
- Error support: $I = \{i \in 1 \dots n, E \neq 0 \mod m_i\}$
- Impact of the error: $\Pi_F = \prod_{i \in I} m_i$

Detects up to r errors:

If $X' = X + E$ with $X \in C, \#I \leq r$,

then $X' > \Pi_k$.

- Redundancy $r = n - k$, distance: $r + 1$
- $\Rightarrow$can correct up to $\left\lfloor \frac{r}{2} \right\rfloor$ errors in theory
- More complicated in practice...

Correction

- $\forall i \notin I : E \mod m_i = 0$
- E is a multiple of Π_V: $E = Z\Pi_V = Z\prod_{i \notin I} m_i$
- $\gcd(E, \Pi) = \Pi_V$

Mandelbaum 78: rational reconstruction

$$\begin{aligned} X = X' - E &= X' - Z\Pi_v \\ \frac{X}{\Pi} &= \frac{X'}{\Pi} - \frac{Z}{\Pi_F} \end{aligned}$$

$\Rightarrow |\frac{X'}{\Pi} - \frac{Z}{\Pi_F}| \leq \frac{1}{2\Pi_F^2}$

$\Rightarrow \frac{Z}{\Pi_F} = \frac{E}{\Pi}$ is a convergent of $\frac{X'}{\Pi}$

$\Rightarrow$ Rational reconstruction of $X' \mod \Pi$

$\Rightarrow$ Extended Euclidean Algorithm interrupted

Correction capacity

Mandelbaum 78:

- 1 symbol = 1 residue
- Polynomial time algorithm if $e \leq (n-k)\frac{\log m_{\min} - \log 2}{\log m_{\max} + \log m_{\min}}$
- worst case: exponential (random perturbation)

Goldreich Ron Sudan 99 weighted residues $\Rightarrow$equivalent

Guruswami Sahai Sudan 00 invariably polynomial time

Correction capacity

Mandelbaum 78:

- 1 symbol = 1 residue
- Polynomial time algorithm if $e \leq (n-k)\frac{\log m_{\min} - \log 2}{\log m_{\max} + \log m_{\min}}$
- worst case: exponential (random perturbation)

Goldreich Ron Sudan 99 weighted residues $\Rightarrow$equivalent

Guruswami Sahai Sudan 00 invariably polynomial time

Interpretation:

- Errors have variable weights depending on their impact $\Pi_F = \prod_{i \in I} m_i$
- Example: $m_1 = 2, m_2 = 3, m_3 = 101$
  - 1 error $\mod 2$ or $\mod 3$, can be corrected, not $\mod 101$
  - limits to $|X| < 2$, where $|X| < 17$ is sufficient

Analogy with Reed Solomon

Dornstetter 87 Berlekamp/Massey $\Leftrightarrow$ extended Euclidean Alg.

Gao02 Reed-Solomon decoding by extended Euclidean Alg.

- Chinese Remaindering over $K[X]$
- $m_i = X - a_i$
- Encoding = evaluation in a_i
- Decoding = interpolation
- Correction = Extended Euclidean algorithm

Analogy with Reed Solomon

Dornstetter 87 Berlekamp/Massey $\Leftrightarrow$ extended Euclidean Alg.

Gao02 Reed-Solomon decoding by extended Euclidean Alg.

- Chinese Remaindering over $K[X]$
- $m_i = X - a_i$
- Encoding = evaluation in a_i
- Decoding = interpolation
- Correction = Extended Euclidean algorithm

$\Rightarrow$Generalization for m_i of degrees > 1
$\Rightarrow$Variable impact, depending on the degree of m_i
$\Rightarrow$Necessary unification [Sudan 01,...]

Generalized point of view: amplitude code

Over a Euclidean ring $\mathcal{A}$ with a Euclidean function ν, multiplicative and sub-additive, ie such that

$$\begin{aligned} \nu(ab) &= \nu(a)\nu(b) \\ \nu(a+b) &\leq \nu(a)+\nu(b) \end{aligned}$$

eg.

- over $\mathbb{Z}$: $\nu(x) = |x|$
- over $K[X]$: $\nu(P) = 2^{\deg(P)}$

Definition

Error impact between x and y: $\Pi_F = \prod_{i|x\neq y[m_i]} m_i$

Error amplitude: $\nu(\Pi_F)$

Amplitude codes

Distance

$$\begin{array}{rccc} \Delta : & \mathcal{A} \times \mathcal{A} & \rightarrow & \mathbb{R}_+ \\ & (x, y) & \mapsto & \displaystyle\sum_{i | x \neq y[m_i]} \log_2 \nu\left(m_i\right) \end{array}$$

$$\Delta(x, y) = \log_2 \nu(\Pi_F)$$

Definition ((n, k) amplitude code)

Given $\{m_i\}_{i \leq m}$ *pairwise rel. prime, and* $\kappa \in \mathbb{R}_+$ *The set*

$$C = \{x \in \mathcal{A} : \nu(x) < \kappa\},$$

$n = \log_2 \prod_{i \leq m} m_i, k = \log_2 \kappa$. *is a (n,k) amplitude code.*

Definition ((n, k) amplitude code)

Given $\{m_i\}_{i \leq m}$ *pairwise rel. prime, and* $\kappa \in \mathbb{R}_+$ *The set*

$$C = \{x \in \mathcal{A} : \nu(x) < \kappa\},$$

$n = \log_2 \prod_{i \leq m} m_i, k = \log_2 \kappa$. *is a (n,k) amplitude code.*

Property (Quasi MDS)

$\forall (x, y) \in C$

$$\Delta(x, y) > n - k - 1$$

$\Rightarrow$correction capacity = maximal amplitude of an error that can be corrected

Definition ((n,k) amplitude code)

Given $\{m_i\}_{i\leq m}$ *pairwise rel. prime, and* $\kappa \in \mathbb{R}_+$ *The set*

$$C = \{x \in \mathcal{A} : \nu(x) < \kappa\},$$

$n = \log_2 \prod_{i\leq m} m_i, k = \log_2 \kappa$. *is a (n,k) amplitude code.*

Property (Quasi MDS)

$\forall (x,y) \in C,\ \mathcal{A} = K[X]$

$$\Delta(x,y) \geq n - k + 1$$

$\sim$ *Singleton bound*

$\Rightarrow$correction capacity = maximal amplitude of an error that can be corrected

Advantages

- Generalization over any Euclidean ring
- Natural representation of the amount of information
- No need to sort moduli
- Finer correction capacities

Advantages

- Generalization over any Euclidean ring
- Natural representation of the amount of information
- No need to sort moduli
- Finer correction capacities
- **Adaptive decoding:** taking advantage of all the available redundancy
- **Early termination:** with no a priori knowledge of a bound on the result

Interpretation of Mandelbaum's algorithm

Remark

Rational reconstruction $\Rightarrow$ *Partial Extended Euclidean Algorithm*

Property

The Extended Euclidean Algorithm, applied to (Π, E) *and to* $(X' = X + E, \Pi)$*, performs the same first iterations until* $r_i < \Pi_V$*.*

$$\begin{array}{c|c} u_{i-1}\Pi + v_{i-1}E = \Pi_v & u_{i-1}\Pi + v_{i-1}X' = r_{i-1} \\ u_i\Pi + v_iE = 0 & u_i\Pi + v_iX' = r_i \end{array}$$

$$\Rightarrow v_iX = r_i$$

Amplitude decoding, with static correction capacity

Amplitude based decoder over R

Input: Π, X'
Input: $\tau \in \mathbb{R}_+ \mid \tau < \frac{\nu(\Pi)}{2}$: bound on the maximal error amplitude
Output: $X \in R$: corrected message s.t. $\nu(X)4\tau^2 \leq \nu(\Pi)$
begin

$u_0 = 1, v_0 = 0, r_0 = \Pi;$
$u_1 = 0, v_1 = 1, r_1 = X';$
$i = 1;$
while $(\nu(r_i) > \nu(\Pi)/2\tau)$ **do**

Let $r_{i-1} = q_i r_i + r_{i+1}$ be the Euclidean division of r_{i-1} by r_i;
$u_{i+1} = u_{i-1} - q_i u_i;$
$v_{i+1} = v_{i-1} - q_i v_i;$
$i = i + 1;$

return $X = \frac{r_i}{v_i}$

- reaches the quasi-maximal correction capacity

Amplitude decoding, with static correction capacity

Amplitude based decoder over R

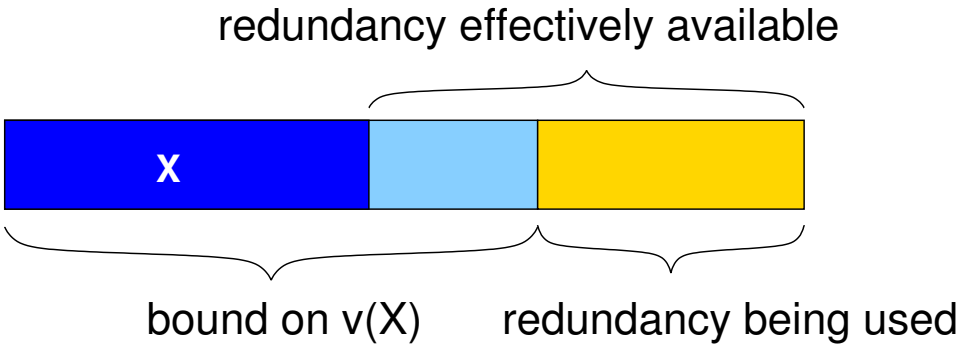
Input: Π, X'
Input: $\tau \in \mathbb{R}_+ \mid \tau < \frac{\nu(\Pi)}{2}$: bound on the maximal error amplitude
Output: $X \in R$: corrected message s.t. $\nu(X)4\tau^2 \leq \nu(\Pi)$
begin
  $u_0 = 1, v_0 = 0, r_0 = \Pi;$
  $u_1 = 0, v_1 = 1, r_1 = X';$
  $i = 1;$
  while $(\nu(r_i) > \nu(\Pi)/2\tau)$ **do**
    Let $r_{i-1} = q_i r_i + r_{i+1}$ be the Euclidean division of r_{i-1} by r_i;
    $u_{i+1} = u_{i-1} - q_i u_i;$
    $v_{i+1} = v_{i-1} - q_i v_i;$
    $i = i + 1;$
  return $X = \frac{r_i}{v_i}$

- reaches the quasi-maximal correction capacity
- requires an *a priori* knowledge of τ
  $\Rightarrow$How to make the correction capacity adaptive?

Adaptive approach

Multiple goals:

- With a fixed n, the correction capacity depends on a bound on $\nu(X)$
  $\Rightarrow$pessimistic estimate
  $\Rightarrow$how to take advantage of all the available redundancy?



A first adaptive approach: divisibility check

Termination criterion in the Extended Euclidean alg.:

- $u_{i+1}\Pi + v_{i+1}E = 0$
  $\Rightarrow E = -u_{i+1}\Pi / v_{i+1}$
  $\Rightarrow$test if v_j divides Π
- check if X satisfies: $\nu(X) \leq \frac{\nu(\Pi)}{4\nu(v_j)^2}$
- But several candidates are possible
  $\Rightarrow$discrimination by a post-condition on the result

A first adaptive approach: divisibility check

Termination criterion in the Extended Euclidean alg.:

- $u_{i+1}\Pi + v_{i+1}E = 0$
  $\Rightarrow E = -u_{i+1}\Pi / v_{i+1}$
  $\Rightarrow$test if v_j divides Π
- check if X satisfies: $\nu(X) \leq \frac{\nu(\Pi)}{4\nu(v_j)^2}$
- But several candidates are possible
  $\Rightarrow$discrimination by a post-condition on the result

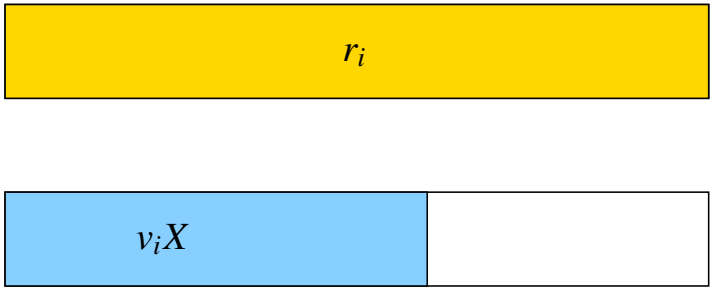
Example

m_i	3	5	7
x_i	2	3	2

- $x = 23$ *with 0 error*
- $x = 2$ *with 1 error*

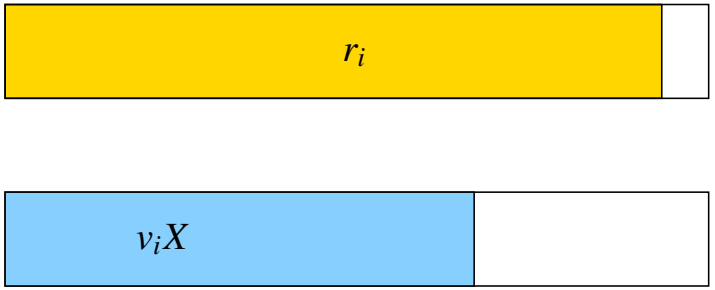
Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$



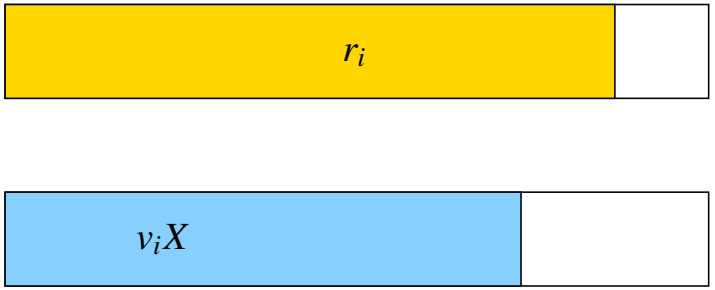
Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$



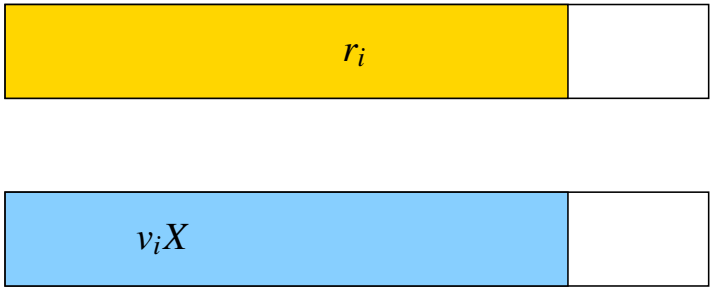
Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$



Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

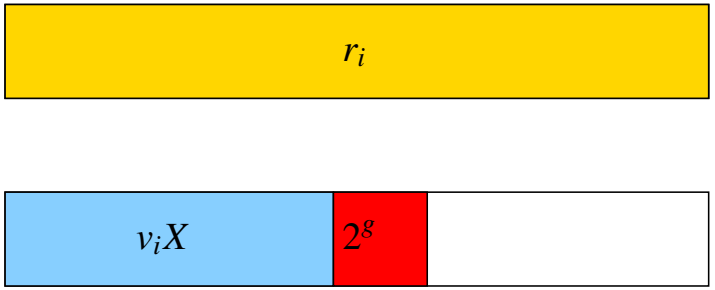


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.

Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

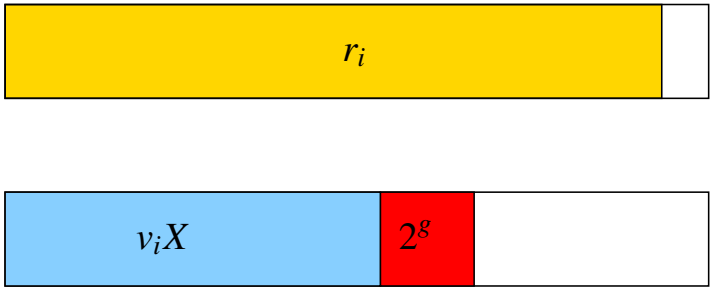


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

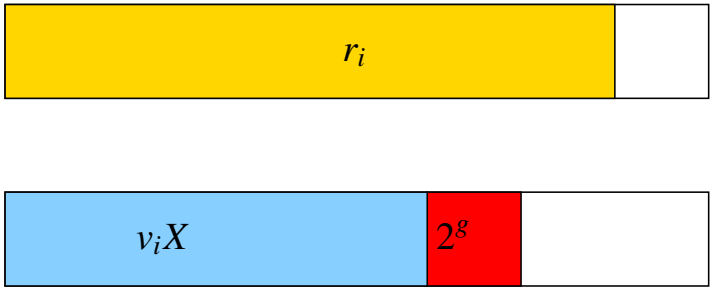


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

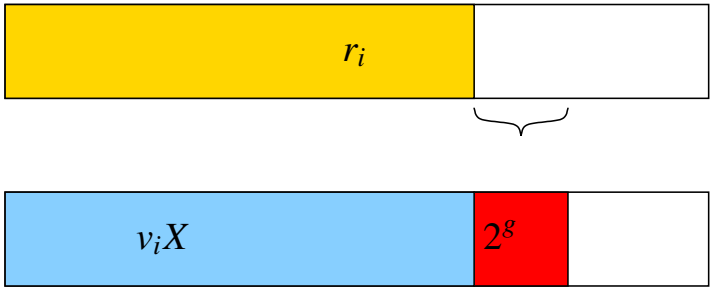


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

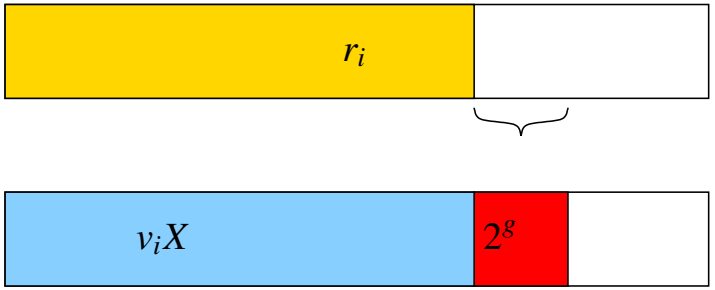


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$



$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Property

- *Loss of correction capacity: very small in practice*
- *Test of the divisibility for the remaining candidates*
- *Strongly reduces the number of divisibility tests*

Experiments

Size of the error	10	50	100	200	500	1000
$g = 2$	1/446	1/765	1/1118	2/1183	2/4165	1/7907
$g = 3$	1/244	1/414	1/576	2/1002	2/2164	1/4117
$g = 5$	1/53	1/97	1/153	2/262	1/575	1/1106
$g = 10$	1/1	1/3	1/9	1/14	1/26	1/35
$g = 20$	1/1	1/1	1/1	1/1	1/1	1/1

Table: Number of remaining candidates after the gap detection: c/d means d candidates with a gap $> 2^g$, and c of them passed the divisibility test. $n \approx 6001$ (3000 moduli), $\kappa \approx 201$ (100 moduli).

Experiments

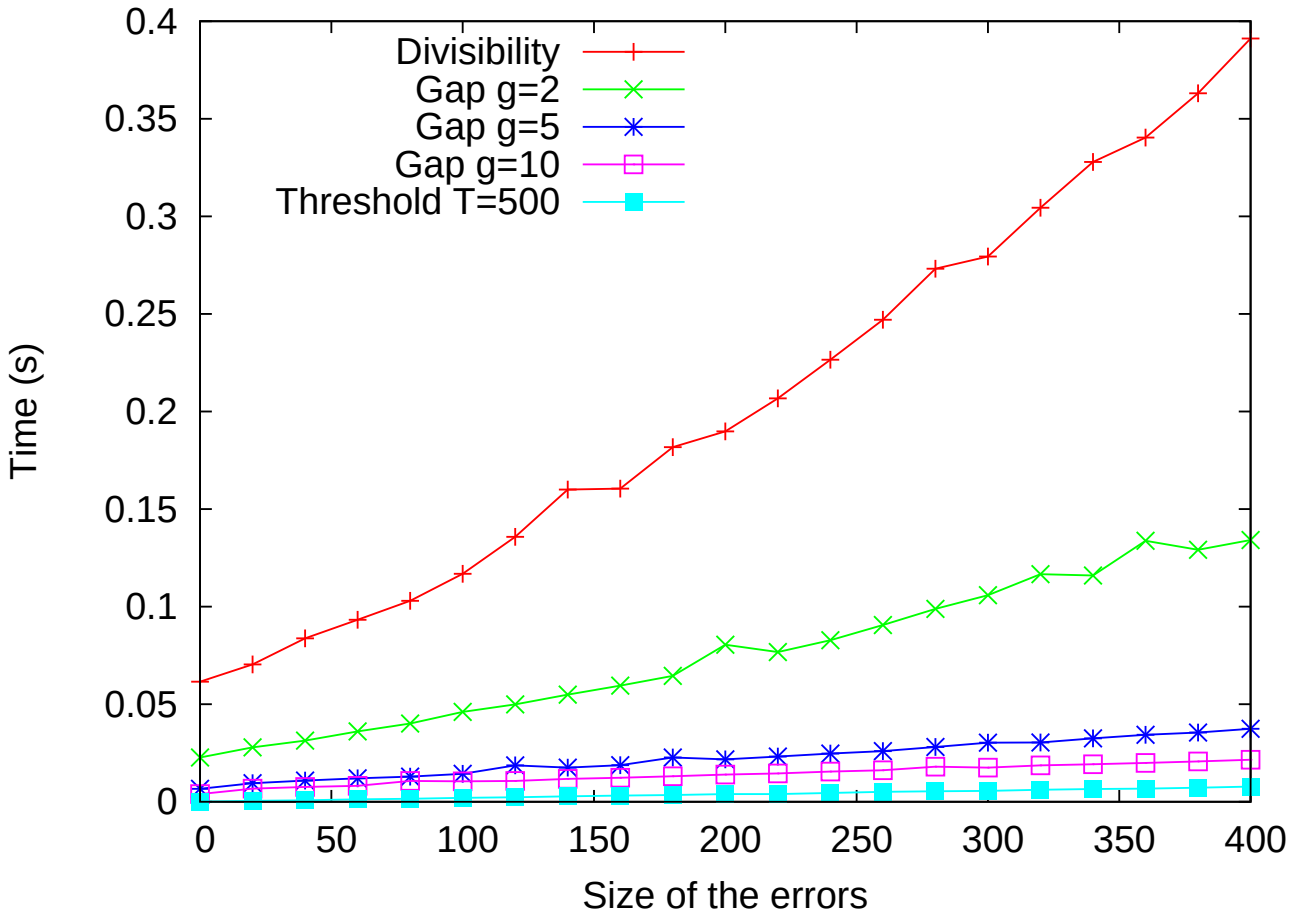


Figure: Comparison for $n \approx 26\,016$ ($m = 1300$ moduli of 20 bits), $\kappa \approx 6001$ (300 moduli) and $\tau \approx 10007$ (about 500 moduli).

Experiments

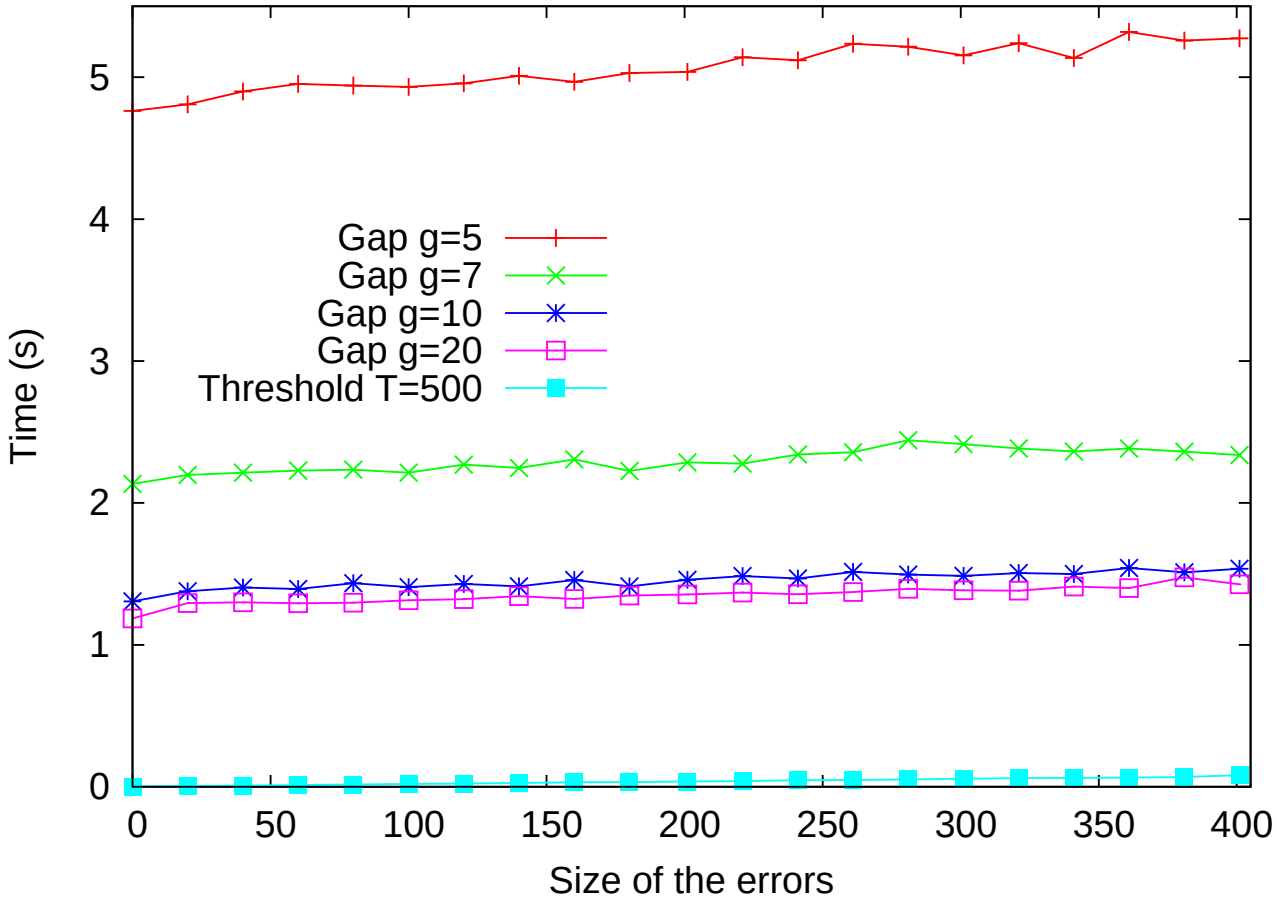


Figure: Comparison for $n \approx 200\,917$ ($m = 10000$ moduli of 20 bits), $\kappa \approx 17\,0667$ (8500 moduli) and $\tau \approx 10498$ (500 moduli).

Gap: Euclidean Algorithm down to the end $\Rightarrow$overhead

Dense rational function interpolation with errors (Cauchy interpolation)

$$y_i = \frac{f(x_i)}{g(x_i)}$$

Rational function interpolation: Pade approximant

- Find $h \in K[X]$ s.t. $h(x_i) = y_i$ (interpolation)
- Find f, g s.t. $hg = f \mod \prod(X - x_i)$ (Pade approx)

Dense rational function interpolation with errors (Cauchy interpolation)

$$y_i = \frac{f(x_i)}{g(x_i)}$$

Rational function interpolation: Pade approximant

- Find $h \in K[X]$ s.t. $h(x_i) = y_i$ (interpolation)
- Find f, g s.t. $hg = f \mod \prod(X - x_i)$ (Pade approx)

Introducing an error of impact $\Pi_F = \prod_{i \in I}(X - x_i)$:

$$hg\Pi_F = f\Pi_F \mod \prod(X - x_i)$$

Property

If $n \geq \deg f + \deg g + 2e$, one can interpolate with at most e errors