CCA2 and partial key recovery attack on PALOMA

Daniel J. Bernstein & Tanja Lange

thanks to Jolijn Cottaar, Kathrin Hövelmanns, Alex Pellegrini, and Silvia Ritsch for discussions

19 July 2024

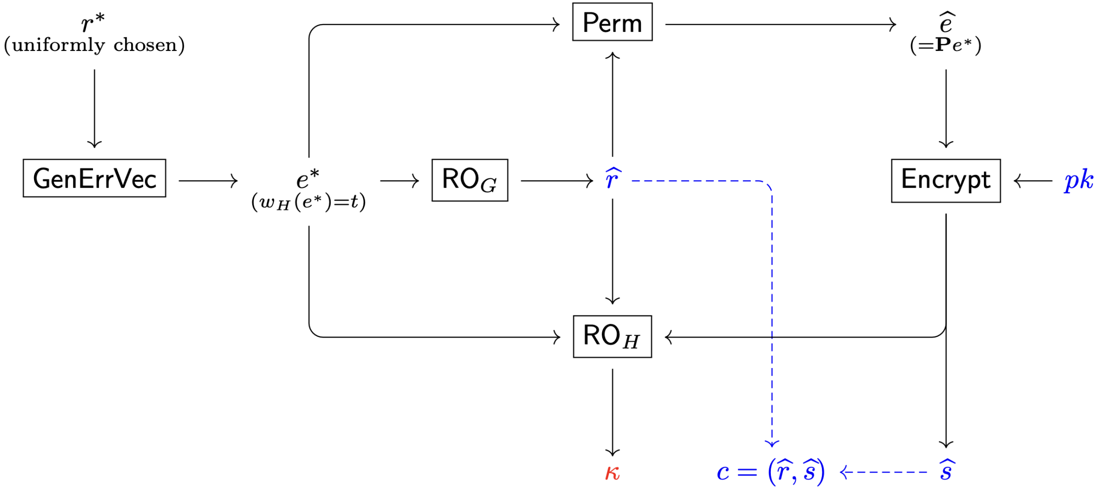
Ingredients in PALOMA

- PALOMA is a code-based cryptosystem using Goppa codes.
- Parameters: m, n, and t.
  - $m = 13$.
  - $n < 2^m$ is code length.
  - t is number of errors code can efficiently correct.
- pk is a $mt \times (m - mt)$ matrix M over $\mathbb{F}_2$.
- pk expands to $mt \times n$ matrix $\hat{H} = [I|M]$.
- Encryption: $\hat{s} = \hat{H}\hat{e}$, for $\mathrm{wt}(\hat{e}) = t$.

Ingredients in PALOMA

- PALOMA is a code-based cryptosystem using Goppa codes.
- Parameters: m, n, and t.
  - $m = 13$.
  - $n < 2^m$ is code length.
  - t is number of errors code can efficiently correct.
- pk is a $mt \times (m - mt)$ matrix M over $\mathbb{F}_2$.
- pk expands to $mt \times n$ matrix $\hat{H} = [I|M]$.
- Encryption: $\hat{s} = \hat{H}\hat{e}$, for $\mathrm{wt}(\hat{e}) = t$.
- Decryption uses Goppa decoder to retrieve $\hat{e}$ from $\hat{s}$.
- Assumption: $\hat{H} = SHP'$ hides structured Goppa matrix H (P' random permutation matrix, S invertible matrix to get $\hat{H} = [I|M]$).

PALOMA encapsulation



(a) $\kappa, c = (\widehat{r}, \widehat{s}) \leftarrow \mathsf{Encap}(pk)$

Image credit: PALOMA Team

PALOMA decapsulation

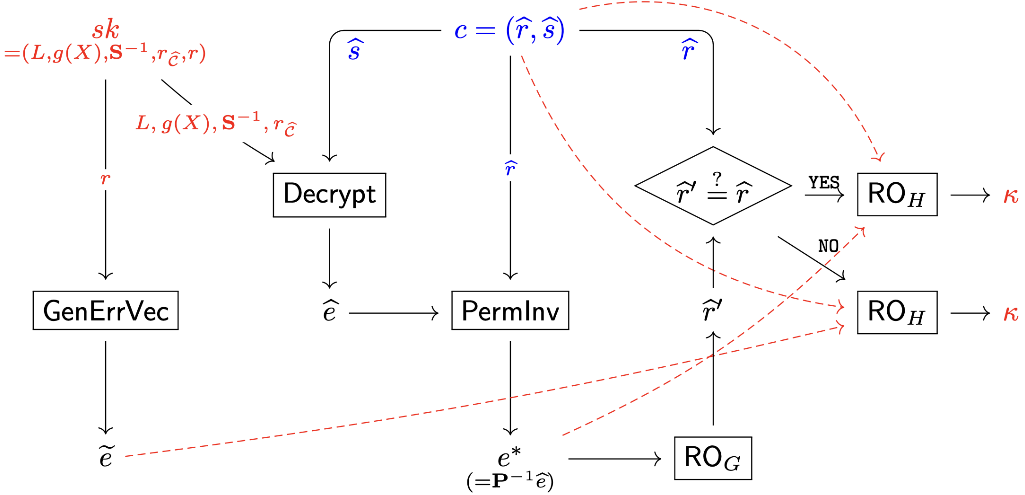


Image credit: PALOMA Team

(b) $\kappa \leftarrow \mathsf{Decap}(sk; c = (\widehat{r}, \widehat{s}))$

Reaction attacks

- Goes back to turn of century
  - “Reaction Attacks Against Several Public-Key Cryptosystems” (Hall, Goldberg, Schneier)
  - “Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem” (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.

Reaction attacks

- Goes back to turn of century
  - "Reaction Attacks Against Several Public-Key Cryptosystems" (Hall, Goldberg, Schneier)
  - "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem" (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.

Reaction attacks

- Goes back to turn of century
  - "Reaction Attacks Against Several Public-Key Cryptosystems" (Hall, Goldberg, Schneier)
  - "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem" (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.
- Goppa decoder decodes **up to and including** t errors, fails for more.
- Change $\hat{s} = \hat{H}\hat{e}$ to $s' = \hat{s} + h_1$, where h_1 is the first column of $\hat{H}$.

Reaction attacks

- Goes back to turn of century
  - "Reaction Attacks Against Several Public-Key Cryptosystems" (Hall, Goldberg, Schneier)
  - "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem" (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.
- Goppa decoder decodes **up to and including** t errors, fails for more.
- Change $\hat{s} = \hat{H}\hat{e}$ to $s' = \hat{s} + h_1$, where h_1 is the first column of $\hat{H}$.
- s' is encryption of $e' = \hat{e} + e_1$.

Reaction attacks

- Goes back to turn of century
  - "Reaction Attacks Against Several Public-Key Cryptosystems" (Hall, Goldberg, Schneier)
  - "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem" (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.
- Goppa decoder decodes **up to and including** t errors, fails for more.
- Change $\hat{s} = \hat{H}\hat{e}$ to $s' = \hat{s} + h_j$, where h_j is the j-th column of $\hat{H}$.
- s' is encryption of $e' = \hat{e} + e_j$.

Reaction attacks

- Goes back to turn of century
  - "Reaction Attacks Against Several Public-Key Cryptosystems" (Hall, Goldberg, Schneier)
  - "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem" (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.
- Goppa decoder decodes **up to and including** t errors, fails for more.
- Change $\hat{s} = \hat{H}\hat{e}$ to $s' = \hat{s} + h_j$, where h_j is the j-th column of $\hat{H}$.
- s' is encryption of $e' = \hat{e} + e_j$.
- Decryption works iff e' has weight $\leq t$, i.e., if position j flipped from 1 to 0.

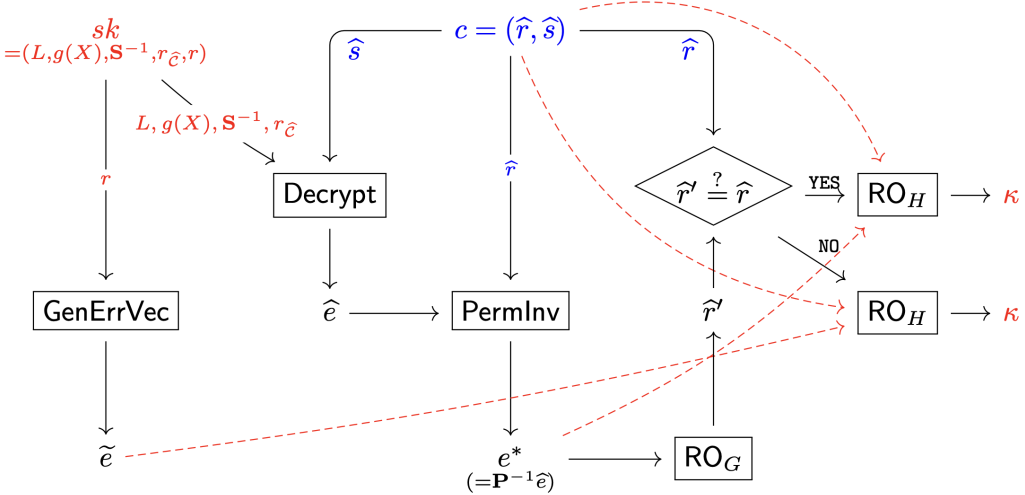
Reaction attacks

- Goes back to turn of century
  - “Reaction Attacks Against Several Public-Key Cryptosystems” (Hall, Goldberg, Schneier)
  - “Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem” (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.
- Goppa decoder decodes **up to and including** t errors, fails for more.
- Change $\hat{s} = \hat{H}\hat{e}$ to $s' = \hat{s} + h_j$, where h_j is the j-th column of $\hat{H}$.
- s' is encryption of $e' = \hat{e} + e_j$.
- Decryption works iff e' has weight $\leq t$, i.e., if position j flipped from 1 to 0.
- Learn $\hat{e}$ in at most $n - 1$ steps.

Reaction attacks

- Goes back to turn of century
  - "Reaction Attacks Against Several Public-Key Cryptosystems" (Hall, Goldberg, Schneier)
  - "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece cryptosystem" (Verheul, Doumen, Tilborg)
- Send specially crafted ciphertext, watch for reaction.
- Learn encrypted message (for codes) or key (lattices) from adaptive queries.
- Goppa decoder decodes **up to and including** t errors, fails for more.
- Change $\hat{s} = \hat{H}\hat{e}$ to $s' = \hat{s} + h_j$, where h_j is the j-th column of $\hat{H}$.
- s' is encryption of $e' = \hat{e} + e_j$.
- Decryption works iff e' has weight $\leq t$, i.e., if position j flipped from 1 to 0.
- Learn $\hat{e}$ in at most $n - 1$ steps.
- Our attack with Alex Pellegrini from 13 April against the PALOMA software used the reaction that some decryption attempts crashed the program.

PALOMA decapsulation



$(b)\ \kappa \leftarrow \mathsf{Decap}(sk; c = (\widehat{r}, \widehat{s}))$

Image credit: PALOMA Team

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.

1

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.
- $\kappa = \mathrm{RO}_H(e^*, \hat{r}, \hat{s})$ with $\hat{r} = \mathrm{RO}_G(e^*)$ is computable for guessed e^*.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.
- $\kappa = \mathrm{RO}_H(e^*, \hat{r}, \hat{s})$ with $\hat{r} = \mathrm{RO}_G(e^*)$ is computable for guessed e^*.
- Let $e = (00\cdots0), r = \mathrm{RO}_G(e)$.
- For j in $0, 1, 2, \ldots, n$
  - if decapsulation of $(r, \hat{s} + h_j)$ returns $\mathrm{RO}_H(e, r, \hat{s} + h_j)$ then we know $\hat{e}_j = 0$.
- Now know $\hat{e}_j = 0$ for 40 – 70% of all j.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.
- $\kappa = \mathrm{RO}_H(e^*, \hat{r}, \hat{s})$ with $\hat{r} = \mathrm{RO}_G(e^*)$ is computable for guessed e^*.
- Let $e = (00\cdots0)$, $r = \mathrm{RO}_G(e)$.
- For j in $0, 1, 2, \ldots, n$
  - if decapsulation of $(r, \hat{s} + h_j)$ returns $\mathrm{RO}_H(e, r, \hat{s} + h_j)$ then we know $\hat{e}_j = 0$.
- Now know $\hat{e}_j = 0$ for $40 - 70\%$ of all j.
- $\hat{s} + h_i + h_j$ matches e' of weight t iff exactly one of positions i and j is 1.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.
- $\kappa = \mathrm{RO}_H(e^*, \hat{r}, \hat{s})$ with $\hat{r} = \mathrm{RO}_G(e^*)$ is computable for guessed e^*.
- Let $e = (00\cdots0), r = \mathrm{RO}_G(e)$.
- For j in $0, 1, 2, \ldots, n$
  - if decapsulation of $(r, \hat{s} + h_j)$ returns $\mathrm{RO}_H(e, r, \hat{s} + h_j)$ then we know $\hat{e}_j = 0$.
- Now know $\hat{e}_j = 0$ for 40 – 70% of all j.
- $\hat{s} + h_i + h_j$ matches e' of weight t iff exactly one of positions i and j is 1.
- Use pairs of columns to identify all positions in original $\hat{e}$.
  Obtain e^* using $\hat{r}$.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.
- $\kappa = \mathrm{RO}_H(e^*, \hat{r}, \hat{s})$ with $\hat{r} = \mathrm{RO}_G(e^*)$ is computable for guessed e^*.
- Let $e = (00\cdots0), r = \mathrm{RO}_G(e)$.
- For j in $0, 1, 2, \dots, n$
  - if decapsulation of $(r, \hat{s} + h_j)$ returns $\mathrm{RO}_H(e, r, \hat{s} + h_j)$ then we know $\hat{e}_j = 0$.
- Now know $\hat{e}_j = 0$ for 40 – 70% of all j.
- $\hat{s} + h_i + h_j$ matches e' of weight t iff exactly one of positions i and j is 1.
- Use pairs of columns to identify all positions in original $\hat{e}$.
  Obtain e^* using $\hat{r}$. Our attack software takes 0.5 – 7 minutes.

[1] Can prove, but relies on details on decoder.

CCA attack on PALOMA

- Attack seems to be stopped: we don't know e' and $\hat{r}$ is obtained from e^* by one-way function RO_G.
- Observation:[1] Goppa decoder often returns $(00\cdots0)$ on random input.
- Any permutation of $(00\cdots0)$ remains $(00\cdots0)$.
- $\kappa = \mathrm{RO}_H(e^*, \hat{r}, \hat{s})$ with $\hat{r} = \mathrm{RO}_G(e^*)$ is computable for guessed e^*.
- Let $e = (00\cdots0), r = \mathrm{RO}_G(e)$.
- For j in $0, 1, 2, \ldots, n$
  - if decapsulation of $(r, \hat{s} + h_j)$ returns $\mathrm{RO}_H(e, r, \hat{s} + h_j)$ then we know $\hat{e}_j = 0$.
- Now know $\hat{e}_j = 0$ for $40 - 70\%$ of all j.
- $\hat{s} + h_i + h_j$ matches e' of weight t iff exactly one of positions i and j is 1.
- Use pairs of columns to identify all positions in original $\hat{e}$.
  Obtain e^* using $\hat{r}$. Our attack software takes $0.5 - 7$ minutes.
- Same recovery as with Pellegrini. New: **valid** fake ciphertexts, predicting κ.

[1] Can prove, but relies on details on decoder.

Binary Goppa code

Let $q = 2^m$. A binary Goppa code is defined by

- a list $L = (\alpha_1, \ldots, \alpha_n)$ of n distinct elements in $\mathbb{F}_q$, called support.
- a square-free polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t with $g(\alpha_i) \neq 0$ for all $1 \leq i \leq n$. $g(x)$ is called Goppa polynomial.

The corresponding binary Goppa code is

$$\left\{ \mathrm{c} \in \mathbb{F}_2^n \,\middle|\, S(\mathrm{c}) = \frac{c_1}{x - \alpha_1} + \frac{c_2}{x - \alpha_2} + \cdots + \frac{c_n}{x - \alpha_n} \equiv 0 \bmod g(x) \right\}$$

- Congruence mod g defines $t \times n$ parity check-matrix over $\mathbb{F}_q$.
- Use explicit basis of $\mathbb{F}_q/\mathbb{F}_2$ to get $nt \times n$ matrix H.
- Restrict code words to having entries in $\mathbb{F}_2$.
- Code has length n, dimension $k \geq n - mt$ and minimum distance $d \geq 2t + 1$.

KeyGen in PALOMA

PALOMA chooses

$$g(x) = \prod_{\alpha \in T} (x - \alpha)$$

for $T \subseteq \mathbb{F}_q \setminus \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ with $|T| = t$. Hence, $g(x)$ splits completely over $\mathbb{F}_q$.

PALOMA KeyGen, main secret is string r:

1. $(\alpha_1, \alpha_2, \ldots, \alpha_q) = \text{SHUFFLE}_r(\mathbb{F}_q)$.
2. $L = (\alpha_1, \alpha_2, \ldots, \alpha_n),\ \ T = (\alpha_{n+1}, \alpha_{n+2}, \ldots, \alpha_{n+t})$.
3. Compute g and parity-check matrix H.
4. Pick random permutation matrix P', compute HP' & bring to systematic form, repeat this step if fails.

Secrets are L, g, and P'; sk includes S with $\hat{H} = SHP' = [I|M]$.
Public key is M, the rightmost $n - mt$ columns of $\hat{H}$.
P' effectively changes order of elements in L.

Partial key recovery attack

- Goppa codes can efficiently correct up to t errors.
- Let $(\alpha'_1, \alpha'_2, \ldots, \alpha'_n) = P'L$.
- Observation:[2] Decoder used in PALOMA has exception:

$$He_j \text{ decodes to } (00\cdots 0) \text{ iff } \alpha'_j = 0.$$

[2] Also easily explained from details on decoder.

Partial key recovery attack

- Goppa codes can efficiently correct up to t errors.
- Let $(\alpha'_1, \alpha'_2, \ldots, \alpha'_n) = P'L$.
- Observation:[2] Decoder used in PALOMA has exception:

$$He_j \text{ decodes to } (00\cdots0) \text{ iff } \alpha'_j = 0.$$

- Let $e = (00\cdots0), r = \mathrm{RO}_G(e)$.
- Attack algorithm:
  1. For j in $1, 2, 3, \ldots, n$:
     - If decapsulation of (r, h_j) returns $\mathrm{RO}_H(e, r, h_j)$: return "$\alpha'_j = 0$".
  2. Return "0 is not in support".
- This takes at most n steps and will find the position of 0 if included.

[2] Also easily explained from details on decoder.

Bonus slides

Patterson decoding of c + e in $\Gamma(L, g)$

$$s(x) = \sum_{i=1}^{n} (c_i + e_i)/(x - \alpha_i)$$

Patterson decoding of c + e in $\Gamma(L, g)$

$$s(x) = \sum_{i=1}^{n} (c_i + e_i)/(x - \alpha_i) \equiv \left(\sum_{i=1}^{n} e_i \prod_{j \neq i} (x - \alpha_j) \right) / \prod_{i=1}^{n} (x - \alpha_i) \bmod g(x).$$

Patterson decoding of c + e in $\Gamma(L, g)$

$$s(x) = \sum_{i=1}^{n} (c_i + e_i)/(x - \alpha_i) \equiv \left(\sum_{i=1}^{n} e_i \prod_{j \neq i} (x - \alpha_j) \right) / \prod_{i=1}^{n} (x - \alpha_i) \bmod g(x).$$

- Put $f(x) = \prod_{i=1}^{n} (x - \alpha_i)^{e_i}$ with $e_i \in \{0, 1\}$, then $f'(x) = \sum_{i=1}^{n} e_i \prod_{j \neq i} (x - \alpha_j)^{e_j}$.
- Thus $s(x) \equiv f'(x)/f(x) \bmod g(x)$. We want to find f.

Patterson decoding of $\mathrm{c}+\mathrm{e}$ in $\Gamma(L,g)$

$$s(x)=\sum_{i=1}^{n}(c_i+e_i)/(x-\alpha_i)\equiv\left(\sum_{i=1}^{n}e_i\prod_{j\neq i}(x-\alpha_j)\right)/\prod_{i=1}^{n}(x-\alpha_i)\bmod g(x).$$

- Put $f(x)=\prod_{i=1}^{n}(x-\alpha_i)^{e_i}$ with $e_i\in\{0,1\}$, then $f'(x)=\sum_{i=1}^{n}e_i\prod_{j\neq i}(x-\alpha_j)^{e_j}$.
- Thus $s(x)\equiv f'(x)/f(x)\bmod g(x)$. We want to find f.
- Split $f(x)$ into odd and even terms: $f(x)=A^2(x)+xB^2(x)$ with $f'(x)=B^2(x)$.
- Thus

$$B^2(x)\equiv f(x)s(x)\equiv(A^2(x)+xB^2(x))s(x)\bmod g(x)$$
$$B^2(x)(x+1/s(x))\equiv A^2(x)\bmod g(x)$$

- Put $v(x)\equiv\sqrt{x+1/s(x)}\bmod g(x)$, then $A(x)\equiv B(x)v(x)\bmod g(x)$.
- Can compute $v(x)$ from $s(x)$.
- Use XGCD on v and g, stop when $\deg(A)\leq\lfloor t/2\rfloor,\deg(B)\leq\lfloor(t-1)/2\rfloor$ in

$$A(x)=B(x)v(x)+h(x)g(x).$$

Extended Patterson decoder

PALOMA uses extended Patterson decoder for reducible g, dealing with $\gcd(g, s) \neq 1$.

- Let $\tilde{s} = 1 + xs$ and $g_1 = \gcd(g, s), g_2 = \gcd(g, \tilde{s}), g_{12} = g/(g_1 g_2)$.
- Compute $\tilde{s}_2 = \tilde{s}/g_2$ and $s_1 = s/g_1$.
- Replace $u(x) = x + 1/s(x)$ by

$$u = g_1 \tilde{s}_2/(g_2 s_1) \bmod g_{12}$$

Extended Patterson decoder

PALOMA uses extended Patterson decoder for reducible g, dealing with $\gcd(g, s) \neq 1$.

- Let $\tilde{s} = 1 + xs$ and $g_1 = \gcd(g, s), g_2 = \gcd(g, \tilde{s}), g_{12} = g/(g_1 g_2)$.
- Compute $\tilde{s}_2 = \tilde{s}/g_2$ and $s_1 = s/g_1$.
- Replace $u(x) = x + 1/s(x)$ by

$$u = g_1 \tilde{s}_2/(g_2 s_1) \bmod g_{12}$$

- Deal with complication of computing $v = \sqrt{u} \bmod g_{12}$ for reducible g_{12}.
- Let half-gcd return A', B', put

$$f(x) = (A' g_2)^2 + x(B' g_1)^2.$$

Extended Patterson decoder

PALOMA uses extended Patterson decoder for reducible g, dealing with $\gcd(g, s) \neq 1$.

- Let $\tilde{s} = 1 + xs$ and $g_1 = \gcd(g, s), g_2 = \gcd(g, \tilde{s}), g_{12} = g/(g_1 g_2)$.
- Compute $\tilde{s}_2 = \tilde{s}/g_2$ and $s_1 = s/g_1$.
- Replace $u(x) = x + 1/s(x)$ by

$$u = g_1 \tilde{s}_2/(g_2 s_1) \bmod g_{12}$$

- Deal with complication of computing $v = \sqrt{u} \bmod g_{12}$ for reducible g_{12}.
- Let half-gcd return A', B', put

$$f(x) = (A' g_2)^2 + x(B' g_1)^2.$$

- Put $e = (00 \cdots 0)$.
- For j in $1, 2, \ldots, n$: if $f(\alpha_j) = 0$ put $e = e + e_j$.

Extended Patterson decoder

PALOMA uses extended Patterson decoder for reducible g, dealing with $\gcd(g, s) \neq 1$.

- Let $\tilde{s} = 1 + xs$ and $g_1 = \gcd(g, s), g_2 = \gcd(g, \tilde{s}), g_{12} = g/(g_1 g_2)$.
- Compute $\tilde{s}_2 = \tilde{s}/g_2$ and $s_1 = s/g_1$.
- Replace $u(x) = x + 1/s(x)$ by

$$u = g_1 \tilde{s}_2/(g_2 s_1) \bmod g_{12}$$

- Deal with complication of computing $v = \sqrt{u} \bmod g_{12}$ for reducible g_{12}.
- Let half-gcd return A', B', put

$$f(x) = (A' g_2)^2 + x(B' g_1)^2.$$

- Put $e = (00 \cdots 0)$.
- For j in $1, 2, \ldots, n$: if $f(\alpha_j) = 0$ put $e = e + e_j$.
- Random polynomial has 0 roots in L with probability $\approx (1 - 1/q)^n$.