Adaptive attacks against FESTA without input validation or constant-time implementation

Tomoki Moriya[1], Hiroshi Onuki[2], Maozhi Xu[3], and Guoqing Zhou[3]

[1]School of Computer Science, University of Birmingham
[2]Department of Mathematical Informatics, The University of Tokyo
[3]School of Mathematical Sciences, Peking University

PQCRYPTO 2024

Result

FESTA: An isogeny-based PKE proposed by Basso, Maino, and Pope
"*FESTA: Fast Encryption from Supersingular Torsion Attacks*" (ASIACRYPT 2023)

Result

FESTA: An isogeny-based PKE proposed by Basso, Maino, and Pope
"*FESTA: Fast Encryption from Supersingular Torsion Attacks*" (ASIACRYPT 2023)

Proposed adaptive attacks for FESTA variants in which there are no

1. input validation
2. constant-time implementation

Background

Elliptic curve

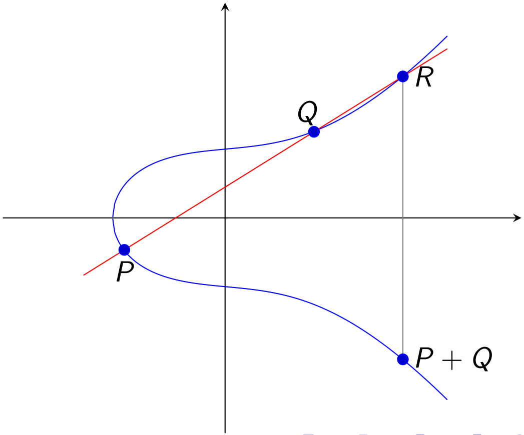
p: a prime
k: a field of characteristic p
$E: y^2 = x^3 + ax + b \ /k$
E is an elliptic curve $/k \Leftrightarrow 4a^3 + 27b^2 \neq 0$

Elliptic curve

p: a prime
k: a field of characteristic p
$E: y^2 = x^3 + ax + b \ / k$
E is an elliptic curve $/k \Leftrightarrow 4a^3 + 27b^2 \neq 0$

- E has a commutative group structure.

- $E[N] := \{P \in E \mid [N]P = \overbrace{P + \cdots + P}^{N} = 0\}$
  $E[N] \cong (\mathbb{Z}/N\mathbb{Z})^2$ for N with $\gcd(N, p) = 1$.

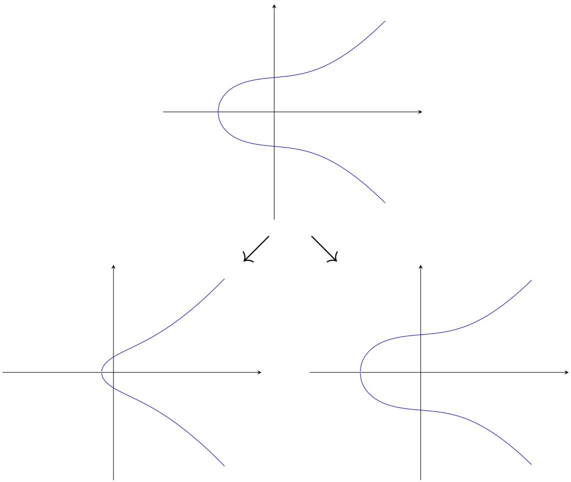


Isogeny

E_1, E_2: elliptic curves

$\phi\colon E_1 \to E_2$ is an isogeny $\Leftrightarrow$ ϕ is

- a morpihsm (rational function),
- a group morphism,
- surjective
- with a finite kernel.



Isogeny

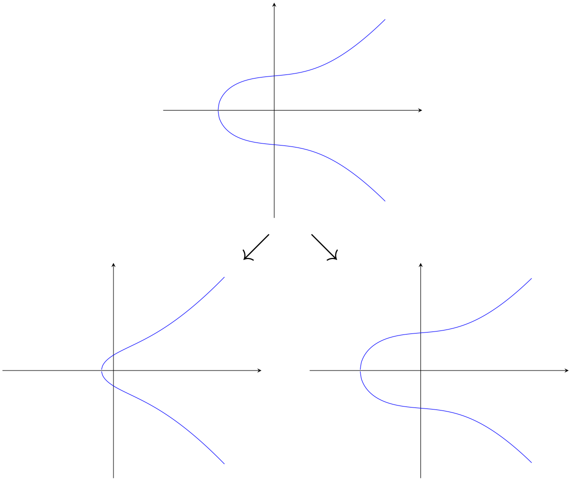
E_1, E_2: elliptic curves

$\phi\colon E_1 \to E_2$ is an isogeny $\Leftrightarrow$ ϕ is

- a morpihsm (rational function),
- a group morphism,
- surjective
- with a finite kernel.

(Suppose that ϕ is separable.)

- $\deg \phi := \# \ker \phi$
- $\hat{\phi}\colon E_2 \to E_1$ is the dual isogeny of ϕ
  $\Leftrightarrow$ $\phi \circ \hat{\phi} = [\deg \phi]$ and $\hat{\phi} \circ \phi = [\deg \phi]$.



Isogeny graph

Vertices: Elliptic curves
Edges: Isogenies

Isogeny graph

Vertices: Elliptic curves
Edges: Isogenies

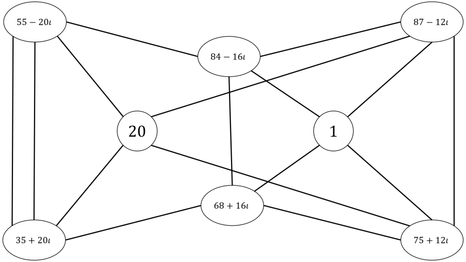


Figure: $p = 97$, isogenies of degree 3

Isogeny graph

Vertices: Elliptic curves
Edges: Isogenies

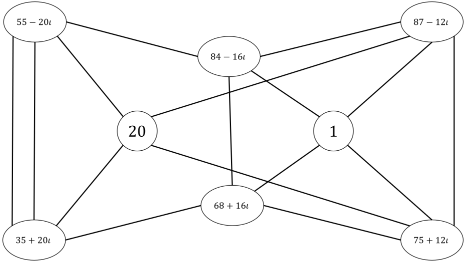


Figure: $p = 97$, isogenies of degree 3

isogeny	$\longleftrightarrow$	path
$\deg \phi$	$\longleftrightarrow$	"length" of the path
$\hat{\phi}$	$\longleftrightarrow$	the backtracking path of ϕ

Isogeny computation (random walking):

Isogeny computation and Isogeny Problem

Isogeny computation (random walking):

E: an elliptic curve
G: a finite subgroup of E

Isogeny computation and Isogeny Problem

Isogeny computation (random walking):

E: an elliptic curve
G: a finite subgroup of E

$\xrightarrow[\text{[Vélu (1971)]}]{\text{Vélu's formula}}$

An isogeny $\phi\colon E \to E'$ with $\ker \phi = G$

Isogeny computation and Isogeny Problem

Isogeny computation (random walking):

E: an elliptic curve
G: a finite subgroup of E

$\xrightarrow[\text{[Vélu (1971)]}]{\text{Vélu's formula}}$

An isogeny $\phi\colon E \to E'$ with $\ker \phi = G$

Isogeny Problem (path-finding):

Isogeny computation and Isogeny Problem

Isogeny computation (random walking):

E: an elliptic curve
G: a finite subgroup of E

$\xrightarrow[\text{[Vélu (1971)]}]{\text{Vélu's formula}}$

An isogeny $\phi\colon E \to E'$ with $\ker \phi = G$

Isogeny Problem (path-finding):

Isogeneous elliptic curves E, E'

Isogeny computation and Isogeny Problem

Isogeny computation (random walking):

E: an elliptic curve
G: a finite subgroup of E

$\xrightarrow[\text{[Vélu (1971)]}]{\text{Vélu's formula}}$

An isogeny $\phi\colon E \to E'$ with $\ker \phi = G$

Isogeny Problem (path-finding):

An isogeny $\phi\colon E \to E'$ or $G := \ker \phi$

$\xleftarrow[\text{hard}]{\text{Isogeny Problem}}$

Isogeneous elliptic curves E, E'

Isogeny Problem with torsion points information (1/2)

Remind that $E[N] \cong (\mathbb{Z}/N\mathbb{Z})^2$.
$\{P, Q\}$: a basis of $E[N]$

Isogeny Problem with torsion points information (1/2)

Remind that $E[N] \cong (\mathbb{Z}/N\mathbb{Z})^2$.
$\{P, Q\}$: a basis of $E[N]$

CSSI Problem [Jao and De Feo (PQCRYPTO 2011)]

(E, P, Q) and $(E', \phi(P), \phi(Q)) \rightsquigarrow \phi$

Isogeny Problem with torsion points information (1/2)

Remind that $E[N] \cong (\mathbb{Z}/N\mathbb{Z})^2$.
$\{P, Q\}$: a basis of $E[N]$

CSSI Problem [Jao and De Feo (PQCRYPTO 2011)]

(E, P, Q) and $(E', \phi(P), \phi(Q)) \rightsquigarrow \phi$

This problem can be solved in polynomial time. (**the SIDH attacks**)

- Castryck and Decru "*An Efficient Key Recovery Attack on SIDH*" (EUROCRYPT 2023)
- Maino, Martindale, Panny, Pope and Wesolowski "*A Direct Key Recovery Attack on SIDH*" (EUROCRYPT 2023)
- Robert "*Breaking SIDH in polynomial time*" (EUROCRYPT 2023)

Isogeny Problem with torsion points information (2/2)

Put $N = 2^b$.

Isogeny Problem with torsion points information (2/2)

Put $N = 2^b$.

CIST Problem [Basso, Maino and Pope (ASIACRYPT 2023)]

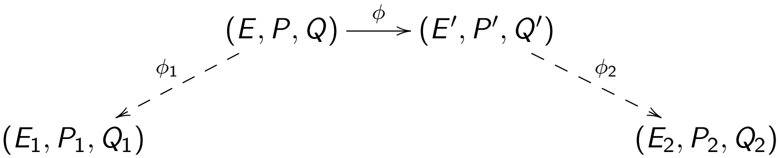
$(E, E', P, Q, P', Q', \mathcal{M}_b) \rightsquigarrow \phi$

$\mathcal{M}_b$: a commutative subgroup of $\mathrm{GL}_2(\mathbb{Z}/2^b\mathbb{Z})$

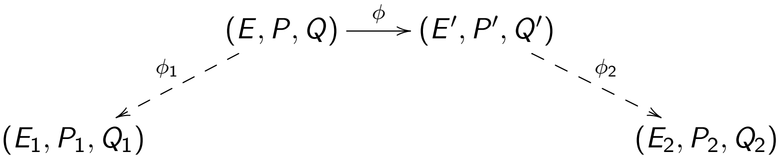
P', Q': points such that, for $\mathbf{A} \in \mathcal{M}_b$,

$$\begin{pmatrix} P' \\ Q' \end{pmatrix} = \mathbf{A} \begin{pmatrix} \phi(P) \\ \phi(Q) \end{pmatrix}$$

FESTA trapdoor function (1/3)



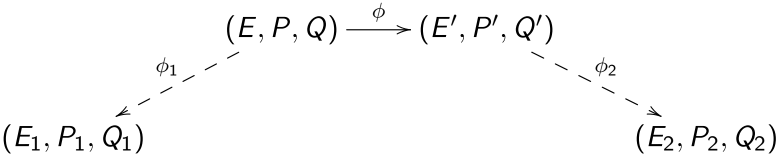
FESTA trapdoor function (1/3)



(P', Q'): masked by $\mathbf{A} \in \mathcal{M}_b$
(P_1, Q_1) and (P_2, Q_2): masked by $\mathbf{B} \in \mathcal{M}_b$

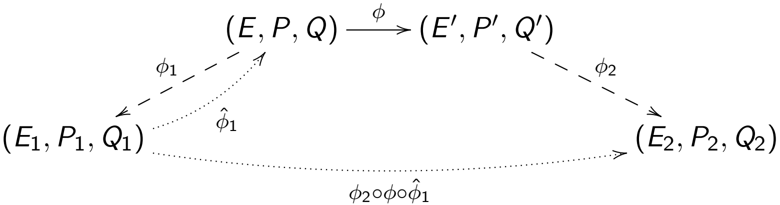
$$\begin{pmatrix} P' \\ Q' \end{pmatrix} = \mathbf{A} \begin{pmatrix} \phi(P) \\ \phi(Q) \end{pmatrix}, \quad \begin{pmatrix} P_1 \\ Q_1 \end{pmatrix} = \mathbf{B} \begin{pmatrix} \phi_1(P) \\ \phi_1(Q) \end{pmatrix}, \quad \begin{pmatrix} P_2 \\ Q_2 \end{pmatrix} = \mathbf{B} \begin{pmatrix} \phi_2(P') \\ \phi_2(Q') \end{pmatrix}.$$

FESTA trapdoor function (2/3)



Trapdoor:

FESTA trapdoor function (2/3)

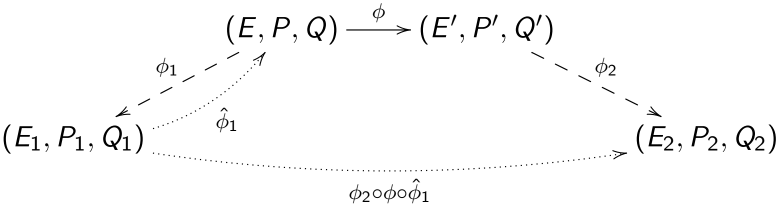


Trapdoor:

Let ${}^t(P_2', Q_2') = (\deg \phi_1) \cdot \mathbf{A}^{-1} \cdot {}^t(P_2, Q_2)$.
Since $\mathbf{AB} = \mathbf{BA}$, we have

$$\begin{pmatrix} P_2' \\ Q_2' \end{pmatrix} = (\phi_2 \circ \phi \circ \hat{\phi}_1) \begin{pmatrix} P_1 \\ Q_1 \end{pmatrix}.$$

FESTA trapdoor function (2/3)



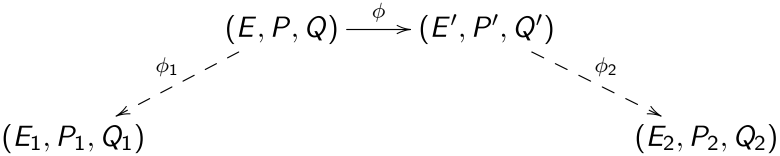
Trapdoor:

Let ${}^{t}(P_2', Q_2') = (\deg \phi_1) \cdot \mathbf{A}^{-1} \cdot {}^{t}(P_2, Q_2)$.
Since $\mathbf{AB} = \mathbf{BA}$, we have

$$\begin{pmatrix} P_2' \\ Q_2' \end{pmatrix} = (\phi_2 \circ \phi \circ \hat{\phi}_1) \begin{pmatrix} P_1 \\ Q_1 \end{pmatrix}.$$

$\longrightarrow$ One who knows $\mathbf{A}$ can obtain ϕ_1, ϕ_2 and $\mathbf{B}$.

FESTA trapdoor function (3/3)

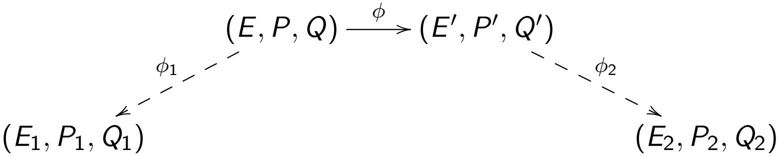


Trapdoor function

Public key: (E, P, Q, E', P', Q')
Secret key: $\mathbf{A}$
Input: $\phi_1, \phi_2, \mathbf{B}$
Output: $(E_1, P_1, Q_1, E_2, P_2, Q_2)$

Inverse map: Hard to be computed without $\mathbf{A}$

FESTA trapdoor function (3/3)



Trapdoor function

Public key: (E, P, Q, E', P', Q')
Secret key: $\mathbf{A}$
Input: $\phi_1, \phi_2, \mathbf{B}$
Output: $(E_1, P_1, Q_1, E_2, P_2, Q_2)$

Inverse map: Hard to be computed without $\mathbf{A}$

Input validation: If $\mathbf{B}$ does not belong to $\mathcal{M}_b$, then the recipient rejects $\phi_1, \phi_2, \mathbf{B}$.

Adaptive attack against FESTA variants

Attack scenario

Bob (sender) tries to obtain the secret key **A** of Alice (recipient).

Attack scenario

Bob (sender) tries to obtain the secret key **A** of Alice (recipient).

$\rightarrow$ Bob sends an incorrect output ot to Alice.

Attack scenario

Bob (sender) tries to obtain the secret key **A** of Alice (recipient).

$\rightarrow$ Bob sends an incorrect output ot to Alice.

$\rightarrow$

- Alice succeeds in using the SIDH attacks.
- Alice fails to use the SIDH attacks.

Attack scenario

Bob (sender) tries to obtain the secret key **A** of Alice (recipient).

$\rightarrow$ Bob sends an incorrect output ot to Alice.

$\rightarrow$

- Alice succeeds in using the SIDH attacks.
- Alice fails to use the SIDH attacks.

$\rightarrow$ Assume that Bob can distinguish the above two cases.

$$O'(\mathrm{ot}) = \begin{cases} 1 & \text{(if Alice succeeds in using the SIDH attacks)} \\ 0 & \text{(if Alice fails to use the SIDH attacks)} \end{cases}$$

Attack scenario

Bob (sender) tries to obtain the secret key **A** of Alice (recipient).

→ Bob sends an incorrect output ot to Alice.

→

- Alice succeeds in using the SIDH attacks.
- Alice fails to use the SIDH attacks.

→ Assume that Bob can distinguish the above two cases.

$$O'(\mathrm{ot}) = \begin{cases} 1 & \text{(if Alice succeeds in using the SIDH attacks)} \\ 0 & \text{(if Alice fails to use the SIDH attacks)} \end{cases}$$

→ He obtains **A** (solves the CIST problem).

Attack scenario

Bob (sender) tries to obtain the secret key **A** of Alice (recipient).

$\rightarrow$ Bob sends an incorrect output ot to Alice.

$\rightarrow$

- Alice succeeds in using the SIDH attacks.
- Alice fails to use the SIDH attacks.

$\rightarrow$ Assume that Bob can distinguish the above two cases.

$$O'(\mathrm{ot}) = \begin{cases} 1 & \text{(if Alice succeeds in using the SIDH attacks)} \\ 0 & \text{(if Alice fails to use the SIDH attacks)} \end{cases}$$

$\rightarrow$ He obtains **A** (solves the CIST problem).

Main theorem

Theorem

Consider the CIST problem.
If there is an oracle O such that, for $\mathbf{B_1}, \mathbf{B_2} \in \mathrm{GL}_2(\mathbb{Z}/2^b\mathbb{Z}) \neq \mathcal{M}_b$,

$$O(\mathbf{B_1}, \mathbf{B_2}) = \begin{cases} 1 & (\textit{if } \mathbf{AB_1} = \mathbf{B_2A}) \\ 0 & (\textit{if } \mathbf{AB_1} \neq \mathbf{B_2A}) \end{cases},$$

then there is a polynomial-time algorithm to compute $\mathbf{A}$.

Main theorem

Theorem

Consider the CIST problem.
If there is an oracle O such that, for $\mathbf{B_1}, \mathbf{B_2} \in \mathrm{GL}_2(\mathbb{Z}/2^b\mathbb{Z}) \neq \mathcal{M}_b$,

$$O(\mathbf{B_1}, \mathbf{B_2}) = \begin{cases} 1 & (\textit{if } \mathbf{AB_1} = \mathbf{B_2 A}) \\ 0 & (\textit{if } \mathbf{AB_1} \neq \mathbf{B_2 A}) \end{cases},$$

then there is a polynomial-time algorithm to compute $\mathbf{A}$.

It is easy to see that

$$O(\mathbf{B_1}, \mathbf{B_2}) = O'((E_1, E_2, \mathbf{B_1} \cdot {}^t(\phi_1(P), \phi_1(Q)), \mathbf{B_2} \cdot {}^t(\phi_2(P'), \phi_2(Q')))).$$

Main theorem

Theorem

Consider the CIST problem.
If there is an oracle O such that, for $\mathbf{B_1}, \mathbf{B_2} \in \mathrm{GL}_2(\mathbb{Z}/2^b\mathbb{Z}) \neq \mathcal{M}_b$,

$$O(\mathbf{B_1}, \mathbf{B_2}) = \begin{cases} 1 & (\textit{if } \mathbf{AB_1} = \mathbf{B_2A}) \\ 0 & (\textit{if } \mathbf{AB_1} \neq \mathbf{B_2A}) \end{cases},$$

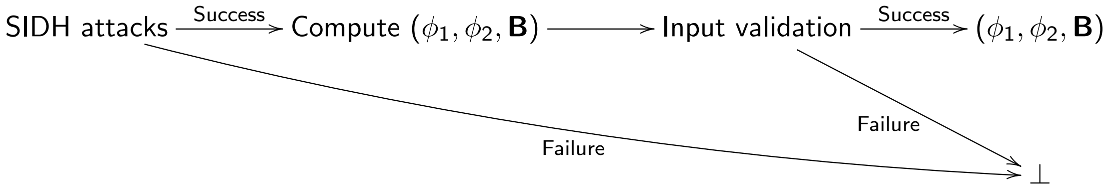
then there is a polynomial-time algorithm to compute $\mathbf{A}$.

It is easy to see that

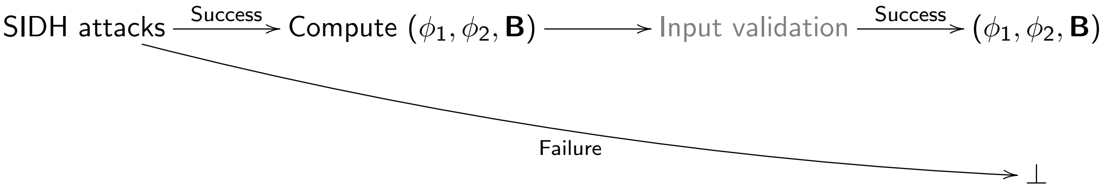
$$O(\mathbf{B_1}, \mathbf{B_2}) = O'((E_1, E_2, \mathbf{B_1} \cdot {}^t(\phi_1(P), \phi_1(Q)), \mathbf{B_2} \cdot {}^t(\phi_2(P'), \phi_2(Q')))).$$

How do we construct the oracle O'?
Note: O' is NOT a decryption oracle because of the input validation.

How to construct O'

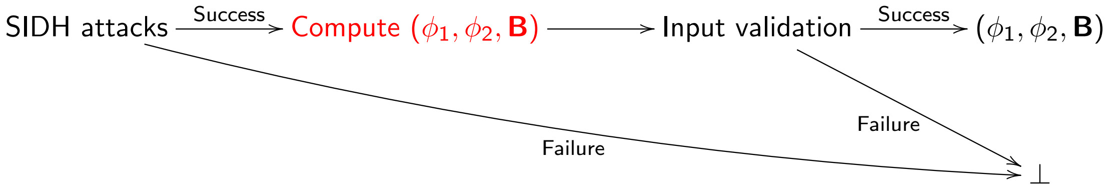


How to construct O'



1. No input validation
   - "Wrong use" of the FESTA trapdoor function
   - Other schemes based on the CIST problem (e.g., IS-CUBE, LIT-SiGamal)

How to construct O'



1. No input validation
   - "Wrong use" of the FESTA trapdoor function
   - Other schemes based on the CIST problem (e.g., IS-CUBE, LIT-SiGamal)
2. Non-constant-time implementation

Summary

There is an adaptive attack for a FESTA variant if it has no

1. input validation
2. constant-time implementation

Summary

There is an adaptive attack for a FESTA variant if it has no

1. input validation
2. constant-time implementation

Thank you for listening! Any questions?

Sketch of the proof of the main theorem (1/3)

Theorem

Consider the CIST problem.
If there is an oracle O such that, for $\mathbf{B_1}, \mathbf{B_2} \in \mathrm{GL}_2(\mathbb{Z}/2^b\mathbb{Z}) \neq \mathcal{M}_b$,

$$O(\mathbf{B_1}, \mathbf{B_2}) = \begin{cases} 1 & (\textit{if } \mathbf{AB_1} = \mathbf{B_2A}) \\ 0 & (\textit{if } \mathbf{AB_1} \neq \mathbf{B_2A}) \end{cases},$$

then there is a polynomial-time algorithm to compute $\mathbf{A}$.

Sketch of the proof of the main theorem (1/3)

Theorem

Consider the CIST problem.
If there is an oracle O such that, for $\mathbf{B_1}, \mathbf{B_2} \in \mathrm{GL}_2(\mathbb{Z}/2^b\mathbb{Z}) \neq \mathcal{M}_b$,

$$O(\mathbf{B}_1, \mathbf{B}_2) = \begin{cases} 1 & (\textit{if } \mathbf{AB}_1 = \mathbf{B}_2\mathbf{A}) \\ 0 & (\textit{if } \mathbf{AB}_1 \neq \mathbf{B}_2\mathbf{A}) \end{cases},$$

then there is a polynomial-time algorithm to compute $\mathbf{A}$.

We show the proof of this theorem in the case that $\mathcal{M}_b$ is the set of circulant matrices.

Sketch of the proof of the main theorem (2/3)

Put

$$\mathbf{A} = \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}, \quad \gamma^2 - \delta^2 = 1, \quad \gamma = \sum_{i=0}^{b-1} \gamma_i 2^i, \quad \delta = \sum_{i=0}^{b-1} \delta_i 2^i.$$

Sketch of the proof of the main theorem (2/3)

Put

$$\mathbf{A} = \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}, \quad \gamma^2 - \delta^2 = 1, \quad \gamma = \sum_{i=0}^{b-1} \gamma_i 2^i, \quad \delta = \sum_{i=0}^{b-1} \delta_i 2^i.$$

Define

$$O_{\mathrm{coeff}}(\varepsilon_1, \varepsilon_2) = O\left(\mathbf{B} + \begin{pmatrix} \varepsilon_1 & 0 \\ \varepsilon_2 & 0 \end{pmatrix}, \mathbf{B} + \begin{pmatrix} 0 & 0 \\ \varepsilon_1 & \varepsilon_2 \end{pmatrix}\right).$$

Sketch of the proof of the main theorem (2/3)

Put

$$\mathbf{A} = \begin{pmatrix} \gamma & \delta \\ \delta & \gamma \end{pmatrix}, \quad \gamma^2 - \delta^2 = 1, \quad \gamma = \sum_{i=0}^{b-1} \gamma_i 2^i, \quad \delta = \sum_{i=0}^{b-1} \delta_i 2^i.$$

Define

$$O_{\text{coeff}}(\varepsilon_1, \varepsilon_2) = O\left(\mathbf{B} + \begin{pmatrix} \varepsilon_1 & 0 \\ \varepsilon_2 & 0 \end{pmatrix}, \mathbf{B} + \begin{pmatrix} 0 & 0 \\ \varepsilon_1 & \varepsilon_2 \end{pmatrix}\right).$$

Lemma

$$O_{coeff}(\varepsilon_1, \varepsilon_2) = \begin{cases} 1 & (\textit{if } \varepsilon_1\gamma + \varepsilon_2\delta = 0) \\ 0 & (\textit{if } \varepsilon_1\gamma + \varepsilon_2\delta \neq 0) \end{cases}$$

Sketch of the proof of the main theorem (3/3)

k = 0:

Sketch of the proof of the main theorem (3/3)

k = 0:

$$O_{\text{coeff}}(2^{b-1}, 0) = \begin{cases} 1 & (\text{if } 2^{b-1}\gamma = 0 \iff \gamma_0 = 0) \\ 0 & (\text{if } 2^{b-1}\gamma \neq 0 \iff \gamma_0 = 1) \end{cases}$$

Sketch of the proof of the main theorem (3/3)

k = 0:

$$O_{\text{coeff}}(2^{b-1}, 0) = \begin{cases} 1 & (\text{if } 2^{b-1}\gamma = 0 \iff \gamma_0 = 0) \\ 0 & (\text{if } 2^{b-1}\gamma \neq 0 \iff \gamma_0 = 1) \end{cases}$$

$\longrightarrow$ We can detect (γ_0, δ_0). (Note that $\gamma_0^2 - \delta_0^2 \equiv 1 \pmod 2$.)

Sketch of the proof of the main theorem (3/3)

$\mathbf{k = 0}$:

$$O_{\text{coeff}}(2^{b-1}, 0) = \begin{cases} 1 & (\text{if } 2^{b-1}\gamma = 0 \iff \gamma_0 = 0) \\ 0 & (\text{if } 2^{b-1}\gamma \neq 0 \iff \gamma_0 = 1) \end{cases}$$

$\longrightarrow$ We can detect (γ_0, δ_0). (Note that $\gamma_0^2 - \delta_0^2 \equiv 1 \pmod 2$.)

$\mathbf{2 \leq k \leq b - 2}$:

Assume that we already have $\gamma^{(k-1)}$ and $\delta^{(k-1)}$ and $\delta_0 = 0$.

- $-2^{b-k-1}\delta^{(k-1)} \cdot \gamma + 2^{b-k-1}\gamma^{(k-1)} \cdot \delta = \delta_k 2^{b-1}$

Sketch of the proof of the main theorem (3/3)

k = 0:

$$O_{\text{coeff}}(2^{b-1}, 0) = \begin{cases} 1 & (\text{if } 2^{b-1}\gamma = 0 \iff \gamma_0 = 0) \\ 0 & (\text{if } 2^{b-1}\gamma \neq 0 \iff \gamma_0 = 1) \end{cases}$$

$\longrightarrow$ We can detect (γ_0, δ_0). (Note that $\gamma_0^2 - \delta_0^2 \equiv 1 \pmod 2$.)

2 ≤ k ≤ b − 2:

Assume that we already have $\gamma^{(k-1)}$ and $\delta^{(k-1)}$ and $\delta_0 = 0$.

- $-2^{b-k-1}\delta^{(k-1)} \cdot \gamma + 2^{b-k-1}\gamma^{(k-1)} \cdot \delta = \delta_k 2^{b-1}$

  $\longrightarrow O_{coeff}(-2^{b-k-1}\delta^{(k-1)}, 2^{b-k-1}\gamma^{(k-1)})$ reveals δ_k.

Sketch of the proof of the main theorem (3/3)

k = 0:

$$O_{\mathrm{coeff}}(2^{b-1}, 0) = \begin{cases} 1 & (\text{if } 2^{b-1}\gamma = 0 \iff \gamma_0 = 0) \\ 0 & (\text{if } 2^{b-1}\gamma \neq 0 \iff \gamma_0 = 1) \end{cases}$$

$\longrightarrow$ We can detect (γ_0, δ_0). (Note that $\gamma_0^2 - \delta_0^2 \equiv 1 \pmod 2$.)

$2 \le k \le b - 2$:

Assume that we already have $\gamma^{(k-1)}$ and $\delta^{(k-1)}$ and $\delta_0 = 0$.

- $-2^{b-k-1}\delta^{(k-1)} \cdot \gamma + 2^{b-k-1}\gamma^{(k-1)} \cdot \delta = \delta_k 2^{b-1}$

  $\longrightarrow$ $O_{coeff}(-2^{b-k-1}\delta^{(k-1)}, 2^{b-k-1}\gamma^{(k-1)})$ reveals δ_k.

- $1 - {\gamma^{(k-1)}}^2 + {\delta^{(k-1)}}^2 = \gamma^2 - \delta^2 - {\gamma^{(k-1)}}^2 + {\delta^{(k-1)}}^2 \equiv \gamma_k 2^{k+1} \pmod{2^{k+2}}$

Sketch of the proof of the main theorem (3/3)

k = 0:

$$O_{\text{coeff}}(2^{b-1}, 0) = \begin{cases} 1 & (\text{if } 2^{b-1}\gamma = 0 \iff \gamma_0 = 0) \\ 0 & (\text{if } 2^{b-1}\gamma \neq 0 \iff \gamma_0 = 1) \end{cases}$$

$\longrightarrow$ We can detect (γ_0, δ_0). (Note that $\gamma_0^2 - \delta_0^2 \equiv 1 \pmod 2$.)

$2 \leq k \leq b - 2$:

Assume that we already have $\gamma^{(k-1)}$ and $\delta^{(k-1)}$ and $\delta_0 = 0$.

- $-2^{b-k-1}\delta^{(k-1)} \cdot \gamma + 2^{b-k-1}\gamma^{(k-1)} \cdot \delta = \delta_k 2^{b-1}$

  $\longrightarrow$ $O_{coeff}(-2^{b-k-1}\delta^{(k-1)}, 2^{b-k-1}\gamma^{(k-1)})$ reveals δ_k.

- $1 - \gamma^{(k-1)^2} + \delta^{(k-1)^2} = \gamma^2 - \delta^2 - \gamma^{(k-1)^2} + \delta^{(k-1)^2} \equiv \gamma_k 2^{k+1} \pmod{2^{k+2}}$

  $\longrightarrow$ $1 - \gamma^{(k-1)^2} + \delta^{(k-1)^2} \mod 2^{k+1}$ reveals γ_k.