

McEliece scheme with quasi-dyadic separable binary Goppa codes

Gerhard Hoffmann

CASSED

15th April 2010

Outline of Part I: McEliece cryptosystem

- 1 Before we begin
- 2 McEliece cryptosystem (1978)
 - The ingredients
 - The recipe
 - The Dilemma

Outline of Part I: McEliece cryptosystem

- 1 Before we begin
- 2 McEliece cryptosystem (1978)
 - The ingredients
 - The recipe
 - The Dilemma

Outline of Part I: McEliece cryptosystem

- 1 Before we begin
- 2 McEliece cryptosystem (1978)
 - The ingredients
 - The recipe
 - The Dilemma

Outline of Part I: McEliece cryptosystem

- 1 Before we begin
- 2 McEliece cryptosystem (1978)
 - The ingredients
 - The recipe
 - The Dilemma

Outline of Part I: McEliece cryptosystem

- 1 Before we begin
- 2 McEliece cryptosystem (1978)
 - The ingredients
 - The recipe
 - The Dilemma

Outline of Part II: Asking for trouble

- Don't make the adversary happy
- Compact keys are only useful in compact form
- What about performance?
- Mission impossible?

Outline of Part II: Asking for trouble

- Don't make the adversary happy
- Compact keys are only useful in compact form
- What about performance?
- Mission impossible?

Outline of Part II: Asking for trouble

- Don't make the adversary happy
- Compact keys are only useful in compact form
- What about performance?
- Mission impossible?

Outline of Part II: Asking for trouble

- Don't make the adversary happy
- Compact keys are only useful in compact form
- What about performance?
- Mission impossible?

Outline of Part III: Goppa codes

3 Goppa codes

- Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again

4 Irreducible binary Goppa codes

5 Separable binary Goppa codes

- A striking property
- It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
- H is a Cauchy matrix!
- Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part III: Goppa codes

- 3 Goppa codes
 - Goppa codes: what was that again?
 - Generalized Reed-Solomon Codes
 - Alternant Codes
 - Classical Goppa codes
 - Syndrome function
 - Some remarks
 - The canonical parity check matrix for GRS again
- 4 Irreducible binary Goppa codes
- 5 Separable binary Goppa codes
 - A striking property
 - It's a kind of magic
 - Parity check matrix for separable binary Goppa codes
 - Reminder: Cauchy matrices
 - H is a Cauchy matrix!
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part IV: The dyadic and quasi-dyadic world

- 6 The (quasi-) dyadic world
 - Dyadic matrices
 - Sylvester-Hadamard matrices
 - Sylvester-Hadamard matrices: examples
 - Some properties
 - Dyadic convolution of two vectors
 - Dyadic convolution: what for?
 - The Fast Walsh-Hadamard transform (FWHT)
 - FWHT visualized
 - Let's summarize

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part V: QD

7 QD

- Can we have the cake and eat it, too?
- Putting the pieces together
- Oh no: $\text{char}(\mathbb{F}) = 2!$
 - The trick
 - The trick: an example
- Quasi-dyadic matrices
 - QD: Assumptions
 - QD: Select
 - QD: Select and permute
 - QD: Select, permute and scale
 - QD: Co-Trace
 - QD: Co-Trace continued
 - QD: Co-Trace continued
 - QD: Final step
- QD: Some remarks

Outline of Part VI: Decoding Goppa codes as alternant codes

8 Decoding Goppa codes as alternant codes

- Some definitions
- The syndrome
- The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
- Let's summarize
- Separable binary Goppa codes again
- The magic again
- The strategy again
- Adapting the decoder
- Under attack
- Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

8 Decoding Goppa codes as alternant codes

- **Some definitions**
- The syndrome
- The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
- Let's summarize
- Separable binary Goppa codes again
- The magic again
- The strategy again
- Adapting the decoder
- Under attack
- Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Decoding Goppa codes as alternant codes

- 8 Decoding Goppa codes as alternant codes
 - Some definitions
 - The syndrome
 - The key equation
 - Reminder: Extended Euclidian Algorithm
 - Euclidian Algorithm continued
 - Applying the Euclidian Algorithm
 - Let's summarize
 - Separable binary Goppa codes again
 - The magic again
 - The strategy again
 - Adapting the decoder
 - Under attack
 - Warning

Outline of Part VI: Flexible QD

9 Flexible QD

- FQD: Step 1
- FQD: Step 2
- FQD: Step 3
- FQD: Step 4
- FQD: Step 5
- FQD: Step 6
- FQD: Sketch of proof
- FQD: Remarks

Outline of Part VI: Flexible QD

- 9 Flexible QD
 - FQD: Step 1
 - FQD: Step 2
 - FQD: Step 3
 - FQD: Step 4
 - FQD: Step 5
 - FQD: Step 6
 - FQD: Sketch of proof
 - FQD: Remarks

Outline of Part VI: Flexible QD

- 9 Flexible QD
 - FQD: Step 1
 - FQD: Step 2
 - FQD: Step 3
 - FQD: Step 4
 - FQD: Step 5
 - FQD: Step 6
 - FQD: Sketch of proof
 - FQD: Remarks

Outline of Part VI: Flexible QD

- 9 Flexible QD
 - FQD: Step 1
 - FQD: Step 2
 - FQD: Step 3
 - FQD: Step 4
 - FQD: Step 5
 - FQD: Step 6
 - FQD: Sketch of proof
 - FQD: Remarks

Outline of Part VI: Flexible QD

- 9 Flexible QD
 - FQD: Step 1
 - FQD: Step 2
 - FQD: Step 3
 - FQD: Step 4
 - FQD: Step 5
 - FQD: Step 6
 - FQD: Sketch of proof
 - FQD: Remarks

Outline of Part VI: Flexible QD

9 Flexible QD

- FQD: Step 1
- FQD: Step 2
- FQD: Step 3
- FQD: Step 4
- FQD: Step 5
- FQD: Step 6
- FQD: Sketch of proof
- FQD: Remarks

Outline of Part VI: Flexible QD

9

Flexible QD

- FQD: Step 1
- FQD: Step 2
- FQD: Step 3
- FQD: Step 4
- FQD: Step 5
- FQD: Step 6
- FQD: Sketch of proof
- FQD: Remarks

Outline of Part VI: Flexible QD

- 9 Flexible QD
 - FQD: Step 1
 - FQD: Step 2
 - FQD: Step 3
 - FQD: Step 4
 - FQD: Step 5
 - FQD: Step 6
 - FQD: Sketch of proof
 - FQD: Remarks

Outline of Part VI: Flexible QD

- 9 Flexible QD
 - FQD: Step 1
 - FQD: Step 2
 - FQD: Step 3
 - FQD: Step 4
 - FQD: Step 5
 - FQD: Step 6
 - FQD: Sketch of proof
 - FQD: Remarks

Outline of Part VII: HyMES

10 HyMES

- HyMES with QD/FQD

Outline of Part VII: HyMES

- 10 HyMES
 - HyMES with QD/FQD

Outline of Part IX: Anything else?

- Summary
- Mission accomplished?
- Questions?
- Material used and further reading

Outline of Part IX: Anything else?

- Summary
- Mission accomplished?
- Questions?
- Material used and further reading

Outline of Part IX: Anything else?

- Summary
- Mission accomplished?
- Questions?
- Material used and further reading

Outline of Part IX: Anything else?

- Summary
- Mission accomplished?
- Questions?
- Material used and further reading

Part I

McEliece cryptosystem

Before we begin

- ▶ There are quite a number of slides.
- ▶ A lot of them are only there for your convenience as a quick reminder.

What you should remember:

- ▶ McEliece with binary separable Goppa codes.
- ▶ Such Goppa codes and quasi-dyadic matrices are related.
- ▶ Together they provide short McEliece keys.

McEliece cryptosystem (1978)

- ▶ First cryptosystem using error correcting codes as trapdoor.
- ▶ Trapdoor: efficient error correcting algorithm for Goppa codes.
- ▶ Since 1978 unbroken in its original version.


How it works: The ingredients

- ▶ $n, t \in \mathbb{N}$, where $t \ll n$
 - ▶ Binary (n, k) code \mathcal{G} with minimum distance $d \geq 2t + 1$
 - ▶ $G' : k \times n$ generator matrix of \mathcal{G}
 - ▶ $S : k \times k$ random binary non-singular matrix
 - ▶ $P : n \times n$ random permutation matrix
-
- ▶ **Public key:** (G, t) with $G := SG'P$
 - ▶ **Private key:** $(S, \mathcal{D}_{\mathcal{G}}, P)$, where $\mathcal{D}_{\mathcal{G}}$ decoder for \mathcal{G}

How it works: The recipe

Apply P^{-1}

Apply \mathcal{D}_G


$$\mathbf{c}P^{-1} = (\mathbf{m}S)G' \oplus \mathbf{z}P^{-1}$$

$$\mathbf{m}SG' = \mathcal{D}_G(\mathbf{c}P^{-1})$$

$$\mathbf{m} = (\mathbf{m}SG')_J (G'_{.J})^{-1} S^{-1}$$

Use $(G'_{.J})^{-1}$ and S^{-1}

How it works: The recipe

Apply P^{-1}

Apply \mathcal{D}_G

$$\mathbf{c}P^{-1} = (\mathbf{m}S)G' \oplus \mathbf{z}P^{-1}$$

$$\mathbf{m}SG' = \mathcal{D}_G(\mathbf{c}P^{-1})$$

$$\mathbf{m} = (\mathbf{m}SG')_J (G'_{\cdot J})^{-1} S^{-1}$$

Use $(G'_{\cdot J})^{-1}$ and S^{-1}

How it works: The recipe

Apply P^{-1}

Apply \mathcal{D}_G

$$\mathbf{c}P^{-1} = (\mathbf{m}S)G' \oplus \mathbf{z}P^{-1}$$

$$\mathbf{m}SG' = \mathcal{D}_G(\mathbf{c}P^{-1})$$

$$\mathbf{m} = (\mathbf{m}SG')_J (G'_{.J})^{-1} S^{-1}$$

Use $(G'_{.J})^{-1}$ and S^{-1}

The Dilemma

Although the McEliece scheme based on Goppa codes turns out to be quite efficient in terms of speed, and is unbroken since its invention, one major drawback still remains:

Downside

The sizes of the public matrices are too big, typically around hundreds of KB or even some MB.

Goal

Reduce this size, but don't sacrifice speed and security.

Part II

Asking for trouble

Don't make the adversary happy

One way to reduce the key size is to put more structure into the cryptosystem. But we have to be very careful ...

More structure means more information!

If it's done the wrong way ...

Good news for the adversary!

Compact keys are only useful in compact form

Compact keys are definitely what we want, but ...

**If we have to unfold the keys to become operational,
we would gain not that much!**

What about performance?

Compact keys are definitely what we want, but ...

Their processing speed shouldn't be much worse than RSA.

Mission impossible?

To solve those problems all at once seems to be really tough.

So let's try!

Part III

Goppa codes

Goppa codes have shown their quality

Since its invention, the original McEliece scheme using general Goppa codes is unbroken.

McEliece with other codes has failed.

Therefore, let's try Goppa codes as starting place.

Goppa codes: what was that again?

Goppa codes \subsetneq Alternant codes \subsetneq GRS codes

General assumptions:

- ▶ $n \leq q \in \mathbb{N}_+$
- ▶ $\mathbf{L} = (L_0, \dots, L_{n-1}) \in \mathbb{F}_q^n$, pairwise distinct
- ▶ $h(X) := \prod_{i \in n} (X - L_i) \in \mathbb{F}_q[X]$
- ▶ $g(X) \in \mathbb{F}_q[X]$, $g(L_i) \neq 0$ for all i

Generalized Reed-Solomon Codes

$GRS_k(L, g) :=$

$$\left\{ \mathbf{c} \in \mathbb{F}_q^n : \exists f(X) \in \mathbb{F}_q[X]_{<k} : \sum_{i \in n} c_i \prod_{j \neq i} (X - L_j) \equiv fg \pmod{h} \right\}$$

for $1 \leq k \leq n$.

Alternant Codes: restricting GRS to subfields

- ▶ $m \in \mathbb{N}_+$, some extension degree
- ▶ $\mathbf{L} = (L_0, \dots, L_{n-1}) \in \mathbb{F}_{q^m}^n$, pairwise distinct
- ▶ $g(X) \in \mathbb{F}_{q^m}[X]$, $g(L_i) \neq 0$ for all i

$$\text{Alt}_{k,q}(L, g) := \text{GRS}_k(L, g) \cap \mathbb{F}_q^n$$

Classical Goppa codes: special alternant codes

- ▶ Alternant code as before, but:
- ▶ $\boxed{\deg(g(X)) := n - k}$

$$\begin{aligned}
 GO_q(L, g) &:= \left\{ \mathbf{c} \in \mathbb{F}_q^n : \sum_{i \in n} c_i \prod_{j \neq i} (X - L_j) \equiv 0 \pmod{g} \right\} \\
 &= \left\{ \mathbf{c} \in \mathbb{F}_q^n : \sum_{i \in n} \frac{c_i}{(X - L_j)} \equiv 0 \pmod{g} \right\}
 \end{aligned}$$

- ▶ $g(X)$ is called *Goppa polynomial* and,
- ▶ $\mathbf{L} = (L_0, \dots, L_{n-1})$ its *support*.

Syndrome function

Assume $q = p^m$ for some prime p (usually $p = 2$) and $t = \deg(g(X))$.

Syndrome function

$$S : \mathbb{F}_p^n \longrightarrow \mathbb{F}_q[X]/(g)$$

$$\mathbf{c} \mapsto \sum_{i \in n} \frac{c_i}{X - L_i} \pmod{g}$$

We see that

$$GO_p(L, g) = \{\mathbf{c} \in \mathbb{F}_p^n : S(\mathbf{c}) \equiv 0 \pmod{g}\}$$

Some remarks

Theorem

The Goppa code $GO_p(L, g)$, where $\deg(g(X)) = t < n$

- ▶ has length $n = \#L = \#\{L_0, \dots, L_{n-1}\}$
- ▶ dimension k satisfying $n - mt \leq k \leq n - t$
- ▶ and minimum distance $d \geq t + 1$.

Note that the last property holds for Goppa codes in general.
As we will see, the estimation can be improved in certain cases.

The canonical parity check matrix for GRS again

The parity check matrix H for a GRS code, which restricts to a Goppa code, has the following form:

H for GRS code, which restricts to Goppa code

$$H = VD = \begin{bmatrix} 1 & 1 & \dots & 1 \\ L_0 & L_1 & \dots & L_{n-1} \\ L_0^2 & L_1^2 & \dots & L_{n-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ L_0^{t-1} & L_1^{t-1} & \dots & L_{n-1}^{t-1} \end{bmatrix} \begin{bmatrix} g(L_0)^{-1} & 0 & \dots & 0 \\ 0 & g(L_1)^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & g(L_{n-1})^{-1} \end{bmatrix}$$

Irreducible binary Goppa codes

Usual case: $g(X) \in \mathbb{F}_q[X]$, irreducible, $q = 2^m$.

Because:

- ▶ We have $d \geq 2t + 1$ in this case, so we can correct up to t errors.
- ▶ There are efficient decoders for these **irreducible binary** Goppa codes (Patterson).
- ▶ For general Goppa codes, we have no decoders correcting more than $t/2$ errors.

Separable binary Goppa codes

- ▶ Again: $g(X) \in \mathbb{F}_q[X]$, $q = 2^m$
- ▶ But now $g(X)$ is assumed to have no multiple roots:

$$g(X) = (X - z_0)\dots(X - z_{t-1}) \in \mathbb{F}_{2^m}[X]$$

- ▶ A binary Goppa code whose $g(X)$ has no multiple roots is called a *separable* Goppa code.

A striking property

It turns out that for separable Goppa codes we have:

$$GO_2(L, g) = GO_2(L, g^2)$$

- ▶ That means: $g(X)$ and $g(X)^2$ generate the same space!

It's a kind of magic

Strategy:

Public generator matrix G is based **on** $g(X)$.

Decoder \mathcal{D}_G is based **on** $g(X)^2$.

Remember:

- ▶ Then \mathcal{D}_G can correct up to $2t$ errors.
- ▶ Breaking a system based on G resp. $g(X)$ is not enough!

We will see that in more detail soon.

Parity check matrix for separable binary Goppa codes

Theorem (Tzeng, Zimmermann, 1975)

- ▶ $g(X) = (X - z_0)\dots(X - z_{t-1}) \in \mathbb{F}_{q^m}[X]$
- ▶ without multiple zeros
- ▶ $\mathbf{L} = \{L_0, \dots, L_{n-1} \in \mathbb{F}_{q^m} : g(L_i) \neq 0\}$
- ▶ admits a parity check matrix of the following form:

$$H(z, L) = (H_{ij}) = \begin{bmatrix} \frac{1}{z_0 - L_0} & \frac{1}{z_0 - L_1} & \cdots & \frac{1}{z_0 - L_{n-1}} \\ \frac{1}{z_1 - L_0} & \frac{1}{z_1 - L_1} & \cdots & \frac{1}{z_1 - L_{n-1}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{z_{t-1} - L_0} & \frac{1}{z_{t-1} - L_1} & \cdots & \frac{1}{z_{t-1} - L_{n-1}} \end{bmatrix} \in \mathbb{F}_{q^m}^{t \times n}$$

Reminder: Cauchy matrices

- ▶ $\mathbf{z} = (z_0, \dots, z_{t-1}) \in \mathbb{F}^t$, pairwise distinct
- ▶ $\mathbf{L} = (L_0, \dots, L_{n-1}) \in \mathbb{F}^n$, pairwise distinct
- ▶ $\mathbf{z} \cap \mathbf{L} = \emptyset$

$$C(\mathbf{z}, \mathbf{L}) = (C_{ij}) = \begin{bmatrix} \frac{1}{z_0 - L_0} & \frac{1}{z_0 - L_1} & \cdots & \frac{1}{z_0 - L_{n-1}} \\ \frac{1}{z_1 - L_0} & \frac{1}{z_1 - L_1} & \cdots & \frac{1}{z_1 - L_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{z_{t-1} - L_0} & \frac{1}{z_{t-1} - L_1} & \cdots & \frac{1}{z_{t-1} - L_{n-1}} \end{bmatrix} \in \mathbb{F}^{t \times n}$$

H is a Cauchy matrix!

A binary separable Goppa code allows a compact representation, but:

- ▶ Code structure is apparent.
- ▶ Usual hiding tricks destroy the Cauchy structure.
- ▶ Compact representation doesn't seem to be suitable: What about mG with insufficient memory ?

Let's summarize

- ▶ Our goal: reduce the public key sizes of McEliece.
- ▶ We tried with **binary separable Goppa codes**.
- ▶ They have a striking property: $GO_2(L, g) = GO_2(L, g^2)$
- ▶ They admit a parity check matrix in Cauchy form, but it's not obvious how to perform mG efficiently.
- ▶ Looks like a dead end, but we will see how to use it soon.

Part IV

The (quasi-) dyadic world

The (quasi-) dyadic world

- ▶ Dyadic matrices are highly structured.
- ▶ mG highly efficient in case G is dyadic.

- ▶ We will see below how to connect dyadic to Cauchy matrices.

Dyadic matrices

Signature $\mathbf{h} := (h_0, \dots, h_{n-1}) = (A, B, C, D, E, F, G, H)$

Highly regular, always $2^n \times 2^n$

$$\Delta(\mathbf{h}) := (h_{i \oplus j}) := \begin{array}{cccccccc} A & B & C & D & E & F & G & H \\ B & A & D & C & F & E & H & G \\ C & D & A & B & G & H & E & F \\ D & C & B & A & H & G & F & E \\ E & F & G & H & A & B & C & D \\ F & E & H & G & B & A & D & C \\ G & H & E & F & C & D & A & B \\ H & G & F & E & D & C & B & A \end{array}$$

Dyadic matrices are symmetric, even point-symmetric.

Sylvester-Hadamard matrices

Let \mathbb{F} be a field with $\text{char}(\mathbb{F}) \neq 2$, $r = 2^k$.

- ▶ The Sylvester-Hadamard matrix $\mathbf{H}_k \in \mathbb{F}^{r \times r}$ is recursively defined as:

$$\mathbf{H}_0 = [1]$$

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{H}_{k-1} \\ \mathbf{H}_{k-1} & -\mathbf{H}_{k-1} \end{bmatrix}, k > 0$$

$$\mathbf{H}_0^{-1} = [1]$$

$$\mathbf{H}_k^{-1} = \frac{1}{2} \begin{bmatrix} \mathbf{H}_{k-1}^{-1} & \mathbf{H}_{k-1}^{-1} \\ \mathbf{H}_{k-1}^{-1} & -\mathbf{H}_{k-1}^{-1} \end{bmatrix}, k > 0$$

Sylvester-Hadamard matrices: examples

To give some examples:

$$\mathbf{H}_0 = [+1]$$

$$\mathbf{H}_1 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$\mathbf{H}_2 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

etc.

Some properties

Let \mathbb{F} be a field with $\text{char}(\mathbb{F}) \neq 2$.

Let $\mathbf{h} \in \mathbb{F}^r$, $\mathbf{M} \in \mathbb{F}^{r \times r}$, dyadic, where $r = 2^k$.

▶ $\mathbf{H}_k^{-1} \mathbf{M} \mathbf{H}_k$ is diagonal.

▶ $\mathbf{H}_k^{-1} \Delta(\mathbf{h}) \mathbf{H}_k = \text{diag}(\mathbf{h} \mathbf{H}_k)$

Dyadic convolution of two vectors

Let $\mathbf{u}, \mathbf{v} \in \mathbb{F}^r$, $r = 2^k$, $\text{char}(\mathbb{F}) \neq 2$. The (unique) $\mathbf{w} \in \mathbb{F}^r$ with

$$\Delta(\mathbf{u})\Delta(\mathbf{v}) = \Delta(\mathbf{w})$$

is called the **dyadic convolution** of \mathbf{u} and \mathbf{v} .

$$\begin{aligned} \text{diag}(\mathbf{u}\mathbf{H}_k)\text{diag}(\mathbf{v}\mathbf{H}_k) &= (\mathbf{H}_k^{-1}\Delta(\mathbf{u})\mathbf{H}_k)(\mathbf{H}_k^{-1}\Delta(\mathbf{v})\mathbf{H}_k) \\ &= \mathbf{H}_k^{-1}\Delta(\mathbf{w})\mathbf{H}_k \\ &= \text{diag}(\underbrace{\mathbf{w}\mathbf{H}_k}_{=:\mathbf{z}}) \end{aligned}$$

$$\implies \mathbf{w} = \mathbf{z}\mathbf{H}_k^{-1} = 2^{-k}\mathbf{z}\mathbf{H}_k$$

Dyadic convolution: what for?

Let \mathbf{m} a message, $G = \Delta(\mathbf{g})$ a dyadic generator matrix.

- ▶ Compute $\hat{\mathbf{m}} \leftarrow \mathbf{m}\mathbf{H}_k$.
- ▶ Compute $\hat{\mathbf{g}} \leftarrow \mathbf{g}\mathbf{H}_k$.
- ▶ Compute $\hat{\mathbf{c}} \leftarrow \hat{\mathbf{m}} \cdot * \hat{\mathbf{g}}$ (*Hadamard product* : $\hat{c}_i = \hat{m}_i \hat{g}_i \forall i$)
- ▶ Compute $\mathbf{c} \leftarrow \hat{\mathbf{c}}\mathbf{H}_k$.
- ▶ Finally $\mathbf{c} \leftarrow 2^{-k}\mathbf{c}$

For mG only the first row of G is needed.

$\mathbf{m}\mathbf{H}_k$ can be done efficiently by the so-called *Fast Walsh-Hadamard transform* (FWHT).

The Fast Walsh-Hadamard transform (FWHT)

Let $r = 2^k$ with $k = 3$.

$$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{pmatrix} \underbrace{\begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix}}_{\mathbf{H}_3} = \begin{bmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mathbf{H}_2 + \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \mathbf{H}_2 \\ \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mathbf{H}_2 - \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \mathbf{H}_2 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} x_{1,0} \\ x_{1,1} \\ x_{1,2} \\ x_{1,3} \end{pmatrix} \mathbf{H}_2 \\ \begin{pmatrix} x_{1,4} \\ x_{1,5} \\ x_{1,6} \\ x_{1,7} \end{pmatrix} \mathbf{H}_2 \end{bmatrix}$$

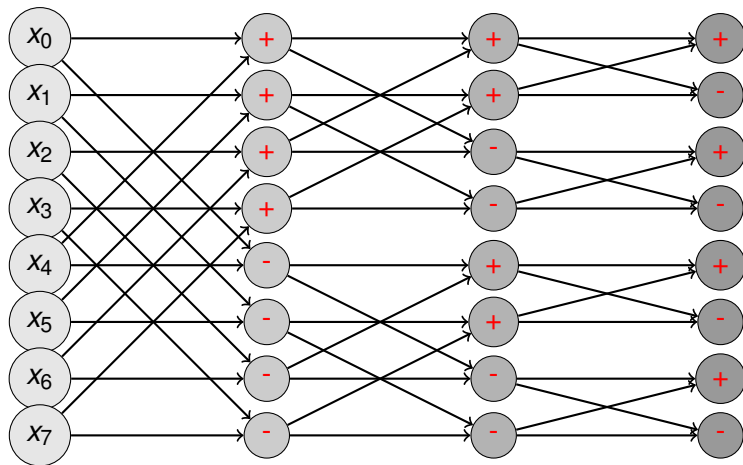
where

$$x_{1,i} = x_i + x_{i+4}$$

$$x_{1,i+4} = x_i - x_{i+4}$$

for $i \in \{0, 1, 2, 3\}$. Recurse. Complexity: $\mathcal{O}(r \log(r))$.

FWHT visualized



Let's summarize

Dyadic matrices: The pros

- ▶ Very compact representation: the first row (= signature) is enough.
- ▶ Efficient mG even in compact representation.

Dyadic matrices: The cons

- ▶ We still don't know how to base a crypto system on them.
- ▶ We always made the assumption $\text{char}(\mathbb{F}) \neq 2$. But in cryptography, we would like to have $\text{char}(\mathbb{F}) = 2$.

Part V

QD

QD

By Rafael Misoczki and Paulo S.L.M. Barreto.

Idea

Connect binary separable Goppa codes and dyadic matrices.

That means: look for dyadic Cauchy matrices.

Can we have the cake and eat it, too?

Theorem (P. Barreto)

A dyadic Cauchy matrix is only possible over fields \mathbb{F}_q , $q = 2^m$. In this case, $H = (H_{ij})$ can be generated by any suitable $h \in \mathbb{F}_q^n$ satisfying

$$\begin{aligned} \frac{1}{h_{i \oplus j}} &= \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0} \\ z_i &= \frac{1}{h_i} + \omega \\ L_j &= \frac{1}{h_j} + \frac{1}{h_0} + \omega \\ H_{ij} &= h_{i \oplus j} = \frac{1}{z_i \oplus L_j} \end{aligned}$$

and $\omega \in \mathbb{F}_q$ just arbitrary.

Putting the pieces together

Solving the equations above gives us:

- ▶ Separable Goppa codes for security.
- ▶ Compact McEliece keys.
- ▶ Efficiency via FWHT.

Any drawbacks? **Yes!**

- ▶ The theorem above requires $\text{char}(\mathbb{F}) = 2$!
- ▶ Cryptosystems cannot securely defined by parity-check matrices in Cauchy form.

We will see now how to address these points.

Oh no: $\text{char}(\mathbb{F}) = 2!$

- ▶ FWHT & dyadic convolution $\Rightarrow \text{char}(\mathbb{F}) \neq 2$
- ▶ Dyadic Cauchy matrix $\Rightarrow \text{char}(\mathbb{F}) = 2$

Trick by Daniel Bernstein:

- ▶ Just lift FWHT to characteristic 0

Remember:

$$\mathbb{F}_{2^m} \simeq (\mathbb{Z}/2\mathbb{Z})[X]/P(X)$$

for some irreducible $P(X)$ of degree m .

The trick

$$\blacktriangleright \mathbb{F}_{2^m} \simeq (\mathbb{Z}/2\mathbb{Z})[X]/P(X) \rightsquigarrow \mathbb{Z}[X]$$

In our case

$$\blacktriangleright \mathbb{F}_2 \simeq (\mathbb{Z}/2\mathbb{Z}) \rightsquigarrow \mathbb{Z}$$

Do FWHT now in \mathbb{Z} and reduce the end result modulo 2.

- ▶ Remember multiplying two binary matrices for the first time? Chances are good you applied the trick.
- ▶ Drawback of the trick: arithmetic in $\mathbb{Z} \Rightarrow$ more memory.

The trick: an example

Usually in \mathbb{F}_2 with only *AND* and *XOR*:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Viewed in \mathbb{Z} :

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 5 & 4 & 4 & 2 & 2 \\ 3 & 3 & 2 & 0 & 1 \\ 4 & 3 & 3 & 2 & 2 \\ 3 & 3 & 3 & 1 & 0 \\ 3 & 3 & 2 & 1 & 1 \end{bmatrix}}_{\text{reduce mod 2}}$$

Quasi-dyadic matrices

- ▶ Fully dyadic parity-check matrices not an option.
- ▶ Next better level: quasi-dyadic matrices.

Still quite regular

Not necessarily square, consisting of dyadic submatrices of the same dimensions: quasi-dyadic.

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>
<i>B</i>	<i>A</i>	<i>D</i>	<i>C</i>	<i>F</i>	<i>E</i>	<i>H</i>	<i>G</i>
<i>I</i>	<i>J</i>	<i>L</i>	<i>M</i>	<i>N</i>	<i>P</i>	<i>Q</i>	<i>R</i>
<i>J</i>	<i>I</i>	<i>M</i>	<i>L</i>	<i>P</i>	<i>N</i>	<i>R</i>	<i>Q</i>

QD: Assumptions

Choose the following data:

- ▶ $p = 2^s$ for some s
- ▶ $q = p^d = 2^m$ for some d with $m = ds$
- ▶ A code length n
- ▶ A design number of correctable errors t with $n = lt$ for some $l > d$

QD: Select

- ▶ Let $N = kt$ for some k
- ▶ $N \gg n$
- ▶ $N \leq q/2$

- ▶ QD generates a **fully dyadic** $N \times N$ parity-check matrix H for a binary Goppa code $GO_q(L, g)$
- ▶ QD selects a submatrix $\hat{H} \in \mathbb{F}_q^{t \times N}$

QD: Select and permute

$$\hat{H} := [B_0 \mid \dots \mid B_{N/t-1}] \in \mathbb{F}_q^{t \times N}$$

- ▶ Each $B_i \in \mathbb{F}_q^{t \times t}$ and dyadic
- ▶ Select randomly l distinct blocks $B_{i_0}, \dots, B_{i_{l-1}} \in \mathbb{F}_q^{t \times t}$
- ▶ Select l dyadic permutations $\Pi^{j_0}, \dots, \Pi^{j_{l-1}} \in \mathbb{F}_2^{t \times t}$

$$\hat{H}' := [B_{i_0} \Pi^{j_0} \mid \dots \mid B_{i_{l-1}} \Pi^{j_{l-1}}] \in (\mathbb{F}_q^{t \times t})^l$$

QD: Select, permute and scale

- ▶ Aside: dyadic permutation

A dyadic matrix $\Pi^i \in \Delta(\{0, 1\}^n)$, whose signature is the i th row of the identity matrix.

- ▶ Build $\hat{H}'\Sigma$ and co-trace it.
- ▶ Co-tracing $\hat{H}'\Sigma$ means to map it to binary.
- ▶ But note: $\hat{H}'\Sigma$ is quasi-dyadic, and that structure shall be maintained.

QD: Co-Trace

Example:

- ▶ Let $u_0 = (1, 1, 0, 1)$, $u_1 = (0, 1, 0, 1) \in \mathbb{F}_2^4$
- ▶ $T = \begin{bmatrix} u_0 & u_1 \\ u_1 & u_0 \end{bmatrix} \in \mathbb{F}_2^{2 \times 2}$ and dyadic

QD: Co-Trace continued

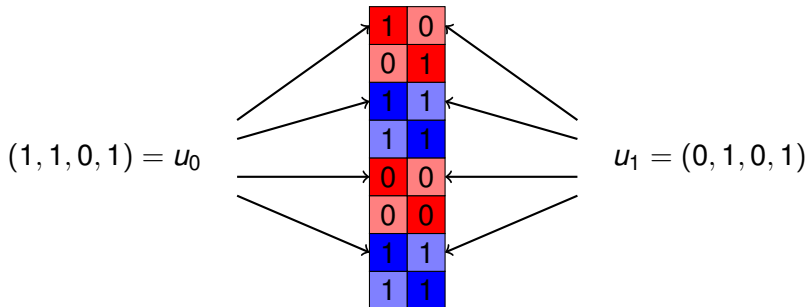
- ▶ Usual trace construction would be wrong:

$$T = \begin{bmatrix} u_0 & u_1 \\ u_1 & u_0 \end{bmatrix} = \begin{bmatrix} (1, 1, 0, 1) & (0, 1, 0, 1) \\ (0, 1, 0, 1) & (1, 1, 0, 1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

- ▶ Quasi-dyadic structure is lost.

QD: Co-Trace continued

- Interleave instead:



QD: Final step

- ▶ Let $H' := T'_d(\hat{H}' \Sigma)$ denote the co-traced $\hat{H}' \Sigma$
- ▶ $H' \in (\mathbb{F}_p^{t \times t})^{d \times l}$
- ▶ Finally, let H denote H' in systematic form.

- ▶ The parity-check matrix H defines code of length n and dimension $k = n - dt$ over \mathbb{F}_p
- ▶ It is still quasi-dyadic with $t \times t$ dyadic submatrices.

- ▶ It can be stored in an area a factor t smaller than a general matrix.

QD: Some remarks

The QD construction has some drawbacks:

- ▶ Complicated.
- ▶ Highly dependent on chosen parameters.
- ▶ $N \gg n$

All those points will be addressed later with the FQD construction.

Part VI

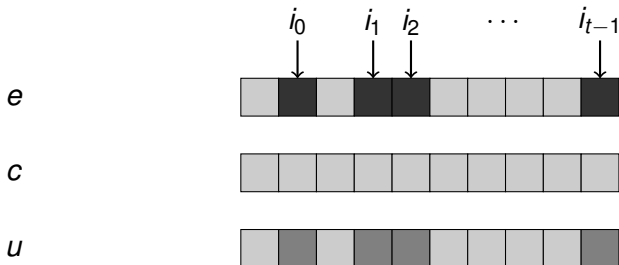
Decoding Goppa codes as alternant codes

Decoding Goppa codes as alternant codes

- ▶ Goppa codes are special alternant codes.
- ▶ Therefore, alternant decoders can decode Goppa codes.

Decoding Goppa codes: The setup

- ▶ $r = \deg(g(X))$, even
- ▶ received codeword: $u \in \mathbb{F}_2^n$
- ▶ $u = c + e$
- ▶ error positions: $1 \leq i_0, \dots, i_{t-1} \leq n$, where $t \leq r/2$



Some definitions

Error locator polynomial:

$$\sigma(X) = \prod_{j=0}^{t-1} (1 - L_j X) \in \mathbb{F}_{2^m}[X]$$

► Property: $\sigma(L_j) = 0 \iff e_{j_i} \neq 0$

Error evaluator polynomial:

$$\omega(X) = \sum_{k=0}^{t-1} g(L_{i_k})^{-1} \prod_{\substack{j=0 \\ j \neq k}}^{t-1} (1 - L_j) \in \mathbb{F}_{2^m}[X]$$

The syndrome

- ▶ Find the L_i , which are the roots of $\sigma(X)$.
- ▶ Because the L_i are pairwise distinct, they give the error locations.

$$uH^T = eH^T = [0 \dots e_{i_0} \dots e_{i_{t-1}} \dots 0] H^T = [S_0 \dots S_{r-1}]$$

Syndrome

$$S(X) := \sum_{k=0}^{r-1} S_k X^k$$

The key equation

- ▶ $\sigma(X)$ and $\omega(X)$ are related by the so-called key equation:

Theorem

$$\omega(X) = \sigma(X)S(X) \pmod{X^r}$$

Solving the key equation gives $\sigma(X)$. It is usually done by the extended Euclidian algorithm.

Reminder: Extended Euclidian Algorithm

- ▶ Let $f(X), h(X) \in \mathbb{F}[X]$ with $\deg(h(X)) \leq \deg(f(X))$
- ▶ Let $r_{-1}(X) := f(X)$ and $r_0 := h(X)$

$$r_{-1}(X) = q_1(X)r_0(X) + r_1(X) \quad \deg(r_1) < \deg(r_0)$$

$$\dots \quad \vdots \quad \vdots$$

$$r_{k-2}(X) = q_k(X)r_{k-1}(X) + r_k(X) \quad \deg(r_k) < \deg(r_{k-1})$$

$$\dots \quad \vdots \quad \vdots$$

$$r_{s-1}(X) = q_{s+1}(X)r_s(X)$$

Euclidian Algorithm continued

Theorem

$$r_k(X) = a_k(X)f(X) + b_k(X)h(X) \quad (k \geq -1)$$

$$a_k(X) = -q_k(X)a_{k-1}(X) + a_{k-2}(X) \quad (k \geq 1)$$

$$b_k(X) = -q_k(X)b_{k-1}(X) + b_{k-2}(X) \quad (k \geq 1)$$

where

$$a_{-1}(X) = 1, \quad a_0(X) = 0$$

$$b_{-1}(X) = 1, \quad b_0(X) = 0$$

Applying the Euclidian Algorithm

Apply the algorithm until $\deg(r_k) < r/2$ and $\deg(r_{k-1}) \geq r/2$.

Theorem

- ▶ $\sigma(X) = b_k(0)^{-1} b_k(X)$
- ▶ $\omega(X) = b_k(0)^{-1} r_k(X)$

Let's summarize

- ▶ Step 1: Find the syndrome polynomial $S(X) = \sum_{k=0}^{r-1} S_k X^k$
- ▶ Step 2: Solve the key equation $\omega(X) = \sigma(X)S(X) \pmod{X^r}$
- ▶ Step 3: Find the L_{ij} with $\sigma(L_{ij}) = 0$
- ▶ Step 4: Correct the received vector \mathbf{u}

Separable binary Goppa codes again

- ▶ $GO_2(L, g) = GO_2(L, g^2)$
- ▶ The canonical parity-check matrix H_0 based on $g(X)^2$ is this:

$$H_0 = VD = \begin{bmatrix} 1 & 1 & \dots & 1 \\ L_0 & L_1 & \dots & L_{n-1} \\ L_0^2 & L_1^2 & \dots & L_{n-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ L_0^{t-1} & L_1^{t-1} & \dots & L_{n-1}^{t-1} \\ L_0^t & L_1^t & \dots & L_{n-1}^t \\ \vdots & \vdots & \ddots & \vdots \\ L_0^{2t-1} & L_1^{2t-1} & \dots & L_{n-1}^{2t-1} \end{bmatrix} \begin{bmatrix} g(L_0)^{-2} & 0 & \dots & 0 \\ 0 & g(L_1)^{-2} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & g(L_{n-1})^{-2} \end{bmatrix}$$

The magic again

- ▶ $GO_2(L, g) = GO_2(L, g^2)$

That means:

- ▶ H_0 is related to **any** other parity-check matrix H by an equation of the form:

$$H_0 = SH$$

Note:

- ▶ An alternant decoder based on H_0 can correct up to $2 * \mathbf{deg}(\mathbf{g}(\mathbf{X}))$ errors.
- ▶ An alternant decoder based on H only up to $\mathbf{deg}(\mathbf{g}(\mathbf{X}))$ errors.

The strategy again

► $GO_2(L, g) = GO_2(L, g^2)$

Public parity-check matrix H is based **on $g(\mathbf{X})$**

Decoder \mathcal{D}_g is based **on $g(\mathbf{X})^2$** resp. **on H_0**

Adapting the decoder

- ▶ $H_0 = SH$ for some regular matrix S

Once we have S , we work with SH instead of H :

$$u(SH)^T = e(SH)^T = [0 \dots e_{i_0} \dots e_{i_{t-1}} \dots 0] \quad (SH)^T = [S'_0 \dots S'_{r-1}]$$

- ▶ The decoder can correct now up to $2r$ errors without any change.

Under attack

- ▶ $GO_2(L, g) = GO_2(L, g^2)$
- ▶ Faugère et al. launched an attack against QD (2010)
- ▶ The attack **failed!**
- ▶ Their alternant decoder could only correct $t/2$ errors.

Note:

- ▶ The attack might work, if the private decoder were based on H and not on H_0 !

Warning

You might ask how a generalized Cauchy matrix for $g(X)^2$ might look like.

$$C(z, L) = (C_{ij}) = \begin{bmatrix} \frac{1}{z_0 - L_0} & \frac{1}{z_0 - L_1} & \cdots & \frac{1}{z_0 - L_{n-1}} \\ \frac{1}{(z_0 - L_0)^2} & \frac{1}{(z_0 - L_1)^2} & \cdots & \frac{1}{(z_0 - L_{n-1})^2} \\ \frac{1}{z_1 - L_0} & \frac{1}{z_1 - L_1} & \cdots & \frac{1}{z_1 - L_{n-1}} \\ \frac{1}{(z_1 - L_0)^2} & \frac{1}{(z_1 - L_1)^2} & \cdots & \frac{1}{(z_1 - L_{n-1})^2} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{z_{t-1} - L_0} & \frac{1}{z_{t-1} - L_1} & \cdots & \frac{1}{z_{t-1} - L_{n-1}} \\ \frac{1}{(z_{t-1} - L_0)^2} & \frac{1}{(z_{t-1} - L_1)^2} & \cdots & \frac{1}{(z_{t-1} - L_{n-1})^2} \end{bmatrix} \in \mathbb{F}^{2t \times n}$$

Do not base H on that. The attack might work!

Part VII

Flexible QD

Flexible QD (FQD)

QD generates quasi-dyadic matrices in Cauchy form, but has restrictions:

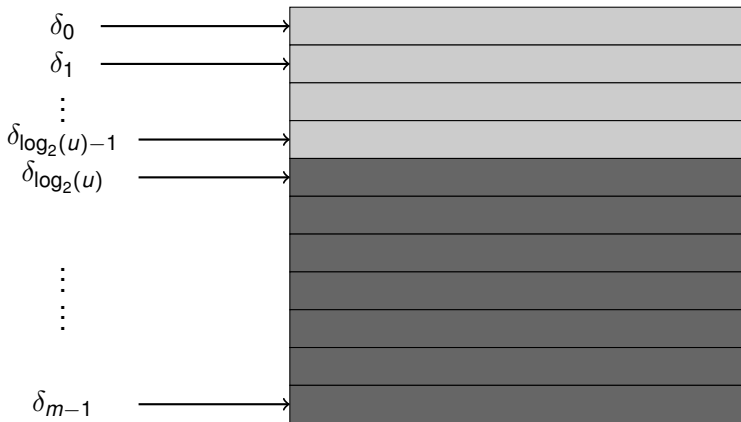
- ▶ It generates a fully dyadic $N \times N$ matrix, with $N \gg n$
- ▶ If $N = 2^m - t$ this is not good in terms of CFS.

Goal:

- ▶ Improve QD: make $n = 2^m - t$ possible.

FQD: Step 1

- ▶ Generate a $m \times m$ random binary nonsingular matrix M



FQD: Step 2

- ▶ Generate a $u \times u$ dyadic matrix using the upper part of M

$$z_{i_0} = \bigoplus_{b=0}^{\log_2(u)-1} i_0[b] \delta_b \quad (0 \leq i_0 < u)$$

- ▶ $i_0[b]$ denotes the $(b+1)$ -th bit of i in binary form
- ▶ δ_b denotes the $(b+1)$ -th row from the top of M
($0 \leq b < \log_2(u) - 1$)

FQD: Step 3

- ▶ Choose distinct Δ'_{i_1} and Δ_{j_1} from a linear combination of the bottom $m - \log_2(u)$ rows of M
- ▶ Shift now the previous $u \times u$ dyadic matrix by Δ'_{i_1} and Δ_{j_1} :

$$z_{i_1 u + i_0} = z_{i_0} \oplus \Delta'_{i_1} \quad (1 \leq i_1 < \lceil t/u \rceil)$$

$$L_{j_1 u + j_0} = z_{j_0} \oplus \Delta_{j_1} \quad (0 \leq j_1 < \lceil n/u \rceil)$$

The next slide will give an idea what is going on ...

FQD: Step 4

$$\begin{array}{cccccccc}
 z_0 & z_1 & \dots & z_{U-1} & \dots & z_0 \oplus \Delta_{j_1} & z_1 \oplus \Delta_{j_1} & \dots & z_{U-1} \oplus \Delta_{j_1} & \dots \\
 z_1 & z_0 & \dots & z_{U-2} & \dots & z_1 \oplus \Delta_{j_1} & z_0 \oplus \Delta_{j_1} & \dots & z_{U-2} \oplus \Delta_{j_1} & \dots \\
 z_3 & z_2 & \dots & z_{U-3} & \dots & z_3 \oplus \Delta_{j_1} & z_2 \oplus \Delta_{j_1} & \dots & z_{U-3} \oplus \Delta_{j_1} & \dots \\
 \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots & \dots \\
 z_{U-1} & z_{U-2} & \dots & z_0 & \dots & z_{U-1} \oplus \Delta_{j_1} & z_{U-2} \oplus \Delta_{j_1} & \dots & z_0 \oplus \Delta_{j_1} & \dots \\
 \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots & \dots \\
 z_0 \oplus \Delta_{i_1} & z_1 \oplus \Delta_{i_1} & \dots & z_{U-1} \oplus \Delta_{i_1} & \dots & \vdots & \vdots & \dots & \vdots & \dots \\
 z_1 \oplus \Delta_{i_1} & z_0 \oplus \Delta_{i_1} & & z_{U-2} \oplus \Delta_{i_1} & \dots & & & & & \dots \\
 z_3 \oplus \Delta_{i_1} & z_2 \oplus \Delta_{i_1} & \dots & z_{U-3} \oplus \Delta_{i_1} & \dots & \dots & \vdots & \dots & \vdots & \dots \\
 \vdots & \vdots & \vdots & \vdots & \dots & & & & & \dots \\
 z_{U-1} \oplus \Delta_{i_1} & z_{U-2} \oplus \Delta_{i_1} & \dots & z_0 \oplus \Delta_{i_1} & \dots & & & & & \dots \\
 \vdots & \vdots & \vdots & \vdots & \dots & \dots & \vdots & \dots & \vdots & \dots
 \end{array}$$

FQD: Step 5

Note that $z_0 = 0$, so we have:

$$\begin{array}{cccccccccccc}
 z_0 & z_1 & \dots & z_{U-1} & L_0 & L_1 & \dots & L_{U-1} & L_U & \dots & L_{n-1} \\
 z_1 & z_0 & \dots & z_{U-2} & L_1 & L_0 & \dots & L_{U-2} & L_{U+1} & \dots & L_{n-2} \\
 z_3 & z_2 & \dots & z_{U-3} & L_2 & L_3 & \dots & L_{U-3} & & \dots & \\
 \vdots & \vdots & & \vdots & \vdots & \vdots & \dots & & & & \vdots \\
 z_{U-1} & z_{U-2} & \dots & z_0 & L_{U-1} & L_{U-2} & \dots & L_0 & \dots & \dots & \\
 \vdots & \vdots & & \vdots & \vdots & \vdots & \dots & \vdots & \dots & & \\
 z_0 \oplus \Delta_{i_1} & z_1 \oplus \Delta_{i_1} & \dots & z_{U-1} \oplus \Delta_{i_1} & & & & & & & \\
 z_1 \oplus \Delta_{i_1} & z_0 \oplus \Delta_{i_1} & & z_{U-2} \oplus \Delta_{i_1} & & & & & & & \\
 & & & & & & & & & & \\
 z_3 \oplus \Delta_{i_1} & z_2 \oplus \Delta_{i_1} & \dots & z_{U-3} \oplus \Delta_{i_1} & \dots & \vdots & \dots & \vdots & & & \vdots \\
 \vdots & \vdots & \vdots & \vdots & & & & & & & \vdots \\
 z_{U-1} \oplus \Delta_{i_1} & z_{U-2} \oplus \Delta_{i_1} & \dots & z_0 \oplus \Delta_{i_1} & & & & & & & \\
 \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \dots & \vdots & & & \vdots
 \end{array}$$

FQD: Step 6

Having seen those slides, maybe you believe that the final matrix $H = (h_{ij})$ is quasi-dyadic then:

$$H = (h_{ij}) = \left(\frac{1}{z_i \oplus L_j} \right) = \begin{bmatrix} \frac{1}{z_0 \oplus L_0} & \frac{1}{z_0 \oplus L_1} & \frac{1}{z_0 \oplus L_2} & \cdots & \frac{1}{z_0 \oplus L_{n-1}} \\ \frac{1}{z_1 \oplus L_0} & \frac{1}{z_1 \oplus L_1} & \frac{1}{z_1 \oplus L_2} & \cdots & \frac{1}{z_1 \oplus L_{n-1}} \\ \frac{1}{z_2 \oplus L_0} & \frac{1}{z_2 \oplus L_1} & \frac{1}{z_2 \oplus L_2} & \cdots & \frac{1}{z_2 \oplus L_{n-1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{z_{t-1} \oplus L_0} & \frac{1}{z_{t-1} \oplus L_1} & \frac{1}{z_{t-1} \oplus L_2} & \cdots & \frac{1}{z_{t-1} \oplus L_{n-1}} \end{bmatrix}$$

If not, the next slide contains a sketch of the calculations.

FQD: Sketch of proof

► Using $z_0 = 0$; $z_i \oplus z_j = z_{i \oplus j}$; $\Delta_{i_1} \oplus \Delta_{j_1} = \Delta_{i_1 \oplus j_1}$; we have:

$$\begin{aligned}
 \frac{1}{h_{ij}} &= \underbrace{z_{i_1} u + i_0}_i \oplus \underbrace{L_{j_1} u + j_0}_j &= (z_{i_0} \oplus \Delta_{i_1}) \oplus (z_{j_0} \oplus \Delta_{j_1}) &= \\
 &= (z_{i_0} \oplus z_{j_0}) \oplus (\Delta_{i_1} \oplus \Delta_{j_1}) &= z_{i_0 \oplus j_0} \oplus \Delta_{i_1 \oplus j_1} &= \\
 &= L_{(i_1 \oplus j_1)u + (i_0 \oplus j_0)} &= \underbrace{L_{(i_1 u + i_0)}}_i \oplus \underbrace{L_{(j_1 u + j_0)}}_j &= \\
 &= z_0 \oplus L_{i \oplus j} &= \frac{1}{h_{i \oplus j}} &=
 \end{aligned}$$

FQD: Remarks

- ▶ $t \leq u \Rightarrow z_{i_1 u + i_0}$ can be ignored
- ▶ $\lceil t/u \rceil u > t \Rightarrow$ remove $\lceil t/u \rceil u - t$ rows
- ▶ $\lceil n/u \rceil u > n \Rightarrow$ remove $\lceil n/u \rceil u - n$ columns
- ▶ Size: $t \times n$
- ▶ Now like in QD: co-tracing and systematic form.

Part VIII

HyMES

HyMES (Hybrid McEliece Scheme)

HyMES is a hybrid implementation of the McEliece scheme by Bhaskar Biswas and Nicolas Sendrier. It has two modifications compared to the original McEliece scheme:

Higher information rate by putting some data into the error pattern (similar to Niederreiter).

Reduced public key size by using a generator matrix in echelon form.

HyMES with QD/FQD

We use HyMES at the moment as our basic platform, but we are about to include some modifications.

- ▶ Compact McEliece keys with based on QD/FDQ.

Future improvements might be:

- ▶ Documentation in a "literate programming" style.
- ▶ Assembler routines for speed.
- ▶ Ranking/Unranking algorithm.

Part IX

Anything else?

Summary

- ▶ Compact McEliece keys are possible with binary separable Goppa codes.
- ▶ These codes allow for a compact representation in form of quasi-dyadic parity-check matrices.
- ▶ The compact representation of the parity-check matrices can be used 'as-is'.
- ▶ At the moment, there is no attack known against McEliece based on quasi-dyadic codes.
- ▶ Special thanks to Paulo Barreto for a lot of helpful mails.

Mission accomplished?

Well, there are people who also believed it ...

... but at the moment things are looking good!

Questions?

Material used and further reading

- ▶ Paulo S.L.M Barreto
The fast Walsh-Hadamard transform (Draft)
- ▶ Paulo S.L.M Barreto
Post-Quantum Cryptography
- ▶ Rafael Misoczki and Paulo S.L.M. Barreto
Compact McEliece Keys from Goppa Codes
- ▶ Kazukuni Kobara
Flexible Quasi-Dyadic Code-Based Public-Key Encryption and Signature
- ▶ Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich
Algebraic Cryptanalysis of McEliece Variants with Compact Keys
- ▶ Steven Roman, Coding and Information Theory
- ▶ F.J. MacWilliams, N.J.A. Sloane
The Theory of Error-correcting Codes

- ▶ Robert J. McEliece, The Theory of Information and Coding, Encyclopedia of Mathematics and its Applications
- ▶ K.K. TZENG, K. Zimmermann
On Extending Goppa Codes to Cyclic Codes
- ▶ Dana Randall
Efficient Generation of Random Nonsingular Matrices